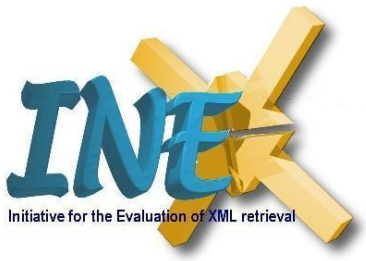


INEX'12 Workshop Pre-proceedings

Shlomo Geva, Jaap Kamps, Ralf Schenkel (editors)

September 17–20, 2012
Rome, Italy
http://inex.mmci.uni-saarland.de/

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 They are not peer reviewed, are not quality controlled, and contain known errors in content and editing.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.

Preface

Welcome to the eleventh 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 eleventh 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'12 was an exciting year for INEX in which we joined forces with CLEF and run our workshop as part of the CLEF labs in order to facilitate knowledge transfer between the evaluation forums. In total five research tracks were included, which studied different aspects of focused information access:

Linked Data Track investigating retrieval over a strongly structured collection of documents based on DBpedia and Wikipedia. The *Ad Hoc Search Task* has informational requests to be answered by the entities in DBpedia/Wikipedia. The *Faceted Search Task* asks for a restricted list of facets and facet-values that will optimally guide the searcher toward relevant information.

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.

Social Book Seach Track investigating techniques to support users in searching and navigating books, metadata and complementary social media. The *Social Book Search 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.

Tweet Contextualization 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."

The aim of the INEX'12 workshop is to bring together researchers who participated in the INEX'12 campaign. During the past year participating orga-

nizations 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 (Social Book Search Track), Microsoft Research (Social Book Search Track), the DBpedia (Linked Data Track), and the Wikimedia Foundation (Linked Data, Relevance Feedback, Snippet Retrieval, and Tweet Contextualization Tracks).

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!

September 2012

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 Barcelona)
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

Linked Data

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

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)

Social Books 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 Lugano)
Michael Preminger (Oslo and Akershus University College)

Tweet Contextualization

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

Table of Contents

Social Book Search Track.

Tweet Contextualization Track.

Back matter.

Overview of the INEX 2012 Linked Data Track

Qiuyue Wang[1], Jaap Kamps[2], Georgina Ramírez Camps[3], Maarten Marx[2], Anne Schuth[2], Martin Theobald[4], Sairam Gurajada[4], and Arunav Mishra[4]

[1]Renmin University of China, Beijing, China
[2]University of Amsterdam, Amsterdam, The Netherlands
[3]Universitat Pompeu Fabra, Barcelona, Spain
[4]Max Planck Institute for Informatics, Saarbrücken, Germany

Abstract. This paper provides an overview of the Linked Data Track that was newly introduced to the set of INEX tracks in 2012.

1 Introduction

The goal of the new Linked Data Track was to investigate retrieval techniques over a combination of textual and highly structured data, where rich textual contents from Wikipedia articles serve as the basis for retrieval and ranking, while addtional RDF properties carry key information about semantic relations among entities that cannot be captured by keywords alone. Our intension in organizing this new track thus follows one of the key themes of INEX, namely to explore and investigate if and how structural information could be exploited to improve the effectiveness of *ad-hoc* retrieval. In particular, we were interested in how this combination of data could be used together with structured queries to help users navigate or explore large sets of results (a task that is well-known from *faceted search* systems), or to address Jeopardy-style natural-language clues and questions (known, for example, from recent *question answering* settings over linked data collections, see for example [6]). The Linked Data Track thus aims to close the gap between IR-style keyword search and semantic-web-style reasoning techniques, with the goal to bring together different communities and to foster research at the intersection of Information Retrieval, Databases, and the Semantic Web.

As its core collection, the Linked Data Track employs a fusion of XML-ified Wikipedia articles with RDF properties from both DBpedia [4] and YAGO2 [5], the latter of which contain the article entity as either their subject (first argument) or object (second argument). The core data collection was based on the popular MediaWiki format[1], where we additionally replaced all Wiki-markup by syntactically valid XML tags, attributes, and CDATA sections. In addition, all internal Wikipedia links (including the article entity itself) have been enriched with links to both their corresponding DBpedia and YAGO2 entities (as far as available). In addition, participants were explicitly encouraged to make use of

[1] `http://dumps.wikimedia.org/enwiki/20110722/`

more RDF facts available from DBpedia and YAGO2, in particular for processing the reasoning-related faceted search and Jeopardy topics. For INEX 2012, we explored three different retrieval tasks:

- The classic **Ad-hoc Retrieval Task** investigates informational queries to be answered mainly by the textual contents of the Wikipedia articles.
- The **Faceted Search Task** employs a hand-crafted hierarchy of facets and facet-values obtained from DBpedia that aim to guide the searcher toward relevant information.
- The new **Jeopardy Task** employs natural-language Jeopardy clues which are manually translated into a semi-structured query format based on SPARQL with keyword filter conditions.

2 Data Collection

The new Wikipedia-LOD (v1.1) collection is hosted by the Max Planck Institute for Informatics and has been made available for download in May 2012 from the following link: `http://www.mpi-inf.mpg.de/inex-lod/wikipedia-lod-2012/`

The collection consists of 3 compressed tar.gz files and contains an overall amount of 3.1 Million individual XML articles. The uncompressed size of the collection is 61 GB. A detailed DTD file that describes the structure of the XML collection is also available from the above URL. Each Wikipedia-LOD article consists of a mixture of XML tags, attributes, and CDATA sections, containing infobox attributes, free-text contents, describing the entity or category that the article captures, and a section with both DBpedia and YAGO2 properties that are related to the article's entity. All sections contain links to other Wikipedia articles (including links to the corresponding DBpedia and YAGO2 resources), Wikipedia categories, and external Web pages.

Figure 1 shows an example of an XML-ified Wikipedia article about the entity `Albert_Einstein` by depicting the two main sections of the article:

i) the Wikipedia section, containing an XML-ified infobox, enhanced links pointing to DBpedia and YAGO2, and Wikipedia text contents with more XML markup, and
ii) the Linked Data section with RDF triples imported from both DBpedia and YAGO2 that contain the entity `Albert_Einstein` as either their subject or object.

Wikipedia To WikiXML Parser. For converting the raw Wikipedia articles into our XML format, we used a parser derived from the wiki2xml parser [3] provided by MediaWiki [1]. The parser generates an XML file from the raw Wikipedia article (originally in Wiki markup) by transforming infobox information to a proper XML representation, comprehending links with DBpedia and YAGO2 entities, and finally annotating each article with a list of RDF properties from the DBpedia and YAGO2 knowledge sources.

```
<lodxml xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' xml:lang='en'
  xmlns:xhtml='http://www.w3.org/1999/xhtml' encoding='UTF-8'>
     <article title='Albert_Einstein'>
      <wikipedia>

        <template type='Metadata'>                              Meta Data
           <tag name='id'>736</tag>
           <tag name='title'>Albert_Einstein</tag>
        </template>

        <infobox type='person'>                          InfoBox Property
           <tag name='name'>Albert Einstein</tag>
           <tag name='image'>Einstein 1921 portrait2.jpg</tag>
           .....
        </infobox>
        .........
        <paragraph>Albert Einstein was born in<space/>

        <link>                                                    LOD Links
           <wikilink href='http://en.wikipedia.org/wiki/Ulm'>Ulm</wikilink>
              <dbpedia href='http://dbpedia.org/resource/Ulm'/>
              <yago ref='Ulm'/>
        </link>
        ...........
     </wikipedia>

     <dbpediaproperties>                                  DBPedia Property
        <property name='http://dbpedia.org/property/birthPlace'>
           <object name='http://dbpedia.org/resource/German_Empire'/>
        </property>
        ..............
     </dbpediaproperties>

     <yagoproperties>                                                YAGO2 Property
          <property name='isKnownFor'><object name='Einstein_field_equations'/></property>
          <property name='type'><object name='wikicategory_German_pacifists'/></property>
          <property name='type'><object name='wikicategory_German_philosophers'/></property>
          <property name='type'><object name='wikicategory_German_physicists'/></property>
     </yagoproperties>

   </article>
</lodxml>
```

Fig. 1. XML-ified Wikipedia articles with DBpedia and YAGO2 properties

Collection Statistics. The Wikipedia-LOD collection currently contains 3.1 Million XML documents in 3 compressed tar.gz files counting to the size of 61 GB in uncompressed form. Table 1 provides more detailed numbers about different properties of the collection.

Linked Data Sources. In addition to the new core collection, which is based on XML-ified Wikipedia articles, the Linked Data Track explicitly encourages (but does not require) the use of current Linked Open Data dumps for DBpedia (v3.7) and YAGO2, which are available from the following URLs:

- DBpedia v3.7 (created in July 2011):
  `http://downloads.dbpedia.org/3.7/en/`
- YAGO2 core and full dumps (created on 2012-01-09):
  `http://www.mpi-inf.mpg.de/YAGO2-naga/YAGO2/`

Property	Count
XML Documents	3,164,041
XML Elements	1,173,255,397
Wikipedia Category Articles	266,134
Wikipedia Entity Articles	2,053,050
Wikipedia Entity Articles with Infoboxes	907,304
Other Wikipedia Articles	844,857
Resolved DBpedia Links	36,941,795
Resolved YAGO2 Links	32,941,667
Intra-Wiki Links	22,235,753
External Web Links	7,214,827
Imported DBpedia Properties	168,374,863
Imported YAGO2 Properties	23,634,511

Table 1. Wikipedia-LOD (v1.1) Collection Statistics

DBpedia and YAGO2 are two comprehensive, common-sense knowledge bases providing structured information that has been semi-automatically extracted mostly from Wikipedia infoboxes and categories. Both knowledge bases focus on extracting attribute-value pairs from Wikipedia infoboxes and category lists, which serve as basis for applying various information extraction techniques. They also contain geo-coordinates, links between Wikipedia pages, redirection and disambiguation pages, external links, and much more. Each Wikipedia page corresponds to a resource in DBpedia and YAGO2. The connection between the data sets is given in the "wikipedia_links_en.nt" file from DBpedia. The following entry, for example,

```
<http://dbpedia.org/resource/AccessibleComputing>
<http://xmlns.com/foaf/0.1/page>
<http://en.wikipedia.org/wiki/AccessibleComputing>
```

connects the DBpedia entity with the URI `http://dbpedia.org/resource/AccessibleComputing` with the Wikipedia page that is available under the URI `http://en.wikipedia.org/wiki/AccessibleComputing`.

The Linked Data Track was explicitly intended to be an "open track" and thus invited participants to include more Linked Data sources (see, for example, `http://linkeddata.org`) or other sources that go beyond "just" DBpedia and YAGO2. Any inclusion of further data sources was welcome, however, workshop submissions and follow-up research papers should explicitly mention these sources when describing their approaches.

3 Retrieval Tasks and Topics

3.1 Ad-hoc Task and Faceted Search Tasks

The Ad-hoc Task is to return a ranked list of results (Wikipedia pages) estimated relevant to the user's information need, which is typically formulated into a

keyword query. Given an exploratory 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 results to quickly identify the results of interest. It presents the user a list of facet-values to refine the query. After the user choosing from the suggested facet-values, the result list is narrowed down and 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. One of the key issues in faceted search systems is to recommend appropriate facet-values to help the user quickly identify what he/she really wants in the large set of results. The task aims to investigate different techniques of recommending facet-values.

This year, we did not ask participants to submit ad-hoc or faceted search topics. We generated and collected the topics from the following three sources. Firstly, we built a three-level hierarchy of topics as described in [7]. For example,

```
Vietnam
  Vietnam war
  Vietnam war movies
  Vietnam war facts
  Vietnam food
     Vietnam food recipes
     Vietnam food blog
  Vietnam travel
    Vietnam travel national park
    Vietnam travel airports
```

The topics on the top level are general topics, e.g., "Vietnam". We randomly created 5 general topics, i.e. "Vietnam", "guitar", "tango", "bicycle", and "music". For each general topic, we typed it into Google, and from Google's online suggestions, we chose 3 subtopics. For example, when you type in "Vietnam", Google may suggest "Vietnam war", "Vietnam food" or "Vietnam travel", and so on, which can be viewed as subtopics to "Vietnam". Furthermore, for each subtopic, we selected 2 sub-subtopics using Google Suggest again. Thus we formed a three-level hierarchy of topics, with 5 general topics, 15 subtopics and 30 sub-subtopics. Since the relevant answers for a topic can be treated as the union of the relevant answers of all its subtopics, only the leaf-level topics, i.e. 30 sub-subtopics need to be assessed. So we put the 30 sub-subtopics to the Ad-hoc Task and 20 non-leaf level topics to the Faceted Search Task. The relevance results for the ad-hoc topics will serve as the relevant results to their corresponding faceted search topics.

Secondly, we selected 20 topics from INEX 2009 and 2010 Ad-hoc Tracks to compare the performance of different data collections. Since we want to select challenging topics, we took 40 worst performed topics (with lowest average precisions) from the INEX 2009 Ad-hoc Track and 30 worst performed topics from the INEX 2010 Ad-hoc Track, and then randomly selected 10 topics from each set. In this process, we also found some natural general topics, "Normandy", "museum" and "social networ", which have multiple subtopics among the 20

topics that we collected. So we added the 3 topics to the set of faceted search topics.

Thirdly, to compare the performance of structured queries that were used in Jeopardy Task and unstructured queries, we added all the 90 keyword titles of Jeopardy topics into the set of ad-hoc topics. In total, we collected 140 ad-hoc topics and 23 faceted search topics, which are in the same format as that in previous years [8].

3.2 Jeopardy Task

The new Jeopardy Task investigated retrieval techniques over a set of 90 natural-language Jeopardy-style clues and questions, which have been manually translated into SPARQL query patterns that were enhanced with keyword-based filter conditions. Specifically, we investigated a data model, where every entity (in DBpedia or YAGO2) is associated with the Wikipedia article (contained in the Wikipedia-LOD v1.1 collection) that describes this entity. An XML file with 90 Jeopardy-style topics was made available available for download in June 2012 under the following URL:
`http://www.mpi-inf.mpg.de/inex-lod/LDT-2012-jeopardy-topics.xml`

For example, topic no. 2012301 from the current set of Jeopardy topics looks as follows:

```
<topic id="2012301" category="LAKES">
    <jeopardy_clue>Niagara Falls has its source of origin
    from this lake. </jeopardy_clue>
    <keyword_title>Niagara Falls source lake</keyword_title>
    <sparql_ft>
      Select ?q Where {
         <http://dbpedia.org/resource/Niagara_Falls>
               <http://dbpedia.org/property/watercourse> ?o .
         ?o <http://dbpedia.org/ontology/origin> ?q .
         Filter FTContains(?o, "river water course niagara") .
         Filter FTContains(?q, "lake origin of")}
    </sparql_ft>
</topic>
```

The `<jeopardy_clue>` element contains the original Jeopardy clue as a natural-language sentence; the `<keyword_title>` element contains a set of keywords that have been manually extracted from this title and will be reused as part of the Ad-hoc Retrieval Task; and the `<sparql_ft>` element contains a formulation of the natural-language sentence into a corresponding SPARQL pattern. The `<category>` attribute of the `<topic>` element may be used as an additional hint for disambiguating the query.

In the above query, the DBpedia entity `http://dbpedia.org/resource/Niagara_Falls` has been marked as the subject of the first triplet pattern, while both the object of the first triplet pattern and the subject and object of the second

triplet pattern are unknown. The two `FTContains` filter conditions however restrict both these subjects and objects to entities that should be associated with the keywords "river water course niagara" and"lake origin" via the content of their corresponding Wikipedia articles, respectively. The result of this query is exactly one target entity, namely the DBpedia resource `http://dbpedia.org/resource/Lake_Erie`.

Since this particular variant of processing SPARQL queries with full-text filter conditions is not a default functionality of current SPARQL engines (and queries should not be run against a standard RDF collection such as DBpedia or YAGO2 alone), participants were encouraged to develop individual solutions to index both the RDF and textual contents of the Wikipedia-LOD collection in order to process these queries. Adding full-text search to SPARQL queries is an ongoing research issue. While initial implementations and syntax proposals exist (see for example [2]), we are not aware of any SPARQL engine that currently allows for associating and indexing entire text documents along with RDF resources. We also remark that this particular LOD data model differs from most current SPARQL full-text approaches, as we impose keyword conditions over individual entities (resources) rather than entire facts (triplets).

4 Run Submissions

All run submissions were to be uploaded via the INEX website via the URL: `https://inex.mmci.uni-saarland.de/`. The due date for the submission of all LOD runs was July 14, 2012.

4.1 Ad-Hoc and Jeopardy Tasks

For the Ad-hoc and Jeopardy Tasks, each run must contain a maximum of 1,000 results per topic, ordered by decreasing value of relevance. For the Ad-hoc Task, each result is a Wikipedia article uniquely identified by its page ID. For the Jeopardy Task however, each query result could be a set of entities (identified by their corresponding Wikipedia page IDs) in case that the select clause contains more than one query variables. For relevance assessment and evaluation of the results, we require submission files to be in the familiar TREC format, with each row representing a single query result. In case the select clause contains more than one query variable as in a Jeopardy topic, the row should consist of a comma- or semicolon-separated list of target entity ID's. This list of entities must reflect the order of query variables as specified by the select clause of the Jeopardy topic.

```
<qid> Q0 <page_id_list> <rank> <rsv> <run_id>
```

Where:

- 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 a comma- or semicolon-separated list the ID's of the resulting Wikipedia page(s).
– The fourth column is the rank of the result.
– The 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 your group AND for the method used. Run tags must contain 12 or fewer letters and numbers, with NO punctuation, to facilitate labeling graphs with the tags.

An example submission thus may look as follows:

```
2012301 Q0 12 1 0.9999 2012UniXRun1
2012301 Q0 997 2 0.9998 2012UniXRun1
2012301 Q0 9989 3 0.9997 2012UniXRun1
```

Here we have three results for topic "2012301". The first result is the entity (i.e. Wikipedia page) with ID "12". The second result is the entity with ID "997", and the third result is the entity with ID "9989".

4.2 Faceted Search Task

For the Faceted Search Task, the organizers will provide a result file, which contains a result list of maximum 2000 results for each general topic. Based on the reference result file, a run submitted by a participant should be a XML file conforming to the following DTD, which contains a hierarchy of recommended facet-values for each topic, in which each node represents a facet-value and all of its children constitute the newly recommended facet-value list when the user selects this facet-value to refine the query. The maximum fan-out of each node in the hierarchy is restricted to be 20.

```
<!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>
```

Where:

– The root element is `<run>`, which has an ID type attribute, `rid`, representing the unique identifier of the 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. All the possible facet-value pairs are from the triples in DBpedia or YAGO2.
– The <fv>'s can be nested to form a hierarchy of facet-values.

An example submission is:

```
<run rid=2012UniXRun1>
<topic tid=2012001>
<fv f=dbpedia-owl:date v=1955-11-01>
  <fv f=dbpedia-owl:place v=dbpedia:South_Vietnam>
    <fv f=rdf:type v=dbpedia-owl:MilitaryConflict/>
    <fv f=rdf:type v=dbpedia-owl:Country/>
  </fv>
  <fv f=dbpedia-owl:place v=dbpedia:North_Vietnam>
    <fv f=rdbpprob:capital v=dbpedia:Ho_Chi_Minh_City/>
  </fv>
</fv>
...
</topic>
<topic tid=2012002>
...
</topic>
...
</run>
```

Here for the topic "2012001", the faceted search system first recommends the facet-value condition "dbpedia-owl:date = 1955-11-01" among other facet-value conditions, which are its siblings. If the user selects this condition to refine the query, the system will recommend a new list of facet-value conditions, which are "dbpedia-owl:place = dbpedia:South_Vietnam" and "dbpedia-owl:place = dbpedia:North_Vietnam". If the user then selects "dbpedia-owl:plac = dbpedia:North_Vietnam", the system will recommend the facet-value condition "rdbprob:capital = dbpedia:Ho_Chi_Minh_City". Note that no facet-value condition may occur twice on a path in the hierarchy.

5 Relevance Assessments and Evaluation Metrics

In total 20 ad-hoc search runs were submitted by 7 participants, i.e., *Ecole des Mines de Saint-Etienne* (EMSE), *Kasetsart University*, *Renmin University of China*, *University of Otago*, *Oslo University College*, *University of Amsterdam*, *Norwegian University of Science and Technology* (NTNU), and 5 valid Jeopardy runs were submitted by 2 participants, i.e., *Kasetsart University* and *Max-Planck Institute for Informatics* (MPI).

Assessment was done using the Amazon Mechanical Turk. We did not assess the 20 topics from the INEX 2009 and 2010 Ad-hoc Tracks as we could use the

assessment results done in previous years. We assessed the 30 sub-subtopics and 50 Jeopardy topics randomly selected from the 90 ones. For each sub-subtopic, we pooled all the submitted runs in a round-robin manner, and then picked up the top 200 results to be assessed. For each selected Jeopardy topic, we pooled the results in the same way and picked up the top 100 results to be assessed as in general Jeopardy Task can be viewed a known-item search.

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 and Jeopardy runs. For the Faceted Search Task, we use the same metrics as that used in last year [**?**] to evaluate the runs.

6 Results

6.1 Ad-hoc and Jeopardy Task Results

As mentioned above, 140 ad-hoc topics were collected from three different sources: sub-subtopics, old topics from INEX 2009 and 2010, and keyword titles of Jeopardy topics. Among them, the 30 sub-subtopics, 20 old topics and 50 Jeopardy topics have assessment results. In this section, we will first present the evaluation results over the whole set of ad-hoc topics for all the submitted runs, and then analyze the effectiveness of the runs for each of the three sets of topics.

There are 20 runs submitted to the Ad-hoc Task by 7 participating groups. For each group, we selected its best performing run in terms of MAP, since MAP averages reasonably well over all topic types. Table 2 shows an overview of the 7 best performing runs from different groups. Over all topics, the best scoring run is from the Renmin University of China with a MAP of 0.2776 and also highest 1/rank, P@5, P@10, P@20 and P@30. Second best scoring team is University of Otago (0.2721). Third best scoring team is Ecole des Mines de Saint-Etienne (0.2609). Interpolated precision against recall is plotted in Fig 2, which shows little differences among the 3-4 best performing runs. The best performing runs are quite similar actually.

Table 3 shows the results over the 30 sub-subtopics. Since University of Amsterdam did not submit any results on sub-subtopics, there are only 6 instead of 7 runs in the table. We see that Renmin University of China (0.33365), University of Otago (0.3081), and Ecole des Mines de Saint-Etienne (0.2991) are still the 3 best performing groups.

Table 4 shows the results over the 20 old topics from INEX 2009 and 2010 Ad-hoc Tracks, now again evaluated by MAP. There are only 6 runs in the table since Oslo University College did not submit any results on this set of topics. We see that Renmin University of China still performs the best in terms of MAP (0.0936), and University of Amsterdam runs the second with the best 1/rank and P@5. The MAPs are commonly very low for this set of topics. This is no surprise since these are "hard" topics from previous years.

Table 5 shows the results over only the Jeopardy topics, now evaluated by the mean reciprocal rank (1/rank). There are 7 groups submitted results to the

Run	MAP	1/rank	P@5	P@10	P@20	P@30
Renmin-LDT2012_adhoc_ruc_comb07	**0.2776**	**0.7778**	**0.452**	**0.389**	**0.3235**	**0.2823**
Otago-ou2012pr09	0.2721	0.745	0.444	0.382	0.323	0.279
EMSE-run-085	0.2609	0.7131	0.444	0.367	0.3055	0.2663
NTNU-run1	0.2459	0.7145	0.436	0.372	0.3015	0.255
Amsterdam-inex12LDT.adhoc.baseline_LM	0.2187	0.7481	0.3829	0.2929	0.2114	0.1729
Kasetsart-kas16-PHR	0.1074	0.6718	0.3783	0.313	0.2489	0.2152
Oslo-result.fil	0.0046	0.037	0	0	0	0.0333

Table 2. Best performing runs (only showing one run per group) based on MAP over all the assessed ad-hoc topics.

Run	MAP	1/rank	P@5	P@10	P@20	P@30
Renmin-LDT2012_adhoc_ruc_comb15	**0.3365**	**0.8511**	**0.6067**	**0.6**	**0.5617**	**0.5167**
Otago-ou2012pr09	0.3081	0.8522	0.62	0.58	0.5467	0.4989
EMSE-run-086	0.2991	0.7356	0.58	0.5667	0.535	0.5067
NTNU-run1	0.2693	0.8122	0.6	0.5533	0.505	0.46
Kasetsart-kas16-EXT	0.1312	0.543	0.3154	0.3231	0.3173	0.3013
Oslo-result.fil	0.0046	0.037	0	0	0	0.0333

Table 3. Best performing runs (only showing one run per group) based on MAP over the 30 sub-subtopics

Run	MAP	1/rank	P@5	P@10	P@20	P@30
Renmin-LDT2012_adhoc_ruc_comb1	**0.0936**	**0.6845**	**0.33**	**0.29**	**0.2225**	**0.195**
Amsterdam-inex12LDT.adhoc.baseline_LM	0.0895	0.7146	0.34	0.29	0.22	0.1867
Otago-ou2012pr10	0.0836	0.5717	0.31	0.26	0.205	0.1783
EMSE-run-085	0.0782	0.5916	0.3	0.225	0.1875	0.1633
NTNU-run1	0.0724	0.5794	0.23	0.24	0.18	0.1517
Kasetsart-kas16-EXT	0.0585	0.3756	0.1625	0.1313	0.1125	0.1021

Table 4. Best performing runs (only showing one run per group) based on MAP over the 20 INEX 2009 and 2010 ad-hoc topics.

Run	MAP	1/rank	P@5	P@10	P@20	P@30
Renmin-LDT2012_adhoc_ruc_comb07	**0.3195**	**0.7655**	**0.416**	**0.306**	**0.231**	**0.188**
Amsterdam-inex12LDT.adhoc.baseline_LM	0.2704	0.7615	0.4	0.294	0.208	0.1673
Otago-ou2012pr09	0.3264	0.741	0.396	0.318	0.233	0.188
NTNU-run1	0.3014	0.7099	0.42	0.316	0.228	0.1733
Kasetsart-kas16-PHR	0.1434	0.7	0.18	0.16	0.085	0.0633
EMSE-run-085	0.3157	0.6979	0.424	0.316	0.235	0.186
MPI-submission	0.1618	0.5991	0.2732	0.1829	0.1061	0.0772

Table 5. Best performing runs (only showing one run per group) based on 1/rank over the 50 Jeopardy topics.

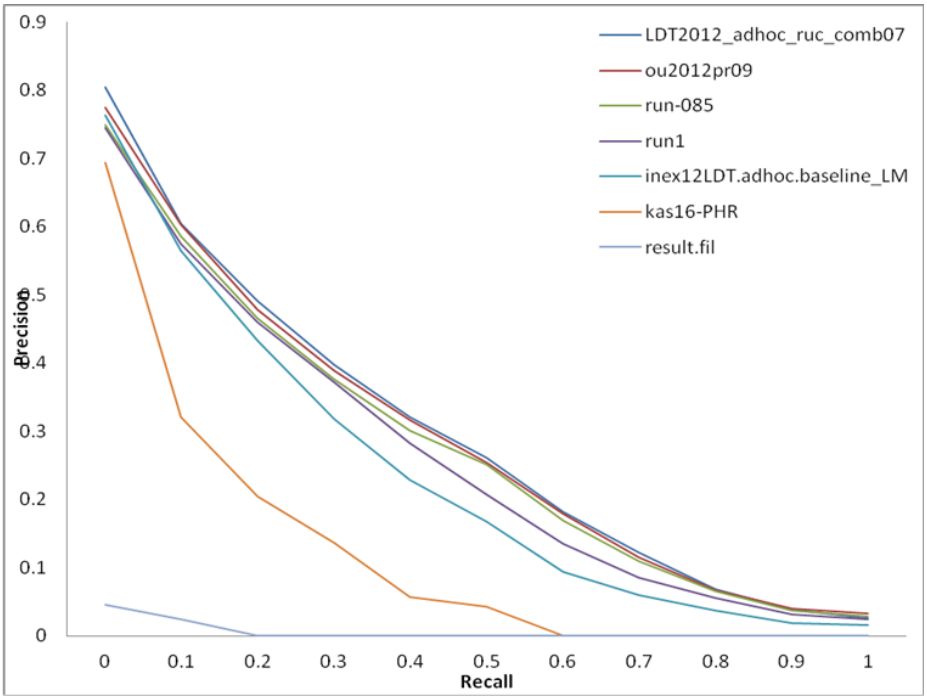


Fig. 2. Best run by each participating institute measured with MAP

Jeopardy topics, even though some of them submitted the runs to the Jeopardy task not to the Ad-hoc Task. We observe that Renmin University of China (0.7655) runs the first in terms of the mean reciprocal rank (1/rank), but University of Otago (0.741) has the best MAP. The second best scoring team in terms of 1/rank is University of Amsterdam.

7 Conclusions and Future Work

The Linked Data Track, which was a new track in INEX 2012, was organized towards our goal to close the gap between IR-style keyword search and semantic-web-style reasoning techniques. The track thus continues one of the earliest guiding themes of INEX, namely to investigate whether structure may help to improve the results of ah-hoc keyword search. As a core of this effort, we introduced a new document collection, coined *Wikipedia-LOD v1.1*, of XML-ified Wikipedia articles which were additionally annotated with RDF-style resource-property pairs from both DBpedia and YAGO2. This document collection serves as the basis for three tasks: i) the Ad-hoc Retrieval Task, ii) the Faceted Search Task, and iii) a new Jeopardy Task, which were all held as part of this year's Linked Data Track. We believe that this track encourages further research to-

wards applications that exploit semantic annotations over large text collections and thus facilitates the development of effective retrieval techniques for the same.

References

1. Media Wiki. `http://www.mediawiki.org/wiki/MediaWiki`.
2. SPARQL FullText, W3C Working Draft. `http://www.w3.org/2009/sparql/wiki/Feature:FullText`.
3. Wikipedia To XML Extension. `http://www.mediawiki.org/wiki/Extension:Wiki2xml`.
4. S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. DBpedia: A Nucleus for a Web of Open Data. In *ISWC/ASWC*, pages 722–735, 2007.
5. J. Hoffart, F. M. Suchanek, K. Berberich, E. Lewis-Kelham, G. de Melo, and G. Weikum. YAGO2: exploring and querying world knowledge in time, space, context, and many languages. In *WWW (Companion Volume)*, pages 229–232, 2011.
6. V. Lopez, V. S. Uren, M. Sabou, and E. Motta. Is Question Answering fit for the Semantic Web?: A survey. *Semantic Web*, 2(2):125–155, 2011.
7. A. Schuth and M. Marx. SPARQL FullText, W3C Working Draft. `http://staff.science.uva.nl/ marx/pub/INEX/facetedtaskproposal.pdf`.
8. Q. Wang, G. Ramírez, M. Marx, M. Theobald, and J. Kamps. Overview of the INEX 2010 Data Centric Track. In *INEX*, volume 7424 of *Lecture Notes in Computer Science*, pages 118–137. Springer, Heidelberg, 2011.

ENSM-SE at INEX 2012: Basic Experiments

Philippe Beaune, Michel Beigbeder, and Mihaela Juganaru-Mathieu

École Nationale Supérieure des Mines de Saint-Étienne
Institut Henri Fayol
158 cours Fauriel, F 42023 SAINT ETIENNE CEDEX 2, France
beaune@emse.fr, mbeig@emse.fr, mathieu@emse.fr

1 Introduction

Our objective in the INEX 2012 campaign was to integrate the semantic tags and the linked data in our proximity retrieval model. This model was sucessfully used in previous INEX campaigns and obtained good results, particularly in 2007 with the second place in the Ad Hoc Track Focused Task [1], and in 2010 with the first place in the Ad Hoc Track Relevant in Context Task [2]

Though we had several discomfitures with the collection because i) there were several versions of the collection, the last one available at the end of June, one week before the initial run submission deadline, ii) the different versions were difficult to follow because they were not clearly identified, iii) not every documents were well formed according to the XML format, iv) the provided DTD gives little information on the actual structure and its semantics, v) the documents contains many semantic annotations but the underlying ideas used to generate them are not documented making them difficult to apprehend. We present in section 2 how we processed the documents to alleviate the problems with the DTD.

Thus we only have been able to do some basic experiments presented in section 3. In section 4 we present our work in progress.

2 Collection preparation

The collection comes with 3 164 040 documents, of which 4 749 are not well formed according to the XML format. We deleted these documents in our experiments as they only represent 0,15% of the collection.

Structure was extremely difficult to apprehend with the provided DTD (`wikipedia-lod-xml.dtd`) because almost every elements can contain any other one. Here is a small extract of this DTD:

```
 10 <!ELEMENT wikipedia ( heading | list | paragraph | table | hr | list |
preblock )* >
 11
 12 <!ELEMENT heading ANY >
 13 <!ATTLIST heading level CDATA #IMPLIED >
 14
 15 <!ELEMENT list ( listitem+ ) >
```

```
16 <!ATTLIST list type NMTOKEN #REQUIRED >
17
18 <!ELEMENT listitem ANY >
19
20 <!ELEMENT paragraph ANY >
```

Some XML elements (such as `wikipedia` and `list`) are well defined because they could only contain a small number of meaningful elements. But 46 of the 70 XML tags defined in this DTD can contain any content, such as `heading`, `listitem` and `paragraph`.

With this DTD the following extract can be a part of a valid document:

```
[...]
<heading>
   <listitem>
      <paragraph>
          <heading>
[...]
```

where the structure has no sense using the usual meaning of the words *heading*, *paragraph* and so on.

So we decided to build a new collection where each document validates the very simple following DTD:

```
<!ELEMENT article ( title, CDATA ) >
<!ELEMENT title CDATA>
```

Some elements were deleted, for example `yagoproperties` and `dbpediaproperties`. For the other elements we only kept their textual content. We also ignored all the attributes except the attribute `@name`, whose value was kept as text. This operation was done with `xsltproc` and processing the whole collection lasted more than 17 hours.

We also tried to use `TreeTagger`[3] but it was too slow to process the whole collection because each document needed around one second to be processed.

Finally, the collection and its very simple structure was indexed with `zettair`[1] with the light stemmer on, lasting 40 minutes.

3 Runs

Three runs were allowed for participants in INEX 2012. Two of our runs were produced with *zettair*, the first one, *Emse-085*, used a language model with a Dirichlet smoothing. The second one, *Emse-086*, used the well known BM25 model with $k_1 = 1.2$, $k_3 = +\infty$ and $b = 0.75$. Both these runs were produced within 30 seconds for the 140 queries.

The third run, *Emse-087* used our proximity model developed for the previous INEX campaigns [4, 5], and its execution needed 2 minutes and 45 seconds.

For the present we do not have the assessments so no evaluation was performed.

[1] http://www.seg.rmit.edu.au/zettair/

4 Perspectives

4.1 Proximity model

Our proximity model works with the following type of structured documents:
document $\leftarrow$ *(part)*$^+$
part $\leftarrow$ *text*
part $\leftarrow$ *(part)*$^+$
part $\leftarrow$ *title* $\oplus$ *(part)*$^+$
For plain text our model computes a score based on a fuzzy neighbouring parameterized function. For a document composed of a concatenation of parts, the score is the sum of the part scores. For a document/part with a title, title words are considered as close to any word of the part content.

4.2 First choice

The provided DTD doesn't permit us to easily construct a collection fulfilling the above description.The title of the documents was easy to extract, but as the part titles and the parts themselves are not nested, extracting these titles to insert them in their corresponding part is not possible in XSLT [6]. So we considered the XML documents as:
document $\leftarrow$ *title* $\oplus$ *text*
and we applied our model in this simplified case.

4.3 Future works

We detected that the tag `heading` could be the title of parts, but the parts themselves are not explicit and clearly delimited. We will construct a new collection fulfilling our document model using a high level programming language using the library `libxml` and build the nesting based on the assumption that the attribute `@level` of the tag `heading` indicates the actual nesting.

We will also consider the tags `yagoproperties` and `dbpediaproperties` as parts of the newer documents. This work is in progress.

References

1. Fuhr, N., Kamps, J., Lalmas, M., Malik, S., Trotman, A.: Overview of the inex 2007 ad hoc track. In Fuhr, N., Kamps, J., Lalmas, M., Trotman, A., eds.: INEX. Volume 4862 of Lecture Notes in Computer Science., Springer (2007) 1–23
2. Arvola, P., Geva, S., Kamps, J., Schenkel, R., Trotman, A., Vainio, J.: Overview of the inex 2010 ad hoc track. In Geva, S., Kamps, J., Schenkel, R., Trotman, A., eds.: INEX. Volume 6932 of Lecture Notes in Computer Science., Springer (2010) 1–32
3. Schmid, H.: Probabilistic Part-of-Speech Tagging Using Decision Trees. In: International Conference on New Methods in Language Processing. (September 1994)
4. Beigbeder, M., Imafouo, A., Mercier, A.: ENSM-SE at INEX 2009 : Scoring with proximity and semantic tag information. **6203** (2009) 49–58

5. Beigbeder, M.: Focused retrieval with proximity scoring. In Shin, S.Y., Ossowski, S., Schumacher, M., Palakal, M.J., Hung, C.C., eds.: SAC, ACM (2010) 1755–1759
6. Møller, A., Olesen, M.O., Schwartzbach, M.I.: Static validation of xsl transformations. ACM Trans. Program. Lang. Syst. **29**(4) (August 2007)

Running SPARQL-Fulltext Queries Inside a Relational DBMS

Arunav Mishra[1], Sairam Gurajada[1], and Martin Theobald[1]

Max Planck Institute for Informatics, Saarbrücken, Germany

Abstract. We describe the indexing, ranking, and query processing techniques we implemented in order to process a new kind of SPARQL-fulltext queries that were provided in the context of the INEX 2012 Jeopardy task.

1 Introduction

The INEX 2012 Linked Data track provides a new data collection that aims to combine the benefits of both text-oriented and structured retrievals settings in one unified data collection. For the rapid development of a new query engine that could handle this particular combination of XML markup and RDF-style resource/property-pairs, we decided to opt for a relational database system as storage back-end, which allows us to index the collection and to retrieve both the SPARQL- and keyword-related conditions of the Jeopardy queries under one common application layer. To process the 90 queries of the benchmark, we keep two main index structures, one of which is based on a recent dump of DBpedia core triples, and another one, which is based on keywords extracted from the INEX Wikipedia-LOD collection. Additionally, our engine comes with a rewriting layer that translates the SPARQL-based query patterns into SQL queries, thus formulating joins over both the DBpedia triples and the keywords extracted from the XML articles.

2 Document Collection and Queries

2.1 Document Collection

Each XML document in the Wikipedia-LOD collection combines structured and unstructured information about a Wikipedia entity (or so-called "resource") in one common XML format. The structured part ("properties") of the document represents factual knowledge, which was obtained from querying DBpedia and YAGO for facts containing this entity as either their subject or object together with the property itself, while the semistructured part (WikiText) is XML-ified content that was obtained from the Wikipedia article describing the entity. Specifically, the main components of each XML document in the collection can be divided into the following blocks:

1. **Meta Data:** Describes the meta information of the document like title, author, etc. and provides a unique Wikipedia entity ID.

Query1: What is a famous Indian Cuisine dish that mainly contains rice, dhal, vegetables, roti and papad

`SELECT ?s WHERE { ?s ?p <http://dbpedia.org/resource/Category:Indian_cuisine> . FILTER FTContains (?s, "rice dhal vegetables roti papad") . }`	`http://dbpedia.org/resource/Thali`

Query2: Which mountain range is bordered by another mountain range and is a popular sightseeing and sports destination?

`SELECT ?p WHERE { ?m <http://dbpedia.org/ontology/border> ?p . ?p <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpedia.org/ontology/MountainRange> . FILTER FTContains(?p, "popular sightseeing and sports destination") . }`	`http://dbpedia.org/resource/Alps`

Table 1. Example benchmark queries in SPARQL-fulltext format

2. **WikiText:** This portion of the document includes text from the Wikipedia article in well-formed XML syntax. The Wikipedia were translated from the common MediaWiki [1] format. Additionally XML tags for all infobox attributes (and corresponding values) were included to replace all Wiki markup by proper XML tags. The inter-Wiki links which point to the other Wikipedia entities are extended to include the links to both the YAGO and DBpedia resources of the Wikipedia target pages. Each link has three sub-links: `wiki-link`, `yago-link`, and `dbpedia-link`. The `wiki-link` attribute is a regular hyperlink to the target Wikipedia article, while the `yago-link` and `dbpedia- link` attributes contain pointers to the respective sources in the RDF collections of DBpedia and YAGO2.
3. **Properties:** Finally, each document at the end includes a list of all the properties from the DBpedia and YAGO facts about the entity.

2.2 Benchmark Queries

The translated queries are syntactically conforming to the SPARQL standard, with an additional `FTContains` operator that provides the flexibility to add keyword constraints to the query. The new `FTContains` operator takes two parameters, namely the variable name in the SPARQL component and a set of associated keywords. Table 1 shows two such natural-language queries and their SPARQL translations containing `FTContains` operators for the keyword components. For instance, in the SPARQL translation of Query 1, we have the original SPARQL component as `?s ?p <http://dbpedia.org/resource/Category:Indian_cuisine>` and the keyword component as {`rice dhal vegetables roti papad`}. In addition, the subject "?s" is bound by the keyword component as specified in the `FTContains` function.

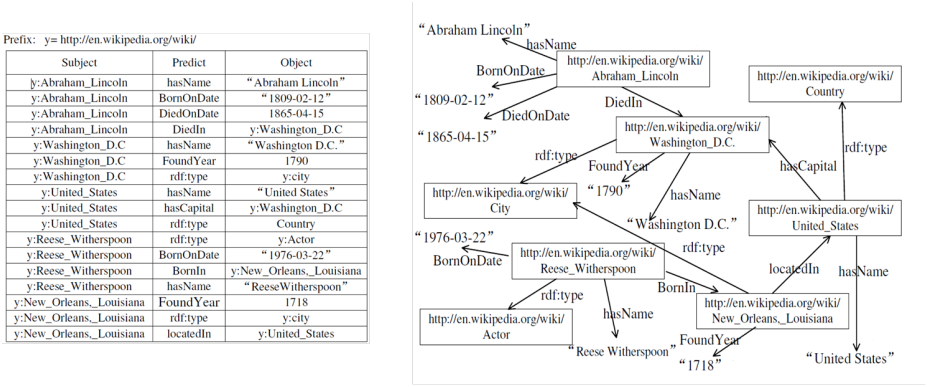


Fig. 1. Example showing an RDF graph and the corresponding triplet representation in a relational database *(picture taken from [14])*

3 Document Parsing and Index Creation

We employed a regular SAX parser to parse the 3.1 Million XML articles whose general XML structure is still based on that of the original articles. That is, these articles contain a metadata header with information about the ID, authors, creation date and others, usually also an infobox with additional semistructured information consisting of attribute-value pairs that describe the entity, and of course rich text contents consisting of unstructured information and more XML markup about the entity that is captured by such an article. Our keyword indexer uses the basic functionality of TopX 2.0 [11], which includes Porter stemming, stopword removal and BM25 ranking, but stores the resulting inverted lists for keywords into a relational database instead of TopX's proprietary index structures.

4 Data Model for the Document Collection

In this section we describe the storage backend for both the structured and unstructured data parts of our document collection. Our query engine employs two relational tables to import the core tables of DBpedia and the keywords extracted from the Wikipedia LOD collection [2] in one unified database schema. Thus, the SPARQL queries with fulltext filter conditions of the above form can directly be rewritten to SQL queries with various join conditions over these two tables.

4.1 Storing RDF Data in a Single Relational Table

An RDF data collection can be defined as a collection of triples (a subject or resource, a predicate or property, and an object or value). In this paper, we refer to the RDF data

Column	Type
N3ID	NUMBER
Subject	VARCHAR2(1024)
Predicate	VARCHAR2(1024)
Object	VARCHAR2(1024)

Table 2. Schema of the DBpediaCore table

Index Name	Attributes
DBpediaIDX_Obj	(Object,Subject,Predicate)
DBpediaIDX_Sub	(Subject,Predicate,Object)
DBpediaIDX_Prd	(Predicate,Object,Subject)

Table 3. Indices built over the DBpediaCore table

as a collection of SPO triples, each containing exactly one such subject (S), predicate (P), and object (O).

As our storage back-end, we use a relational database (in our case Oracle 11g), to store the RDF data we imported from the core dump of DBpedia v3.7 (see http://downloads.dbpedia.org/3.7/en/).A SPARQL query then conceptually is transformed into a sub-graph matching problem over this large RDF graph. The RDF data model (a collection of triples) can be viewed as a directed, labeled multi-graph (RDF graph) where every vertex corresponds to a subject or object, and each directed edge from a subject to an object corresponds to a predicate. There may be multiple edges directed from a subject towards an object. Thus an RDF graph becomes a multi-graph.

In our query engine, RDF data is treated as a large list of triples and stored in a relational database. Figure 1(b) shows a fragment of an RDF graph constructed over Wikipedia, and Figure 1(a) shows a corresponding relational table representation in the database. Keeping this in mind, and maintaining the simplicity of the model, we use one relational table to store all SPO triples. In our system, we refer to this table as the DBpediaCore table, and its schema is described in Table 2. This route is pursued similarly by many so-called triplet-stores like Jena [12], Sesame [4] and Oracle [6] and RDF-3X [8]. Though there are other advanced and more complex approaches, like vertical partitioning or the exploitation of property tables, our goal was satisfied by this relatively simple idea.

4.2 Indices on the RDF Table

Representing RDF data as a relational table opens up for us all kinds of optimization techniques used by the database community. One such technique is to build indices over the DBpediaCore table to improve the data retrieval operations on it. Since we translate a SPARQL query with fulltext conditions into conjunctive SQL queries (described in the following section), we employ a large amount of self joins over this table. These self joins could occur on any of the three columns described in Table 2. Thus we build three multi-attribute indices, using each of the SPO columns of the table as key and the remaining two attributes as depending data values, to accelerate the lookup times over this table for the case when each of the attributes is provided as a constant by the SPARQL query. Table 3 describes these three indices built over the DBpediaCore table.

Column	Type
Entity_ID	VARCHAR2(1024)
Term	VARCHAR2(1024)
Score	NUMBER

Table 4. Table schema of the Keywords table

Index Name	Attributes
Keywords_Entity_IDX	(Entity_ID, Term, Score)
Keywords_Term_IDX	(Term, Entity_ID, Score)

Table 5. Indices built over the Keywords table

4.3 Storing Keywords in a Single Relational Table

As per traditional IR, a fulltext search allows for identifying documents that satisfy a keyword query, and optionally sorting the matching documents by their relevance to the query. The most common approaches use fairly simple TF/IDF counts or Boolean retrieval to measure the content similarity of a fulltext keyword query to the documents in the data collection. In our engine, we store the unstructured text contents of all the documents in the collection as another table in the relational database. This table essentially contains three columns, relating a keyword (or term) with the documents in which it occurs and the similarity scores calculated for this particular term and document based on the underlying scoring model.

We define a term as a sequence of characters that occur grouped together in some document and thus yield a useful semantic unit for processing. To obtain this set of terms from the fulltext content of a Wikipedia article, we use a variant of the TopX indexer [11], which applies a standard white-space tokenizer, Porter stemming, and stopword removal to the unstructured text inputs. For this purpose, we treat all CDATA sections and attribute values of the Wikipedia-LOD articles as one flat collection of text; that is we treat the XML collection as an unstructured set of Wikipedia text documents. We use a default BM25 scoring model (using $k = 2.0$ and $b = 0.75$) to calculate the relevance score of a term in a document. In our approach, we create a Keywords table to store all the terms occurring in the unstructured Wikipedia fulltext collection. The schema of this table is shown in Table 4. The Entity_ID column essentially stores the *Uniform Resource Identifier* (URI) of the DBpedia entities. It may be noted that every document in our collection also corresponds to a DBpedia entity. So we prefer to use the RDF prefix defined by DBpedia to represent an entity, which is ⟨http://dbpedia.org/resource/entity⟩. To obtain a mapping from a DBpedia entity to the Entity_ID's from a Wikipedia page, we maintain a hashmap that maps every Wikipedia page to its corresponding DBpedia entity URI. Thus every statement of the Keywords table represents a term that is mapped to an entity in DBpedia together with its content similarity score to the entity's Wikipedia page.

4.4 Indices on the Keywords Table

We can easily imagine that the Keywords table would be very large (containing more than a billion entires), considering the number of terms extracted from almost the entire Wikipedia encyclopedia. The Keywords table basically maps every term occurring in Wikipedia to a DBpedia entity. Taking the size of this table into consideration, data

retrieval operations on the table becomes costly and inefficient. Also processing conjunctive queries over multiple self joins becomes infeasible unless correct indices are built over the table. So we build two more multi-attribute indices over this table as shown in Table 5.

5 Scoring

5.1 Keywords Scoring and retrieval

We use the TopX 2.0 indexer to create per-term inverted lists from the plain (unstructured) text content of the Wikipedia documents, which each corresponds to an entity in DBpedia. The exact BM25 variant we used for ranking an entity e by a string of keywords S in an `FTContains` operator is given by the following formula:

$$score(e,\ \texttt{FTContains}(e,S)) = \sum_{t_i \in S} \frac{(k_1+1)\, tf(e,t_i)}{K + tf(e,t_i)} \cdot \log\left(\frac{N - df(t_i) + 0.5}{df(t_i) + 0.5}\right)$$

$$\text{with} \quad K = k_1 \left((1-b) + b \frac{len(e)}{avg\{len(e') \mid e' \ in \ collection\}}\right)$$

Where:

1) N is the number of XML articles in Wikipedia LOD collection.
2) $tf(e,t)$ is the term frequency of term t in the Wikipedia LOD article associated with entity e.
3) $df(t)$ is the document frequency of term t in the Wikipedia LOD collection.
4) $len(e)$ is the length (sum of tf values) of the Wikipedia LOD article associated with entity e.

We used the values of $k_1 = 2.0$ and $b = 0.75$ as the BM25-specific tuning parameters (see also [7] for tuning BM25 on earlier INEX settings).

5.2 Translating the Keyword-Scores to SPO Triples

Recently there has been a lot of work on identifying appropriate entity scoring and ranking models. Many techniques use language models [13, 9, 10] while other approaches try to adopt more complex measures form IR. Ranking for structured queries have been intensively investigated for XML [3], and, to a small extent, also for restricted forms of SQL queries [5]. While some of these approaches carry over to large RDF collections and expressive SPARQL queries as discussed earlier, the LOD track makes a significant simplifying assumption: since every DBpedia or YAGO entity is associated (via an explicit link) with the Wikipedia article (in XML format) that describes this entity, the entities can directly be referred to and ranked by the keywords that are imposed over the corresponding Wikipedia article. We thus believe that it is a good idea to translate the scores to the RDF entities from a keyword-based search on the fulltext part of the Wikipedia LOD collection. As discussed earlier, every entity in our collection is essentially a document containing all the DBpedia and YAGO properties about this entity

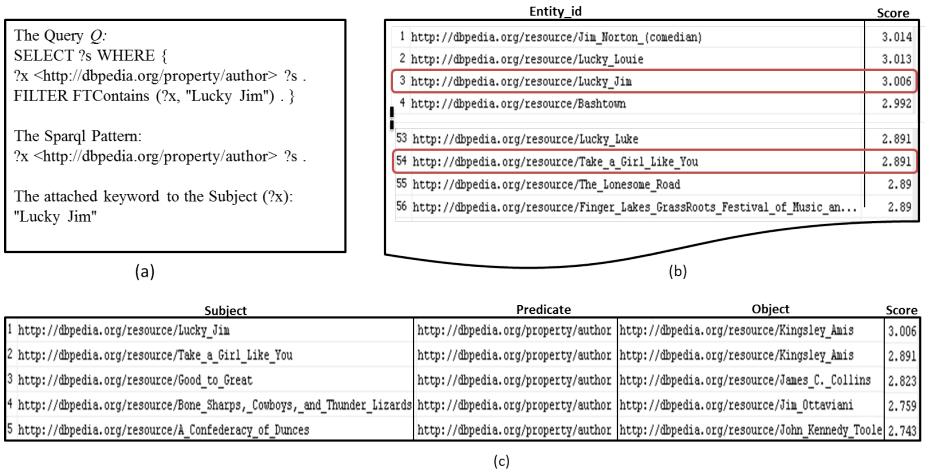


Fig. 2. Example showing the translation of keyword scores to the entity-relationship graph

together with the fulltext content from the corresponding Wikipedia page. Thus we perform a keyword search on this fulltext content by using the fulltext condition specified by the `FTContains` operator (as discussed earlier). From this search, we get a ranked entity list containing the candidates for the answer to the given query. This is can be best explained by an example as shown in Figure 2 .

Figure 2(a) shows an example query containing a simple SPARQL query with one triplet and a fulltext condition that is bound to the subject of the SPARQL query. Figure 2(b) shows fragments of top-100 results obtained by fulltext search on the unstructured part of the collection with keywords “Lucky Jim”. As illustrated, we already have the relevant candidates for the answer from the keyword search due to the satisfactory performance of our BM25 scoring scheme applied to score the keywords. The scored entities in the candidate list is again checked for the graph pattern of the SPARQL query in Figure 2(a) and the final top-k entities are retrieved from the entities which qualify for the graph pattern. Figure 2(c) shows the retrieved results after searched for the pattern in the graph.

6 Rewriting SPARQL-Fulltext Queries to SQL

6.1 Basic Idea of a Conjunctive Query

Conjunctive queries have high practical relevance because they cover a large part of queries issued on relational databases and RDF stores. That is, many SQL and SPARQL queries can be written as conjunctive queries. In our scenario, we are required to generate an appropriate translation of a given SPARQL query with multiple fulltext conditions into an SQL query. Our system-generated queries are essentially conjunctive queries over multiple instances of the `DBpediaCore` and `Keywords` tables (as described earlier).

To capture the idea behind translating a given SPARQL query with fulltext conditions into a conjunctive SQL query, let us take an example query Q taken from the Jeopardy benchmark as shown in Table 6. Q contains three triple patterns $T1$, $T2$ and $T3$, and two fulltext conditions $K1$ and $K2$. Each K_i contains a variable occurring in a triple pattern and is bound to a string by an `FTContains` operator. To begin translation, firstly, every attached string is tokenized and stemmed into the format of terms stored in the `Term` column of the `Keywords` table. For every generated token k_j where $j \geq 0$ from each K_i , an instance of the `Keywords` table is taken, where the `Terms` column of the instance is restricted to k_j. Similarly for every triple pattern T_m , we take an instance of the `DBpediaCore` table t_i, where $i \geq 0$. In the example, instance t_1 represents triple $?sub$ $?o$ $?s$, instance t_2 represents the triple $?x$ $?r$ $?s$, and instance t_3 represents the triple $?sub$ `rdf:type` ⟨`http://dbpedia.org/ontology/Place`⟩. These instances t_i are further restricted on their `Subject` or `Object` by the `Entity_ID` of the k_j instance of the `Keywords` table as specified by the fulltext condition. Finally, the instances t_i are restricted by each other on `Subject`, `Predicate` or `Object` as per the logic of the joins in the SPARQL query. The translation of Q into a conjunctive SQL query, as described above, is shown in Table 7.

7 Materialization SQL Joins, Temporary Tables, and Join Orders

In the previous section, we presented a translation method for any given (conjunctive) SPARQL query with fulltext conditions into a conjunctive SQL query. But, there are a few problems with such a direct translation. Firstly, this form of translation does not handle empty results returned from any intermediate join in the desired way. For example, lets say we encounter with a very unlikely keyword term which is missing in the `Keywords` table. Then an empty result will be returned for that instance, and, since we issue a conjunctive query with all AND conditions, the overall result of the query will also be empty. Secondly, this type of query with many (self-)joins is inefficient and difficult to optimize. During query processing, we rely on the Oracle optimizer for selecting an appropriate query plan. It becomes difficult for the optimizer to choose the best plan due to the redundancy of data in the `DBpediaCore` table, i.e., due to multiple occurrences of an entity as a `Subject` or `Object` and an unknown amount of occurrences of a term in the `Keywords` table. Due to apparently insufficient statistics on these large tables (although we had analyzed the tables and enabled this feature in

A manually translated query in SPARQL syntax:
SELECT ?sub WHERE {
?x ?r ?s .
?sub ?o ?s .
?sub rdf:type ⟨ http://dbpedia.org/ontology/Place ⟩ .
FILTER FTContains (?x, “Canarian”).
FILTER FTContains (?sub, “place islands country”) . }

Table 6. A given SPARQL-fulltext query of the INEX 2012 Jeopardy task

The automatically converted conjunctive SQL query:
SELECT DISTINCT t1.SUBJECT AS sub FROM dbpediacore t2, dbpediacore t1, dbpediacore t3, keywords k40, keywords k41, keywords k42, keywords k3
WHERE
t2.OBJECT = t1.OBJECT
AND t1.SUBJECT = t3.SUBJECT
AND t3.PREDICATE = 'http://www.w3.org/1999/02/22-rdf-syntax-ns#type'
AND t3.OBJECT = 'http://dbpedia.org/ontology/Place'
AND k0.TERM = 'place' AND k1.TERM = 'island'
AND k2.TERM = 'countri'
AND k3.TERM = 'canarian'
AND t1.SUBJECT = k0.ENTITY_ID
AND t1.SUBJECT = k1.ENTITY_ID
AND t1.SUBJECT = k2.ENTITY_ID
AND t2.SUBJECT = k3.ENTITY_ID

Table 7. SQL translation of the SPARQL query of Table 6

the optimizer), we found the Oracle optimizer to often choose a bad query plan, which initially resulted in individual SQL queries that did not even finish after several days.

7.1 SQL Joins

Most join queries contain at least one join condition, either in the FROM (for outer joins) or in the WHERE clause. The join condition compares two columns, each from a different table. To execute a join, the database system combines pairs of rows, each containing one row from each table, for which the join condition evaluates to true. The columns in the join conditions need not also appear in the select list.

To execute a join of three or more tables, Oracle first joins two of the tables based on the join conditions comparing their columns and then joins the result to another table based on the join conditions containing columns of the previously joined tables and the new table. Oracle continues this process until all tables are joined into the result. The optimizer determines the order in which Oracle joins tables based on the join conditions, indexes on the tables, and any available statistics for the tables.

FULL OUTER JOIN A FULL OUTER JOIN does not require each record in the two joined tables to have a matching record. The joined table retains each record even if no other matching record exists. Where records in the FULL OUTER JOIN'ed tables do not match, the result set will have NULL values for every column of the table that lacks a matching row. For those records that do match, a single row will be produced in the result set (containing fields populated from both tables). This join can be used to solve the first problem mentioned above. All instances of the `Keywords` table representing a fulltext condition K_i can undergo a FULL OUTER JOIN on the `Entity_ID` attribute. Thus we will have results from the keywords table even if one of the instance matches any tuples or has a NULL as a value.

```
CREATE TABLE KEYS0 AS  SELECT/*+ORDERED*/ * FROM
( SELECT/*+ORDERED*/ DISTINCT ENTITY_ID , MAX(SCOREss) AS SCORE FROM
(  SELECT DISTINCT K1.ENTITY_ID  AS ENTITY_ID , (NVL(k1.SCORES,0)+1  + NVL(k2.SCORES,0)+1 +
NVL(k3.SCORES,0)+1 ) AS SCORESS FROM
(SELECT DISTINCT ENTITY_ID  , SCORE  AS SCORES  FROM KEYWORDS WHERE TERM='island') K1
FULL OUTER JOIN
(SELECT DISTINCT ENTITY_ID  , SCORE  AS SCORES  FROM KEYWORDS WHERE TERM='countri') K2
ON  k1.ENTITY_ID = k2.ENTITY_ID
FULL OUTER JOIN
(SELECT DISTINCT ENTITY_ID  , SCORE  AS SCORES  FROM KEYWORDS WHERE TERM='place') K3
ON  k1.ENTITY_ID = k3.ENTITY_ID
ORDER BY SCORESS DESC)
GROUP BY ENTITY_ID ORDER BY SCORE DESC )
```

```
CREATE TABLE KEYS1 AS  SELECT/*+ORDERED*/ * FROM
( SELECT/*+ORDERED*/ DISTINCT ENTITY_ID , MAX(SCOREss) AS SCORE FROM
(  SELECT DISTINCT K1.ENTITY_ID  AS ENTITY_ID , (NVL(k1.SCORES,0)+1 ) AS SCORESS FROM
(SELECT DISTINCT ENTITY_ID  , SCORE  AS SCORES  FROM KEYWORDS WHERE TERM='canarian')
K1
ORDER BY SCORESS DESC)GROUP BY ENTITY_ID ORDER BY SCORE DESC )
```

Fig. 3. Queries to create the `Keys` temporary tables

INNER JOIN An INNER JOIN is the most common join operation used in applications and can be regarded as the default join type. INNER JOIN creates a new result table by combining column values of two tables (A and B) based upon the join predicate. The query compares each row of A with each row of B to find all pairs of rows which satisfy the join predicate. When the join predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row. The result of the join can be defined as the outcome of first taking the Cartesian product (or cross join) of all records in the tables (combining every record in table A with every record in table B). It returns all records which satisfy the join predicate. This join returns the same result as the "equality" join as described in the previous section but gives more flexibility to the optimizer to select different types of joins like hash joins, merge joins, or nested loop joins. We can replace all the equality join with Oracle's INNER JOIN. In some queries this trick shows considerable improvement in query processing time by Oracle's query engine.

7.2 Materializing Temporary Tables

One big conjunctive query forces the Oracle optimizer to choose from a very large number of possible query execution plans, and it turns out that—at least in our setting—it often chooses an inefficient plan. For example, in a big conjunctive query, the optimizer often chose to join instances `DBpediaCore` tables before restricting the relevant entities with the given conditions. These types of query plans proved to be highly expensive. Thus, to prevent the optimizer from taking such inappropriate decisions, we materialize temporary tables by separately joining the `Keywords` table instances and

```
CREATE TABLE tab3 as select subject , predicate , object  , ( NVL(KEYS0.score,0) ) AS SCORE  from
(SELECT * FROM  dbpediacore3  t1
WHERE  t1.PREDICATE='http://www.w3.org/1999/02/22-rdf-syntax-ns#type'
AND  t1.OBJECT='http://dbpedia.org/ontology/Place')temp
INNER JOIN
KEYS0
ON temp.SUBJECT=keys0.entity_id
```

```
CREATE TABLE tab2 as select subject , predicate , object  , ( NVL(KEYS0.score,0) ) AS SCORE  from
(SELECT * FROM  dbpediacore3  t3 )temp
INNER JOIN
KEYS0
ON temp.SUBJECT=keys0.entity_id
```

```
CREATE TABLE tab1 as select subject , predicate , object  , ( NVL(KEYS1.score,0) ) AS SCORE  from
(SELECT * FROM  dbpediacore3  t2 )temp
INNER JOIN
KEYS1
ON temp.SUBJECT=keys1.entity_id
```

Fig. 4. Queries to create the TAB temporary tables

the `DBpediaCore` table instances. This acts as a strong support for the optimizer to select better query plans for smaller intermediate queries and store their results into temporary tables which are later joined together to retrieve the final result.

7.3 Evaluating the Join Order and Forcing Orders via Optimizer Hints

There are some simple techniques by which we can determine the join order of the tables. One such technique is to maintain an IDF index containing the most frequent terms that occur in the collection. This index has a very simple and intuitive schema `Features(Term, IDF)`. The first column represents a term and the second column represents its Inverse Document Frequency (IDF). It can be intuitively be seen that a frequent term will have lower IDF and a select query will result in bigger intermediate joins. At the same time, if a term is absent in the feature index, then it can be assumed to be infrequent and thus have a high IDF value. Every instance of the `Keywords` table

```
SELECT /*+Ordered*/  sub , MAX(SCORE) AS SCORE_MAX FROM
 (SELECT /*+ORDERED*/ DISTINCT tab3.SUBJECT AS sub , ( NVL(tab3.score ,0 )+1
+  NVL(tab1.score ,0 )+1 +  NVL(tab2.score ,0 )+1 +  0 ) AS score FROM
tab3 , tab1 , tab2
WHERE tab1.OBJECT = tab2.OBJECT
AND tab2.SUBJECT = tab3.SUBJECT
) GROUP BY sub ORDER BY SCORE_MAX DESC
```

Fig. 5. The final SELECT query to obtain the answer

```
drop table keys0
drop table keys1
drop table tab3
drop table tab1
drop table tab2
```

Fig. 6. Queries to DROP the temporary tables

can now be joined in increasing order of the IDF values of their respective term, thus ensuring the smaller tables to be joined first. This order of joining can be enforced on the Oracle optimizer by adding so-called optimizer hints to the queries.

A hint is a code snippet that is embedded into an SQL statement to suggest to Oracle how the statement should be executed. There are many hints provided to assist the optimizer. In our case, we found the `Ordered` hint to force the optimizer to join tables in the specified order written in the FROM clause of the query. So our translator algorithm writes the `Keywords` table instances in the appropriate order in the FROM clause of the translated SQL query.

8 The Rewriting Algorithm

We can now develop an overall rewriting algorithm by putting together all the afore described steps as follows.

1. Load the features index containing frequent terms and their IDF values into main memory.
2. Tokenize and stem the `FTContains` fulltext conditions and decide the order of joins among the keywords from the features index.
3. Create temporary $Keys_i$ tables for each fulltext condition: these contain the results of the outer join over the `Keywords` table instances constrained by the terms. This is shown in Figure 3.
4. Create temporary Tab_i tables for each triplet pattern: these contain the results of the inner join over the `DBpediaCore` table instances which are additionally joined with $Keys_i$ temporary tables for each `FTContains` fulltext condition in the query. This is shown in Figure 4.
5. Assign a default score of 1 to all triples in absence of a fulltext condition: in absence of a fulltext condition on any triplet pattern, a default score of 1 is assigned to all the triples as a constant score for each triplet condition (as discussed earlier).
6. Final query: the main select query combines the Tab_i temporary tables via an inner join; the join logic is based on the joins given in the original SPARQL query. This is shown in Figure 5.
7. Finally, drop the temporary tables $Keys_i$ and Tab_i. This is shown in Figure 6.

9 INEX Results

Since a detailed evaluation of the run submissions was not available at the time this paper was submitted, we defer a discussion of the results until to the INEX workshop at the CLEF conference in September.

10 Conclusions

We presented an approach for storing structured RDF data and unstructured data in relational database. We also presented the necessary indices required to efficiently process queries over this relational schema. Our approach converts a SPARQL query with fulltext conditions into unions of conjunctive SQL queries by materializing temporary tables. These temporary tables store intermediate results from inner or outer joins over our relations, based on given conditions in the query. We also presented a simple yet effective way to rank entities by translating scores from keywords. Finally, we showed the advantages of predeciding the join orders of tables and techniques to enforce them in the Oracle optimizer

References

1. Media Wiki. http://en.wikipedia.org/wiki/MediaWiki.
2. Overview of the INEX 2012 Linked Data Track.
3. S. Amer-Yahia and M. Lalmas. XML search: languages, INEX and scoring. *SIGMOD Record*, 35, 2006.
4. J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *LNCS*, volume 2342, Sardinia, Italy, 2002.
5. S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic information retrieval approach for ranking of database query results. *ACM Trans. Database Syst.*, 31, 2006.
6. E. I. Chong, S. Das, G. Eadon, and J. Srinivasan. An efficient SQL-based RDF querying scheme. In *VLDB '05*, 2005.
7. C. L. A. Clarke. Controlling overlap in content-oriented XML retrieval. In *SIGIR*, 2005.
8. T. Neumann and G. Weikum. The RDF-3X engine for scalable management of RDF data. *The VLDB Journal*, 19(1), 2010.
9. D. Petkova and W. B. Croft. Hierarchical Language Models for Expert Finding in Enterprise Corpora. In *ICTAI '06*, 2006.
10. P. Serdyukov and D. Hiemstra. Modeling Documents as Mixtures of Persons for Expert Finding. In *LNCS*, volume 4956, 2008.
11. M. Theobald, A. Aji, and R. Schenkel. TopX 2.0 at the INEX 2009 Ad-Hoc and Efficiency Tracks. In *INEX*, 2009.
12. K. Wilkinson, C. Sayers, H. Kuno, and D. Reynolds. Efficient RDF Storage and Retrieval in Jena2. In *Proc. First International Workshop on Semantic Web and Databases*, 2003.
13. C. Yang, Y. Cao, Z. Nie, J. Zhou, and J.-R. Wen. Closing the Loop in Webpage Understanding. *IEEE Transactions on Knowledge and Data Engineering*, 22, 2010.
14. L. Zou, J. Mo, L. Chen, M. T. Özsu, and D. Zhao. gStore: answering SPARQL queries via subgraph matching. *PVLDB*, 4(8), 2011.

NTNU at the INEX 2012 Linked Data Track

Robert Neumayer and Krisztian Balog

Norwegian University of Science and Technology,
Department of Computer and Information Science, Trondheim, Norway,
{Robert.Neumayer,Krisztian.Balog}@idi.ntnu.no

Abstract. We describe our participation in the INEX 2012 Linked Data track. This track is set out to investigate retrieval techniques over a combination of textual and structured data. The data collection used, Wikipedia-LOD, comprises of Wikipedia articles, enriched with RDF properties from both DBpedia and YAGO2.

This year, we focused on the 'classic' ad-hoc retrieval task, i.e., informational queries to be answered mainly by the textual contents of Wikipedia articles. Our approach builds on previous research done on entity search in RDF collections and our main aim was to investigate how well these models generalise to this new setting.

We submitted three runs, all of which were modifications of a structured retrieval model based on language models (i.e., mixture of language models). These models were found to perform very well on the ad-hoc entity ranking task in RDF data. For runs 1 and 3 we considered the top 500 properties found in the collection. For run 2 we considered two fields: (i) a 'catchall' field, collapsing all textual content associated with a given entity, and (ii) a name field, representing the entity's name (which we found to be strong evidence for an entity's relevance in past work). For runs 2 and 3 we applied additional NLP techniques for query extension/expansion.

Initial results suggest that the techniques that were shown to improve retrieval performance are not necessarily beneficial in this new setup. One possible reason for that is that entities have much more text associated with them; before, only RDF properties were used, while here the contents of Wikipedia articles are considered too. We will investigate this issue in more detail in our future work.

Integrated Retrieval over Structured and Unstructured Data

Qiuyue Wang[1,2], Jinglin Kang[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, 402817493@qq.com

Abstract. We report our experiment results on the INEX 2012 Linked Data Track. We participated in the ad hoc and jeopardy tasks. As the new data collection on INEX 2012 Linked Data Track features a combination of unstructured and structured data, our first attempt is to investigate different strategies of combining the retrievals over structured and unstructured data, and compare the combined approaches with the traditional unstructured ones. In this paper, we discussed three types of combination strategies and we experimented two of them on the track. The experiment results show that ……

1 Introduction

Though Web is best known as an enormous collection of unstructured documents, it also contains a huge amount of structured data, like HTML tables, data stored in Deep Web databases, increasingly published RDF data due to the efforts of Linked Data community, and so on. With more and more structured data became accessible to end users, more intelligent search on the Web is expected. There are increasing interests on semantic search on the Web, i.e. leveraging the semantics in structured data to improve the Web search.

The new data collection of INEX 2012 Linked Data Track is a fusion of Wikipedia articles and their corresponding RDF data from DBpedia and YAGO2. Each Wikipedia article corresponds to an entity/resource in DBpedia and YAGO2, while DBpedia and YAGO2 contain structured data extracted from each article, e.g. properties of entities and relationships with other entities. It can be viewed as an integrated collection of unstructured and structured data covering a wide range of topics. With such a data collection, we intended to investigate different strategies of combining retrievals over unstructured and structured data so that the performance would be better than that of unstructured retrieval or structured retrieval only. Basically, there are three types of combination strategies. (1) **Parallel combination**. Retrieve the structured and unstructured data separately, and then combine the two result lists. (2) **Unstructured-structured serial combination**. Retrieve the unstructured data first. The top-k results, which correspond to entity nodes in the RDF graph, then spread their activations over the RDF graph. Thus, some relevant results which do not contain query terms may be retrieved. (3) **Structured-unstructured**

serial combination. Retrieve the structured data first. The top-k returned entities or subgraphs are then analyzed so that the original query could be better understood. For example, the query is expanded with more effective terms, or is reformulated in terms of related entities and so on. The newly transformed query is then used to retrieve the unstructured data more accurately.

Due to the limit of time, we only experimented on the first two strategies. Firstly, we indexed the unstructured and structured data separately, and used language modeling approaches to retrieve them respectively. We treat the unstructured run as our baseline. Then we combined the unstructured and structured runs using weighted sum approach. Secondly, we used the unstructured run as the input to the algorithm of spreading actions on the RDF graph, and submitted the new ranked list of results after spreading activation.

The results show that ……

2 Combined Retrieval Strategies

In this section, we first present the retrieval models that we used to retrieve unstructured data and structured data respectively, and then discuss different strategies of combining the retrievals over unstructured and structured data.

Given a keyword query and unstructured document collection, we can employ any traditional IR models to retrieve relevant documents. In this paper, we use the language modeling approach since it has the state of art performance among other retrieval models.

For a structured data collection, there are various approaches proposed to look for relevant answers for a keyword query [1]. One of the key problems in structured retrieval is that return units are not predefined as in document retrieval. There are various ways to generate all possible results. Then the results are ranked using either traditional content-based models, e.g. TF-IDF, vector space model, or content-structure-based ranking models. However, there is still lack of a general evaluation campaign for comparing all these retrieval models for keyword search on structured data. So it is very hard to draw any conclusions on these various approaches. In this paper, we simply define the retrieval units of structured retrieval to be entities, which actually correspond to Wikipedia articles in the collection of Linked Data Track. To retrieve entities on RDF graphs, we first aggregate all information about an entity together, i.e. the entity's properties, subjects, objects, etc., and index it as a pseudo document identified by the entity's ID. Then we employ the language modeling approach to rank each pseudo document with respect to the given keyword query.

2.1 Parallel Combination

2.2 Unstructured-Structured Serial Combination

2.3 Structured-Unstructured Serial Combination

3 Experimental Results

Due to the limit of time, we only experimented on the first two strategies. In this section, we discuss the experiment results on the INEX 2012 Linked Data Track.

3.1 Implementation

We indexed the unstructured and structured data separately both using Indri with Krovetz stemmer and a short stop word list *{a, about, an, and, as, at, by, in, of, on, or, that, the, to}*. Remember that we generate a pseudo document for each entity in the structured data set, and index this pseudo document for the entity.

3.2 Results

The evaluation results have not been released by the time when the author wrote this abstract.

4 Conclusions and Future Work

5 Acknowledgements

The research work is supported by the "HGJ" National Science and Technology Major Project of China under Grant No. 2010ZX01042-001-002-002.

References

1. Y. Chen, W. Wang, Z. Liu, and X. Lin, Keyword search on structured and semi-structured data. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data* (SIGMOD '09), Carsten Binnig and Benoit Dageville (Eds.). ACM, New York, NY, USA, 1005-1010.
2. C. Rocha, D. Schwabe, and M. P. Aragao, A hybrid approach for searching in the semantic web. In *Proceedings of the 13th international conference on World Wide Web* (WWW '04). ACM, New York, NY, USA, 374-383.
3. M. Fernandez, V. Lopez, M. Sabou, V. Uren, D. Vallet, E. Motta, and P. Castells, Semantic Search Meets the Web. In *Proceedings of the 2008 IEEE International Conference on Semantic Computing* (ICSC '08). IEEE Computer Society, Washington, DC, USA, 253-260.

Overview of the INEX 2012 Relevance Feedback Track

Timothy Chappell[1] and Shlomo Geva[2]

[1] Queensland University of Technology
timothy.chappell@qut.edu.au
[2] Queensland University of Technology
s.geva@qut.edu.au

Abstract. The INEX 2012 Relevance Feedback track provided participating organisations with an evaluation system designed to simulate the user of a search engine. Participants provided their own search systems designed to interface with the evaluation platform and receive live feedback from the simulated user showing which parts, if any, of the current document were considered by the user to be relevant.
This version of the track was run in a very different manner compared to the INEX 2011 and 2010 versions of the Relevance Feedback track in an attempt to increase participation and strengthen the quality of the evaluations. While the former goal was not met, the new format of the track allowed the entire Wikipedia collection[5] to be used, as opposed to the small subsets used in 2010 and 2011.
We present the evaluation methodology, its implementation, and experimental results obtained for thirteen submissions from two participating organisations.

1 Introduction

This paper presents an overview of the INEX 2012 Relevance Feedback track. The track was designed to facilitate the development of search engine modules that incorporate focused relevance feedback. The INEX Wikipedia Collection[5], a 50.7GB collection of 2,666,190 Wikipedia articles in XML format was used as the data collection for the track. The search topics and assessments used were collected for the INEX 2009 and 2010 Ad Hoc tracks[8][1].

Organisations participated by supplying executables that would communicate with a supplied evaluation platform through standard operating system I/O pipes. The evaluation platform would provide the search topics and, for each document provided to it by the search module, reply with relevant passages. The search module can then make use of this information to rerank the remaining documents as necessary.

After each topic has been searched, the evaluation platform uploads the document IDs returned by the search module for each topic in the form of a *trec*

eval[2]-compatible submission. The submission is evaluated on a remote server against relevance assessments for the topics and the results are sent back to the evaluation platform.

The evaluation platform had two modes, training and evaluation, with a different set of topics for each. Training mode would run the module over a smaller set of 10 topics and, while the submission would still be uploaded to the remote server, the results would be returned to the user but not recorded as a submission. In contrast, evaluation mode uses a larger set of 50 topics and records all runs submitted. Hence, users can provide submissions to the track simply by executing evaluation runs with the platform.

2 Focused Feedback

This track covers the use of focused feedback, a relevance feedback model wherein users specify segments of the document (usually through some form of selection or highlighting tool) considered relevant to the search topic. This allows users to give more flexible feedback when only portions of the current document are relevant to their search.

More information about focused feedback is available in [6].

3 Evaluation

Submissions to the Relevance Feedback track are evaluated from the perspective of a user searching for information on a number of topics. The user reads each document returned by the search system and highlights sections that are relevant to the current topic. If the document returned is not relevant at all, the user simply skips this document and asks for a new one. Hence, the search system has an opportunity to rerank the unseen documents at every step along the way, taking into account new information about what the user is searching for. However, as the user's search experience is the ultimate indicator of search performance, documents the user has already seen are considered frozen and can not be reranked after they have been presented.

To simulate the user, relevance judgments from the Ad Hoc tracks from INEX 2009 [8] and 2010 [1] are used to provide information about which segments of documents are relevant to which topics. These assessments consist of offset-length pairs, each indicating that the specified segment in the given document is relevant to the given topic. The evaluation platform uses these assessments, returning the segments that match the given document. The relevance feedback modules can then rerank the remaining documents in the collection with information from this and from previous feedback to produce more relevant results for the remaining documents to be presented to the user.

At the end of a run, the evaluation platform compiles the documents, in the order they were presented, into a *trec eval*-compatible submission file, which is

uploaded to a remote server where the evaluation is performed. The results are then returned to the user. This serves to keep the relevance judgments secret; though only to an extent as the relevance judgments for the Ad Hoc track are publicly available and it is trivial to convert them to TREC format.

In the 2010 and 2011 versions of the Relevance Feedback track, the evaluation platform would provide the relevance feedback plugin with the offset and length of each segment of relevant text. This was changed in 2012 to reduce the need for the entire, uncompressed collection to be available to the relevance feedback plugin. Instead, the direct text from the documents, stripped of XML tags, was passed to the relevance feedback plugin. This made it more practical to create Relevance Feedback submissions without a copy of the uncompressed Wikipedia collection or a copy of the archive that makes random access within the collection feasible. As the default form the Wikipedia collection is distributed in (.tar.bz2) is not suitable for random access, it is difficult and time-consuming to extract individual documents as they are required. This step will also make it more feasible to create Relevance Feedback tasks based on other large collections, such as ClueWeb09[3].

The topics used for this collection were the topics for the INEX 2009 [8] and 2010 [1] Ad Hoc tracks. After stripping out the topics that had no relevance judgments attached, the first ten were used as the training set. Out of the remaining topics, every second topic was used to make up the evaluation set until all fifty slots were filled.

4 Task

4.1 Overview

Track participants were tasked with creating relevance feedback modules that would interface with the provided evaluation platform and respond with results in answer to queries. With each result, the evaluation platform would respond with relevant passages from each document and the relevance feedback module would have the opportunity to rerank the remaining results in that topic to deliver better results.

In past iterations of the track, these relevance feedback modules were implemented as dynamic plugins written in Java. These plugins were provided by the track participants as submissions. This approach, while effective at preventing approaches like tuning to specific topics, came with a number of drawbacks. It restricted the implementation environment to Java. In addition, because it would not be feasible for the users to submit their own index of the collection (which can be hundreds of megabytes large) or index the Wikipedia collection in its entirety at the time of evaluation, only subsets of the Wikipedia collection were, making it more difficult to gather realistic performance information.

In the 2011 iteration of the Relevance Feedback track, the same system was used; however, participants were also provided with a Java plugin capable of

interfacing with generic platform-dependent executables over pipes. This made it possible to implement relevance feedback modules in more languages but brought with it issues of compatibility as the resulting module had to be evaluated on a specific operating system and hardware architecture.

The 2012 iteration made an attempt to rectify these issues, keeping the pipe communication aspect from the binary interface plugin but otherwise heavily changing the way the track was run. In the 2012 Relevance Feedback track, participants create a relevance feedback module in whatever language they choose and the only restriction is that the module run on their own hardware. The evaluation platform was rewritten for the new task and would communicate directly with the relevance feedback module over pipes and make submissions to a remote server, set up specifically for the track. Making a submission with the new system was as simple as running the evaluation platform (in the correct mode.)

Separate training and evaluation modes were included to allow participants to test their relevance feedback modules with the code without needing to make a submission. Every evaluation submission was recorded by the server to ensure that, while participants could still tune their code to the evaluation topics, all the results of doing so would be recorded.

4.2 Submission format

When the track was first opened, the evaluation platform was made available from the INEX website.

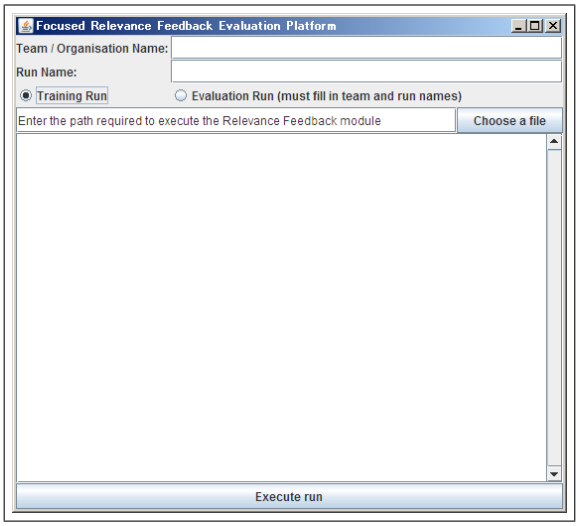


Fig. 1. Evaluation Platform for the INEX 2012 Relevance Feedback track

On supplying a valid path to the relevance feedback module, the evaluation platform would open up an I/O pipe to the module and begin working through the selected topic set.

Choosing the *Evaluation Run* mode would run the module through the 50 topic Evaluation Mode set.

Participating organisations created relevance feedback module executables that adhered to the following specifications, as described in the documentation provided with the evaluation platform:

4.3 Relevance Feedback module interface protocol

The evaluation platform and the module communicate using a pipe, a standard feature of all modern operating systems. Hence, any programming language capable of creating an executable that can read from standard input and write to standard output would be suitable for creating a relevance feedback module for the task.

Each message from the evaluation platform or the relevance feedback module will be in the form of a single line of text ending in a linefeed character. The meaning of the line of text will be derived from the context in which it is submitted.

The evaluation platform communicates first, providing a topic line. This line will either contain the text of the topic or the text EOF, signalling to the module that the evaluation is over and it may exit. The module will respond with a document line. This line will contain either a document ID or the text EOF, signalling to the evaluation platform that the module has finished presenting documents for the current topic and is ready to move on to the next topic. If a document ID is presented, the evaluation platform will respond with feedback.

Feedback will be provided in the form of a line with a number indicating the number of passages of relevant text found in the document. If that number was 0, the document was not relevant and the module should provide the next document ID. Otherwise, the evaluation platform will immediately follow up the number with that many passages of feedback text, each on a single line. After all the lines of feedback have been sent, the module is expected to respond with another document.

4.4 Relevance Feedback module interface format

The topic line supplied by the evaluation platform will be in ASCII text, stripped of characters outside the 32-127 range. The line will be no more than 127 characters long, including the linefeed.

The document ID line returned by the module should contain a number in ASCII text, corresponding to the document ID within the Wikipedia collection of the document to return.

The 'lines of feedback' line returned by the evaluation platform in response to a document ID line will be a number in ASCII text containing the number of segments of relevant text in the document. The feedback will then be followed by lines of text, one for each segment of feedback. The line will be no more than 1048575 characters long, including the linefeed. This, too, will be in ASCII text, stripped of characters outside the 32-127 range.

5 Results

5.1 Submissions

Two groups made a total of 15 submissions to the INEX 2012 Relevance Feedback track, up from four submissions from two groups in 2011. This may be partly due to the new format making it easier to make many submissions as the need for each submission to be packaged into its own Java archive and uploaded was no longer present.

Queensland University of Technology made five submissions using an experimental relevance feedback mode in TopSig[4]. This was originally planned to be the relevance run for the INEX 2012 Relevance Feedback track but due to time constraints this was not possible. The TopSig runs, apart from the baseline run which did not make use of feedback at all, simply used the feedback text as a new query and reranked the remaining documents found by the initial query each time. More information about the signature approach used by TopSig can be found in [7].

The baseline TOPSIG run consisted of an untuned 1024-bit signature search without using collection statistics or relevance feedback returning 100 documents per topic. Subsequent TOPSIG runs incorporated the simple feedback system described earlier. TOPSIG-RF1 reranked the remaining documents not yet presented to the user by using the last line of feedback presented as a new search query. TOPSIG-RF2 kept the same approach but increased the number of documents returned to 1000. TOPSIG-RF3 increased the signature size to 2048 bits and TOPSIG-RF4 changed the feedback approach to use all of the feedback presented instead of the last line. As this is the first experiment performed with using active relevance feedback for signature searching in TopSig, preliminary results are only experimental.

The Universidad Autónoma Metropolitana made 10 submissions using Indri[9] as a base and employing a Markov random field to rerank results with relevance feedback. The BASE-IND run consists of a run with Indri without incorporating relevance feedback while the MF and LF runs consist of the results when adding the 20 most frequent and least frequent terms respectively from the feedback to the query. The RRMRF runs are also based on Indri but employ the Markov random field for reranking. The 100D, 300D and 1000D runs are the results from returning 100, 300 and 1000 documents respectively per topic. The

L values represent the lambda parameter within the reranking approach. More details are available in the Universidad Autónoma Metropolitana's track paper.

5.2 Evaluation

Two sets of topics were made available, not directly to participants but through the evaluation platform. The training set used the first 10 topics from the INEX 2009 Ad Hoc track while the evaluation set used 50 topics chosen from every 2nd topic from the INEX 2009 and 2010 Ad Hoc tracks, excluding the topics used for the training set. Topics without associated relevance judgments were removed from the set beforehand.

All of the submissions were run through *trec eval*[2] using default settings. The results of each run were also presented to the submitter immediately after submission.

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.

Unlike in the previous incarnations of the relevance feedback track, the evaluation platform did not come with the option of producing no-feedback runs. However, both participating organisations created runs that did not utilise feedback, showing where feedback has improved the results of these runs.

5.3 Comparisons

The following tables show the results of each submission in terms of average precision and R-precision.

The charts compare groups of submissions by exact precision. The y axis shows the proportion of relevant documents retrieved and the x axis shows the total number of documents retrieved. Figure 2 shows a comparison of the exact precision of each of the UAM runs submitted. Figure 3 shows a comparison of each QUT run, while figure 4 gives a comparison of the best feedback and non-feedback runs from each group.

Group	Submission	Average Precision	R-Precision
UAM	BASE-IND	0.1015	0.1828
UAM	BASE-INDQE-20tMF	0.0775	0.1396
UAM	BASE-INDQE-20tLF	0.0395	0.0718
UAM	BASE-INDQE-20tMFandLF	0.0728	0.1364
UAM	RRMRF-100D-L03	0.094	0.1612
UAM	RRMRF-100D-L05	0.0946	0.1595
UAM	RRMRF-300D-L03	0.1002	0.1769
UAM	RRMRF-300D-L05	0.1004	0.1805
UAM	RRMRF-1000D-L03	0.1015	0.1824
UAM	RRMRF-1000D-L05	0.1015	0.1824
QUT	TOPSIG	0.1393	0.2059
QUT	TOPSIG-RF1	0.1459	0.2028
QUT	TOPSIG-RF2	0.2015	0.2509
QUT	TOPSIG-RF3	0.2352	0.2747
QUT	TOPSIG-RF4	0.2408	0.2763

Table 1. Average precision and R-precision for submitted runs

Submission	@5	@10	@15	@20	@30	@100	@200	@500	@1000
BASE-IND	0.456	0.41	0.36	0.327	0.3007	0.1844	0.1166	0.0557	0.0292
BASE-INDQE-20tMF	0.396	0.33	0.2907	0.264	0.228	0.1232	0.0789	0.0406	0.0239
BASE-INDQE-20tLF	0.308	0.198	0.148	0.125	0.0967	0.0412	0.0241	0.0114	0.0063
BASE-INDQE-20tMFandLF	0.392	0.304	0.2653	0.237	0.204	0.1136	0.0741	0.0396	0.0225
RRMRF-100D-L03	0.448	0.406	0.3733	0.348	0.3107	0.1846	0.0923	0.0369	0.0185
RRMRF-100D-L05	0.452	0.404	0.372	0.351	0.312	0.1846	0.0923	0.0369	0.0185
RRMRF-300D-L03	0.452	0.398	0.3587	0.337	0.3027	0.1876	0.117	0.0512	0.0256
RRMRF-300D-L05	0.46	0.42	0.368	0.349	0.3	0.1708	0.1157	0.0512	0.0256
RRMRF-1000D-L03	0.456	0.41	0.36	0.328	0.3007	0.1848	0.1166	0.0557	0.0292
RRMRF-1000D-L05	0.456	0.41	0.36	0.328	0.3007	0.1848	0.1166	0.0557	0.0292
TOPSIG	0.448	0.42	0.3827	0.366	0.332	0.232	0.116	0.0464	0.0232
TOPSIG-RF1	0.496	0.44	0.4067	0.384	0.3593	0.232	0.116	0.0464	0.0232
TOPSIG-RF2	0.524	0.46	0.4173	0.398	0.3747	0.242	0.1733	0.0933	0.0569
TOPSIG-RF3	0.56	0.504	0.4813	0.465	0.4187	0.2614	0.1906	0.1049	0.0623
TOPSIG-RF4	0.568	0.52	0.4773	0.459	0.42	0.2656	0.1923	0.1054	0.0623

Table 2. Exact precision of submitted runs

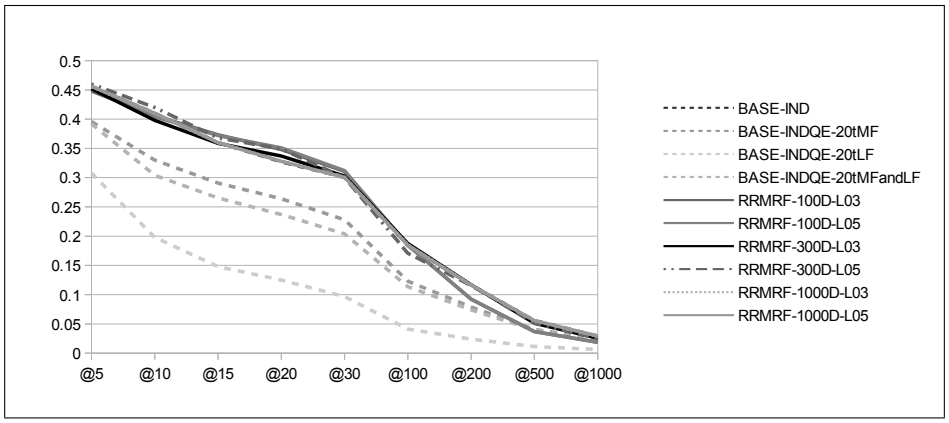


Fig. 2. Exact precision of submissions by the Universidad Autónoma Metropolitana

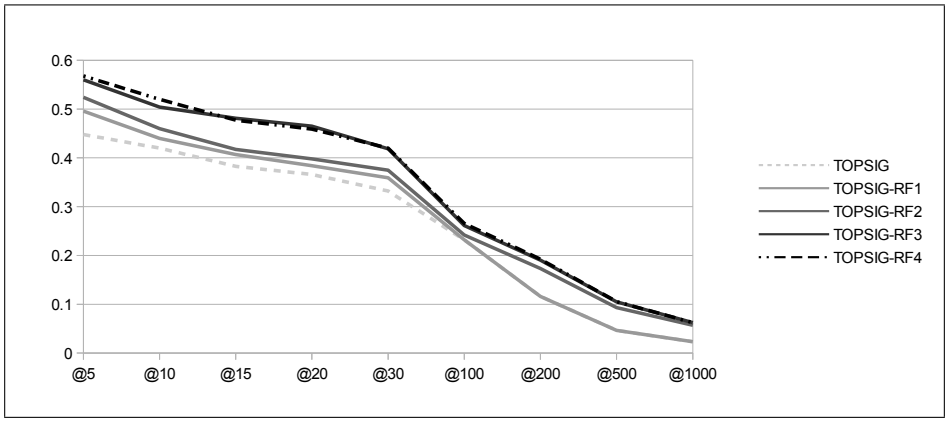


Fig. 3. Exact precision of submissions by the Queensland University of Technology

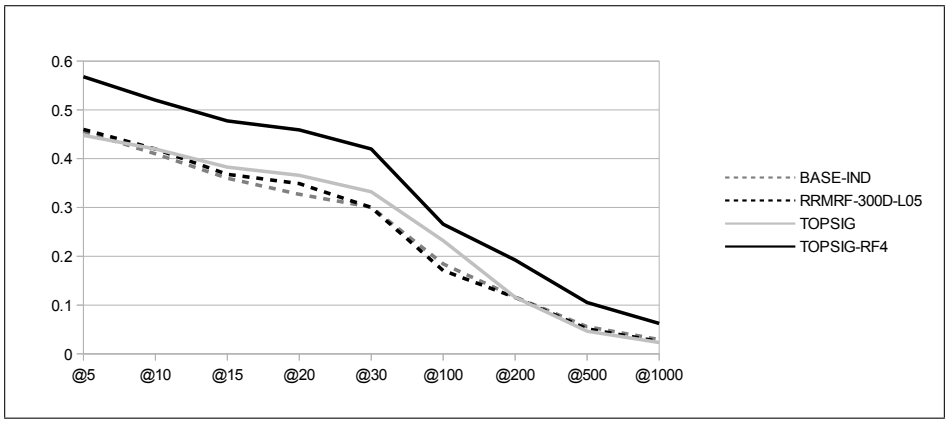


Fig. 4. Exact precision comparison: best non-RF and best RF submissions from each participating organisation

6 Conclusion

We have presented the Relevance Feedback track at INEX 2012.

It is difficult to compare results between different incarnations of the track. While the results are far worse in 2012 from an objective perspective, the large changes in the way the tracks were run between the two years can account for this. In the 2010 and 2011 versions of the Relevance Feedback track, only subsets of the Wikipedia collection were used and these subsets heavily favoured relevant documents. As the burden of finding the results has shifted more to the search systems in the 2012 version of the track the overall results have also declined. The search systems presented at the INEX 2012 Relevance Feedback track are not necessarily worse than those presented in 2011.

While the number of submissions has increased since INEX 2011, the number of participants has not. Lowering the barriers to entry have not resulted in the increased interest in the Relevance Feedback track that was expected. Part of this may be due to the lack of a strong reference submission. In the INEX 2010 and 2011 iterations of the Relevance Feedback track, a relevance feedback module with complete was provided to participants in advance, to be used as a base for other submissions if desired. No equivalent was provided for the INEX 2012 Relevance Feedback track which may have discouraged participation.

7 Acknowledgements

We would like to thank all the participating organisations for their contributions and hard work.

References

1. P. Arvola, S. Geva, J. Kamps, R. Schenkel, A. Trotman, and J. Vainio. Overview of the INEX 2010 Ad Hoc track. *Comparative Evaluation of Focused Retrieval*, pages 1–32, 2011.
2. C. Buckley. trec eval IR evaluation package, 2004.
3. J. Callan, M. Hoy, C. Yoo, and L. Zhao. Clueweb09 data set. *boston. lti. cs. cmu. edu, Jan*, 2009.
4. T. Chappell and S. Geva. TopSig: Topological signature indexing and search engine. *http://www.topsig.org/*, 2012.
5. Ludovic Denoyer and Patrick Gallinari. The Wikipedia XML Corpus. *SIGIR Forum*, 2006.
6. S. Geva and T. Chappell. Focused relevance feedback evaluation. *Simulation of Interaction*, page 9, 2010.
7. S. Geva and C.M. De Vries. Topsig: Topology preserving document signatures. 2011.
8. S. Geva, J. Kamps, M. Lethonen, R. Schenkel, J. Thom, and A. Trotman. Overview of the INEX 2009 Ad Hoc track. *Focused Retrieval and Evaluation*, pages 4–25, 2010.
9. 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.

UAM at INEX 2012 Relevance Feedback Track: Using a Probabilistic Method for Ranking Refinement*

Esaú Villatoro-Tello, Christian Sánchez-Sánchez, Héctor Jiménez-Salazar, Wulfrano A. Luna-Ramírez, and Carlos Rodríguez-Lucatero

Language and Reasoning Group, Information Technologies Dept.,
Universidad Autónoma Metropolitana (UAM), Unidad Cuajimalpa, Mexico City.
{evillatoro,csanchez,hjimenez,wluna,crodriguez}@correo.cua.uam.mx

Abstract. This paper describes the system developed by the Language and Reasoning Group of UAM for the Relevance Feedback track of INEX 2012. The presented system focuses on the problem of ranking documents in accordance to their relevance. It is mainly based on the following hypotheses: *(i)* current IR machines are able to retrieve relevant documents for most of general queries, but they can not generate a pertinent ranking; and *(ii)* focused relevance feedback could provide more and better elements for the ranking process than isolated query terms. Based on these hypotheses, our participation at INEX 2012 aimed to demonstrate that using some query-related relevance feedback it is possible to improve the final ranking of the retrieved documents.

1 Introduction

Information Retrieval (IR) deals with the representation, storage, organization, and access to information items[1] [1]. Given some query, formulated in natural language by a user, the IR system is suppose to retrieve and sort according to their relevance degree documents satisfying user's information needs [4].

The word *relevant* means that retrieved documents should be semantically related to the user information need. Hence, one main problem of IR is determining which documents are, and which are not relevant. In practice this problem is usually regarded as a *ranking* problem, whose goal is to define an ordered list of documents such that documents similar to the query occur at the very first positions.

Over the past years, IR Models, such as: Boolean, Vectorial, Probabilistic and Language models have represented a document as a set of representative keywords (i.e., index terms) and defined a *ranking* function (or retrieval function)

* This work was done under partial support of CONACyT (project grant 153315) and SEP-PROMEP (project grant 48510294(UAM-C-CA-31)). We also thank UAM for their assistance.

[1] Depending on the context, items may refer to text documents, images, audio or video sequences.

to associate a relevance degree for each document with its respective query [1, 4]. In general, these models have shown to be quite effective over several tasks in different evaluation forums (CLEF[2] and TREC[3]). Nevertheless, these retrieval systems still fail at retrieving most of the relevant documents to a given query in the first positions. The latter is due to the fact that modelling user intentions from queries is, in general, a highly subjective and difficult task, hence, post-processing and ranking refinement strategies have been adopted [12–15].

Post-retrieval techniques aim at refining retrieval results by means of feature re-weighting, query modification, document re-ranking and relevance feedback. The common idea is to interact with the user in order to learn or to improve a model of the underlying user's information need. Acceptable results have been obtained with such methods, however, they still have several limitations, including: *i)* the need of extensive user interaction[4]; *ii)* multiple execution of retrieval models; *iii)* the on-line construction of classification methods; *iv)* the lack of contextual information in the post-retrieval processing, which may be helpful for better modelling users' information needs; and *v)* the computational cost that involves processing the entire collection of documents each feedback iteration.

Document re-ranking or ranking refinement in information retrieval has been a widely research topic during the last fifthteen years. There are two main approaches for this task: *i)* indirect re-ranking via some query expansion strategy, and *ii)* direct re-ranking on initial retrieved documents [15]. Normally, query expansion strategies assume that top ranked documents are more likely to be relevant, the terms contained within these documents can be used to augment the original query and then a better ranking can be expected via a second retrieval process. In contrast, direct re-ranking strategies try to improve the ranking of the initial set of retrieved documents by directly adjusting their positions without the need of performing a second retrieval process, normally, this type of strategy use the information contained within the retrieved documents (*e.g.,* inter-document similarities) to generate a better ranking of them. The generated output (*i.e.,* a list of ranked documents) by any of this two strategies would be of obvious benefit to users, for example, direct ranking refinement can be used to improve automatic query expansion since a better ranking in the top retrieved documents can be expected.

1.1 Our approach

Our participation at the INEX 2012 Relevance feedback track proposes using an alternative post-retrieval technique that aims at improving the results provided

[2] Cross Language Evaluation Forum (http://www.clef-campaign.org/).

[3] Text Retrieval Conference (http://trec.nist.gov/).

[4] It is worth mentioning that if available, user interaction should be included in post-retrieval techniques as it is evident that information provided by the user is much more reliable than that obtained by a fully automatic process. Hence, the goal of post-processing techniques should be minimizing users' interaction instead of completely eliminate it from the process.

by a document retrieval system and that overcomes some of the limitations of current post-retrieval methods. Our work classifies as a direct document ranking refinement strategy. In particular we face the problem of re-ranking[5] a list of documents retrieved by some information retrieval system. This problem is motivated by the availability of retrieval systems that present high-recall and low-precision performance, which evidences that the corresponding retrieval system is in fact able to retrieve many relevant documents but has severe difficulties to generate a pertinent ranking of them. Hence, given a list of ranked documents, the problem we approach consists of moving relevant documents to the first positions and displacing irrelevant ones to the final positions in the list.

We propose a solution to the ranking refinement problem based on a Markov Random Field (MRF) [5, 9, 6, 16] that aims at classifying the ranked documents as relevant or irrelevant. Each document in the retrieved list is associated to a binary random variable in the MRF (*i.e.,* a node), the value of each random variable indicates whether a document is considered relevant (when its value is 1) or not (when its value is 0). The MRF considers several aspects: 1) the information provided by the base information retrieval system, 2) similarities among retrieved documents in the list, and 3) information obtained through a relevance feedback process. Accordingly, we reduce the problem of ranking refinement to that of minimizing an energy function that represents a trade-off between document relevance and inter-document similarity. The information provided by the information retrieval system is the base of our method, which is further enriched with contextual and relevance feedback information.

Our motivation for considering context information is that relevant documents to a query will be similar to each other and to its respective query, to some extent; whereas irrelevant documents will be different among them and not as similar to the query as the relevant documents[6]. Relevance feedback information has two main purposes: *i)* to work as a seed generation mechanism for propagating the relevancy/irrelevancy status of nodes (documents) in the MRF, and *ii)* to denote the users' search intention by working as *example texts*.

At this point it is important to mention that, traditionally a relevance feedback process takes as input a set of n documents (tentatively relevant) and generates as output a set of k isolated terms (tentatively relevant to the query) which are further employed for a query expansion process. For our purposes we will employ all the information contained in the feedback (called *example texts*) since by doing this we have showed [12–14] that it is possible to make a more accurate approximation of the users' search intention (*i.e.,* to become into a more explicit representation the implicit information contained in the query).

The proposed MRF does not require of multiple executions of IR models, nor training classification methods, and it can work without user intervention;

[5] Also known as the problem of *Ranking Refinement*.

[6] Keep in mind that irrelevant documents will be similar to the query in some degree since such documents were obtained by an IR system through that query in the first place.

therefore, our MRF overcomes the main limitations of current post-processing techniques.

1.2 Structure of the paper

The rest of the paper is organized as follows. Section 2 introduces the proposed Markov Random Field for ranking refinement in document retrieval. Section 3 describes the experimental platform used to evaluate and compare our ranking strategy. Section 4 presents the experimental results. Finally, section 5 depicts our conclusions.

2 System Description

A general outline of the proposed method is given in Figure 1. Given a query, the IR system retrieves from a given collection of documents a list of files sorted according to a relevance criteria. From this list, some relevant documents are selected based on a relevance feedback approach[7]. For each document in the list, the textual features are extracted. The text contained in each document in the list, the query given by the user, and a subset of information selected via relevance feedback, are combined to produce a re-ordered list. This re-ranking is obtained based on a Markov random field (MRF) model that separates the relevant documents from irrelevant ones, generating a new list by positioning the relevant documents first, and the others after. Next we give a brief review of MRFs, and then we describe in detail each component of the proposed method.

2.1 Markov Random Fields

Markov Random Fields (MRF) are a type of undirected probabilistic graphical models that aim at modelling dependencies among variables of the problem in turn [5, 9, 6, 16]. MRFs have a long history within image processing and computer vision [7]. They were first proposed for denoising digital images [5, 9, 6, 16] and since then a large number of applications and extensions have been proposed.

MRF modelling has appealing features for problems that involve the optimization of a configuration of variables that have interdependencies among them. Accordingly, MRFs allow the incorporation of contextual information in a principled way. MRFs rely on a strict probabilistic modelling, yet they allow the incorporation of prior knowledge by means of potential functions. For those reasons, in this paper we adopted an MRF model for refining the initial ranking of a set of documents retrieved by some IR system. The rest of this sections summarizes the formalism of MRFs.

An MRF is a set of random variables $F = \{f_1, ..., f_N\}$ indexed by sites or nodes where the following conditions hold:

$$P(f_i) \geq 0, \forall f_i \in F \tag{1}$$

[7] In the context of the Relevance Feedback track from INEX, we were given as feedback *relevant passages* instead of full documents though the Evaluation Platform.

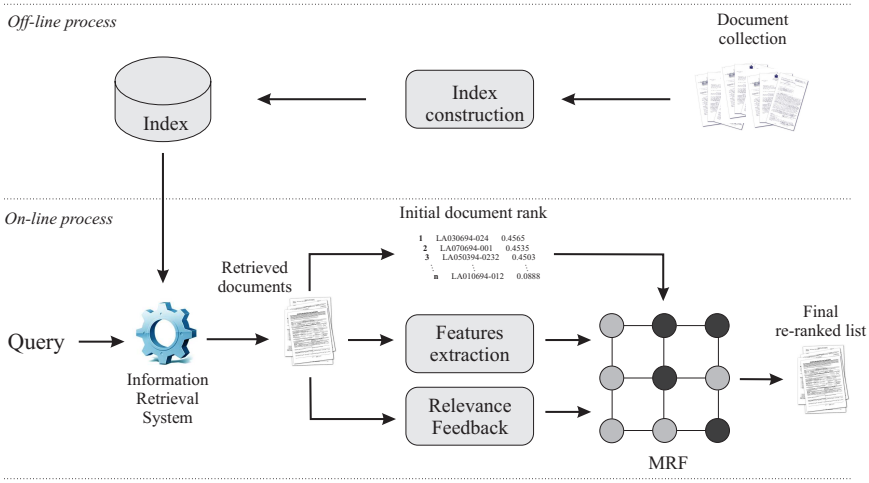


Fig. 1. Block diagram of the proposed ranking refinement method employed in the INEX 2012

$$P(f_i|f_{S-\{i\}}) = P(f_i|\mathcal{N}(f_i)) \tag{2}$$

where $\mathcal{N}(f_i)$ is the set of neighbours of f_i according to the neighbouring system $\mathcal{N}$. Formula 1 is the so called positivity condition and avoids negative probability values, whereas expression 2 states that the value of a random variable depends only on the set of neighbours of that variable.

It has been shown that an MRF follows a Gibbs distribution [3], where a Gibbs distribution of the possible configurations of F with respect to $\mathcal{N}$ has the following form:

$$P(F) = Z^{-1} \times e^{-\frac{1}{T}E(F)} \tag{3}$$

where Z is a normalization constant and the T is the so called temperature parameter (a common choice is $T = 1$) and $E(F)$ is an energy function of the following form:

$$E(F) = \sum_{c \in C} V_c(f) = \sum_{\{i\} \in C_1} V_1(f_i) + \sum_{\{i,j\} \in C_2} V_2(f_i, f_j) + \dots \tag{4}$$

where "..." denotes possible potentials V_c defined over higher order neighbourhoods $C_3, C_4 \dots, C_K$; each C_i defines a neighbourhood system of order i between the nodes of the MRF. Often the set F is considered the union of two subsets of random variables $X \cup Y$; where X is the set of observed variables and Y is the set of output variables, which state we would like to predict. Potentials V_c are problem dependent and commonly learned from data.

One of the main problems in MRFs is that of selecting the most probable configuration of F (*i.e.,* an assignment of values to each variable f_i of the field). Such configuration is determined by the configuration of F that minimizes expression 4, for which a diversity of optimization techniques have been adopted [5, 9, 6, 16].

2.2 Proposed Model

In our case we consider a MRF in which each node corresponds to a document in the list. Each document is represented as a random variable with 2 possible values: *relevant* and *irrelevant*. We consider a fully connected graph, such that each node (document) is connected to all other nodes in the field; that is, we defined a neighbourhood scheme in which each variable is adjacent to all the others. Given that the number of documents in the list is relatively low (100, 300 and 1000 in the experiments), to consider a complete graph is not a problem computationally, and allows us to consider the relations between all documents in the list.

For representing the documents, and evaluating the internal and external similarities, we consider all the words contained in each document (except stopwords), it is worth mentioning that we did applied a stemming process to all documents. To describe the documents we used a binary bag of words (BOW) representation, in which each vector element represents a word from the collection vocabulary; and the *example texts* are represented in the same manner. The internal and external similarities are considered via the energy function described next.

2.3 Energy Function

The energy function of the MRF combines two factors: the similarity between the documents in the list (*internal* similarity); and external information obtained from the original order and the similarity of each document with the provided feedback (*external* similarity). The internal similarities correspond to the interaction potentials and the external similarities to the observation potentials. The proposed energy function takes into account both aspects and is defined as follows:

$$E(F) = \lambda V_c(f) + (1 - \lambda)V_a(f) \tag{5}$$

Where V_c is the interaction potential and it considers the similarity between random variable f and its neighbours, representing the support that neighboring variables give to f. V_a is the observation potential and represents the influence of external information on variable f. The weight factor λ favours V_c ($\lambda > 0$), V_a ($\lambda = 0$), or both ($\lambda = 0.5$).

V_c is defined as:

$$V_c(f) = \begin{cases} \bar{Y} + (1 - \bar{X}) \; if \; f = irrelevant \\ \bar{X} + (1 - \bar{Y}) \; if \; f = relevant \end{cases} \tag{6}$$

Where $\bar{Y}$ represents the average distance between variable f and its neighbours with irrelevant value. $\bar{X}$ represents the average distance between variable f and its neighbors with relevant value. The distance metric used to measure the similarity between variables is defined as: $1 - dice(f, g)$, where $dice(f, g)$

represents the *Dice* coefficient [8], and is defined as: $dice(f,g) = \frac{2|f \cap g|}{|f \cup g|}$. V_a is defined as follows:

$$V_a(f) = \begin{cases} (1 - dist(f,e)) \times g(posinv(f)) \; iff = irrelevant \\ \\ dist(f,e) \times g(pos(f)) \qquad\qquad iff = relevant \end{cases} \tag{7}$$

The V_a potential is obtained by combining two factors. The first indicates how similar, $dist(f,e)$, or different, $1-dist(f,e)$ is the f variable with the *example texts* (e) (*i.e.,* the information provided by the feedback). Where $dist(f,e)$ is defined as: $1 - dice(f,e)$. The second is a function that converts the position in the list given by a base IR machine to a real value. The function used $g(x) = exp(x/100)/exp(5)$ [2][8]. The function $pos(f)$ returns the position of the document f in the original list, $posinv(f)$ returns the inverse position of the f variable in this list.

Having described each potential, the proposed energy function is defined as:

$$E(F) = \begin{cases} \lambda\bar{Y} + (1 - \bar{X}) + (1 - \lambda)[1 - dist(f,e)) \times g(posinv(f)] \; iff = irrelevant \\ \\ \lambda\bar{X} + (1 - \bar{Y}) + (1 - \lambda)dist(f,e) \times g(pos(f)) \qquad iff = relevant \end{cases} \tag{8}$$

The initial configuration of the MRF is obtained by relevance feedback. That is, the subset of documents that contain relevant passages selected via relevance feedback are initialized as relevant, and all other documents as irrelevant. Then, the MRF configuration of minimum energy (MAP) is obtained via stochastic simulation using the ICM algorithm. At the end of this optimization process, each variable (document) has a value of relevant or irrelevant. Based on these values, a new re-ordered list is produced, by positioning first the relevant documents according to the MRF, and then the not-relevant ones.

3 Experimental Setup

In this section we describe the experimental setup that we employed for the proposed method during the INEX 2012 competition. A brief description of the base IR system used is given as well as its configuration. Besides this, we describe an additional ranking refinement strategy employed in our submitted runs, as well as the documents collection and the evaluation measures.

3.1 Base IR System

As we have mentioned before, our ranking refinement strategy does not depend on any particular IR system. However, in order to perform our experiments

[8] The intuitive idea of this function is such that it first increases slowly so that the top documents have a small potential, and then it increases exponentially to amplify the potential for those documents in the bottom of the list.

we employed as base IR system the well known information retrieval system LEMUR-INDRI. This system is part of the Lemur Project[9] started in 2000 by the Center for Intelligent Information Retrieval (CIIR) at the University of Massachusetts, Amherst, and the Language Technologies Institute (LTI) at Carnegie Mellon University. Particularly the LEMUR-INDRI toolkit is a search engine that provides state-of-the-art text search facilities, a rich structured query language for different text collections, and is considered as a robust system capable of producing comparable results to new IR schemes.

For all our experiments, the collections were indexed by this tool using a probabilistic language model. For this purpose, collections were preprocessed by applying stop word elimination as well as a stemming process. For our experiments we employed a list of 571 stop words available in the CLEF site[10]. Additionally, for the stemming process we employed the well known Porter algorithm [10].

As baseline results we considered the performance obtained under this configuration employing the LEMUR-INDRI search engine.

3.2 Query Expansion via Relevance Feedback

A query expansion via relevance feedback process is a controlled technique which main goal is to reformulate a query. In other words, a relevance feedback strategy is normally a previous step for a query expansion (QE) process. The basic idea is to select a set of k words which are related to a set of documents that have been previously retrieved and tagged as relevant by some user. Further, this words are added to the original query [11]. In order to apply a relevance feedback process it is necessary to perform a first search (*i.e.,* a first retrieval process) which generates an ordered list of documents. Afterwards, the user selects from the first positioned documents those that he considers as relevant (*i.e.,* the user establishes the documents' relevance). This relevance judgements that the user just gave to the documents are employed to compute a new set of values that indicate in a more accurate form the impact of each word in the original query[11].

As an alternative solution, we performed some experiments applying a QE process. For this, every time some feedback was given, we selected the k most frequent/less frequent words for its addition to the original query in order to perform a new retrieval process. Among the disadvantages of QE is the computational cost implied, since it is necessary to perform a second retrieval process. Besides this, relevance feedback strategies have shown to be sensitive to the quality of the added words, since adding an *irrelevant word* could be very harmful for the IR system

[9] http://www.lemurproject.org

[10] Cross Language Evaluation Forum (http://www.clef-campaign.org/).

[11] When the relevant documents are identified by some automatic process, it is assumed that documents placed at the top positions of the list are in fact relevant, and the new set of words that will be added to the query are automatically selected; this type of feedback is known as *blind relevance feedback*.

3.3 Data set

In the framework of the INEX 2012 Relevance Feedback track we were provided with the Wikipedia XML Corpus as the test collection. This collection was created from the October 8, 2008 dump of the English Wikipedia articles, and contains 2,666,190 articles, which represent more that than 50 GiB of disk space.

3.4 Evaluation

The evaluation of results was carried out using a measure that has demonstrated its pertinence to compare IR systems, namely, the Mean Average Precision (MAP). MAP is defined as follows:

$$MAP = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \left(\frac{\sum_{r=1}^{m} P_i(r) \times rel_i(r)}{n} \right)$$

Where $P_i(r)$ is the precision at the first r documents, $rel_i(r)$ is a binary function which indicates if document at position r is relevant or not for the query i; n is the total number of relevant documents for the query i, m is the number of relevant documents retrieved and Q is the set of all queries.

Intuitively, this measure indicates how well the system puts into the first positions relevant documents. It is worth pointing out that since our IR system was configured to retrieve 1000 documents per query, MAP values are measured at 1000 documents.

On the other hand, $P@N$ is defined as the percentage of retrieved relevant items at the first N positions of the result list. Finally $R - Prec$ is defined as the precision at R-th position in the ranking of results for a query that has R relevant documents.

3.5 Experiments definition

The use-case of the INEX 2012 relevance feedback track is as follows: assume a single user searching with a particular query in an information retrieval system that supports relevance feedback. The user highlights relevant passages of text in returned documents (if any) and provides this feedback to the information retrieval system. The IR system re-ranks the remainder of the unseen results list in order to provide more relevant results to the user.

Accordingly, we conducted a series of experiments with the following objectives: *i)* to test the results of the proposed method compared with the traditional re-ranking strategies, *ii)* to evaluate the sensitivity of the method to the model parameters.

We defined 10 different configurations, which are described below:

– BASE-IND: represents the experiment performed using just the INDRI IR machine. For this experiment, if some feedback is provided, the feedback is ignored and the next retrieved document is showed.

– BASE-IND-QE20tMF: this experiment was performed using a QE strategy as re-ranking method. Once an initial documents list is provided by INDRI, the system keep delivering documents until some feedback is provided. If some feedback occurs, our systems reformulates the original query adding the 20 most frequent terms contained in the feedback passages and applies a new retrieval process. The new retrieved documents list is then showed to the user. This procedure is repeated every time some feedback occurs.
– BASE-IND-QE20tLF: this configuration works in a similar form to the previous experiment, although the only difference is that we reformulate the original query by adding the 20 less frequent terms.
– BASE-IND-QE20tMFandLF: this configuration works in a similar form to the previous experiment, although the only difference is that we reformulate the original query by adding the 20 most frequent and the 20 less frequent terms.
– RRMRF-xxxD-Lxx: these experiments represent the runs that employed our proposed markov random field as re-ranking strategy. The first three x's represent the number of documents that were used to construct the field, whereas the second x's represent the lambda (λ) parameter value. This configuration works as follows: once we have retrieved an initial list of documents using INDRI, our system keeps delivering documents until some feedback is provided. If some feedback occurs, our proposed method constructs a virtual *example text* (e) employing all the information contained in the feedback and marks as relevant those documents that provided the feedback. After the iteration process we show to the user the next relevant document. This process repeats every time some feedback is provided.

4 Results

Table 1 shows the evaluation results from all the submitted runs by our team. It is important to mention that the INEX 2012 Relevance Feedback track employed a new methodology for submitting results. During this campaign, participant teams were provided with an Evaluation Platform (EP) that worked as an online tool for providing the queries as well as for providing the feedback (if any) for every document showed to the EP. A total of 50 queries were processed, hence, Table 1 shows the average results obtained across the 50 queries.

Notice that even when the baseline configuration does not obtained a very high performance, obtained results are among the best performances. Remember that the baseline method means that we are using only the output produced by the INDRI IR machine. Therefore, obtained results indicate that INDRI was not able to retrieve a significant number of relevant documents, resulting in low recall levels.

A preliminary analysis indicate us that the configuration of our IR machine was not the most adequate for the type of queries that we processed. Most of the queries consisted on a set of general terms that do not necessarily represent a query formulated in natural language. We believe that using a boolean or a

Experiment	MAP	R-Prec	P@5	Recall
BASE-IND	**0.1015**	**18.28%**	45.60%	**25.93%**
BASE-IND-QE20tMF	0.0775	13.96%	39.60%	21.25%
BASE-IND-QE20tLF	0.0395	7.18%	30.80%	5.61%
BASE-IND-QE20tMFandLF	0.0728	13.64%	39.20%	20.03%
RRMRF-100D-L0.3	0.0940	16.12%	44.80%	16.40%
RRMRF-100D-L0.5	0.0946	15.95%	45.20%	16.40%
RRMRF-300D-L0.3	0.1002	17.69%	45.20%	22.76%
RRMRF-300D-L0.5	0.1004	18.05%	**46.00%**	22.76%
RRMRF-1000D-L0.3	**0.1015**	18.24%	45.60%	**25.93%**
RRMRF-1000D-L0.5	**0.1015**	18.24%	45.60%	**25.93%**

Table 1. Official Evaluation results obtained in the framework of the INEX 2012 Relevance Feedback track

vectorial model instead of a probabilistic one could provide better results in terms of the *recall* measure.

As can be observed, results obtained by the configurations that employed a query expansion strategy as re-ranking mechanism obtained the worst set of results. This indicate that terms considered during the query reformulation were somehow irrelevant even when they were provided by an user.

Finally, notice that our proposed method it is able to provide a better ranking when using 300 documents and lambda 0.5, since it provides better results at the first five positions of the final list (P@5). In general, we can observe that using few documents and a high value of lambda our method is able to produce acceptable results (almost similar to those obtained when using 1000 documents). however, the main limitation of our system was the INDRI initial bad performance (low *recall* values). It is important to mention, as established in [14], the proposed Markov random field depends on having high *recall* levels.

5 Conclusions

This paper proposed a method for improving the ranking of a list of retrieved documents by a IR system. Based on a relevance feedback approach, the proposed method integrates the similarity between the documents in the list (*internal* similarity); and external information obtained from the original order, the query and the provided feedback (*external* similarty), via a MRF to separate the relevant and irrelevant documents in the original list.

Experiments were conducted in the framework of the INEX 2012 Relevance Feedback track. For our experiments we avoid using any specialized external resources, since we were interested in evaluating the pertinence of the method employing only textual (document's words) features. Results showed that considering few documents and providing more importance to the internal similarities among documents, the proposed method is able to reach an acceptable performance. An initial analysis indicates that for this collection, it is necessary to

employ as IR method a traditional boolean or vectorial model in order to improve the recall levels of the IR machine, which is an important condition for the proposed method to work properly.

References

1. Baeza-Yates R., and Ribeiro-Neto B. (1999) *Modern Information Retrival.* Addison Wesley.
2. Chávez O., Sucar L. E., and Montes M. (2010). Image Re-ranking based on Relevance Feedback Combining Internal and External Similarities. In *The FLAIRS Conference*, Daytona Beach, Florida, USA.
3. Geman S., and Geman D. (1987) Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of Images. In *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms.* pp. 564-584.
4. Grossman D. A., and Frieder O. (2004) *Information Retrieval, Algorithms and Heuristics.* Springer,2nd edition.
5. Kemeny J., Snell J.L. and Kanpp A.W. (1976) Denumerable Markov Chains. New York-Heidelberg-Berlin, Springer Verlag.
6. Lauritzen S. L. (1996). Graphical Models. Oxford University Press, New York NY.
7. Li S. Z. (2001) Markov Random Field Modeling in Image Analysis. 2nd. Edition, Springer.
8. Mani, I. (2001). Automatic Summarization. In *Natural Language Processing*, Vol. 3 . John Benjamins Publishing Co.
9. Pearl J. (1988) Probabilistic Reasoning in Intelligent Systems. Morgan Kaufman, San Mateo CA.
10. Porter M. F. (1997) An Algorithm for suffix stripping. Morgan Kaufman Publishers Inc. pp. 313-316.
11. Salton G., and Buckley C. (1990) Improving Retrieval Performance by Relevance Feedback. In *Journal of the American Society for Information Science*: 41(4), pp. 288-297
12. Villatoro-Tello E. , Montes-y-Gómez M., and Villaseñor-Pineda L. (2009) A Ranking Approach based on Example Texts for Geographic Information Retrieval. In *Post-proceedings of the 9th Workshop of the Cross Language Evaluation Forum CLEF 2008.* Vol. 5822, pp. 239-250. Lecture Notes in Computer Science. Berlin: Springer-Verlag.
13. Villatoro-Tello E., Villaseñor-Pineda L., and Montes-y-Gómez (2009) M. Ranking Refinement via Relevance Feedback in Geographic Information Retrieval. In *Mexican International Conference on Artificial Intelligence MICAI 2009.* Vol. 5845, pp. 165-176. Lecture Notes in Computer Science. Berlin: Springer-Verlag.
14. Villatoro-Tello E. , Juárez-González A., Montes-y-Gómez M., Villaseñor-Pineda L., and Sucar E. L. (2012) Document Ranking Refinement Using a Markov Random Field Model. In *Journal of Natural Language Engineering* Volume 18, issue 02, pp. 155-185
15. Yang L., Ji D., Zhou G., Nie Y., and Xiao G. (2006) Document Re-ranking Using Cluster Validation and Label Propagation. In *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management.* pp. 690-697.
16. Winkler G. (2006) Image Analysis, Random Fields and Markov Chain Monte Carlo Methods. *Springer Series on Applications of Mathematics*, 27, Springer.

Overview of the INEX 2012 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, s.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 2012 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, details of the assessment and evaluation, and initial participation.

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 goal of the INEX Snippet Retrieval track 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.

This year is the second year in which the INEX Snippet Retrieval track has run. In response to feedback from the first year, search topics have been made more specific, and document-based assessment has been introduced.

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 view the document itself.

Each run must return 20 documents per topic, with a maximum of 180 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 by Schenkel et al. [1]. This corpus contains 2,666,190 documents.

This year there are 35 topics in total. The majority of these topics (25 of 35) have been created specifically for this track, with the goal being to create topics requesting more specific information than is likely to be found in the first few paragraphs of a document. The remaining 10 topics have been reused from the INEX 2010 Ad Hoc Track [2].

Each topic contains a short content only (CO) 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.

For those participants who wished to generate snippets only, and not use their own search engine, a reference run was generated.

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 will be used. In response to feedback from the previous year, both snippet-based and document-based assessment will be used. The documents will first be 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. The documents will then be assessed for relevance based on the full document text, with evaluation based on comparing these two sets of assessments.

Each topic within a submission will be assigned an assessor. The assessor, after reading the details of the topic, read through the 20 returned snippets, and judge which of the underlying documents seem relevant based on the snippets. The assessor will then be presented the full text of each document, and determine whether or not the document was actually relevant.

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 will be shuffled in such a way that each assessment package contains 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 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_1score.

$$\text{PA} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \tag{2}$$

Likewise, negative agreement (NA) is the conditional probability that, given one of the assessors judges a document as irrelevant, 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 Round 1 of the Snippet Retrieval Track

ID	Institute
20	Queensland University of Technology
46	Jadavpur University
65	University of Minnesota Duluth

Participation in the track has been split into two rounds, the first of which has had a compressed schedule. As of this writing, submissions for round 1 have closed, with submissions received from three participating organisations.

4 Conclusion

This paper gave an overview of the INEX 2012 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, the assessment method and evaluation metrics, as well as initial participation in the track.

References

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

2. Arvola, P, Geva, S., Kamps, J., Schenkel, R., Trotman, A., Vainio, J: Overview of the INEX 2010 ad hoc track. In: Geva, S., Kamps, J., Trotman, A. (eds.) Comparative Evaluation of Focused Retrieval. LNCS, pp. 1–32. Springer Berlin / Heidelberg (2011)

The 2012 INEX Snippet and Tweet Contextualization Tasks

Carolyn J. Crouch, Donald B. Crouch, Sai Chittilla,
Supraja Nagalla, Sameer Kulkarni, Swapnil Nawale

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

Abstract. This paper reports on our current experiments involving the Snippet and Tweet Contextualization Tracks of the 2012 INEX competition. Most of this work in snippet generation extends our earlier (2011) approach, described in [4], which produced a top-ranked result. The source of the snippet in these experiments is the top-ranked focused element(s) of the document in question. Another approach is based on using the document itself as the source of the snippet. Having identified the source, the snippet is then generated based on simple basic methodologies described herein. We also describe our experiments in tweet contextualization, a new track for INEX in 2012.

1 Introduction

In both 2009 and 2010, major tracks in the INEX competition centered upon the retrieval of what were referred to as *focused* elements. A focused element is by definition non-overlapping. We were able, as described in [1, 2, 5], to produce a methodology the results of which would rank in the top 10 for all the focused tasks of 2009 and 2010. This approach, described in detail in [4], is recapped briefly below.

To produce good (i.e., highly ranked) focused elements in response to a query, we first perform a document retrieval to identify the articles of interest. Our system is based on the Vector Space Model [7]; basic retrieval functions are performed by Smart [6]. To produce the set of elements corresponding to each article, we use an approach which we call flexible or dynamic element retrieval—Flex for short. (See [3] for details.) Flex allows us to produce the document tree, bottom up, at run time, based on a schema representing the structure of the document, generated during parsing, and a terminal node index of the collection. *Lnu-ltu* term weighting [8] is utilized with inner product to produce a rank-ordered list of all the elements of the document with respect to the query.

These elements are overlapping, so we now apply a focusing strategy to produce the non-overlapping elements of with the document tree. In the 2012 experiments, we use two focusing strategies, namely, the correlation strategy, which chooses the high-

est correlating element along a path as the focused element (without restriction as to element type) and the child strategy, which chooses the terminal element along a path as the focused element (regardless of correlation). The result is a rank-ordered list of focused elements associated with each document.

If we were to select one element that best represents the content of a document with respect to the query, one might do worse than to consider the highest ranking focused element(s) of that document. That element may not prove to be the best choice—it is certainly only one of many choices—but it appeared to us to be a viable source for snippet generation. Our snippet results in 2011 were based on this premise; one result placed in the top 10 of the official rankings despite our failure to produce a clean, easily readable version in each case.

2 INEX 2012: Snippet Generation

The snippet generation algorithms used in our 2012 experiments are similar in basic strategy, varying only in terms of the source of the snippet (focused elements or article), focusing strategy (correlation or child), and ranking algorithm (the first based on a simple function of the number of query terms in the snippet and the second a BLEU approach based on the number of query vs snippet n-grams, as applied to the sentences extracted). One experiment uses the text directly from the article as the snippet. We are currently awaiting INEX evaluation so as to enable assessment of the snippets.

3 INEX 2012: Tweet Generation

Our tweet conceptualization experiments use Indri to retrieve a small set of documents for each query. The corresponding sentences are ranked with respect to their similarity to the query based on several simple approaches, including word n-grams. A 500 word summary is then constructed using the top-ranked sentences in rank order. Two of these runs, evaluated earlier this year, rank 1 and 2 out of 33 in the official ranking with respect to average scores. These early results appear informative but would clearly benefit from increased readability. Future work is directed at this task and at related issues of interest that have not yet been addressed.

References

1. Acquilla, N.: Improving results for the INEX 2009 and 2010 Focused tasks. M.S. Thesis, Department of Computer Science, University of Minnesota Duluth (2011). http://www.d.umn.edu/cs/thesis/acquilla.pdf
2. Banhatti, R.: Improving results for the INEX 2009 Thorough and 2010 Efficiency tasks. M.S. Thesis, Department of Computer Science, University of Minnesota Duluth (2011). http://www.d.umn.edu/cs/thesis/banhatti.pdf
3. Crouch, C.: Dynamic element retrieval in a structured environment. ACM TOIS 24(4), 437-454 (2006).

4. Crouch, C., Crouch D., Acquilla, N., Banhatta, R., Chittilla, S., Nagalla, N., Navenvarapu, R: Focused elements and snippets. In: Geva, et al. (eds). Focused Retrieval of Content and Structure, LNCS 7424, Springer, (2012). [to appear]
5. Narendravarapu, R.: Improving results for the INEX 2009 and 2010 Relevant-in-Context tasks. M.S. Thesis, Department of Computer Science, University of Minnesota Duluth (2011). http://www.d.umn.edu/cs/thesis/ narendravarapu.pdf
6. Salton, G., (ed.): The Smart System—Experiments in Automatic Document Processing. Prentice-Hall (1971).
7. Salton, G., Wong, A., Yang, C.: A vector space model for automatic indexing. Comm. ACM 18(11), 613-620 (1975).
8. Singhal, A., Buckley, C., Mitra, M.: Pivoted document length normalization. Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 21-29, Zurich, Switzerland (1996).

Overview of the INEX 2012 Social Book Search Track

Marijn Koolen[1], Gabriella Kazai[2], Jaap Kamps[1], Michael Preminger[3], Antoine Doucet[4], and Monica Landoni[5]

[1] University of Amsterdam, Netherlands
{marijn.koolen,kamps}@uva.nl
[2] Microsoft Research, United Kingdom
v-gabkaz@microsoft.com
[3] Oslo and Akershus University College of Applied Sciences, Norway
michaelp@hioa.no
[4] University of Caen, France
doucet@info.unicaen.fr
[5] University of Lugano
monica.landoni@unisi.ch

Abstract. The goal of the INEX 2012 Social Book Search Track is to evaluate approaches for supporting users in reading, searching, and navigating book metadata and full texts of digitised books as well as associated user-generated content. The investigation is focused around two tasks: 1) the Social Book Search task investigates the complex nature relevance in book search and the role of user information and traditional and user-generated book metadata for retrieval, 2) the Prove It task evaluates focused retrieval approaches for searching pages in books that support or refute a given factual claim. There are two additional tasks that did not run this year. The Structure Extraction task tests automatic techniques for deriving structure from OCR and layout information, and 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 two search tasks.

1 Introduction

Prompted by the availability of large collections of digitised books, e.g., the Million Book project[6] and the Google Books Library project,[7] the Social Book Search Track[8] 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 digitised books. Toward this goal, the track provides opportunities to explore research questions around five areas:

[6] http://www.ulib.org/
[7] http://books.google.com/
[8] Previously known as the Book Track (2007–2010) and the Books and Social Search Track (2011).

– Evaluation methodologies for book search tasks that combine aspects of retrieval and recommendation,
– Information retrieval techniques for dealing with professional and user-generated metadata,
– Information retrieval techniques for searching collections of digitised books,
– Mechanisms to increase accessibility to the contents of digitised books, and
– Users' interactions with eBooks and collections of digitised books.

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

1. The *Social Book Search* (SBS) task, framed within the scenario of a user searching a large online book catalogue for a given topic of interest, aims at exploring techniques to deal with both complex information needs of searchers—which go beyond topical relevance and can include aspects such as genre, recency, engagement, interestingness, quality and how well-written it is—and complex information sources including user profiles and personal catalogues, and book descriptions containing both professional metadata and user-generated content.
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 the two search tasks, SBS and PI, at INEX 2012. First, in Section 2, we give a brief summary of the participating organisations. The SBS task is described in detail in Section 3, and the PI task in Section 4. We close in Section 5 with a summary and plans for INEX 2013.

2 Participating Organisations

A total of 55 organisations registered for the track (compared with 47 in 2011, 82 in 2010, 84 in 2009, 54 in 2008, and 27 in 2007). At the time of writing, we counted 5 active groups (compared with 10 in 2011 and 2010, 16 in 2009, 15 in 2008, and 9 in 2007), see Table 1.[9]

3 The Social Book Search Task

The goal of the Social Book Search (SBS) task is to evaluate the value of professional metadata and user-generated content for book search on the web. Through social media have extended book descriptions far beyond what is traditionally

[9] SE is biennial and will occur again in 2013.

Table 1. Active participants of the INEX 2012 Social Book Search Track, the task they were active in, and number of contributed runs (SBS = Social Book Search, PI = Prove It , SE = Structure Extraction, ART = Active Reading Task)

ID	Institute	Tasks	Runs
4	University of Amsterdam	SBS, PI	2 SBS, 5 PI
5	University of Michigan	PI	6 PI
54	Royal School of Library and Information Science	SBS	6 SBS
62	LIA, University of Avignon	SBS	5 SBS
100	Oslo and Akershus University College of Applied Sciences	SB, PI	4 SBS, – PI

stored in professional catalogues. Not only are books described in the users' own vocabulary, but are also reviewed and discussed online, and added to personal catalogues of individual readers. This additional information is subjective and personal, and allows users to search for books in different ways. Traditional descriptions have formal and subject access points for identification, known-item search and subject search. Yet readers use many more aspects of books to help them decide which book to read next [5], such as how engaging, fun, educational or well-written a book is. This results in a search task that requires a different model than traditional ad hoc search [3].

The SBS task investigates book requests and suggestions from the LibraryThing discussion forums as a way to model book search in a social environment. The discussions in these forums show that readers frequently turn to others to get recommendations and tap into the collective knowledge of a group of readers interested in the same topics.

As a source book descriptions, the INEX Amazon/LibraryThing collection [1] is used, which contains 2.8 million book descriptions from Amazon, enriched with content from LibraryThing. This collection contains both professional metadata and user-generated content. An additional goal of the SBS 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.

The SBS task aims to address the following research questions:

- Can we build reliable and reusable test collections for social book search based on book requests and suggestions from the LibraryThing discussion forums?
- Can we simulate book suggestions with judgements from Mechanical Turk?
- Can user-dependent evidence improve retrieval performance for social book search.
- Can personal, affective aspects of book search relevance be captured by systems that incorporate user-generated content and user profiles?

– What is the relative value of social and controlled book metadata for book search?

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.

3.2 Task description

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.

The SBS 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 identify 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.

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 assigned to books. To

make sure there is enough high-quality metadata from traditional library catalogues, we extended the data set with library catalogue records from the Library of Congress and the British Library. We only use library records of ISBNs that are already in the collection. 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).[10] Both the LoC records and the BL records are in MARCXML[11] format. We obtained MARCXML records for 1.76 million books in 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.

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.

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

[10] For more information see: http://www.loc.gov/aba/cataloging/subject/

[11] MARCXML is an XML version of the well-known MARC format. See: http://www.loc.gov/standards/marcxml/

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	

of information needs for a large collection of online book records. We use a selection of these forum topics to evaluate systems participating in the SBS task.

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*. The books suggested by members in the thread 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. In the rest of this paper, we use the term *suggestion* for books identified in the Touchstone lists in forum topics. Since all suggestions are made by forum members, we assume they are valuable judgements for the relevance of books. We first describe the topic selection procedure and then how we used LibraryThing user profiles to assign relevance values to the suggestions.

Topic selection Topic selection was done the same as last year **(author?)** [4]. We crawled close to 60,000 topic threads and selected threads where at least one book is suggested and the first message contains a book request. First, we identified topics where the topic title reflects the information need expressed in it. For this we used the topic titles as queries and ran them against a full-text index of the A/LT collection. We consider a title to be a decent reflection of

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

the information need if the full-text index found it least one suggestion in the top 1000 results. This left 6510 topics. Next, we used short regular expressions to select messages containing any of a list of phrases like *looking for*, *suggest*, *recommend*. From this set we randomly selected topics and manually selected those topics where the initial message contains an actual request for book recommendations, until we had 89 new topics. We also labeled each selected topic with topic type (requests for books related to *subject*, *author*, *genre*, *edition* etc.) and genre information (*Fiction*, *Non-fiction* or both).

We included the 211 topics from the 2011 Social Search for Best Books task and adjusted them to the simpler topic format of this year. All genre labels were changed to either *Fiction* or *Non-fiction*. The label *Literature* was changed to *Fiction* and all other labels were changed to *Non-fiction*. Specificity labels and examples were removed.

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. Parekh's Rethinking Multiculturalism:
    Cultural Diversity and Political Theory (which I just finished) in the
    end left me unconvinced, 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 skepticism, but still... Anyway, I've read Sen, Rawls,
```

```
    Habermas, and Nussbaum, 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>non-fiction</genre>
</topic>
```

We think this set represents a broad range of book information needs. 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. Like last year, 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 a crowdsourcing task where participants have to search the Amazon/LibraryThing collection for relevant books based on the topic narrative, and we pool the queries they type, and provide the most common query to INEX participants.

User profiles and personal catalogues We can distinguish different relevance signals in these suggestions if we compare them against the books that the topic creator added to her personal catalogue before (pre-catalogued) or after (post-catalogued) starting the topic. We obtained user profiles for each of the topic creators of the topics selected for evaluation and distributed these to the participants. Each profile contains a list of all the books a user has in her personal catalogue, with per book the date on which it was added, and the tags the user assigned to the book. The profiles were crawled at least 4 months after the topic threads were crawled. We assume that within this time frame all topic creators had enough time to decide which suggestions to catalogue.

Catalogued suggestions The list of books suggested for a topic can be split into three subsets. The subset of books that the topic creator had already catalogued before starting the topic (Pre-catalogued suggestions, or Pre-CSs), the subset of books that the topic creator catalogued after starting the topic (Post-catalogued suggestions or Post-CSs) and the subset that the topic creator had not catalogued at the time of crawling the profiles (Non-catalogued suggestions, or Non-CSs).

Members sometimes suggest books that the topic creator already has in her catalogue. In this case, the suggestion is less valuable for the topic creator, but still a sign that for the topic creator that the suggestion makes sense. Similarly, if a topic creator does not catalogue a suggestion, before or after creating the topic, we consider this a signal that the topic creator found the suggestion not valuable enough. In both cases, the suggestion is still a valuable relevance judgement in itself that goes beyond mere topical relevance [3]. In contrast, when the topic creator adds a suggestion to her catalogue after starting the topic (topic creation

is the first signal that she has that particular information need), we assume the suggestion is of great value to the topic creator.

Self-supplied suggestions Some of the books in the Touchstone list are suggestions by the topic creator herself. One reason for these suggestions could be that the creator wants to let others know which books she already knows or has read. Another reason could be that she discovered these books but considered them not good enough for whatever reason. A third reason could be that she discovered these books and wants the opinions of others to help her decide whether it is good enough or not. Because it is hard to identify the reason for a self-supplied suggestion, we consider these suggestions as not relevant, except for the self-supplied suggestions the topic creator later added to her personal catalogue. In this case, the post-cataloguing action is a signal that creator eventually considered it good enough.

Touchstone Suggestions as Judgements This year we used a topic set of 300 topics, including the 211 topics from last year and the 89 new topics. We also provided user profiles of the topic creators as context for generating recommendations. These profiles contain information on which books the user has catalogued and on which the date.

Because we want to focus on suggestions that the topic creator is most interested in, we filtered the 300 topics and retained only those topics where the creator added at least one of the suggested books to her personal catalogue on or after the date she created the forum topic. This resulted in a subset of 96 topics, which is used for evaluation (Section 3.7). The next section describes our method for generating relevance judgements.

3.6 From Suggestions to Relevance Judgements

A system presenting a user with book suggested on the forum fulfils the library objective of helping users to find or locate relevant items, whereas a system presenting the user with books she will add to her catalogue, we argue that it fulfils the library objective of helping her *choose* which of the relevant items to obtain [6]. Based on this correspondence to library cataloguing objective, we assign higher relevance values to books that are post-catalogued than to other suggestions.

We use the following terminology:

Creator The topic creator, who has the information need and formulated the request.

Suggestion Suggestions are books mentioned in the messages of the topic thread, and that are identified via Touchstone.

Suggestor The forum member who first suggests a book. The thread is parsed from first to last message, and the first member to mention the book is considered the suggestor. Note that this can be the topic creator. Perhaps

she suggests books because she wants others to comment on them or because she wants to show she already knows about these books.

Pre-catalogued Suggestion a suggestion that the creator catalogues before starting the topic.

Post-catalogued Suggestion a suggestion that the creator catalogues after having started the topic.

Non-catalogued Suggestion a suggestion that the creator did not catalogue before or after having started the topic.

To operationalise the suggestions as relevance judgements, we use different relevance values (rv):

Highly relevant (rv=4) Post-catalogued Suggestions are considered the best suggestions, regardless of who the suggestor is.

Relevant (rv=1) Pre- and Non-catalogued suggestions where the suggestor is not the creator. Suggestions from others that the creator already has are good suggestions in general (perhaps not useful for the creator, but still relevant to the request).

Non-relevant (rv=0) Pre- and Non-catalogued suggestions that the creator suggested herself, i.e., the suggestor is the creator. These are either books the creator already has (pre-catalogued) or may be negative examples (*I'm not looking for books like this*), or are mentioned for some other reason. The creator already knows about these books.

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 Full set of 300 topics and the PCS subset of 96 topics with post-catalogued suggestions are given in Table 3.

We first compare the suggestions for the full topic set with those of 2011. The two sets of suggestions are similar in terms of minimum, median and mean number of suggestions per topic. The maximum has increased somewhat this year. Split over genres we see that the Fiction topics tend to get more suggestions than Non-Fiction topics. Topics where creators explicitly mention both fiction and non-fiction recommendation are welcome—denoted Mix—are more similar to Non-Fiction topics in terms of maximum and median number of suggestions, but closer to Fiction topics in terms of mean number of suggestions.

If we zoom in on the PCS topics, we see they have a larger number of suggestions per topic than the Full set, with a mean (median) of 16.2 (9). Most of the suggestions are not catalogued by the topic creator and made by others (RV_1). In most topics there is at least one book first mentioned by the topic creator (RV_0 has a median of 1), and only a small number suggestions are post-catalogued by the creator (RV_4 has a mean of 1.7 and median of 1). What does it mean that the PCS topics get more suggestions than the other topics in the Full set? One

Table 3. Statistics on the number of recommended books for the 101 topics from the LT discussion groups

RV	# topics	# suggest.	min.	max.	mdn.	mean	std. dev.
Full							
2011	211	2377	1	79	7	11.3	12.5
2012	300	3533	1	101	7	11.8	14.5
Fiction	135	2143	1	101	9	15.9	18.0
Non-fiction	146	1098	1	56	5	7.5	7.8
Mix	19	292	1	59	7	15.4	16.2
PCS							
RV_{0+1+4}	96	1558	1	98	9	16.2	17.7
RV_0	96	194	0	21	1	2.0	3.8
RV_1	96	1200	0	80	7	12.5	16.1
RV_4	96	164	1	14	1	1.7	1.6

Table 4. Statistics on the number of topics per genre for the full set of 300 topics and the 96 topics with PCSs

	Topic set	
Genre	Full	PCS
All	300	96
Fiction	135 (45%)	49 (51%)
Non-fiction	146 (49%)	36 (38%)
Mix	19 (6%)	9 (9%)
Subject	207 (69%)	60 (63%)
Author	36 (12%)	16 (17%)
Genre	64 (21%)	21 (22%)
Known-item	15 (5%)	5 (5%)

reason might be that with more suggestions, there is a larger a priori probability that the topic creator will catalogue at least one of them. Another, related, reason is that a larger number of suggestions means the list of relevant books is more complete, which could make the topic creator more confident that she can make an informed choice. Yet another reason may be that PCS topics are dominated by Fiction topics, which have more suggestions than the Non-Fiction topics.

In Table 4 we show the number of Fiction, Non-Fiction and Mix topics in the Full and PCS topics sets. In the Full set, there are a few more Non-Fiction topics (146, or 49%) than Fiction topics (135 or 45%), with only 19 (6%) Mix topics. In the PCS set, this is the other way around, with 49 Fiction topics (51%), 36 Non-fiction topics (38%) and 9 Mix topics (9%). This partly explains why the PCS topics have more suggestions. Post-cataloguing tends to happen more often in topic threads related to fiction.

Judgements from Mechanical Turk To get a better understanding of the nature of book suggestions and book selection, we plan to gather rich relevance judgements from Mechanical Turk that cover different aspects of relevance. Workers will judge the relevance of books based on the book descriptions in the collection and the topic statement from the LT forum. Instead of asking them to judge the overall relevance of books, we plan to ask them to identify different relevance aspects of the information need and to judge the books on each of these aspects separately. Additionally, we ask them to identify which part of the description (title, subject headings, reviews or tags) is useful to determine the relevance of the book for each relevance aspect in the request. Of course, workers are not able to judge books on the user-dependent (personal, affective relevance aspects) of the topic creator. For these aspect we would need judgements from the topic creator herself. One possibility is to approach topic creator on the forums or via private messages to they LT profile.

We are currently in the process of setting up the Mechanical Turk experiment and hope to have results for the final report in the official proceedings.

ISBNs and intellectual works Each record in the collection corresponds to an ISBN, and each ISBN corresponds to a particular intellectual work. 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 of the same work 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 ignore these distinctions to make evaluation easier. This turns edition-related topics into known-item topics.

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.[12]

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 any time, some ISBNs map to multiple work IDs even though they represent the same intellectual work. LibraryThing can detect such cases but, to avoid making mistakes, leaves it to members to merge them. The

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

Table 5. Evaluation results for the official submissions. Best scores are in bold

Run	MRR	nDCG@10	P@10	R@10	R@1000
p54.run2.all-topic-fields.all-doc-fields	0.3069	**0.1492**	0.1198	0.1527	0.5736
p54.run3.all-topic-fields.QIT.alpha0.99	0.3066	0.1488	0.1198	0.1527	0.5736
p4.inex2012SBS.xml_social.fb.10.50	**0.3616**	0.1437	0.1219	0.1494	**0.5775**
p62.B_IT30_30	0.3410	0.1339	**0.1260**	**0.1659**	0.5130
p4.inex2012SBS.xml_social	0.3256	0.1297	0.1135	0.1476	0.5588
p62.mrf-booklike	0.3584	0.1295	0.1250	0.1514	0.5242
p54.run5.title.II.alpha0.94	0.2558	0.1173	0.1073	0.1289	0.4891
p62.IOT30	0.2933	0.1141	0.1240	0.1503	0.5864
p62.IT30	0.2999	0.1082	0.1187	0.1426	0.5864
p54.run6.title.II.alpha0.97	0.2392	0.0958	0.0823	0.0941	0.4891
p62.lcm-2	0.2149	0.0901	0.0667	0.1026	0.5054
p100.sb_g0	0.2394	0.0884	0.0844	0.1145	0.5524
p54.run4.title.QIT.alpha0.65	0.1762	0.0875	0.0719	0.0949	0.4891
p100.sb_g_ttl_nar0	0.1581	0.0740	0.0594	0.0939	0.4634
p54.run1.title.all-doc-fields	0.1341	0.0678	0.0583	0.0729	0.4891
p100.sb_2xsh_ttl_nar0	0.0157	0.0057	0.0021	0.0022	0.0393
p100.sb_2xsh0	0.0199	0.0042	0.0021	0.0020	0.0647

fraction of works with multiple ISBNs is small so we expect this problem to have a negligible impact on evaluation.

3.7 Evaluation

This year four teams together submitted 17 runs. The Oslo and Akershus University College of Applied Sciences (OAUCAS) submitted 4 runs, the Royal School of Library and Information Science (RSLIS) submitted 6 runs, the University of Amsterdam (UAms) submitted 2 runs and the LIA group of the University of Avignon (LIA) submitted 5 runs.

The official evaluation measure for this task is nDCG@10. It takes graded relevance values into account and concentrates on the top retrieved results. The set of PCS topics and corresponding suggestions form the official topics and relevance judgements for this year's evaluation. The results are shown in Table 5.

The best performing run is *p54.run2.all-topic-fields.all-doc-fields* by **RSLIS**, which used all topic fields combined against an index containing all available document fields.

The best run by **UAms** is *p4.inex2012SBS.xml_social.fb.10.50*, which uses only the topic titles and ran against an index containing the title information fields (title, author, edition, publisher, year) and the user-generated content fields (tags, reviews and awards). Blind relevance feedback was applied using the top 50 terms from the top 10 initial retrieval results.

The best run by **LIA** is *p62.B_IT30_30*.

The best run by **OAUCAS** is *p100.sb_g0*.

We note that the best run does not use any information from the user profiles. The best performing run that incorporates user profile information is the second best run, *p54.run3.all-topic-fields.QIT.alpha0.99* by RSLIS. Like the best performing run, it uses all topic fields against all document fields, but re-ranks the results list based on the LT profile of the topic creator. Retrieved books that share a lot of tags associated books already present in the user's catalog are regarded as a more appropriate match. The final retrieval score is a linear combination of the original content-based score and the cosine similarity between a tag vector containing the tag counts from a user's personal catalog and the tag vectors of the retrieved books.

The run *p4.inex2012SBS.xml_social.fb.10.50* achieves the highest MRR score (0.3616), which means that on average, it retrieves the first relevant book at or above rank 3. The nine best systems achieve a P@10 score just above 0.1, which means on average they have one suggestion in the top 10 results. Most systems are able to retrieve an average of around 50% of the suggestions in the top 1000 results.

Note that the three highest scores for P@10 (0.1260, 0.1250 and 0.1240) correspond with the 4th, 6th and 8th highest scores for nDCG@10. The highest nDCG@10 score corresponds to the 5th highest P@10 score. This could mean that top performing system is not better than the other systems at retrieving suggestions in general, but that it is better at retrieving PCSs, which are the most important suggestions. The top two runs have similar nDCG@10 scores and the same P@10 scores and retrieve more PCSs in the top 10 (36 over all 96 topics) than the other runs, the best of which retrieves only 26 PCSs in the top 10, over all 96 topics. The full topic statement is a more effective description of the books that the topic creator will catalogue than the topic title alone.

In sum, systems that incorporate user profile information have so far not been able to improve upon a plain text retrieval baseline. The best systems for retrieving PCSs use the full topic statement.

Recall that the evaluation is done on a subset of the Full set of 300 topics. In Section 3.5 we found that the PCS topics have more suggestions per topic than the rest of the topics in the Full set, and that the fraction of Fiction topics is also higher in the PCS set. To what extent does this difference in genre and number of suggestions result in differences in evaluation?

We compare the system rankings of the official relevance judgements (PCS topics with differentiated relevance value, denoted PCS(RV_{0+1+4}) with two alternative sets. One based on the same topics but with all suggestions mapped to relevance value $rv = 1$ (denoted PCS(RV_{flat}) and the other is the set of judgements for the Full set of 300 topics, where all suggestions were also all mapped to relevance value $rv = 1$, denoted Full(RV_{flat}. The PCS(RV_{flat} set allows us to see whether the differentiation between suggestions affects the ranking. The comparison between PCS(RV_{flat} and Full(RV_{flat} can show whether the different topic selection criteria lead to different system rankings.

Table 6 shows the Kendall's Tau (column 2) and Tau$_{AP}$ (column 3) ranking correlations over the 18 official submissions for nDCG@10. The Tau$_{AP}$ ranking

Table 6. Kendall's Tau and Tau$_{AP}$ system ranking correlations between the relevance judgements of Full and PCS topics. PCS(RV_{flat} represents judgements where all suggestions have $RV = 1$, PCS(RV_{0+1+4} represents the suggestion with differentiated relevance values.

Qrels	τ	τ_{AP}
Full(RV_{flat} / PCS(RV_{flat}	0.91	0.79
Full(RV_{flat} / PCS(RV_{0+1+4}	0.85	0.73
PCS(RV_{flat} / PCS(RV_{0+1+4}	0.91	0.93

correlation puts more weight on the top-ranked systems [7], emphasising how well the evaluations agree on ranking the best systems. The standard Kendall Tau correlation is very strong (> 0.9) between Full(RV_{flat} and PCS(RV_{flat}, suggesting the topic selection plays little role. The correlation between PCS(RV_{flat} and PCS(RV_{0+1+4} is also very high, furthermore suggesting that the differentiation between suggestions has no impact on the ranking. However, the Tau$_{AP}$ correlations show that disagreement between Full(RV_{flat} and PCS(RV_{flat} is bigger among the top ranked system than the on the lower scoring systems. The two PCS sets have very strongly correlated system rankings. From this we conclude that the differentiation between suggestions in terms of relevance value has little impact, but that the PCS topics are somewhat different in nature than the other topics.

4 The Prove It (PI) Task

The goal of this task was to investigate the application of focused retrieval approaches to a collection of digitised 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[13], digitised by Microsoft. The corpus is made up of books of different genre, including history books, biographies, literary studies, religious texts and teachings, reference

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

works, encyclopaedias, 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">
     <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

In recent years we have had a topic-base of 83 topics, 21 of which we have collected relevance judgments for using crowdsourcing through the Amazon Mechanical Turk infrastructure [2].

The ambition this year has been two-fold:

– To increase the number of topics
– To further develop the relevance judgment method, so as to combat the effect of the statement complexity on the assessment consistency.

For the second point above, we have been attempting to divide each topics into its primitive aspects (a process we refer to as "aspectization"). To this end we developed a simple web-application with a database back-end, to allow anyone to aspectize topics. This resulted in 30 topics

For each page being assessed for confirmation / refutation of a topic, the assessor is presented with a user interface similar to Figure 2

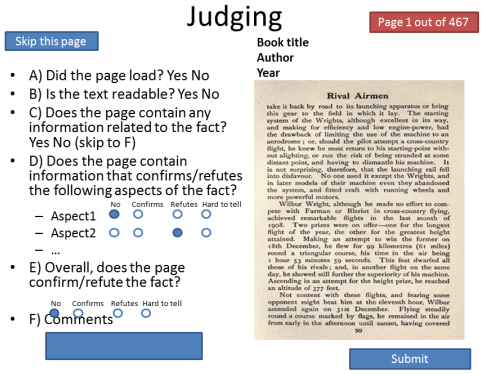


Fig. 2. An illustration of the planned assessment interface

This means that we go from a discrete (confirms / refute / none) assessment to a graded assessment, where a page may e.g. be assessed by a certain as 33 percent confirming a topic, if one of three aspects is judged as confirmed by him/her for that page.

For the current assessment we have prepared 30 topics, for which the number of aspects range from 1 (very simple statements) to 6 per topic with an average of 2,83 aspects per topic.

4.3 Collected Relevance Assessments

At the time of writing this years relevance assessment are still not collected yet.

4.4 Evaluation Measures and Results

Result publication is awaiting the conclusion of the relevance assessment process.

5 Conclusions and plans

This paper presents an overview of the INEX 2012 Social Book Search Track. This year, the track ran two tasks: the Social Book Search task, and the Prove It task.

The Social Book Search (SBS) task changed focus from the relative value of professional and user-generated metadata, to the complexity of book search information needs.

We extended our investigation into the nature of book requests and suggestions from the LibraryThing forums as statements of information needs and relevance judgements. By differentiating between who the suggestor is and whether the topic creator subsequently adds a suggestion to her catalogue or not (post catalogued suggestions), we want to focus even more on the personal, affective aspects of relevance judgement in social book search. We operationalised this by differentiating in relevance values, giving higher values for post-catalogued suggestions than for other suggestions.

Our choice to focus on topics with post-catalogued suggestions (PCS topics) resulted in a topic set that is slightly different from the topics we used last year, where we ignored the personal catalogued of the topic creator and considered all topics that have a book request, a descriptive title and at least one suggestion. The PCS topics have more suggestions on average than other topics, and a larger fraction of them is focused on fiction books. This results in a difference in system ranking, which is mainly due to the different nature of the topics, and not in the differentiation of the relevance values.

In addition to the topic statements extracted from the forum discussions, we extracted user profiles of the topic creators, which contain full catalogue information on which books they have in the personal catalogues, when each book was added to the catalogue and which tags the user assigned to each book. These profiles were distributed along with the topic statements, to allow participants to build systems that incorporate both the topical description of the information need and personal behaviour, preferences and interests of the topic creators.

The evaluation has shown that the most effective systems incorporate the full topic statement, which includes the title of the topic thread, the name of the discussion group, and the full first message that elaborates on the request. However, the best system did not use any user profile information. So far, the best system is a plain full-text retrieval system.

Next year, we continue with the task to further investigate the role of user information. We also plan to enrich the relevance judgements with further judgements on the relevance of books to specific relevance aspects of the information need. For this, we plan to use either Mechanical Turk or approach the topic creators on LibraryThing to obtain more specific judgements directly from the person with the actual information need.

This year the Prove It task has undergone some changes when it comes to assessments. The number of participants for the PI task is still low, which also puts some limitations on what we are able to do collaboratively, but based

on the changes introduced this year which will hopefully give us more useful assessments, we hope to increase the number of participants, further vitalizing the task.

Acknowledgments We are very grateful to Justin van Wees for providing us with the user profiles of the topic creators for this year's evaluation. This research was supported by the Netherlands Organization for Scientific Research (NWO projects # 612.066.513, 639.072.601, and 640.005.001) and by the European Communitys Seventh Framework Program (FP7 2007/2013, Grant Agreement 270404).

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] 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.

[3] Marijn Koolen, Jaap Kamps, and Gabriella Kazai. Social Book Search: The Impact of Professional and User-Generated Content on Book Suggestions. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM 2012)*. ACM, 2012.

[4] Marijn Koolen, Gabriella Kazai, Jaap Kamps, Antoine Doucet, and Monica Landoni. Overview of the INEX 2011 books and social search track. In Shlomo Geva, Jaap Kamps, and Ralf Schenkel, editors, *Focused Retrieval of Content and Structure: 10th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2011)*, volume 7424 of *LNCS*. Springer, 2012.

[5] Kara Reuter. Assessing aesthetic relevance: Children's book selection in a digital library. *JASIST*, 58(12):1745–1763, 2007.

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

[7] Emine Yilmaz, Javed A. Aslam, and Stephen Robertson. A new rank correlation coefficient for information retrieval. In Sung-Hyon Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat-Seng Chua, and Mun-Kew Leong, editors, *SIGIR*, pages 587–594. ACM, 2008.

RSLIS at INEX 2012: Social Book Search Track

Toine Bogers and Birger Larsen

Royal School of Library and Information Science
Birketinget 6, 2300 Copenhagen, Denmark
{tb,blar}@iva.dk

Abstract. In this paper, we describe our participation in the INEX 2012 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 different social features, such as user ratings, tags, and authorship information. We find that the best results are obtained using all available document fields and topic representations. Re-ranking retrieval results works better on shorter topic representations, where there is less information for the retrieval algorithm to work with; longer topic representations do not benefit from our social re-ranking approaches.

Keywords: XML retrieval, social tagging, controlled metadata, book recommendation, re-ranking

1 Introduction

In this paper, we describe our participation in the INEX 2012 Social Book Search track[1]. Our goals for the Social Book Search task were (1) to investigate the contribution of additional controlled metadata provided for this year's task; and (2) to examine the effectiveness of using social features for re-ranking the initial content-based search results. We focus in particular on using techniques from collaborative filtering (CF) to improve our 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, including the effect of the additional controlled metadata sources. 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 2012 Social Book Search track. This collection contains XML

[1] https://inex.mmci.uni-saarland.de/tracks/books/

representations of 2.8 million books, with the book representation data crawled from both Amazon.com and LibraryThing (LT). The 2012 collection is identical to the collection provided for the 2011 track [1] in all but two ways: the collection has been expanded with additional library records from the British Library (BL) and the Library of Congress (LoC). Of the 2.8 million books in the collection, 1.15 million have a BL record and 1.25 have a LoC record. Together these two sources cover 1.82 million of the 2.8 million books in the collection.

We converted the collection's original XML schema into a simplified version to retain only those metadata fields that were most likely to contribute to the successful retrieval of relevant books[2]. After these pre-processing steps, we were left with the following 19 content-bearing XML fields in our collection: <isbn>, <title>, <publisher>, <editorial>, <creator>, <series>, <award>, <character>, <place>, <blurber>, <epigraph>, <firstwords>, <lastwords>, <quotation>, <dewey>, <subject>, <browseNode>, <review>, and <tag>.

We replaced the numeric Dewey codes in the original <dewey> fields by their proper textual descriptions using the 2003 list of Dewey category descriptions[3] 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>. The BL and LoC records were provided in MODS format[4], we mapped this format to the appropriate new XML fields and added them to the book representations.

2.2 Field categories and Indexing

The 19 selected 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. To examine the contribution of the additional BL/LoC controlled metadata we created two versions of the index containing controlled metadata: one with and one without this additional controlled metadata. In addition, we combined all five groups of relevant fields for an index containing all fields. This all-fields index also comes in two variants: one with and one without the BL/LoC metadata. This resulted in a total of eight 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). The all-doc-fields-plus index contains all of the original 2011 fields as well as the BL/LoC metadata.

[2] Please consult [2] for more details on this filtering and conversion process.
[3] Available at http://www.library.illinois.edu/ugl/about/dewey.html
[4] See http://www.loc.gov/standards/mods/ for more information.

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>.

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 harvested from Amazon: <browseNode>, <dewey>, and <subject>. The controlled-metadata-plus index contains the original metadata as well as the BL/LoC metadata.

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 All user reviews belonging to a single book were combined in a single document representation for that book and added to our review index reviews.

We used the Indri 5.1 retrieval toolkit[5] 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 2012 Social Book Search track three sets of topics were released with requests for book recommendations based on textual description of the user's information need: two training sets and a test set. All topic sets were extracted from the LibraryThing forum. The original training set of 43 topics created for the 2011 Social Book Search track came with unverified relevance judgments, so we only used the test set of 2011 as our training set for 2012. This second training set contains 211 topics with relevance judgments derived from the books recommended on the LibraryThing discussion threads of these 211 topics. We used this 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 for 2012 contains 90 additional topics which, combined with the 211 training set topics, were used to rank and compare the different participants' systems at INEX 2012. The results listed in Section 6 were obtained on this combined set of 301 topics. Each topic is represented by several different fields:

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.

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

Group The LibraryThing forum is divided into different groups covering different topics.

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 typically contains the richest description of the topic.

All topic fields We also performed runs with all three fields combined, referred to as all-topic-fields.

In our experiments with the training and the test set, we restricted ourselves to automatic runs using the following title and the all-topic-fields representations (based on our experiments for INEX 2011 [2]).

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 5.1 toolkit. We preferred JM smoothing over Dirichlet smoothing, because previous work has shown that for longer, more verbose queries JM smoothing outperforms Dirichlet smoothing [3], which matches the richer topic descriptions provided in the topic sets.

For the best possible performance, we optimized the λ parameter, which controls the influence of the collection language model, with higher values giving more influence to the collection language model. We varied λ in steps of 0.1, from 0.0 to 1.0 using the training set of topics. We also examined the value of stop word filtering and stemming and use the SMART stop word list and Krovetz stemming in these cases. This resulted in 44 different possible combinations of these three parameters. For each topic we retrieved up 1000 documents and we used NDCG@10 as our evaluation metric [4].

3 Content-based Retrieval

In order to produce a competitive baseline for our experiments with re-ranking based on social features, we conducted a first round of experiments focused on optimizing a standard content-based retrieval approach for each combination of index and topic representations. We found that the best results were always produced with stop word filtering and Krovetz stemming, so all results reported in this paper share these settings. We compared the different index and the different topic representations for a total of 16 different content-based retrieval runs. Table 1 shows the best NDCG@10 results for each run on the training set.

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) with an NDCG@10 score of 0.3058. Retrieving on the all-doc-fields index performs best on both topic sets (all-topic-fields and title). The reviews index is a close second with strong performance on both topic sets. When we compare the two topic sets, we see that the

Table 1. Results of the 16 different content-based retrieval runs on the training set using NDCG@10 as evaluation metric. Best-performing runs for each topic representation are printed in bold.

Document fields	Topic fields	
	title	all-topic-fields
metadata	0.0915	0.2015
content	0.0108	0.0115
controlled-metadata	0.0406	0.0496
controlled-metadata-plus	0.0514	0.0691
tags	0.0792	0.2056
reviews	0.1041	0.2832
all-doc-fields	**0.1129**	**0.3058**
all-doc-fields-plus	0.1120	0.3029

all-topic-fields set consistently outperforms the title topic set. These findings are all in line with our 2011 results [2].

Finally, we observe that the content and controlled-metadata indexes result in the worst retrieval performance across all four topic sets. Adding the extra BL/LoC controlled metadata has a positive effect on retrieving over only controlled metadata: the controlled-metadata-plus index outperforms the controlled-metadata on both topic sets. However, the adding this additional BL/LoC metadata to the index containing all document fields (all-doc-fields-plus) actually causes a small but surprising drop in performance. This suggests that for some topics the existing document fields better describe the documents than the information present in the BL/LoC fields.

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.

However, there are still many topics for which performance is sub-par, with many possible reasons for this performance gap. One explanation could be differences in document field sparsity, which could cause certain indexes to underperform for particular topics. The well-known vocabulary problem [5] could be another explanation, resulting in mismatches between synonymous query and document terms. Finally, content-based matches are no guarantee for high-quality recommendations, merely for on-topic recommendations.

To remedy these problems, we explore the use of social features for re-ranking the content-based search results in this section. We experiment with re-ranking

based on book similarities (Section 4.1) as well as a personalized re-ranking approach (Section 4.2).

4.1 Book similarity re-ranking

Similar books that are equally relevant to a user's request for recommendations might appear at wildly different positions in the results list due to differences in term usage between the documents and the topic description. The goal of our re-ranking approach is to push those relevant documents that did not score well under a content-based approach to a higher position in the ranked results list. To that end we propose calculating a new retrieval score for each book that is a linear combination of (1) the original retrieval score and (2) the combined contributions of all other documents in the results list, weighted by their similarity to the book in question. This means that each of the books j retrieved for a topic contributes a little bit to the final retrieval score of a specific book i, depending on the original retrieval score $score_{org}(j)$ of book j and its similarity $sim(i,j)$ to book i. More similar books and books retrieved at higher ranks contribute more to book i's new re-ranked score $score_{re-ranked}(i)$; others contribute less. Equation 1 shows how we calculate this score:

$$score_{re-ranked}(i) = \alpha \cdot score_{org}(i) + (1-\alpha) \cdot \sum_{j=1, i \neq j}^{n} score_{org}(j) \cdot sim(i,j) \qquad (1)$$

Before re-ranking we apply rank normalization on the retrieved results to map the score into the range $[0,1]$ [6]. The balance between the original retrieval score $score_{org}(i)$ and the contributions of the other books in the results list is controlled by the α parameter, which takes values in the range $[0,1]$. The actual book similarities $sim(i,j)$ can be calculated using different types of social features; we have explored five variants, which are described in more detail below.

User ratings As mentioned earlier, content-based matches are no guarantee for high-quality book recommendations; they merely indicate a strong term overlap between the topic description and the book descriptions. One way of dealing with this problem is to consider one of the social features in the collection that explicitly capture the quality of a book: user ratings. The reviews in the Amazon/LibraryThing collection contain the Amazon user names of the reviewers as well as their ratings on a five-star scale. We extract and use these ratings to calculate the similarities between the different books.

For each book in each of our results lists, we construct an vector of book ratings that contains all the ratings for that book from each reviewer in the Amazon/LibraryThing collection. Missing ratings—in case a reviewer did not review that particular book—receive a score of zero. We combine all item rating vectors in an **IU** ratings matrix where I is the number of books retrieved in all of our results lists combined and U is the number of reviewers in the collection. We normalize the **IU** ratings to compensate for individual differences in rating behavior [7].

Inspired by item-based collaborative filtering [8], we then calculate the cosine similarity between pairs of book vectors (i.e., row vectors). For re-ranking purposes we only need to calculate the book similarities for pairs of books that occur in the same results list. The resulting book similarities are then fed into our re-ranking approach (Eq. 1); we refer to this as IU-similarity.

Amazon's *"similar products"* The Amazon/LibraryThing collection already contains information about similar books: each book representation can contain up to ten <similarproduct> fields which contain the ISBN numbers of similar books, as seen on Amazon under the "*similar products*" section of a book Web page. We also explore the value of these book similarities in our re-ranking approaches, setting the similarity between two books $sim(i, j)$ to 1 if book j is mentioned in the representation of book i (and vice versa), and to 0 otherwise. We refer to this as II-similarity.

How do these "*similar products*" stack up against the ratings-based book similarities? This "*similar products*" data is likely to be a more accurate representation of book similarity based on user ratings as it is calculated over the entire set of user ratings, both with and without reviews [9]. In contrast , the ratings in our **IU** matrix only represent the ratings of a subset of reviews and not the ratings made by users with entering an actual review. However, the "*similar products*" similarities are binary even though the original similarities calculated by Amazon's algorithms were not. Moreover, the "*similar products*" data is likely to be incomplete. Amazon only shows a random selection of 10 similar books each time a book's Web page is generated. This means that the set of similar books during the original crawling of the Amazon/LibraryThing collection represents just a subset of all similarity pairs.

Tags Another source of information for calculating book similarities are the tags assigned to the different books. For this source of book similarities, we construct a **IT** matrix, analogous to our **IU** matrix. In the **IT** matrix, the columns represent the different tags assigned to all the books in our result lists. Each value in **IT** represents the number of times tag t has been assigned to book i. If a tag was not assigned to a book, that cell receives the value 0. The **IT** matrix is then row-normalized. We obtain the similarity between two books by calculating the cosine similarity between their two row vectors. We refer to this as IT-similarity.

Authors Author-book associations represent another way of calculating book similarities: books written by the same author(s) are often similar in style and content. To explore this type of similarity, we construct a **IA** matrix where the columns represent the authors associated with all the books in our result lists. Values in **IA** are binary, with a value of 1 if author a wrote book i, and a 0 otherwise. We obtain the similarity between two books by taking the cosine similarity between their vectors. We refer to this as IA-similarity.

Fusing ratings, tags and authors Instead of picking just one of the aforementioned sources of book similarity, we also experimented with using a combination of user

ratings, tags, and authorship for calculating the book similarities. To this end we construct a combined matrix **IUTA**, which consists of the **IU**, **IT**, and **IA** matrices combined so that each book vectors contains both user ratings, tags, and authorship information. The expectation here is that the different information sources can augment each other's performance. Again, we calculate the similarity between two books by calculating the cosine similarity between their two **IUTA** row vectors. We refer to this as IUTA-similarity.

4.2 Personalized re-ranking

In addition to the one-size-fits-all approach to re-ranking described in Section 4.1, we also explore a personalized re-ranking approach that takes into account the past preferences of the user who originally created the LT topic requesting book recommendations. The goal is to calculate a new personalized score $score_{personalized}(u,i)$ for a LibraryThing user u and a retrieved book i that pushes i up in the rankings if it is similar to other books read by u in the past. The new personalized score is a linear combination of the original retrieval score $score_{org}(i)$ for book i and the similarity between i and the other books in u's profile. Equation 2 shows how we calculate this personalized score:

$$score_{personalized}(u,i) = \alpha \cdot score_{org}(i) + (1-\alpha) \cdot sim_{tag}(u,i) \tag{2}$$

Again, we control the balance the original retrieval score $score_{org}(i)$ and the similarity with the user's past preferences with the α parameter, which takes values in the range $[0,1]$. There are different ways of calculating the similarity $sim(u,i)$ between a user's profile and a book i book similarities: user ratings, tags, authors, or even term overlap between different metadata fields. Tags showed the most promising performance in preliminary experiments, so we construct a tag vector for all tags assigned by the user to books read in the past and calculated the cosine similarity $sim_{tag}(u,i)$ between that vector and the **IT** row vector corresponding to book i. That way, a book that shares a lot of tags with books read by a user in the past will be seen as more similar. We refer to this as pers-similarity.

4.3 Training set results

Table 2 shows the results of the different social re-ranking runs for the optimal α values. We optimized in steps of 0.01. The baseline runs for both topic representations are also included for convenience.

The results of the social re-ranking approaches are very different for the two topic representations. When using the title field for retrieval, all non-personalized re-ranking methods provide impressive boosts over the baseline. The best-performing re-ranking approach here is II-similarity, which uses Amazon's data about "*similar products*". With an NDCG@10 of 0.2429 it increase performance over the baseline by 115%. Typically, most weight is given to the original scores with α values ranging from 0.92 to 0.99, although the other retrieved books do seem to offer a small but valuable contribution, given the performance increases.

Table 2. Results of the 12 different re-ranking runs using NDCG@10 as evaluation metric. The results of the best baseline runs for each topic representation are also included for convenience. Best-performing runs for each topic representation are printed in bold.

Runs	Topic fields			
	title		all-topic-fields	
	NDCG@10	α	NDCG@10	α
Baseline	0.1129	-	**0.3058**	-
IU-similarity	0.1631	0.92	0.3058	1.0
II-similarity	**0.2429**	0.94	0.3058	1.0
IT-similarity	0.1895	0.99	0.3058	1.0
IA-similarity	0.1535	0.96	0.3058	1.0
IUTA-similarity	0.1615	0.97	0.3058	1.0
pers-similarity	0.1293	0.65	0.3058	1.0

A possible explanation for the fact that II-similarity outperforms IU-similarity is that the latter similarities are calculated over an incomplete subset of Amazon user ratings; Amazon's "*similar products*" are likely calculated over all ratings. We can therefore also consider the results using II-similarity as an upper threshold on performance *if* we had all user ratings in the Amazon/LibraryThing collection.

Of the three types of similarity calculated directly on the Amazon/LibraryThing collection—IU-similarity, IT-similarity, and IA-similarity—re-ranking using tag overlap seem to provide the best performance with a score of 0.1895. Surprisingly, the combination of the three sources, IUTA-similarity, does not perform better than the individual sources. This is *not* in line with previous research [10].

However, when using all available topic fields for retrieval (all-topic-fields), social re-ranking does not help at all with all optimal $alpha$ values being equal to 1.0 (which retains only the original retrieval scores. Apparently, using longer query representations makes it that much easier for the retrieval algorithm to find matching book representations so that there is no room for other types of similarities to improve upon this. This suggests that social re-ranking methods have more merit in situations where user tend to use short queries, e.g., like in Web search engines.

Personalized re-ranking does not appear to work as well as non-personalized re-ranking. The most likely explanation for this is that LibraryThing topic creators typically ask for targeted recommendations on books they do not know anything about yet and do not have in their catalog yet. However, re-ranking the results lists towards a user's past books biases the results list to a ranking that is in fact *more* like books they already know about as opposed to new and relevant books.

5 Submitted runs

We selected six automatic runs for submission to INEX[6] based on the results of our content-based and social re-ranking runs. Two of these submitted runs were

[6] Our participant ID was 54.

content-based runs, the other four were social re-ranking-based runs. Since the re-ranking approaches did not benefit using all topic fields, we submitted three re-ranking runs based on the title and all-doc-fields baseline and one re-ranking run based on the all-topic-fields and all-doc-fields run.

Run 1 (title.all-doc-fields) This run used the titles of the test topics and ran this against the index containing all available document fields.

Run 2 (all-topic-fields.all-doc-fields) This run used all topic fields combined and ran this against the index containing all available document fields.

Run 3 (all-topic-fields.pers-similarity.α=0.99) This run applies the personalized re-ranking approach (pers-similarity) to run 2 with α set to 0.99; the value producing the highest NDCG scores yet not equal to 1.0.

Run 4 (title.pers-similarity.α=0.65) This run applies the personalized re-ranking approach (pers-similarity) to run 1 with α set to 0.65, which provided the best results for run 1 on the training set.

Run 5 (title.II-similarity.α=0.94) This run applies the re-ranking approach based on Amazon's "*similar products*" information (II-similarity) to run 1 with α set to 0.94, which provided the best results for run 1 on the training set.

Run 6 (title.IUTA-similarity.α=0.97) This run applies the re-ranking approach based on the combination of the three information sources (IUTA-similarity) to run 1 with α set to 0.97, which provided the best results for run 1 on the training set.

6 Results

The runs submitted to the INEX 2012 Social Book Search track were evaluated using graded relevance judgments. Books suggested by members other than the topic creator are considered relevant suggestions and received the relevance value 1. Books that are added by the topic creator to his/her LibraryThing catalog after creating the topic are considered the best suggestions and receive the relevance value 4. All runs were evaluated using NDCG@10, P@10, MRR, with NDCG@10 as the main metric. Table 3 shows the official evaluation results.

Table 3. Results of the six submitted runs on the test set, evaluated using all 301 topics with relevance judgments extracted from the LibraryThing forum topics. The best run scores are printed in bold.

Run #	Run description	NDCG@10	P@10	MRR
1	title.all-doc-fields	0.0678	0.0583	0.1341
2	all-topic-fields.all-doc-fields	**0.1492**	**0.1198**	**0.3069**
3	all-topic-fields.pers-similarity.α=0.99	0.1488	**0.1198**	0.3066
4	title.pers-similarity.α=0.65	0.0875	0.0719	0.1762
5	title.II-similarity.α=0.94	0.1173	0.1073	0.2558
6	title.IUTA-similarity.α=0.97	0.0958	0.0823	0.2392

We see that, unsurprisingly, the best-performing run on all 301 topics was run 2 with an NCDG@10 of 0.1492. Run 2 used all available topic fields and document fields. Again we see that re-ranking does not improve over the baseline when using all available topic fields. When using the title representation, we see the same performance improvements as on the training set. Run 5, for example, improves over the title baseline by 73.0%.

7 Discussion & Conclusions

On both the training and the test sets the best results were achieved by combining all topic and document fields. This shows continued support for the principle of polyrepresentation [11] which states that combining cognitively and structurally different representations of the information needs and documents will increase the likelihood of finding relevant documents. Adding extra controlled metadata from BL and LoC did not benefit the retrieval results however.

We also experimented with different re-ranking approaches where all the books retrieved in a run were able to contribute the final scores of each separate book by weighting those scores by their similarity to the target book. We examined the usefulness of different information sources for calculating these book similarities, such as user ratings, tags, authorship, and Amazon's "*similar products*" information. We found that all re-ranking approaches are successful when using shorter queries; longer topic representations did not benefit from re-ranking. Although all re-ranking approach improved retrieval results using the title representations as our topics, we found that Amazon's "*similar products*" information—being based on the complete set of Amazon user ratings—provides the best performance.

Personalized re-ranking did not work as well as the non-personalized methods, which is likely due its inappropriate for the recommendation task: the goal is not to find books similar to what the user has read in the past, but new books that are unlike the user's past interests.

References

1. Kazai, G., Koolen, M., Kamps, J., Doucet, A., Landoni, M.: Overview of the INEX 2011 Book and Social Search Track. In: INEX 2011 Workshop pre-proceedings. INEX Working Notes Series (2011) 11–36
2. Bogers, T., Christensen, K.W., Larsen, B.: RSLIS at INEX 2011: Social Book Search Track. In Geva, S., Kamps, J., Schenkel, R., eds.: INEX 2011: Proceedings of the 10th International Workshop of the Initiative for the Evaluation of XML Retrieval. Volume 7424 of Lecture Notes in Computer Science., Berlin, Heidelberg, Springer Verlag (2012) 45–56
3. 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
4. Järvelin, K., Kekäläinen, J.: Cumulated Gain-based Evaluation of IR Techniques. ACM Transactions on Information Systems **20**(4) (2002) 422–446
5. Furnas, G., Landauer, T., Gomez, L., Dumais, S.T.: The Vocabulary Problem in Human-System Communication. Communications of the ACM **30**(11) (1987) 964–971

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. Wang, J., de Vries, A.P., Reinders, M.J.: Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion. In: SIGIR '06: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA, ACM (2006) 501–508
8. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-Based Collaborative Filtering Recommendation Algorithms. In: WWW '01: Proceedings of the 10th International Conference on World Wide Web, New York, NY, USA, ACM (2001) 285–295
9. Linden, G., Smith, B., York, J.: Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing **7**(1) (2003) 76–80
10. Tso-Sutter, K.H.L., Marinho, L.B., Schmidt-Thieme, L.: Tag-aware Recommender Systems by Fusion of Collaborative Filtering Algorithms. In: SAC '08: Proceedings of the 2008 ACM symposium on Applied computing, New York, NY, USA, ACM (2008) 1995–1999
11. Ingwersen, P.: Cognitive Perspectives of Information Retrieval Interaction: Elements of a Cognitive IR Theory. Journal of Documentation **52**(1) (1996) 3–50

Do Social Information Help Book Search?

Ludovic Bonnefoy[1], Romain Deveaud[1] and Patrice Bellot[2]

[1] LIA - University of Avignon
ludovic.bonnefoy@etd.univ-avignon.fr
romain.deveaud@univ-avignon.fr

[2] LSIS - Aix-Marseille University
patrice.bellot@lsis.org

Abstract. In this paper we describe our participation in the INEX 2012 Book Track. The collection enters its second year of age and is composed of Amazon and LibraryThing entries for real books, and their associated user reviews, ratings and tags.
Like in 2011, we tried a simple yet effective approach of reranking books using a social component that takes into account both popularity and ratings. We did experiments using tags as well.

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.

In 2011, 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.

Like we already did last year, we used a Language Modeling approach to retrieval. For our recommendation runs, we used the reviews and the ratings attributed to books by Amazon users. We computed a "social score" for each book, considering the amount of reviews and the ratings. This score is then used to modify the initial ranking obtained by a Markov Random Field baseline that proved to be highly effective last year. We also used tags to build a profile for both a query and the books of the collection which we compared to rank the books.

The rest of the paper is organized as follows. The following Section gives an insight into the document collection whereas Section 2 describes the our retrieval framework. Finally, we describe our runs in Section 3.

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

2 Retrieval Model

2.1 Sequential Dependence Model

We used a language modeling approach to retrieval [2]. We use Metzler and Croft's Markov Random Field (MRF) model [3] to integrate multiword 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:

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

where the features weights are set according to the author's recommendation ($\lambda_T = 0.85$, $\lambda_O = 0.1$, $\lambda_U = 0.05$). f_T, f_O and f_U are the log maximum likelihood estimates of query terms in document D, computed over the target collection with a Dirichlet smoothing.

2.2 Modeling book likeliness

The basic idea behind this likeliness is that if a book has a lot of reviews and if its ratings are generally good, then it must be a very good book.

$$\mathcal{L}(D) = \log(\#reviews(D)) \times \frac{\sum_{r \in \mathcal{R}_D} r}{\#reviews(D)}$$

where $\mathcal{R}_D$ is the set of all ratings given by the users for the book D, and $\#reviews(D)$ is the number of reviews.

We further rerank the books by weighting the previously computed SDM with the likeliness score. The scoring function of a book D given a query Q is thus defined as follows:

$$s(Q, D) = \mathcal{L}(D) \times score_{SDM}(Q, D)$$

2.3 Modeling book thematic relatedness

We want to represent each query Q by a thematic profile and rank books according to their relatedness to it. For this first attempt at using thematic (or

topic) relatedness we choosed to rely exclusively on user tags associated with the books in the collection. We consider as a thematic profile a set of tags weighted according to their significance for Q and we call it a tag profile (TP). As a preprocessing step, a tag profile is associated to each book in the collection. Tags are weighted according to a classic tf.idf measure (where the tf is the number of users who associated the tag to the book).

The main issue is to estimate a tag profile for a query. To construct it, inspired by the pseudo relevance feedback method, we summed the profile of the x top ranked books retrieved by mean of a information retrieval model (more details in runs section). Once the query's tag profile is build, we can compare book's tag profile to it with a vector similarity measure like the cosinus.

Finally, books of the collection are ranked according to the similarity of their profile to the query's one.

3 Runs

We submitted 4 runs for the Social Search for Best Books task. We used Indri[2] for indexing and searching. We did not remove any stopword and used the standard Krovetz stemmer.

mrf-booklike This run is the implementation of the SDM model described in Section 2.1 with the likeliness score.

IOT30 and IT30 Those two runs are based on the tag profile approach presented in Section 2.3. In this approach four parameters have to be fixed : The number x of top ranked books used to build the query's tag profile, the weight given to each tag in query's profile, the information retrieval model used to retrieved books and the similarity measure to compare profiles. For both runs, x is fixed to 30, Indri's language modeling approach is used and the similarity measure is the cosinus angle between vectors.

The last parameter is the weight given to each tag of the query profile. For the IOT30 run, the t_i tag's weight is compute as the sum of its tf.idf weight in each of the top x books returned by Indri:

$$w(t_i) = \sum_{b \in Top_x} tf.idf(t_i, b)$$

where b is one the Top_x books retrieved.

However, we had the intution that all selected books can not contribute equally to the weight of a tag. So, for the IT30 run, we combine the tf.idf of a tag in a book with the relevance of this book according to the retrieval model used in order to penalize contribution of less relevant books:

$$w(t_i) = \sum_{b \in Top_x} tf.idf(t_i, b) \times score(b, Q)$$

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

where $score(b, Q)$ is the measure of relevance of the book b according to Indri.

deduce

B_IT30_30 For the last run we wanted to take advantage of both particularities of mrf-booklike run and a tag profile based one. We combine the mrf-booklike run to the IT30 run by mean of a logistic regression. We trained a model with two classes (relevant or not) and the book scores predicted by both runs as features. Training instances were the 30 top ranked books returned by each run along with their relevance judgment deduce from 2011 qrels.

4 Results

Table 1 shows 2012 official results and 2011 non official results for our 4 runs. The combination of mrf-booklike and IT30 improves the results as expected on 2012 qrels while it increase them dramatically for 2011 qrels. In 2012 our second run is mrf-booklike but it is the worse when evaluated with 2011 qrels which is surprising.

So, according to both officials results in 2012 and non official results based on 2011 qrels we can not answer to our question : *"Do Social Information Help Book Search?"*. It seems to vary a lot depending on evaluation corpus used. In order to explain those big differences we will need to make further experiments and more fine-grained analysis.

Run	nDCG@10	P@10	Recip rank	Recall@10
2012 Qrels - Officials				
Best Run 2012	0.1492	0.1198	0.3069	0.1527
B_IT30_30	0.1339	0.1260	0.3410	0.1659
mrf-booklike	0.1295	0.1250	0.3584	0.1514
IOT30	0.1141	0.1240	0.2933	0.1503
IT30	0.1082	0.1187	0.2999	0.1426
2011 Qrels - Non officials				
B_IT30_30	0.3408	0.2282	0.5398	
Best Run 2011	0.3101	0.2071	0.4811	
IT30	0.2995	0.2105	0.4626	
IOT30	0.2927	0.2081	0.4524	
mrf-booklike	0.2786	0.1890	0.4337	

Table 1. Comparison of our official results at INEX 2012 and non official results for 2011. The runs are ranked according to nDCG@10.

5 Conclusions

In this paper we presented our contributions for the INEX 2012 Book Track. We proposed a simple method for reranking books based on their likeliness and an effective way to take into account user tags. Finally a combination of both methods with a logistic regression approach gives the best results. Results does not allow us to answer on the usefulness of social information for book search despite quite good results.

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. Springer Berlin / Heidelberg, 2011.
2. D. Metzler and W. B. Croft. Combining the language model and inference network approaches to retrieval. *Inf. Process. Manage.*, 40:735–750, September 2004.
3. 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.

Practical Relevance Ranking for 10 Million Books.

Tom Burton-West

Digital Library Production Service, University of Michigan Library, Ann Arbor, Michigan, US
tburtonw@umich.edu

Abstract. In this paper we briefly describe our production environment and some of the open questions about relevance ranking for 10 million books. Then we describe our participation in the Prove It task of the INEX Social Book Search Track. We found that the queries supplied with the Prove It topics were not specific enough to provide good retrieval results. In contrast, the *fact* fields of the topics, when used as queries, provided good retrieval results. However, our query logs show that users are unlikely to enter queries as long as the *fact* fields. We tried to create queries that provided good retrieval results but better represented the queries in our logs. We also experimented with simulating the two-stage search process used in our production system when searching the entire corpus of 10 million books to find relevant books and then searching within the book to find relevant pages. While we succeeded in creating queries that were more specific than those supplied in the Prove It topics, and those queries produced better results, questions remain about how representative these created queries are of real user queries.

1 Introduction

The HathiTrust Digital Library is a digital preservation repository and access platform supported by a partnership of over sixty research institutions. The Digital Library Production Service of the University of Michigan Library supports search services over the full text of more than 10 million books in the repository, through HathiTrust Full-text search. Indexing on this scale presents a number of issues for relevance ranking.

The corpus used in previous INEX Book Tracks and in the 2011 and 2012 "Prove It" task contains the OCR and MARC metadata for about 50,000 public domain books. Since it is likely that most of these books are included among the over 3 mil-

lion public domain books within the HathiTrust repository, this corpus should provide a good test bed for relevance ranking experiments for HathiTrust Full-text search.[1]

1.1 The HathiTrust Full-Text Search System

We use custom-built middleware on top of the open-source Solr/Lucene search platform to index the 10 million books in the repository. In the original design of HathiTrust search, for performance and scalability reasons, we decided on a two tier indexing and searching architecture. Each tier is implemented as a separate Solr instance with its own separate index.[2] The first tier indexes all 10 million books with the unit of indexing the complete book. The second tier uses the page as the unit of indexing, but rather than indexing all 3 billion pages in the repository, documents are indexed on the page level on-demand. Searches are first executed against the index of 10 million books. Once a user clicks on a result, they are taken to a book viewer application. If they search within the book, the book is indexed on-the-fly and added to the separate page-level index.

1.2 Practical Relevance Concerns

Because we have both OCR and high quality MARC metadata, our current relevance ranking in production combines scores from the OCR field and various MARC fields using Solr/Lucene's boosting capability to weight fields. Our current boost values were determined by trial and error. We would like to have a systematic way to determine the optimum relative weights of the OCR and the MARC fields.

We suspect that the default term frequency normalization and length normalization provided by Lucene do not work well with our book length documents. With the exception of a few studies, the work on length normalization for information retrieval has been done on relatively small newswire size documents. Our average document length is around 100,000 words as compared to around 300 for the TREC ad hoc collections or 1,500-1,700 for the Gov2 or ClueWeb collections.

Lucene currently provides an alternative length normalization implementation (SweetSpotSimilarity), and Lucene 4.0 provides alternatives to the current Lucene

[1] Due to the absence of sufficient standard identifiers in the MARC metadata, we were not able to calculate how much of the INEX Prove It corpus is included in the HathTrust repository.

[2] For performance reasons the book-level index is split into 12 shards using Solr's distributed indexing/searching features

ranking model, including BM25 and DFR, which both allow tuning of parameters related to document length.[3] We would like to experiment with these approaches.

In 2007 in the TREC "Million Query Track," researchers at IBM modified Lucene's default length normalization and term frequency normalization with significantly better results on the web length documents in the Gov2 collection used in the Million Query Track [2]. We are interested in determining whether a similar approach would improve relevance ranking for book page size documents in our page-level index.

1.3 The INEX Prove It task

The goal of the Prove It task is to return individual book pages that confirm or refute a factual statement. Determining algorithmically whether a page confirms or refutes a factual statement is a hard problem. As first time participants in the Book Track, we decided to concentrate on retrieving relevant pages in the top of the ranked list and forgo the optional task of identifying whether a page confirms or refutes the fact. Our original goal for the Prove It task was to set up a baseline and then experiment with some of the length normalization approaches discussed above.

2 Preliminary Testing and Experimentation

2.1 Technical Issues

Our indexing, search, and document viewing infrastructure (both for testing and for production) assumes that digitized books are in our repository. For a number of reasons, we couldn't simply insert the INEX Prove It corpus into our repository. We had to do some re-engineering in order to set up a new environment which would allow indexing and running queries against the Prove It corpus. To simulate our production system, we created two indexes of the Prove It corpus, one which indexes the entire book as a single document, and another which indexes individual pages. We mapped MARC metadata into appropriate title, subject, author, etc., fields according to the mapping we use in production. For the Prove It page-level index, we copied the MARC fields (which apply to the whole book) to the indexing unit for each page. Each index has two fields containing the OCR text. The first field uses no stemming or stop words.[4] The second field contains the OCR content stopped with the Lucene default English stop word list and stemmed with the Porter stemmer.

[3] http://searchhub.org/dev/2011/09/12/flexible-ranking-in-lucene-4/

[4] This field simulates our production OCR indexing. In our production index we do not use stop words because we index works in over 400 languages and a stop word in

We also encountered some minor technical issues with corrupted MARC metadata and cleaned up what we could of the MARC metadata before adding it to the indexes.

2.2 Tests and Experiments

We began by running tests against the set of 21 training topics from the previous INEX Prove It task. [5]

To establish a baseline we decided to start by using a default Boolean "OR" operator between the query words and the default Lucene relevance ranking algorithm. Lucene's ranking is a vector-space, tf-idf variation which also includes a "coordination" factor where documents containing a higher percentage of the words in the query get a boost.[6]

We experimented with using different fields of the Prove It task topics. We used the *query* field alone, the *fact* field alone, the *query* and *fact* fields combined, and the *query*, *fact*, and *narrative* fields combined. The *query* fields tend not to include stop words, but the other fields do, so we did a version of each run with and without stopping and stemming. We also experimented with various weighting schemes including down-weighting stemmed versions of the fields, and schemes incorporating the MARC metadata.

It soon became apparent that the relative changes in relevance scores between different treatments using the same field from the topic, such as using stemming and stopping and not using them, or using different weighting schemes, were very small compared to the very large difference between using different fields from the topics. For example, using the *query* field achieved an NDCG@10 of 0.26, while the using the *fact* field achieved NDCG@10 of 0.68 (for the runs using stemming and stopping.)

---

one language is a content word in another. For example the word "die" is a stop word in German but a content word in English. (See http://www.hathitrust.org/blogs/large-scale-search/slow-queries-and-common-words-part-2.) We don't use stemming in the OCR field because stemming is a recall-enhancing process and we want to enhance precision, not recall.

[5] All results reported here use the 2011 qrels (inex11proveit.0_1_2.qrels) which include the additional 535 relevance judgments added by the University of Massachusetts.

[6] http://lucene.apache.org/core/3_6_0/api/core/org/apache/lucene/search/Similarity.html

The queries, (from the *query* field,) are insufficiently specific to retrieve relevant documents, compared to using the *fact* field verbatim as a query. For example, topics 2, 9, 10, 15, 42, 60, and 70 received NDCG@10 scores of 0 for the runs using the *query* field. The *fact* field for topic 70 is: *"Charles Darwin was born in Shrewsbury in 1809."* The *query* field for topic 70 is *"Charles Darwin."* This query does retrieve relevant results, but not in the top 10. Without providing the search engine any clue that the user is looking for a birth date or location, no amount of tuning the relevance algorithm will get these into the top results. Similarly, the *query* field for topic 15, *"Rome capital"* provides neither a clue that the topic is about the re-unification of Italy nor any other clue to the details in the *fact* or *narrative* fields. (The *fact* and *narrative* fields are used in making the relevance judgments). The problems with topic 15, as well as the general problems with the queries being underspecified, were noted by [4] in the INEX 2011 pre-proceedings.

These preliminary results presented us with a dilemma. In order to apply our findings to tune the production system, we wanted to use queries that simulate how real users interact with our production system. However, the *fact* fields supplied with the Prove It topics do not resemble the queries in our logs. The *fact* fields average 21 words. In contrast, in our query logs, the average query length is 2.74 words, and about 77% of our queries are three words or less. Only 10% of our queries are over five words in length and only 1% of queries are 12 words or more. Cummins and O'Riordan [3], found that length normalization needs different tuning parameters for short or long queries. Since we want to experiment with length normalization approaches appropriate for our production system, we do not want to use the *fact* fields which are significantly longer than most of the queries in our production system, as queries for our Prove It experiments.

We did a spot check of a random sample of queries from our logs that were longer than 10 words and found that most of them fell into three categories:

1. Long book titles or titles of government hearings or reports
2. Long quotes, such as "It was the best of times, it was the worst of times, it was the age of wisdom…"
3. Complex Boolean queries

The long quotes tend to be of two types: either they appear to be attempting to find the origin or uses of a well-know phrase such as "a stitch in time saves nine," or they appear to be an attempt to find the book from which a student copied some text (often the quotes are obviously from a textbook). Very few of these long quotes could be interpreted as attempts to verify a fact by copying it verbatim into the search box.

We plan to do user studies to characterize the various search tasks our users engage in when using HathiTrust search. In the absence of such studies, based on the brief log analysis above, and on the literature on interactive information retrieval, we believe that most of our users, who want to verify a fact, don't enter the fact verbatim into the search engine. Instead, we suspect that they start with a relatively short query, (such as those provided by the *query* field of the Prove It topics). When they see that the results are too broad, they add more terms to narrow the scope and bring relevant results to the top of the list.

We decided to simulate this process by recruiting a group of librarians and library programmers to come up with better queries. The intention was to come up with a query a user would eventually use after several iterations of interactive search. Two librarians and two library programmers each contributed queries for 20-21 topics, to provide new queries for all 83 Prove It task topics.

3 Production of New Queries

We asked each query creator to come up with two sets of queries for each topic. The first set was a "reasonable" query designed to bring the most relevant results to the top of the result list. The second set of queries was inspired by recent attempts to simulate interaction [1], and was modeled on the way users search using our production search process. We asked the query creators to give us a two-part query for each topic. The first part is the query they would use when searching our book-level index. The second part is the query that they would use when they searched within the book.

For the first set of queries, a relatively simple change often improved results significantly. For example, the *query* field for topic 9 is [Enchanted Windmill]. This query had an NDCG@10 score of 0. The query creator enclosed the terms in quotes to force the search engine to search for the terms as a phrase: ["Enchanted Windmill"]. This change alone increases the NDCG@10 from 0 to 0.48.

In order to facilitate the process of designing better queries, we modified a copy of our book search application to search the Prove It task page-level index, and to display ranked lists of pages with search terms highlighted. This proved useful in designing the first set of queries.

For the second set of queries, we wanted to provide an interface that executed the first part of the query against the PROVE IT index which indexed the entire book as one unit; and then allowed searching within the book at a page level using the second

part of the query, to simulate our production system. Unfortunately, we did not have enough time to implement an interface to allow the query creators to query the PROVE IT indexes in this way. They had to use our production indexes (limited to Public Domain books) to test their queries. One of the librarians noted that, for a number of queries, the production system had many more relevant documents, and suggested that the queries that worked well in the production system might not work as well in the PROVE IT collection.

4 Submitted Runs

Table 1 shows the results for the submitted runs. Unless otherwise specified, the runs all use the default Lucene English stop word list and the Porter stemmer on the OCR. (The "umich" prefix which is attached to all our run names has been removed for ease of reading.) The runs are described below.

Table 1. Submitted Runs

Run	NDCG@10	MAP	P@10
F	0.68	0.32	0.65
FQ	0.61	0.32	0.60
L	0.53	0.21	0.52
Ldismax_marc	0.53	0.21	0.52
HT25	0.34	0.11	0.33
Ldismax_marc (corrected)	0.30	0.11	0.32
Q	0.26	0.14	0.32

Standard runs:

F: The *fact* field

FQ: The *fact* and *query* fields concatenated

Q: The *query* field

Runs using queries created by librarians and library programmers:

L: The "Librarian" queries. These were created by the librarians and library programmers.

Ldismax_marc: The "Librarian" queries processed using the Solr dismax handler with weighting of the OCR, the stemmed and stopped OCR, and the MARC fields.[7] (See Appendix A for the details of the weighting)

HT25: The HathiTrust simulation. This run attempts to simulate how a user would search our production system. In our production system the default is to show the first 25 books on the first result page. We assume that the user would then click on one or more of the results on the first result page and then search within each book. In these runs, the first part of the query was run against the book-level index and the book ids of the books in the top 25 results were used to limit results to only pages within those books when combined with the second part of the query (which was run against the page-level index.)

Some of the queries created by the librarians and library programmers used the advanced search operators available in the production system, such as combining a search within the title, author, or subject fields with a search within the OCR. Due to time constraints we were not able to modify our software to run these automatically for the submitted runs. They were replaced with searches that did not require the advanced search capabilities.

5 Analysis

5.1 Towards "Realistic" Queries with More Specificity

The "Librarian" queries, run L, significantly improved upon the default queries (*query* field) supplied with the INEX topics. The default (Q) queries had an NDCG@10 score of 0.26, while the "Librarian" (L) queries had an NDCG@10 score of 0.53. However, the improved queries still did worse than the "fact" (F) run which had a score of 0.68. We spent some time trying to determine why the queries based on the *fact* field did so much better than the "Librarian" queries. We compared the NDCG@10 scores for individual topics and looked at the topics that had the greatest differences in the scores between the "Librarian" queries and the *fact* queries. In one case, the "Librarian" query misspelled an important proper name. Correction of that error brought the score for that topic from 0.05 to 0.89.

We also noted that some topics were quite complex and seemed to require a relatively large number of terms in order to get good results, for example topics 2 and 3. When relevance judgments are available for all 83 topics, it may be found that a larg-

[7] Due to a parsing error in the argument processing of our test harness, the submitted run was actually run with the same settings as the L run. Ldismax_marc_corrected is the unsubmitted corrected run that actually used the MARC fields and weighting.

er pool of topics might reduce the effect of particularly difficult or complex topics on the overall scores.

5.2 Simulation of a Two-Stage Search Process.

The HT25 run did very poorly. We investigated several topics where the score for the HT25 run was much worse than for the L run. We found a number of issues (discussed below.) We still believe the idea of simulating a book-level query followed by a page-level query is sound, but we need to further investigate the best way to implement this so that it achieves a reasonable simulation of the production system.

As we did not have time to set up a user interface to simulate our production system, our query creators had to use our production system when working on the HT25 two-part queries. We believe that differences between the production environment and our PROVE IT test environment are the major cause of the poor performance of the HT25 queries. As an example, for topic 0 the book-level query was [“battle of new Orleans” killed wounded] and the page-level query was [killed wounded]. In this case, we suspect that the query creators were misled because they were testing queries in our production system. In our production system, because users are searching the full text of 10 million records, we have set the default operator to a Boolean “AND.” For the PROVE IT book-level search we had the default operator set to a Boolean “OR.” This appears to have been a mistake.

In this particular case, in the production index, because of the default “AND” operator, the book-level query [“battle of new orleans” killed wounded] produced only books containing both the phrase “battle of new Orleans” and the words “killed” and “wounded.” Thus the subsequent page-level query [killed wounded] was only searching within books that at a minimum contained the phrase “battle of new Orleans,” and that page-level query appeared to be sufficiently specific to get good results.

On the other hand, in the PROVE IT book-level index, because of the default “OR” operator, some of the top results contained books that did not contain the phrase “battle of new Orleans”, but contained a huge number of occurrences of the words “killed” and “wounded.” When the page-level query was run against the page-level index, limited to the top 25 books from the book-level query, the pages that had the most occurrences of the words “killed” and “wounded” but did not mention “battle of new orleans” came to the top. A similar problem was found for several other low scoring HT queries, where the page-level query was specific enough in the context of the production environment, but not specific enough in the test environment.

We also discovered that, due to a parsing error in converting the submitted queries to a format suitable for running against Solr, the HT25 query for topic 60 returned 0 results.

6 Conclusions and Future Work

Our original goal was to set up a baseline and then experiment with some of the weighting and length normalization approaches available in Solr 4.0. However, we found we could not use the existing Prove It queries (from the *query* field of the topics) to generate a baseline. As noted in the previous INEX Prove It track, retrieval using the *fact* field produced much better results than using the *query* field. The differences in relevance scores between using these two different fields were very large. In contrast, when using a single field, changes in relevance scores resulting from using different query processing approaches or field weighting approaches were comparatively small. Since our query logs suggest that the *fact* field is not a realistic approximation of typical user behavior, and since our goal was to establish a baseline in order to improve our production system, we focused on trying to create more realistic queries and on simulating user interaction in our production system.

While we succeeded in creating queries that were more specific than those supplied in the Prove It topics, and those queries produced better results, questions remain about how representative these created queries are of real user queries. We plan to do user studies and further analysis of our query logs to determine the extent of fact-checking queries and better understand their characteristics. After that analysis is completed, we will need to determine whether to use the INEX Prove It corpus and our "Librarian" queries as a baseline to experiment with weighting and length normalization or instead to use real queries from our logs. If we use real queries from our logs, we will need to find a way to get relevance judgments.

We would also like to investigate further why using the HT25 queries did so poorly and try to devise a better way to simulate the two-level interactive query process used in our production system.

7 Acknowledgements

The author would like to thank Shevon Desai, Philip Farber, Christina Powell and Peter Ulintz, from the University of Michigan Library who provided the queries used in this work.

Bibliography

1. Azzopardi, Leif, Jarvelin, Kalervo, Kamps, Jaap and Mark D. Smucker. 2011. Report on the SIGIR 2010 workshop on the simulation of interaction. *SIGIR Forum* 44, 2 (January 2011), 35-47
2. Cohen, D., Amitay, E., Carmel, D.: Lucene and juru at trec 2007: 1-million queries track. In: Proceedings of the 16th Text REtrieval Conference (TREC 2007) (November 2007)
3. Cummins, Ronan and O'Riordan, Colm. 2009. The effect of query length on normalisation in information retrieval. In *Proceedings of the 20th Irish conference on Artificial intelligence and cognitive science* (AICS'09), Lorcan Coyle and Jill Freyne (Eds.). Springer-Verlag, Berlin, Heidelberg, 26-32.
4. Feild, H., Cartright, M. and Allan, J. 2011l "The University of Massachusetts Amherst's participation in the INEX 2011 Prove It Track," In the INEX 2011 Workshop Pre-proceedings. Shlomo Geva, Jaap Kamps, Ralf Shenkel eds. IR Publications Amsterdam. INEX Working Notes Series, Volume 2011 Saarbrucken, Germany, December 12-14, 2011.

Appendix: Weighting for Ldismax_marc run

The weighting is given below in Solr edismax query syntax. See http://wiki.apache.org/solr/ExtendedDisMax for an explanation of the syntax. The fields are as follows:

ocr: The OCR content with no stopping or stemming

ocrPorterStop: The OCR content with stopping using Lucene default English stop words and the Porter stemmer

allfiedsProper: A concatenation of all the MARC fields no stopping or stemming

```
{!edismax'
pf='ocr^25000+ocrPorterStop^400+allfieldsProper^50+'
pf3='ocr^2500+ocrPorterStop^40+allfieldsProper^10+'
pf2='ocr^250+ocrPorterStop^25+allfieldsProper^10+'qf='ocr
^25+ocrPorterStop^10+allfieldsProper^10+' mm='2<-1
5<67%25' tie='0.1' }'
```

Using Collaborative Filtering in Social Book Search

Hugo Huurdeman[1], Jaap Kamps[1,2], Marijn Koolen[1], and Justin van Wees[3]

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

Abstract. In this paper we describe our participation in INEX 2012 in the Social Book Search Track and the Linked Data Track. For the Social Book Search Track we compare the impact of query- and user-independent popularity measures and recommendations based on user profiles. Book suggestions are more than just topical relevance judgements and may include personal factors such as interestingness, fun and familiarity and book-related aspects such as quality and popularity. Our aim is to understand to what extent book suggestions are related to user-dependent and -independent aspects of relevance. Our findings are that evidence that is both query- and user-independent is not effective for improving a standard retrieval model using blind feedback. User-dependent evidence, on the contrary, is highly effective, leading to significant improvements. For the Linked Data Track we compare different methods of weighted result aggregation using the DBpedia ontology relations as facets and values. Facets and values are aggregated using either document counts or retrieval scores. The reason to use retrieval scores for facet ranking is that we want the top retrieved results to be summarised by the top ranked facets and values. In addition, we look at the impact of taking overlap in aggregation into account. Facet values that give access to many of the same documents have high overlap. Selecting facet values that have low overlap may avoid frustrating the user.

1 Introduction

In this paper we describe our participation in the INEX 2012 Social Book Search Track and the Link Data Track. For the Social Book Search Track we compare the impact of query- and user-independent popularity measures against recommendations based on user profiles. The web and social media have changed the way people search for books. The availability of user-reviews, ratings and tags allows users to find out more about a book than from the traditional descriptions made by professional cataloguers. This in turn may evoke more complex information needs from users, relating to issues such as how interesting, familiar or funny, educational, engaging, well-written or popular a book is. Some of these issues are user-independent, such as the popularity of a book and to some extent its quality—in the sense of the general opinion of a whole group readers—and

can be derived from data such as the number of people who reviewed, rated or tagged a book. Others are more personal, such interestingness and familiarity, and would require individual user information from user profiles or browsing and purchase history. We combine the user-dependent and -independent evidence with query-dependent evidence from a retrieval system to find out whether book suggestion can benefit from user-dependent evidence.

For the Linked Data Track (LDT), we experiment with different ways of aggregating results. A standard approach is to rank facets and values using document counts. The facets and values that summarise the most retrieval results are considered the best summarisations. We compare this approach with aggregation based on retrieval scores, which prefers facet values that summarise the highest ranked documents. Assuming most of the relevant documents will be in the top ranks, result aggregation based on retrieval scores will be focused on the most relevant documents. The document collection of the LDT is rich in structure and offers multiple ways of summarising search results. We use the DBpedia ontology relations as facets and values for summarisation.

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

2 Social Book Search Track

The effectiveness of user-generated content on social book search may be partly due to its relation to popularity [3]. The amount of user-generated content available for individual books is heavily skewed, with popular books have many more tags, reviews and ratings than more obscure books. Much like the impact of document length on traditional ad hoc search [6], the longer descriptions of popular books have a higher probability of matching query terms and possibly better term distribution statistics as well, with the result that retrieval models favour them over shorter descriptions of less popular books. This prompts the question whether the forum suggestions are merely the most popular among the topically relevant books, or whether personal preferences of the suggestors and topic creators brings in other aspects of relevance as well. If relevance in social book search is merely a combination of topical relevance and popularity, it would seem that book suggestions are mainly user-independent.

We want to compare the effectiveness of popularity priors against recommendations based on user profiles. The goal of our experiments is to investigate whether the impact of user-dependent evidence outweighs the available evidence for popularity, which is both query- and user-independent.

2.1 User-independent Priors

From the book descriptions in the A/LT collection we can derive several indicators of popularity and quality.

We look at the following popularity priors:

– *Length*: document length. Although document length is not directly related to popularity, we assume that descriptions with many tags and reviews are longer than descriptions with no or few tags and reviews.
– *Dirichlet*: without smoothing, language models favour short documents. Dirichlet smoothing introduces an implicit document length bias [7]. As smoothing parameter μ increases, document length becomes less important with respect to term frequency. In other words, documents with high term frequency will be favoured over documents with low term frequency regardless of their document lengths. With equal term frequency, a long document will still score lower than a short document, but the difference is small if μ is higher than the length of either document. The advantage of increasing μ over using a document length prior is that it only prefers longer documents when they have a higher frequency of query terms. With a global document length prior, a very long document with few occurrences of query terms still gets a big boost.
– *NumReviews*: the number of reviews. A large number of reviews means a large number of people know the book and voiced their opinion about it. Note that in constructing the A/LT collection, a maximum of 100 reviews per book were included. Books with at least 100 reviews are all considered equally popular even though the real number of reviews would differentiate between them.
– *SumTag*: the sum of all tag frequencies. The tag frequency of a tag *t*for a book b is the number of users who assigned t to b. We assume that popular books are tag by more users than more obscure books and therefore have a higher total number of tags. Of course, it is possible for a book to receive many tags from a small group of users, but we expect this to be the exception rather than the rule. Only the 50 most frequent tags of a book are included. The tag frequency is unlimited however, and therefore the total number of tags is also not capped.
– *MaxTag*: the frequency of most popular tag. This avoids the problem of conflating cases where many people assign only a few tags each to a book and cases where few people each assign many tags to a book. If the most frequent tag is assigned by n different users, then at least n users know about this book.

Next, we define two quality priors:

– *AvgRating*: average rating. The arithmetic mean over all Amazon ratings for a work.
– *BARating*: The Bayesian average rating. The Bayesian Average (BA) takes into account how many users have rated a work. As more users rates the same work, the average becomes more reliable and less sensitive to outliers. We make the BA dependent on the query, such that the BA of a book is based on books related to the query. The BA of a book b is computed as:

$$BA(b) = \frac{\hat{n} \cdot \hat{m} + \sum_{r \in R(b)} r}{n + \hat{n}} \tag{1}$$

where $R(b)$ is the set of ratings for b $\hat{m}$ is the average unweighted rating over all books in the top 1000 results and $\hat{n}$ is the average number of ratings over all the books in the top 1000.

We crawled a random set of 10,000 books from LibraryThing to obtain popularity information. Each page dedicated to a book contains information on how many members have catalogued it, how popular it is (directly determined by ranking all books by the number of members who catalogued it), how many members have reviewed it and in how many forum discussions it is mentioned (derived from Touchstone mappings).

We use this set to compare the total number of tags and the frequency of the most frequent tag against the number of members who catalogued it. The correlation between these numbers indicates how well our tag-based priors reflect popularity.

2.2 Collaborative Filtering

We want to compare the popularity based measures against methods that take the interests and preferences of the topic creator into account. Specifically, we want to look at collaborative filtering (CF) techniques to exploit the rich data available in the large network of users on LibraryThing. To build a recommender system based on CF, we had to obtain user profiles and personal catalogues of LibraryThing members. We started with a seed list of all the 1,104 users from the topic threads of the 211 topics of the 2011 SB task and crawled their personal catalogues and profiles. Links to other profiles (friends, members with interesting libraries) were extracted to continue the crawl. Because the members who participate in the forums may be different from other members, we also performed crawls based on random sets of 211, 1000 and 10,000 books. In each case, we extracted from each book page on LT the user names who have catalogued that book to generate another seed list. In total, we obtained 89,693 profiles (6% of all profiles) and 5,637,097 book ratings.

We experiment with neighbourhood-based and model-based recommendations and with rated transactions. Rated transactions indicate that a user catalogued a book and how she rated it. The k nearest neighbours ((k-NN) of a user u, denoted $N_i(u)$, are computed using the Pearson correlation of their transaction vectors. The rating r_{ui} of an unseen item i for user u is estimated as:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i(u)} w_{uv} r_{vi}}{\sum_{v \in N_i(u)} |w_{uv}|} \tag{2}$$

where users v are the nearest neighbours who have rated i. For some books, none of the nearest neighbours gave a rating, and k-NN cannot make a rating prediction. In this case, the average of the ratings of all users in our crawl for this book is used. If there is only one user who has rated the book, no reliable average can be obtained and no prediction is made.

Model-based recommender systems learn a predictive model based on the transactions of a user. The Singular Value Decomposition (SVD) method reduces the domain complexity by reducing the number of dimensions in the item space to a smaller set of underlying dimensions which represent the latent topics and user preferences [4].

2.3 Experimental Setup

We used Indri [8] for indexing, removed stopwords and stemmed terms using the Krovetz stemmer. Based on the results from the 2011 Social Search for Best Books task [1] we focus on the social metadata and indexed only user-generated content—Amazon reviews and LibraryThing tags—and book identification fields: title, author, publisher, publication date, dimensions, weight, and number of pages.

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 of the topics as queries, which corresponds to the titles of the topic threads of the LT discussion forums. For the language model our baseline has default settings for Indri (Dirichlet smoothing with $\mu = 2500$). We submitted two runs:

xml_social : a standard LM run on the social metadata index.
xml_social.fb.10.50 : a run on the social metadata index with pseudo relevance feedback using 50 terms from the top 10 results.

For the priors, each of the scores can be turned into a prior probability by dividing it by the sum of scores of all books in the collection. For instance, the document length prior probability is calculated as $P_{Length}(d) = |d|/|D|$, where D is the set of all books in the collection and $|D| = \sum_{d \in D} |d|$. The final document score is then:

$$S_{Length}(d) = P(d|q) \cdot P_{Length}(d) \tag{3}$$

With some priors there are problems with zero scores. A book with no reviews would have a prior probability of zero, which would result in a score $S_{NumReviews} = 0$. To solve this problem, we use the simple smoothing method known as Add-One, which adds one to the number of reviews of each book. The applies to the SumTag, MaxTag, AvgRating and BARating priors. In addition to linear prior probability, we experiment with log priors to compress the score range, thereby reducing the impact of the priors on the ranking. The log SumTag prior is calculated as:

$$P_{Log(SumTag)}(d) = \frac{1 + Log(1 + SumTag(d))}{\sum_{d' \in D} 1 + Log(1 + SumTag(d'))} \tag{4}$$

To rerank the retrieval results with user-dependent evidence from the Collaborative Filtering method, we use a linear combination:

$$S_{CF}(d) = (1 - \lambda) P_{Ret}(d|q) + \lambda P_{CF}(d) \tag{5}$$

Table 1: Evaluation results for the official Social Book Search task runs. Significance levels are 0.05 (°), 0.01 (•) and 0.001 (•).

Run	MRR		nDCG@10		P@10		R@10	
p4.xml_social	0.331		0.130		0.125		0.139	
p4.xml_social.fb.10.50	**0.370**°	**11.8**%	**0.146**	**11.8**%	**0.138**°	**10.3**%	**0.142**	**2.2**%

2.4 Results

The relevance judgements for the SBS task are based on the book suggestions from the LT fourms, and are mapped to three different relevance values: *irrelevant* (rv=0) for suggestions made by the topic creator herself, *relevant* (rv=1) for suggestions by others that the topic creator did not catalogue afterwards, and *highly relevant* (rv=4) for suggestions that the topic creator catalogued after starting the topic. We refer to the latter as post-catalogued suggestions (PCSs).

We first discuss the results of the official submissions (Table 1). Differences between the two runs are tested for statistical significance using a one-tailed Bootstrap test with 100,000 resamples, at significance levels of 0.05 (°), 0.01 (•) and 0.001 (•). The standard run on the xml_social index scores 0.331 on MRR, which means the on average, the first relevant document is found at rank 3. In the 2011 SB task, for which similar topics were used but all suggestions were considered equally relevant, a run on the same index scored 0.2913 on nDCG@10, but with this year's judgements it scores only 0.130. Either the topics this year are harder, or the impact of the difference relevance values is big and the system fails to distinguish between the PCSs and the other suggestions. If we map the PCSs to relevance value $rv = 1$, the nDCG@10 score goes up from 0.130 to 0.171, and if we map all suggestions to $rv = 1$ (similar to operationalisation used for last year's task), it goes up to 0.224. This means that this year's topics are more difficult, but also that the distinction between PCSs and other suggestions has made the task more difficult.

The feedback run improves upon the standard run for all four measures, with significant improvements for MRR and P@10. Adding terms from the top 10 documents leads to a better description of the information need. However, the improvement in nDCG@10, which emphasises the suggestions that the topic creator selects to add to her catalogue, is not significant. For our experiments with popularity and quality priors and recommendations we use the feedback run *p4.xml_social.fb.10.50* as the baseline, which is the highest scoring run of all official submissions on nDCG@10.

The results are shown in Table 2. We start with the quality priors. The ratings have little impact on performance. All variants are able to improve MRR, but on the other measures the improvements are smaller and not significant. The only exception is the plain Bayesian average prior, which is more effective than the others. This suggests that ratings are mainly useful for improving very early precision. The improvement of the BA Rating prior on nDCG@10 suggests that topic creators take ratings into account when selecting books. However, most improvements are not significant. Perhaps ratings do not reflect quality well, or

Table 2: Evaluation results for the Social Book Search task runs. Significance levels are 0.05 (°), 0.01 (•) and 0.001 (•).

Run	MRR		nDCG@10		P@10		R@10	
Baseline	0.362		0.144		0.122		0.149	
Quality priors								
AvgRating	0.377	4.3%	0.143	-0.6%	0.122	0.0%	0.153	2.6%
Log(AvgRating)	0.373	3.2%	0.143	-0.2%	0.125	2.5%	0.152	1.9%
BA Rating	0.379	4.8%	0.151	5.1%	0.126	3.4%	0.158	5.6%
Log(BA Rating)	0.374	3.6%	0.142	-1.3%	0.124	1.7%	0.151	0.9%
Popularity priors								
MaxTag	0.290•	-19.7%	0.082•	-43.3%	0.085•	-29.9%	0.093•	-38.0%
Log(MaxTag)	0.357	-1.2%	0.137	-4.4%	0.116	-5.2%	0.143	-4.4%
SumTags	0.274•	-24.3%	0.080•	-44.1%	0.087•	-28.2%	0.089•	-40.6%
Log(SumTags)	0.371	2.6%	0.145	0.7%	0.119	-2.6%	0.149	-0.1%
NumReviews	0.370	2.4%	0.129	-10.5%	0.110	-9.4%	0.129	-13.9%
Log(NumReviews)	0.403°	11.4%	0.161	12.1%	0.130	6.8%	0.158	5.9%
Length priors								
Length	0.357	-1.2%	0.126	-12.4%	0.112	-8.5%	0.137	-8.4%
Log(Length)	0.379	4.8%	0.149	4.0%	0.121	-0.9%	0.149	-0.3%
Dirichlet $\mu = 5000$	0.357	-1.2%	0.150	4.2%	0.128	5.1%	0.160	7.2%
Dirichlet $\mu = 10000$	0.371	2.7%	0.153	6.7%	0.128	5.1%	0.160	7.2%
Dirichlet $\mu = 15000$	0.355	-1.9%	0.151	5.4%	0.124	1.7%	0.150	0.4%
Recommendation								
k-NN (N=50, λ=0.0001855)	**0.411**•	**13.6%**	**0.181**•	**26.0%**	**0.154**•	**26.5%**	**0.199**•	**32.9%**
SVD (K=100, λ=0.000185)	0.403•	11.3%	0.172•	19.8%	0.149•	22.2%	0.187•	24.9%

quality is not effective as user-independent evidence. In the latter case, it might mean that quality is perceived differently by different users.

Next we discuss the popularity priors. The tag-based priors lead to significant drops in performance when used directly. Curbing their impact by taking the log of the MaxTag or SumTag scores is still not effective. Only the Log(SumTag) prior leads to small but insignificant improvements on MRR and nDCG@10. The number of reviews is more effective. The plain NumReviews prior only improves MRR but hurts performance on the other measures. The compressed score range of the Log(NumReviews) prior is more effective. Performance on all measures improves, with more than 11% improvements for MRR and nDCG@10. The larger improvement for nDCG@10 than for P@10 indicates the reviews are particularly useful for promoting suggestions that the topic creator decides to catalogue. Only the improvement on MRR is significant. This can mean that the number of reviews is a better indicator of popularity than SumTags and MaxTag, or that the topic creator tends to select books for which multiple reviews are available.

The Length prior is only effective when logged, and only improves performance on MRR and nDCG@10, but not significantly. The implicit length prior of the Dirichlet smoothing parameter μ is more stable, and improves performance on all measures for $\mu = 10,000$. With higher values for μ, performance starts to drop. Completely ignoring document length and only considering term frequency and document frequency is not good for performance. Even though promoting

longer document is effective, it is still important to connect term frequency to the amount of text in a document.

Although some popularity and quality ratings can improve performance, any improvements on the official measure nDCG@10 are not significant. Evidence that is both user- and query-dependent seems not effective for social book search.

Finally, we turn to the impact of combining retrieval with recommendation. For the k-NN method we experimented with different neighbourhood sizes (25, 50, and 100 neighbours) and λ values. Typically, the best performance with k-NN is achieved with $20 \leq k \leq 50$ ([2]). We show only the best performing combination, where $k = 50$ and $\lambda = 0.0001855$. For the SVD method, best performance was achieved with 100 dimensions (K=100) and $\lambda = 0.000185$). The recommendations from both SVD and k-NN lead to significant improvements on all measures. User-dependent evidence is highly effective for social book search. The k-NN method performs better than the more complex SVD method.

In sum, evidence based on personal preferences of the user seems much more effective than user-independent evidence based on popularity and quality. The low impact of the quality priors might indicate that quality in book search is more user-dependent. The effectiveness of the number of reviews may be an indicator that popularity can be effective, but also that forum members looking for books only catalogue books for which reviews are available. This is in line with our previous findings that workers on Mechanical Turk, when judging the relevance of books for the same LT forum topics, find it hard to judge books for which no reviews are available Koolen et al. [3]. With the presence of user reviews, the nature of relevance judgements has become more complex and goes beyond mere topical relevance.

3 Linked Data Track

For the Faceted Search Task of the Linked Data Track, systems are required to create a list of both facets and facet values for the explorative search queries contained in the topics of this task. The derived facets should describe relevant information for each of the queries featured in the task, preferably resulting in compact summaries of the available data. Our aim is to experiment with different ways of aggregating results, using either document counts or retrieval scores, and either ignoring or penalising document overlap in the ranking of facet values. The idea behind using retrieval scores for aggregation is that we want to focus on the top ranked results, as the retrieval model ranks documents by relevance, with the most relevant documents in the top of the list. Facet values that summarise many of the top documents give the user easy access to the most relevant documents.

Of course, the point of aggregation is to summarise long lists of results effectively and efficiently, so focussing on facet values that summarise only the top few documents defies the purpose of result aggregation. Good facet value selection requires a careful balance between high coverage and giving access to the most relevant documents.

Experimental Setup

We use Indri [8] with Krovetz stemming and default smoothing (Dirichlet with $\mu = 2500$) for indexing. Up to 2000 documents were retrieved using title fields only. We submitted one run for the Ad Hoc Search Task. For the Faceted Search task we were not able to finish any runs in time for the submission deadline.

The Ad Hoc run is used as the basis for carrying out the Faceted Search Task. We explored possibilities to extract different facets and facet values from the data available in the Wikipedia-LOD collection of the Linked Data Track. The candidate facets consist of the DBpedia relations and properties for each Wikipedia article included in the collection. For our exploration, we are also using additional ontological data available from DBpedia itself.

Facet selection

As a basic approach to performing the selection of facets, we used the concept of '*facet coverage*' [1]. This refers to the number of documents that are summarized by a facets top n values. The aim is to provide compact summaries of the available data using the selected facets, so these facets ideally should cover a high number of documents.

Using the ontology relations of DBpedia, we generated a list of all possible facets for a topic from the available DBpedia properties contained in each Wikipedia-LOD article in the collection. The list of facets includes the top 5 values for each facet, based on the number of documents a value covers, and the top 5 values based on their retrieval scores (originating from the baseline run created using Indri). To select a number of top facets out of the list of all facets for a given query, we are using different methods. One way to select the facets is based on the facet coverage. A disadvantage of this method, however, is that this does not take the overlap between facets into account. Therefore, a second method has been used, *coverageNO*, that focuses on the number of unique documents summarized by the facets top n values (see also [1]).

Based on a recursive selection method, it is possible to create a hierarchical list of facets and facet values. There are some issues with the available data from DBpedia, which influenced the facet selections that we explored in our research. First of all, there is a wide range of properties that are used for DBpedia entities, but not all of them are applied consistently. Furthermore, a substantial number of the top-ranked results from our baseline run do not have DBpedia properties, except for links to other pages, and therefore are not included in the generated facets. Finally, some of the entities have incorrect properties, possibly due to the semi-automatically generated structure of DBpedia, that is based on the user-authored data of Wikipedia. To overcome these limitations, we are also exploring ways to include additional data from DBpedia in the process of selecting facets, for example the ontological structure of DBpedia.[1]

[1] URL: http://mappings.dbpedia.org/server/ontology/classes/

4 Conclusion

In this paper we discussed our participation in the INEX 2012 Social Book Search Track and the Linked Data Track.

For the Social Book Search Track, we experimented with user-dependent and user-independent evidence in the form of document priors—length, book ratings, and numbers of tags and reviews—and user-dependent evidence in the form of recommendations from collaborative filtering approaches. We crawled a large set of user profiles and personal catalogues of LibraryThing members and experimented with neighbourhood-based and model-based recommender systems.

We found that document priors reflecting quality and popularity do not improve performance of a standard language model with blind feedback. The number of reviews of a book is the most effective prior, but does not lead to significant improvements. It is not clear whether the number of reviews is effective because it reflects popularity or because it promotes books for which the searcher can read multiple reviews and therefore make a more informed selection. Our findings suggest that evidence that is both query- and user-independent is not effective for social book search.

In contrast, user-dependent information from recommender systems is highly effective. Both k-nearest neighbour and SVD approaches lead to significant improvements. Although the k-NN method is less complex than SVD, it is the more effective of the two. Our findings suggest that user-dependent evidence is more important than user-independent information.

For the Linked Data Track, our aims are to compare the effectiveness of different result aggregation approach and of ignoring or penalising overlap in the results summarised by the chosen values of a selected facet. We are still implementing this model and the relevance judgements are not yet available, so we have no evaluation results yet.

Acknowledgments This research was supported by the Netherlands Organization for Scientific Research (NWO projects # 612.066.513, 639.072.601, and 640.005.001) and by the European Communitys Seventh Framework Program (FP7 2007/2013, Grant Agreement 270404).

Bibliography

[1] F. Andriaans, M. Koolen, and J. Kamps. The importance of document ranking and user-generated content for faceted search and book suggestions. In S. Geva, J. Kamps, and R. Schenkel, editors, *Focused Retrieval of Content and Structure: 10th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2011)*, volume 7424 of *LNCS*. Springer, 2012.

[2] C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. In Ricci et al. [5], pages 107–144. ISBN 978-0-387-85819-7.

[3] M. Koolen, J. Kamps, and G. Kazai. Social Book Search: The Impact of Professional and User-Generated Content on Book Suggestions. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM 2012)*. ACM, 2012.

[4] Y. Koren and R. M. Bell. Advances in collaborative filtering. In Ricci et al. [5], pages 145–186. ISBN 978-0-387-85819-7.

[5] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011. ISBN 978-0-387-85819-7.

[6] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In SIGIR *'96: Proceedings of the 19th annual international ACM* SIGIR *conference on Research and development in information retrieval*, pages 21–29, New York, NY, USA, 1996. ACM. ISBN 0-89791-792-8. doi: http://doi.acm.org/10.1145/243199.243206.

[7] M. D. Smucker and J. Allan. An investigation of dirichlet prior smoothings performance advantage. Technical report, 2005.

[8] 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.

OUC's participation in the 2012 INEX Book and Linked-Data Tracks

Michael Preminger[1], Ragnar Nordlie[1], David Massey[1], and Nils Pharo[1]

Oslo and Akershus University College of Applied Science

Abstract. In this article we describe the Oslo University College's participation in the INEX 2012 endeavor. This year we participate in the Book Track's "Prove it" (as in 2011) and Social search tasks, as well as the Linked Data track's Ad-hoc task.
In 2011, the OUC submitted retrieval results for the "Prove It" task with traditional relevance detection combined with detection of confrmation based on specificity detected through the Wordnet concept hierarchy. 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 semantic support based on Named entity recognition.
For the Social search task, we wish to examine the utility of the MARC-data (subject heading field) in social searching for readings.
For the Linked data task, we wish to explore the possibility of using links as a query expansion mechanism.

1 The Prove-it task of the Book Track

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.

1.1 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.

1.2 What Is a Proof?

What constitutes a proof is well defined in fields like mathematics and computer science. In connection with a claim or a statement of fact, it is less obvious what demands a passage of text should satisfy in order to be considered proof of the claim. Obviously, we are looking for a passage which expresses a relevant truth

about the claim, but what are the characteristics which signal a sufficient degree of relevance and truthfulness? We might want to identify a trustworthy passage, which in turn might be identified by considering the source of the passage, the degree to which the passage agreed with other passages treating the same claim or fact, or the centrality of the claim to the main content of the text. We might want to identify a concentrated passage, a passage where the largest amount of elements contained in the claim were represented or where they were by some measure most heavily represented. We might look for a definitional passage, which typographically or linguistically showed the characteristics of a definition. Or we might try to identify a "proof" by linguistic characteristics, mostly semantic, which might be of different kinds: certain typical words might be relatively consistently used to speak about a fact or claim in a "proving" manner, writing in a "proving" mode might entail using terms on a certain level of specificity, etc. 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.

1.3 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
- named entities of different kinds are heavily used
- 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.

In 2011, we were 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 determined the level of noun specificity through their place in the WordNet([2]) term hierarchies.

In this year's experiments, we proceed along the same line, trying to detect named entities. This years effort represents a starting point in taking into use named entity detection to assist in identifying confirming pages. Confirmation or proofs will often be about subjects identifiable by a name. Gradually, we first

need to find the limits of current detection of named entities, how easy it is to adapt it to a relatively diverse text mass that the (English part of) our text collection is, and then the approach's effectiveness in detecting proving pages. The two main possibilities in taking NED into use are:

- Detecting of named entities in general: pages that mention many named entities are candidates for being "confirming of something". Other methods are used to find the specific subject of proof. this means we only detect named entities in the book pages.
- Detecting the named entity being the subject of the statement to be proved. This means detecting named entities in the query, and in the.

Even though the latter possibility looks obvious it entails some problems, like polymorphism in identification of entities, which must be approached. This is the main rationale for starting out with the former possibility.

1.4 Ranking According to "Proof 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. Rank order 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 is. 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. Additionally, different contexts would arguably invite different proof emphases. All this advocates for use of other strategies of presenting proving results than ranked lists.

1.5 Indexing and Retrieval Strategies

The point of departure of the strategies discussed here is that confirming or refuting a statement is a simple action of speech that does not require from the book (the context of the retrieved page) to be *about* the topic covering the fact. In this way the "Prove It" task is different than e.g. the one referred to in [3] This means that we do not need the index we build for search purposes to be context-faithful (pages need not be indexed in a relevant book context). It is the formulation of the statement in the book or page that matters.

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

1.6 Indexing

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 catered for creating a simple index of all the words in the corpus, page by page. The pages are treated as separate documents regardless of the book in which they appear. The second aspect is catered for by

1.7 Named entity discovery

Named entity discovery is a natural language processing (NLP) activity. There exist several tools that perform NED. The choice this time fell on the opennlp package of the Apache project. The package was used with default settings (no special training), with the assumption that the big diversity of the book collection is not apt to any significant improvement with respect to the default settings.

1.8 Runs and Results

2 Social Book search

2.1 Introduction

The social book search features two representations of books: the social data, which is a mixture of "Amazon data" (descriptive and social data to facilitate book sale via Amazon) and social encounters as recorded in the libraryThing fora on one hand, and, on the other hand, traditional library data (MARC records) entered by professional catalogers. The main purpose is to find out the relative utility of each of these representations when it comes to automatic book recommendation.

[**?**] has attempted to compare the utility of social data to that of DEWEY classification data (which are also available in the Amazon records). In this paper we try to build upon this, and look at subject headings extracted from the MARC data.

2.2 Indexing and retrieval strategies

The collection has been loaded to a database where all types of data about each book are associated with the book's ISBN. We create an Indri index that includes all the tagged XML information that is extracted from both amazon and the LT fora. To each Indri document (book representation) we also add a section containing the subject headings extracted from the MARC record or records of that book[2].

[2] Some of the books contain MARC records from both the Library of congress as well as the British Library, with a greater diversity of subject headings.

6

```
<DOC>
<DOCNO>0525949283</DOCNO>
<TEXT>
<SH>Balloon ascensions Women balloonists</SH>
<bookdoc><book><isbn>0525949283</isbn><title>The Little
    Balloonist</title><ean>9780525949282</ean><binding>
    Hardcover</binding><label>Dutton Adult</label><
    listprice>$21.95</listprice><manufacturer>Dutton Adult
    </manufacturer><publisher>Dutton Adult</publisher><
    readinglevel/><releasedate/><publicationdate
    >2006-01-19</publicationdate><studio>Dutton Adult</
    studio><edition/><dewey>813.6</dewey><numberofpages
    >224</numberofpages><dimensions><height>70</height><
    width>580</width><length>850</length><weight>75</
    weight></dimensions><reviews><review><authorid>
    A1HA6KZZNDCME9</authorid><date>2007-02-25</date><
    summary>More like 2 1/2 stars...</summary><content>
    History collides with fiction in THE LITTLE BALLOONIST
     set ... may feel that the overall story is a bit
    rushed and that the possible depths that could have
    been conveyed just never emerged.  &lt;br /&gt; &lt;
    br /&gt;COURTESY OF CK2S KWIPS AND KRITIQUES</content
    ><rating>2</rating><totalvotes>2</totalvotes><
    helpfulvotes>1</helpfulvotes></review><review><
    authorid>A1UDDVTG2K1K72</authorid><date>2006-02-07</
    date><summary>Ahh...A Wonderful Love Story for
    Valentine's Day</summary><content>What a love story! A
     perfect present for Valentine's Day if you're still
    looking for any last minute gifts. I bought this book
    on a friend's recommendation and read it from start to
     finish in one evening. A lost love? A rekindled
    romance? All definitely keep the pages turning. And
    Donn's vivid descriptions of Napoleonic France all
    come alive. I highly recommend. &lt;br /&gt; &lt;br /&
    gt;</content><rating>5</rating><totalvotes>0</
    totalvotes><helpfulvotes>0</helpfulvotes></review></
    reviews></browseNode></browseNodes></book></bookdoc>
</TEXT>
</DOC>
```

Every possible element in this XML is made known to the indexing system, so it can be used as an extent e.g. for retrieval time weighting. An obvious strategy here is to weight <SH> at retrieval time when trying to find the effect of the subject headings

One problem we have is that only about two-thirds of our books have MARC records. This means that full comparison of the utility uses less documents. Still,

with many enough topics we may hope that a good number of those do have relevant *and* judged books among those with MARC records[3].

At retrieval time, weighting of subject headings can be done in the following way:

```
<query>
  <number>530</number>
<text>
   #combine ( #weight ( 1.0 #combine( Jesus.text Why.
       text scholarly.text perspective.text historical.
       text From.text ) 2.0 #combine( Jesus.sh Why.sh
        scholarly.sh perspective.sh historical.sh From.
       sh ) ) )
</text>
</query>
```

3 Preliminary runs and results

Preliminary runs were performed in accordance with the description in Section2.2. Table 1 summarizes the results. There seems to be a problem with the basic setting that makes it difficult to assess the contribution of the subject headings. This makes us refrain from further analysis at the present moment, apart from the suspicion that the results may be due to insensitive use of elements from the data collected from each book.

Run name	Query	Element weight	map	ndcg	ndcg_10	recip
sb_g0	Title only	all elements equal	0,0128	0,1713	0	0,0357
sb_2xh	Title only	sh (2) all (1)	0,02	0	0,0045	0,02
sb_g_ttl_nar	Title and narrative	all elements equal	0,065	0,1785	0,0756	0,1615
sb_g_ttl_nar_2xh	Title and narrative	sh (2) all (1)	0,03	0,016	0	0,03

Fig. 1. preliminary results for the Social search book track

4 The linked data track, the Ad-Hoc task

4.1 Introduction

The purpose of the linked data track is to find out how techniques within the semantic web / linked data can be used to improve and enhance retrieval of Wikipedia articles. The data collection is an XML'ified version of a Wikipedia

[3] We do not know it at the time of writing.

subset (about 4.1M articles), where incoming and outgoing links are tagged in terms of RDF-properties (DBpedia), and the article text is also included.

Our experiment is based on a two-stage approach. The initial search is in an index built from the entire corpus. Here each article is only represented by heading and category texts. For each topic, the initial search result (1000 articles) is enhanced by articles that form triples with the initial articles (as subjects or objects). This process results in a set of typically several thousands articles, including the initial set. Those are used to create a smaller "topic-wise index" with more data on each of the retrieved articles. In addition links in the articles are collected to enhance the results with the most popular articles that are not captured in the initial search.

4.2 Indexing and retrieval strategies

Stage one Prior to building the main index, a filter removes from the corpus files that are not considered to be articles. This included files that describe images. Files that had titles with the prefixes 'File:', 'Wikipedia:', 'Category', 'Portal:' and 'Template:' are removed. The aim of this process is to reduce the potential noise such files would create.

Text contained within the following tags is extracted for indexing:

- Title: tag-element with name-attribute 'title' within the metadata-template (TITLE)
- Heading 1: heading-element with level-attribute '2' (H2)
- Heading 2: heading-element with level-attribute '3' (H3)
- Category: property-element with name-attribute 'type' (CAT)

Common headings such as 'References', 'External links' and 'See also' are excluded from the index.

An example of a document ready for indexing by Indri:

```
<DOC>
<DOCNO>1x6bx0ax212420</DOCNO>
<TEXT><TITLE>The Scream</TITLE><H2>Sources of inspiration</H2>
<H2>Thefts</H2><H2>In popular culture</H2><H2>Gallery</H2>
<H3>Depersonalization disorder</H3><CAT>1893 paintings</CAT>
<CAT>Edvard Munch paintings</CAT><CAT>Expressionist paintings</CAT>
<CAT>Modern paintings</CAT><CAT>Symbolist paintings</CAT></TEXT>
</DOC>
```

Our aim is to use the link structure within the Wikipedia articles to enhance the initial search. It was therefore necessary to build a hash that links the title of the article to the file location. A Perl hash with this structure is used:

```
$fileHash{'La_Concorde'} = '1x6bx0ax265017';
$fileHash{'Embassy_of_Barbados_in_Washington,_D.C.'} = '1x6bx0ax31828384';
$fileHash{'Bosnia_and_Herzegovina_Hockey_League'} = '1x6bx0ax25617492';
```

```
$fileHash{'The_Scream'} = '1x6bx0ax212420';
$fileHash{'Belgium_at_the_1924_Summer_Olympics'} = '1x6bx0ax7521518';
$fileHash{'Lambrini'} = '1x6bx0ax5264971';
```

The query against the large index is limited to the text of the description-element of the topic.

An example of a search:

```
<text>
#weight(
10.0 #combine(learn.title major.title bicycle.title
        races.title multi.title affairs.title tour.title
        de.title france.title runs.title milan.title
        san.title remo.title )
 5.0 #combine(learn.cat major.cat bicycle.cat races.cat
        multi.cat affairs.cat tour.cat de.cat france.cat
        runs.cat milan.cat san.cat remo.cat )
 2.5 #combine(learn.h2 major.h2 bicycle.h2 races.h2
                multi.h2 affairs.h2 tour.h2 de.h2 france.h2
                runs.h2 milan.h2 san.h2 remo.h2 )
 1.0 #combine(learn.h3 major.h3 bicycle.h3 races.h3 multi.h3
                affairs.h3 tour.h3 de.h3 france.h3 runs.h3
                milan.h3 san.h3 remo.h3 ) )
 </text>
```

The fields title, cat, h2 and h3 where weighted in falling importance. The top 1,000 results are returned.

Stage two In-links to the articles had the following mark-up:

```
<property name='http://dbpedia.org/ontology/wikiPageWikiLink'>
    <subject name='http://dbpedia.org/resource/List_of_paintings_by_Edvard_Munch'>
    </subject>
</property>
<property name='http://dbpedia.org/ontology/wikiPageWikiLink'>
    <subject name='http://dbpedia.org/resource/Culture_of_Norway'></subject>
</property>
<property name='http://dbpedia.org/ontology/wikiPageWikiLink'>
    <subject name='http://dbpedia.org/resource/Silence_%28Doctor_Who%29'></subject>
</property>
```

While out-links:

```
<property name='http://dbpedia.org/ontology/wikiPageWikiLink'>
    <object name='http://dbpedia.org/resource/Pastel'></object>
</property>
```

```
<property name='http://dbpedia.org/ontology/wikiPageWikiLink'>
    <object name='http://dbpedia.org/resource/Tempera'></object>
</property>
<property name='http://dbpedia.org/ontology/wikiPageWikiLink'>
    <object name='http://dbpedia.org/resource/Oil_painting'></object>
</property>
```

All the in and out links from the 1,000 articles in the result for a topic are stored in a single array together with their frequencies. The 500 most popular links are added to the initial result set.

The combination of the original articles and the most popular linked articles are then indexed. This new index is based on the entire text of the article, i.e. the tags are removed. An example of a document ready for indexing:

```
<DOC>
<DOCNO>1x6bx0ax212420</DOCNO>
<TEXT>
 212420 The Scream The Scream disambiguation The Scream
     jpg 220px The Scream Norwegian Skrik Edvard Munch
     1893 Oil painting Oil tempera and pastel on cardboard
91 73 5 Oslo National Gallery of Norway National Gallery
    The Scream Norwegian Skrik created in 18931910 The
    Scream returns damaged but younger News com au 2008
05 21 is the title of expressionism expressionist
    painting s and prints in a series by Norway Norwegian
    artist Edvard Munch showing an agonized figure against
     a  blood red sky The landscape ...
 </TEXT>
 </DOC>
```

The new smaller index, approximately 1,400 articles, was searched to obtain the final ranking.

4.3 Alternative strategies

The chosen strategy is only one of many. Future experiments could study the importance of these components in the retrieval algorithm:

- Alternative weighting of the indexed fields in the initial (stage one) search
- More fields or different fields in the initial index
- Finding related articles using only in links or out links
- A smaller or larger indexing of the articles in stage two. The index could be enhanced with Google data on incoming link texts. Freebase and other linked data sources could also be used to enhance the index.
- Only the description was used to represent the query. Would additional fields from the topic give improved results?

5 Discussion, Limitation and Further Research

At the same time that the book world becomes more and more digital,as old books are being digitized and new books are increasingly published digitally, information not published in book format becomes more and more "semantic" in the sense that data pieces (as opposed to exclusively documents in the web's first years) are linked together and made available. These two parallel development entail great opportunities in the exploitation of book material for different purposes, of which the topic of this paper is one example.

This paper provides an example of the possibilities and the challenges. Whereas "WordNet specificity", here representing content independent linguistic semantic, is one simple example of information that can be used to systematically extract semantics from written content, other much larger and much more complicated sources of semantics, the semantic web and linked data, are waiting to be used in a similar (or related) way. To explore these possibilities we will need to experiment with more modern texts than what our present test collection contains.

To judge by the results of the runs presented here, this path of research, though promising, still requires a lot of modification and calibration.

Exploring the semantics of a page in a basically statistical 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 represents further progress from the 2010 experiments, but is still rudimentary. Nouns are 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 and such treatment will be more demanding.

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

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[4], we suspect that employing semantics-aware retrieval [5,6], which is closely connected to the development of the Semantic Web [7] 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 Yago2 [8] 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 (or confirmation or support) of factual statements is discussed in light of some rudimental retrieval experiments incorporating semantics. We also discuss the task of proving statement, raising the question whether it is classifiable as a semantics-aware retrieval task. Results are highly inconclusive.

References

1. Kazai, G., Koolen, M., Kamps, J., Doucet, A., Landoni, M.: Overview of the inex 2010 book track: Scaling up the evaluation using crowdsourcing. In: Comparative Evaluation of Focused Retrieval. Volume 6932 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2011) 98–117
2. Fellbaum, C.: WordNet : an electronic lexical database. MIT Press, Cambridge, Mass (1998)
3. Cartright, M.A., Feild, H., Allan, J.: Evidence finding using a collection of books. In: BooksOnline '11 Proceedings of the 4th ACM workshop on Online books, complementary social media and crowdsourcing, Amherst, MA (2011) 11–18
4. VOORHEES, E.M.: The trec question answering track. Natural Language Engineering **7** (2001) 361–378
5. Finin, T., Mayfield, J., Joshi, A., Cost, R.S., Fink, C.: Information retrieval and the semantic web. In: Proc. 38th Int. Conf. on System Sciences, Digital Documents Track (The Semantic Web: The Goal of Web Intelligence). (2005)
6. Mayfield, J., Finin, T.: Information retrieval on the semantic web: Integrating inference and retrieval. In: SIGIR Workshop on the Semantic Web, Toronto. (2003)
7. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American (2001)
8. Hoffart, J., Suchanek, F., Berberich, K., Weikum, G.: Yago2: A spatially and temporally enhanced knowledge base from wikipedia. Special issue of the Artificial Intelligence Journal (2012)

Overview of the INEX 2012 Tweet Contextualization Track

Eric SanJuan[1], Véronique Moriceau[2], Xavier Tannier[2], Patrice Bellot[3], and Josiane Mothe[4]

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

Abstract. The use case of the Tweet Contextualization task is the following: given a new tweet, participating systems must provide some context about the subject of a tweet, in order to help the reader to understand it. In this task, contextualizing tweets consists in answering questions of the form "what is this tweet about?" which can be answered by several sentences or by an aggregation of texts from different documents of the Wikipedia. Thus, tweet analysis, XML/passage retrieval and automatic summarization are combined in order to get closer to real information needs.
This article describes the data sets and topics, the metrics used for the evaluation of the systems submissions, as well as the results that they obtained.

Keywords: Automatic Summarization, Focused Information Retrieval, XML, Twitter, Wikipedia

1 Introduction

The Tweet Contextualization task to be performed by the participating groups of INEX 2012 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 context/summary.

We regard as relevant passages those that both contain relevant information but also contain as little non-relevant information as possible.

For evaluation purposes, we require that a summary uses only elements or passages previously extracted from the document collection. The correctness of summaries is established exclusively based on the support passages and documents. The summaries are evaluated according to:

- Informativeness: the way they overlap with relevant passages,
- Readability, assessed by evaluators and participants.

The paper is organized as follows. Section 2 details the collection of tweets and documents. Section 3 presents the metrics and tools used for evaluation, as well as results obtained by the participants. Finally, section 4 draws some preliminary conclusions.

2 Test data

Organizers provided a document collection extracted form Wikipedia, as well as 1000 topics made of tweets from several different accounts.

2.1 Tweets

About 1000 tweets in English were collected by the track organizers from Twitter® Search API. They were selected among informative accounts (for example, @CNN, @TennisTweets, @PeopleMag, @science...), in order to avoid purely personal tweets that could not be contextualized. Information such as the user name, tags or URLs have been provided. These tweets were available in two formats:

- a full JSON format with all tweet metadata. For example:


```
"created_at":"Wed, 15 Feb 2012 23:32:22 +0000",
"from_user":"FOXBroadcasting",
"from_user_id":16537989,
"from_user_id_str":"16537989",
"from_user_name":"FOX Broadcasting",
"geo":null,
"id":169927058904985600,
"id_str":"169927058904985600",
"iso_language_code":"en",
"metadata":"result_type":"recent",
"profile_image_url":"http://a0.twimg.com/profile_images/...",
"profile_image_url_https":"https://si0.twimg.com/profile_images/...",
"source":"&lt;a href=&quot;http://www.hootsuite.com...",
"text":"Tensions are at an all-time high as the @AmericanIdol
Hollywood Round continues, Tonight at 8/7c. #Idol",
"to_user":null,
"to_user_id":null,
"to_user_id_str":null,
"to_user_name":null
```

– a two-column text format with only tweet id and tweet text. For example:

```
169927058904985600 "Tensions are at an all-time high as the
@AmericanIdol Hollywood Round continues, Tonight at 8/7c. #Idol"
```

63 of these tweets were selected manually by organizers. For each of them, we checked that the document collection contained some information related to the topic of the tweet. This means that all 63 tweets had some contextualization material inside the provided collection.

From the accounts used for extraction of these 63 messages, a number of other tweets were automatically selected, bringing to 1000 the total number of tweets to be contextualized by the participants. This is done to ensure that only fully automatic and robust enough systems could accomplish the task.

However, only the 63 tweets that had been manually collected and checked have been used for informativeness evaluation; only 18 of them have been used for readability evaluation (due to the complexity of this evaluation).

2.2 Document collection

The document collection has been built based on a recent dump of the English Wikipedia from November 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 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.
[...]
</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>
```

2.3 Submission format

Participants could submit up to 3 runs. One run out of the 3 had to be completely automatic: participants had to use only the Wikipedia dump and possibly their own resources (even if the texts of tweets sometimes contain URLs, the Web must not be used as a resource). That is, a participant could not submit more than 3 runs in total.

A submitted summary has the following format:

```
<tid> Q0 <file> <rank> <rsv> <run_id> <text of passage 1>
<tid> Q0 <file> <rank> <rsv> <run_id> <text of passage 2>
<tid> Q0 <file> <rank> <rsv> <run_id> <text of passage 3>
...
```

where:

- The first column `tid` is the topic number.
- The second column is currently unused and should always be `Q0`. It is just a formating requirement used by the evaluation programs to distinguish between official submitted runs and q-rels.
- The third column `file` is the file name (without `.xml`) from which a result is retrieved, which is identical to the `<id>` of the Wikipedia document. It is only used to retrieve the raw text content of the passage, not to compute document retrieval capabilities. In particular, if two results only differ by their document id (because the text is repeated in both), then they will be considered as identical and thus redundant.

– The fourth column rank indicates the order in which passages should be read for readability evaluation, this differs from the expected informativeness of the passage which is indicated by the score rsv in the fifth column. Therefore, these two columns are not necessarily correlated. Passages with highest scores in the fifth column can be scattered at any rank in the result list for each topic.
– The sixth column run_id is called the "run tag" and should be a unique identifier for the participant group and for the method used.
– The remaining column gives the result passage in raw text without XML tags and without formatting characters. The only requirement is that the resulting word sequence appears at least once in the file indicated in the third field.

Here is an example of such an output:

```
167999582578552 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.
167999582578552 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.
167999582578552 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.
```

3 Evaluation

In this task, readability of answers [9] is as important as the informative content. Summaries must be easy to read as well as relevant. Following INEX 2011 Question-Answering task [1], these two properties have been evaluated separately by two distinct measures: informativeness and readability.

This section describes the metrics and tools used to perform the evaluation and gives results obtained by participating systems.

3.1 Baseline System

A baseline XML-element retrieval/summarization system has been made available for participants. This baseline is the same as 2011 QA@INEX task, and has been described in [1]. It relies on the search engine Indri[5] and a fast summarizer algorithm [2]. The system was available to participants through a web interface[6] or a perl API. Its default output has been added to the pool of submitted runs.

[5] http://www.lemurproject.org/
[6] http://qa.termwatch.es

3.2 Submitted Runs

33 valid runs by 13 teams from 10 countries (Canada, Chile, France, Germany, India, Ireland, Mexico, Russia, Spain, USA) were submitted.

This year only three teams used the provided perl API and Indri index of the collection.

The total number of submitted passages is 671,191 (31 596 328 tokens). The median number of distinct passages per tweet is 79.5 and the average is 146.5. Only passages starting and ending by the same 25 characters have been considered as duplicated, therefore short sub-passages could appear twice in longer ones.

3.3 Informativeness Evaluation

Informativeness evaluation has been performed by organizers on a pool of 63 tweets. For each tweet, we took the 60 best passages based on the rsv score in the fith column of the runs from all participants. After removing duplicates per tweet, 16,754 passages were evaluated by organizers. The median number of passages per tweet is 273 and the average is 265.9. Passages have been merged and displayed to the assessor in alphabetical order. Therefore, each passage informativeness has been evaluated independently from others, even in the same summary. The structure and readability of the summary was not assessed in this specific part, and assessors only had to provide a binary judgement on whether the passage was worth appearing in a summary on the topic, or not. 2,801 passages among 16,754 have been judged as relevant, with a median of 50 passages per tweet and an average of 55.1. The average length of a passage is 30.03 tokens.

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 this task, readability of answers [3] is as important as the informative content. Both need to be evaluated. Therefore answers cannot be any passage of the corpus, but at least well formed sentences. As a consequence, informative content of passages cannot be evaluated using standard IR measures since QA and automatic summarization systems do not try to find all relevant passages, but to select those that could provide a comprehensive answer. Several metrics have been defined and experimented with at DUC [4] and TAC workshops [5]. Among them, Kullback-Leibler (KL) and Jenssen-Shanon (JS) divergences have been used [6, 7] to evaluate the informativeness of short summaries based on a bunch of highly relevant documents.

In previous 2010 and 2011 INEX Question Answering tracks, evaluations have been carry out using FRESA package which includes a special lemmatizer. In 2011 we provided the participants with a standalone evaluation toolkit based on

Porter stemmer and implementing a new normalized ad-hoc dissimilarity defined as following:

$$Dis(T,S) = \sum_{t \in T} \frac{f_T(t)}{f_T} \times \left(1 - \frac{\min(\log(P), \log(Q))}{\max(\log(P), \log(Q))}\right) \quad (1)$$

$$P = \frac{f_T(t)}{f_T} + 1 \quad (2)$$

$$Q = \frac{f_S(t)}{f_S} + 1 \quad (3)$$

where T is the set of terms in the reference and for every $t \in T$, $f_T(t)$ is its frequency in the reference and $f_S(t)$ its frequency in the summary.

The idea was to have a dissimilarity which complement has similar properties to usual IR Interpolate Precision measures. Actually, $1-Dis(T,S)$ increases with the Interpolated Precision at 500 tokens where Precision is defined as the number of word n-grams in the reference. The introduction of the log is necessary to deal with highly frequent words.

As previously announced, we used this software to evaluate informativeness and like in INEX QA tracks, we considered as T three different sets based on Porter stemming:

- Unigrams made of single lemmas (after removing stop-words).
- Bigrams made of pairs of consecutive lemmas (in the same sentence).
- Bigrams with 2-gaps also made of pairs of consecutive lemmas but allowing the insertion between them of a maximum of two lemmas.

Bigrams with 2-gaps appeared to be the most robust metric. Sentences are not considered as simple bags of words and the measure is less sensitive to sentence segmentation than simple bi-grams. This is why bigrams with 2-gaps is our official ranking metric for informativeness.

Bigrams with 2-gaps appeared to be the most robust metric in previous INEX QA tracks, however in this edition where topics are real tweets, measures based on bigrams with or without 2-gaps are strongly correlated. Meanwhile the measure based on simple uni-grams is also stable but gives a different ranking. This will be discussed during the CLEF workshop.

Results Results are presented in Table 1. The 3 top ranked runs improved the baseline. Runs with (*) have been submitted as "manual".

Dissimilarity values are very closed, however differences are often statistically significant as shown in table 2.

3.4 Readability evaluation

Human assessment Each participant had to evaluate readability for a pool of summaries of a maximum of 500 words each on an online web interface. Each summary consisted in a set of passages and for each passage, assessors had to tick four kinds of check boxes. The guideline was the following:

Rank	Run	unigram	bigram	with 2-gap
1	178	0.7734	0.8616	0.8623
2	152	0.7827	0.8713	0.8748
3	170*	0.7901	0.8825	0.8848
4	Baseline	0.7864	0.8868	0.8887
5	169	0.7959	0.8881	0.8904
6	168	0.7972	0.8917	0.8930
7	193	0.7909	0.8920	0.8938
8	185	0.8265	0.9129	0.9135
9	171	0.8380	0.9168	0.9187
10	186	0.8347	0.9210	0.9208
11	187	0.8360	0.9235	0.9237
12	154	0.8233	0.9254	0.9251
13	162	0.8236	0.9257	0.9254
14	155	0.8253	0.9280	0.9274
15	153	0.8266	0.9291	0.9290
16	196b	0.8484	0.9294	0.9324
17	196c	0.8513	0.9305	0.9332
18	196a	0.8502	0.9316	0.9345
19	164*	0.8249	0.9365	0.9368
20	197	0.8565	0.9415	0.9441
21	163	0.8664	0.9628	0.9629
22	165	0.8818	0.9630	0.9634
23	150	0.9052	0.9871	0.9868
24	188	0.9541	0.9882	0.9888
25	176	0.8684	0.9879	0.9903
26	149	0.9059	0.9916	0.9916
27	156	0.9366	0.9913	0.9916
28	157	0.9715	0.9931	0.9937
29	191	0.9590	0.9947	0.9947
30	192	0.9590	0.9947	0.9947
31	161	0.9757	0.9949	0.9950
32	177	0.9541	0.9981	0.9984
33	151	0.9223	0.9985	0.9988

Table 1. Informativeness results(official results are "with 2-gap").

- *Syntax* (S): tick the box if the passage contains a syntactic problem (bad segmentation for example),
- *Anaphora* (A): tick the box if the passage contains an unsolved anaphora,
- *Redundancy* (R): tick the box if the passage contains a redundant information, i.e. an information that has already been given in a previous passage,
- *Trash* (T): tick the box if the passage does not make any sense in its context (*i.e.* after reading the previous passages). These passages must then be considered at trashed, and readability of following passages must be assessed as if these passages were not present.
- If the summary is so bad that you stop reading the text before the end, tick all trash boxes until the last passage.

For each summary, the text without tags of the tweet was displayed, thus this year readability was evaluated in the context of the tweet, and passages not related to the tweet could be considered as trash even if there were readable.

Metrics and results To evaluate summary readability, we consider the number of words (up to 500) in valid passages. We used three metrics based on this:

- **Relevancy or Relaxed metric**: a passage is considered as valid if the T box has not been ticked,
- **Syntax**: a passage is considered as valid if the T or S boxes have not been ticked,
- **Structure or Strict metric**: a passage is considered as valid if no box has been ticked.

In all cases, participant runs are ranked according to the average, normalized number of words in valid passages.

A total of 594 summaries from 18 tweets have been assessed. The resulting 18 tweets are included in those used for informativeness assessment. Results are presented in Table 3. The last column gives the number of evaluated summaries for correponding run. Only runs that were evaluated on more that 6 summaries, are ranked following the relaxed metric. Missing evaluations were due to formatting problems, too long passages (more than 500 tokens) or missing summaries in the submitted runs.

4 Conclusion

In 2011 we experimented using the wikipedia to contextualize twitted New York times paper titles. There was a large overlapping between the two vocabularies. This year we selected a larger pool of public factual tweets with a much more diversified vocabulary. The robust baseline we provided was difficult to outperform on the average. This needs further analysis and will be discussed during the workshop. One reason could be that the baseline approach removes all non-nominals from tweet texts, keeping only nouns and adjectives and this can help

in wikipedia search. However, for specific tweets, to retrieve relevant information from the wikipedia, it was necessary to expand the tweet vocabulary or to use tags inside the tweet.

References

1. SanJuan, E., Moriceau, V., Tannier, X., Bellot, P., Mothe, J.: Overview of the INEX 2011 Question Answering Track (QA@INEX). In Geva, S., Kamps, J., Schenkel, R., eds.: Focused Retrieval of Content and Structure: 10th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2011. Volume 7424 of Lecture Notes in Computer Science., Saarbrcken, Germany, Springer Verlag, Berlin, Heidelberg (2012) 188–206
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. Pitler, E., Louis, A., Nenkova, A.: Automatic evaluation of linguistic quality in multi-document summarization. In: ACL. (2010) 544–554
4. Nenkova, A., Passonneau, R.: Evaluating content selection in summarization: The pyramid method. In: Proceedings of HLT-NAACL. Volume 2004. (2004)
5. Dang, H.: Overview of the TAC 2008 Opinion Question Answering and Summarization Tasks. In: Proc. of the First Text Analysis Conference. (2008)
6. Louis, A., Nenkova, A.: Performance confidence estimation for automatic summarization. In: EACL, The Association for Computer Linguistics (2009) 541–548
7. Saggion, H., Torres-Moreno, J.M., 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

	178	152	170	baseline	169	168	193	185	171	186	187	154	162	155	153	196b	196c	196a	164	197	165	163	150	188	176	156	149	157	191	192	161	177	151
178	-	2	-	1	2	2	1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
152	2	-	-	-	-	-	-	3	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
170	-	-	-	-	1	-	-	2	2	2	3	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
baseline	1	-	-	-	-	-	-	2	3	2	2	3	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
169	2	-	1	-	-	-	-	2	1	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
168	2	-	-	-	-	-	-	1	-	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
193	1	-	-	-	-	-	-	1	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
185	3	3	2	2	2	1	1	-	-	1	1	-	-	-	1	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3
171	3	2	2	3	1	-	2	-	-	-	-	-	-	-	-	-	-	-	-	1	3	3	3	3	3	3	3	3	3	3	3	3	3
186	3	3	2	2	2	2	2	1	-	-	-	-	-	-	-	-	-	1	1	2	3	3	3	3	3	3	3	3	3	3	3	3	3
187	3	3	3	2	2	2	2	1	-	-	-	-	-	-	-	-	-	-	-	2	3	3	3	3	3	3	3	3	3	3	3	3	3
154	3	3	2	3	2	2	2	-	-	-	-	-	-	-	2	-	-	-	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3
162	3	3	2	2	2	2	2	-	-	-	-	-	-	-	-	-	-	-	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3
155	3	3	3	3	3	2	3	-	-	-	-	-	-	-	-	-	-	-	1	3	3	3	3	3	3	3	3	3	3	3	3	3	3
153	3	3	3	3	3	2	3	1	-	-	-	2	-	-	-	-	-	-	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3
196b	3	3	3	3	3	3	3	2	-	-	-	-	-	-	-	-	-	3	-	1	3	3	3	3	3	3	3	3	3	3	3	3	3
196c	3	3	3	3	3	3	3	2	-	-	-	-	-	-	-	-	-	-	-	3	3	3	3	3	3	3	3	3	3	3	3	3	3
196a	3	3	3	3	3	3	3	2	-	1	-	-	-	-	-	3	-	-	-	2	3	3	3	3	3	3	3	3	3	3	3	3	3
164	3	3	3	3	3	3	3	2	-	1	-	2	2	1	2	-	-	-	-	2	3	3	3	3	3	3	3	3	3	3	3	3	3
197	3	3	3	3	3	3	3	3	1	2	2	3	3	3	3	1	-	-	-	-	1	2	3	3	3	3	3	3	3	3	3	3	3
165	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	2	1	-	-	2	3	3	3	3	3	3	3	3	3	3
163	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	-	-	3	3	3	3	3	3	3	3	3	3	3
150	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	3	-	-	-	-	1	1	1	1	2	3	3
188	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	-	-	-	-	-	2	1	1	3	3	3
176	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	-	-	-	-	-	3	1	1	3	3	3
156	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	-	-	-	-	-	-	-	-	-	2	2
149	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	1	-	-	-	-	-	-	-	1	3	3
157	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	1	2	3	-	-	-	-	-	2	3	3
191	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	1	1	1	-	-	-	-	-	1	1	
192	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	1	1	1	-	-	-	-	-	1	1	
161	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	3	3	-	1	2	-	-	-	3	3
177	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	3	3	1	1	3	-	-
151	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	3	3	1	1	3	-	-

Table 2. Statistical significance for informativeness evaluation (t-test, 1 : 90%, 2 = 95%, 3 = 99%, $\alpha = 5\%$).

Rank	Run	Relevancy	Syntax	Structure	Nb
1	185	0.7728	0.7452	0.6446	17
2	171	0.6310	0.6060	0.6076	10
3	168	0.6927	0.6723	0.5721	15
4	Baseline	0.6975	0.6342	0.5703	13
5	186	0.7008	0.6676	0.5636	18
6	170*	0.6760	0.6529	0.5611	16
7	165	0.5936	0.6049	0.5442	10
8	152	0.5966	0.5793	0.5433	16
9	155	0.6968	0.6161	0.5315	16
10	178	0.6336	0.6087	0.5289	17
11	169	0.5369	0.5208	0.5181	16
12	193	0.6208	0.6115	0.5145	13
13	163	0.5597	0.5550	0.4983	12
14	187	0.6093	0.5252	0.4847	18
15	154	0.5352	0.5305	0.4748	13
16	196b	0.4964	0.4705	0.4204	16
17	153	0.4984	0.4576	0.3784	14
18	164*	0.4759	0.4317	0.3772	15
19	162	0.4582	0.4335	0.3726	17
20	197	0.5487	0.4264	0.3477	15
21	196c	0.4490	0.4203	0.3441	16
22	196a	0.4911	0.3813	0.3134	15
23	176	0.2832	0.2623	0.2388	13
24	156	0.2933	0.2716	0.2278	9
25	188	0.1542	0.1542	0.1502	11
26	157	0.1017	0.1045	0.1045	13
27	161	0.0867	0.0723	0.0584	14
-	151	0.8728	0.8728	0.8720	5
-	150	0.8493	0.8493	0.7270	3
-	192	0.6020	0.6020	0.6020	2
-	191	0.6173	0.5540	0.5353	3
-	177	0.5227	0.4680	0.4680	3
-	149	0.1880	0.0900	0.0900	4

Table 3. Readability results with the relaxed and strict metric.

Passage retrieval for tweet contextualization at INEX 2012

Ayan Bandyopadhyay[1], Sukomal Pal[2], Mandar Mitra[1], Prasenjit Majumder[3], and Kripabandhu Ghosh[1]

[1] Indian Statistical Institute (Kolkata)
[2] ISM Dhanbad
[3] DAIICT Gandhinagar

Abstract. This paper describes some preliminary results obtained by treating the tweet contextualization task as a passage retrieval task. Each tweet was submitted as a query to the Indri 5.2 search engine after some preprocessing. Either paragraphs or sentences were retrieved in response to a query. Passages retrieved from the same document were concatenated. This approach does not work very well in terms of informativeness: the best of our runs was ranked 23rd out of 33 runs. Further exploration of ways to improve effectiveness is needed.

1 Introduction

The INEX tweet contextualization task at CLEF 2012 is a new task. The aim of this task is to provide some *context* for a given topic tweet [1]. For this task, the context consists of a passage of at most 500 words extracted from a cleaned dump of the English Wikipedia. It is intended to provide some background information that will help a user to better understand the tweet.

In this report, we describe our very preliminary attempts at tweet contextualization. To begin with, we have simply treated contextualization as a passage retrieval task. After some preprocessing, the textual content of a tweet is used as a query to retrieve paragraphs or sentences from the Wikipedia corpus. If multiple passages are retrieved from the same article, they are merged together.

Related work is discussed in the next section (Section 2). Our approach is described in Section 3. Section 4 presents our results and discusses some obvious limitations of our approach. Our plans for further experimentation are outlined in Section 5.

2 Related Works

The tweet contextualization task is introduced by INEX at CLEF 2012. Bellot et al. [1] describes overall report of the INEX 2011. This task is involved with tweet. Tweets are treated as topics here. `http://twitter.com` is one of the

[1] http://twitter.com

popular site of microblogging. Miles Efron [2] reveals an overview of microblog and behavior surrounding it e.g microblog retrieval, entity search, sentiment analysis. According to the passage retrieval point of view Robertson et al. [4] says why we should not use liner equation to merge passages retrieved form the same document. After the passage retrieval answer construction is the next part. Summarization and framing answer has a very important role. Salton et al. [5] says about automatic text summarization using Intra-document passage links. recent text summarization survey by Ani Nenkova et al. [3] helps to know a elaborate description of text summarization.

3 Experimental Setup

We divided each page in the corpus into separate paragraphs using the `<p>` and `</p>` tags. All text contained between these tags was indexed. Each paragraph was also split further into sentences using periods (.), question marks (?) and exclamation marks (!) as sentence delimiters. Stopwords were removed, and Porter's stemmer was used. Some statistics about the processed corpus are given below. Since any period (.) was regarded as an end-of-sentence marker, abbrevi-

Table 1. Comparison of paragraph and sentence level indexing and corpus statistics

	Paragraph Level	**Sentence Level**
Number of paragraph/sentence	8,388,955	26,039,270
Unique terms	2,878,685	2,876,680
Total terms	333,522,647	333,697,767

ations were also split up when the text was indexed at the sentence level. This is why the number of terms (total and distinct) is somewhat different when the same text is indexed at two levels of granularity.

The topic tweets (1142 in all) were provided in two formats: JSON and simple text. We used the simple text format. Stopwords, URLs, the name of the tweeting authority, and the text "RT" were removed. The remaining words were stemmed using Porter's stemmer. Using these preprocessed tweets as queries, and Indri 5.2 as the search engine, we retrieved in turn paragraphs and sentences for each query tweet. A total of three runs were submitted. Details about these runs are given below.

Run1 — Top 50 returned paragraphs were submitted. If multiple paragraphs were retrieved from a document, then those paragraphs were concatenated. The similarity scores of individual paragraphs were simply added together to obtain the score of the concatenated result. Any paragraph longer than 500 words (including those obtained by concatenation) was truncated to the first 500 words.

Run2 — Same as the Run1, except that we started with the top 100 sentences for each query.
Run3 — Same as the Run1, except that the top 100 paragraphs were used.

4 Results

Submitted summaries were evaluated according to their informativeness and readability. Table 2 compares the performance of our submitted runs (Run1, Run2, Run3) with the best run at INEX 2012.

Table 2. Comparison of submitted runs and the best run at INEX 2012

Run Name	Run ID	Rank (out of 33)	Unigram	Bigram	Skip Bigram
Run1	149	26	0.9059	0.9916	0.9916
Run2	150	23	0.9052	0.9871	0.9868
Run3	151	33	0.9223	0.9985	0.9988
Best	178	1	0.7734	0.8616	0.8623

It is clear that the overly simplistic approach that we tried did not perform well with regard to informativeness (they did obtain good readability, however). Out of these runs, the sentence-level run performs best. A number of obvious drawbacks need to be rectified.

- When multiple paragraphs / sentences from a single document are concatenated, their similarity scores are simply added together. This may lead to poor ranking [4]. The score of the combined passage needs to be calculated more carefully.
- We need to be more careful when splitting a paragraph into sentences. In particular, periods used with acronyms and abbreviations should not result in sentence breaks.
- Retrieved passages are arbitrarily truncated at 500 words, without checking for sentence boundaries.

5 Conclusion

As mentioned in Section 2, a number of query-oriented summarisation approaches have been proposed in earlier work. In future work, we intend to explore how these may be applied to the contextualization task. Also, given that the "topics" or tweets are short to start with (at most 140 characters, many of which are taken up by URLs), query expansion is likely to be beneficial. We also hope to investigate query expansion / reformulation techniques as ways to improve informativeness of the generated summaries.

References

1. Patrice Bellot, Timothy Chappell, Antoine Doucet, Shlomo Geva, Jaap Kamps, Gabriella Kazai, Marijn Koolen, Monica Landoni, Maarten Marx, Véronique Moriceau, Josiane Mothe, G. Ramírez, Mark Sanderson, Eric SanJuan, Falk Scholer, Xavier Tannier, Martin Theobald, M. Trappett, Andrew Trotman, and Q. Wang. Report on INEX 2011. *SIGIR Forum*, 46(1):33–42, 2012.
2. Miles Efron. Information search and retrieval in microblogs. *JASIST*, 62(6):996–1008, 2011.
3. Ani Nenkova and Kathleen McKeown. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2-3):103–233, 2011.
4. Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple BM25 extension to multiple weighted fields. In *Proc. CIKM*, pages 42–49. ACM, 2004.
5. Gerard Salton, Amit Singhal, Mandar Mitra, and Chris Buckley. Automatic text structuring and summarization. *Inf. Process. Manage.*, 33(2):193–207, 1997.

A Hybrid Tweet Contextualization System using IR and Summarization

Pinaki Bhaskar, Somnath Banerjee, and Sivaji Bandyopadhyay

Department of Computer Science and Engineering,
Jadavpur University, Kolkata - 700032, India
{pinaki.bhaskar, s.banerjee1980}@gmail.com, sivaji_cse_ju@yahoo.com

Abstract. The article presents the experiments carried out as part of the participation in the Tweet Contextualization (TC) track of INEX 2012. We have submitted three runs. The INEX TC task has two main sub tasks, Focused IR and Automatic Summarization. In the Focused IR system, we first preprocess the Wikipedia documents and then index them using Nutch with NE field. Stop words are removed and all NEs are tagged from each query tweet and all the remaining tweet words are stemmed using Porter stemmer. The stemmed tweet words form the query for retrieving the most relevant document using the index. The automatic summarization system takes as input the query tweet along with the title from the most relevant text document. Most relevant sentences are retrieved from the associated document based on the TF-IDF of the matching query tweet, NEs text and title words. Each retrieved sentence is assigned a ranking score in the Automatic Summarization system. The answer passage includes the top ranked retrieved sentences with a limit of 500 words. The three unique runs differ in the way in which the relevant sentences are retrieved from the associated document.

Keywords: Information Retrieval, Automatic Summarization, Question Answering, Information Extraction, INEX 2012

1 Introduction

With the explosion of information in Internet, Natural language Question Answering (QA) is recognized as a capability with great potential. Traditionally, QA has attracted many AI researchers, but most QA systems developed are toy systems or games confined to laboratories and to a very restricted domain. Several recent conferences and workshops have focused on aspects of the QA research. Starting in 1999, the Text Retrieval Conference (TREC)[1] has sponsored a question-answering track, which evaluates systems that answer factual questions by consulting the documents of the TREC corpus. A number of systems in this evaluation have successfully combined information retrieval and natural language processing

[1] http://trec.nist.gov/

techniques. More recently, Conference and Labs of Evaluation Forums (CLEF)[2] are organizing QA lab from 2010. INEX[3] has also started Question Answering track. Last year, INEX 2011 designed a QA track [1] to stimulate the research for real world application. The Question Answering (QA) task 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 (April 2011). This year they renamed this task as Tweet Contextualization.

Current INEX 2012 Tweet Contextualization (TC) track gives QA research a new direction by fusing IR and summarization with QA. The TC track of INEX 2012 had two major sub tasks. The first task is to identify the most relevant document from the Wikipedia dump, for this we need a focused IR system. And the second task is to extract most relevant passages from the most relevant retrieved document. So we need an automatic summarization system. The general purpose of the task involves tweet analysis, passage and/or XML elements retrieval and construction of the answer, more specifically, the summarization of the tweet topic.

Automatic text summarization [2] has become an important and timely tool for assisting and interpreting text information in today's fast-growing information age. Text Summarization methods can be classified into abstractive and extractive summarization. An Abstractive Summarization ([3] and [4]) attempts to develop an understanding of the main concepts in a document and then expresses those concepts in clear natural language. Extractive Summaries [5] are formulated by extracting key text segments (sentences or passages) from the text, based on statistical analysis of individual or mixed surface level features such as word/phrase frequency, location or cue words to locate the sentences to be extracted. Our approach is based on Extractive Summarization.

In this paper, we describe a hybrid Tweet Contextualization system of focused IR and automatic summarization for TC track of INEX 2012. The focused IR system is based on Nutch architecture and the automatic summarization system is based on TF-IDF based sentence ranking and sentence extraction techniques. The same sentence scoring and ranking approach of [6] and [7] has been followed. We have submitted three runs in the QA track (177, 191 and 192).

2 Related Works

Recent trend shows hybrid approach of tweet contextualization using Information Retrieval (IR) can improve the performance of the TC system. Reference [8] removed incorrect answers of QA system using an IR engine. Reference [9] successfully used methods of IR into QA system. Reference [10] used the IR system into QA and [11] proposed an efficient hybrid QA system using IR in QA.

Reference [12] presents an investigation into the utility of document summarization in the context of IR, more specifically in the application of so-called query-biased summaries: summaries customized to reflect the information need

[2] http://www.clef-initiative.eu//
[3] https://inex.mmci.uni-saarland.de/

expressed in a query. Employed in the retrieved document list displayed after retrieval took place, the summaries' utility was evaluated in a task-based environment by measuring users' speed and accuracy in identifying relevant documents. This was compared to the performance achieved when users were presented with the more typical output of an IR system: a static predefined summary composed of the title and first few sentences of retrieved documents. The results from the evaluation indicate that the use of query-biased summaries significantly improves both the accuracy and speed of user relevance judgments.

A lot of research work has been done in the domain of both query dependent and independent summarization. MEAD [13] is a centroid based multi document summarizer, which generates summaries using cluster centroids produced by topic detection and tracking system. NeATS [14] selects important content using sentence position, term frequency, topic signature and term clustering. XDoX [15] identifies the most salient themes within the document set by passage clustering and then composes an extraction summary, which reflects these main themes. Graph based methods have been also proposed for generating summaries. A document graph based query focused multi-document summarization system has been described by [16], [6] and [7].

In the present work, we have used the IR system as described in [10], [11] and [17] and the automatic summarization system as discussed in [6], [7] and [17]. In the later part of this paper, section 3 describes the corpus statistics and section 4 shows the system architecture of combined TC system of focused IR and automatic summarization for INEX 2012. Section 5 details the Focused Information Retrieval system architecture. Section 6 details the Automatic Summarization system architecture. The evaluations carried out on submitted runs are discussed in Section 7 along with the evaluation results. The conclusions are drawn in Section 8.

3 Corpus statistics

The training data is the collection of documents that has been rebuilt based on recent English Wikipedia dump (November 2011). All notes and bibliographic references have been removed from Wikipedia pages to prepare plain xml corpus for an easy extraction of plain text answers. Each training document is made of a title, an 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. Therefore, the resulting corpus has this simple DTD as shown in table 1.

Test data is made up of 1142 tweets from Twitter. There are two different formats of tweets, one is the full JSON format with all tweet metadata as shown in the table 2 and another is the two-column text format with only tweet id and tweet text as shown in the table 3.

Table 1. The DTD for Wikipedia pages

```
<!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>
```

Table 2. A full JSON format with all tweet metadata of INEX 2012 test corpus

```
"created_at":"Fri, 03 Feb 2012 09:10:20 +0000",
"from_user":"XXX",
"from_user_id":XXX,
"from_user_id_str":"XXX",
"from_user_name":"XXX",
"geo":null,
"id":XXX,
"id_str":"XXX",
"iso_language_code":"en",
"metadata":{"result_type":"recent"},
"profile_image_url":"http://XXX",
"profile_image_url_https":"https://XXX",
"source":"<a href='http://XXX'>",
"text":"blahblahblah",
"to_user":null,
"to_user_id":null,
"to_user_id_str":null,
"to_user_name":null
```

Table 3. A two-column text format with only tweet id and tweet text of INEX 2012 test corpus

Tweet Id	Tweet Text
170167036520038400	"What links human rights, biodiversity and habitat loss, deforestation, pollution, pesticides, Rio +20 and and a sustainable future for all?"

4 System Architecture

In this section the overview of the system framework of the current INEX system has been shown. The current INEX system has two major sub-systems; one is the Focused

IR system and the other one is the Automatic Summarization system. The Focused IR system has been developed on the basic architecture of Nutch[4], which use the architecture of Lucene[5]. Nutch is an open source search engine, which supports only the monolingual Information Retrieval in English, etc. The Higher-level system architecture of the combined Tweet Contextualization system of Focused IR and Automatic Summarization is shown in the Figure 1.

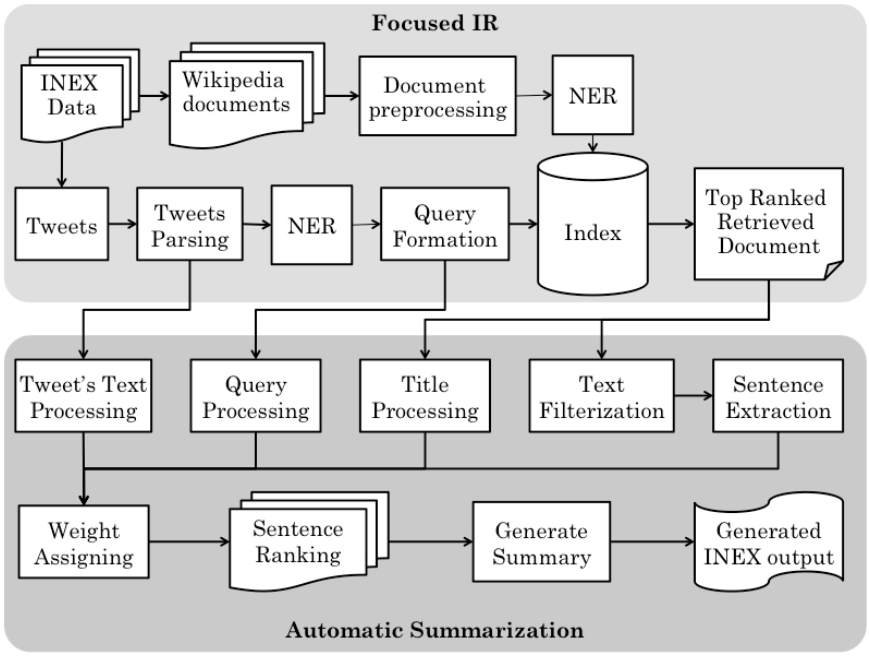


Fig. 1. Higher level system architecture of current INEX system

5 Focused Information Retrieval (IR)

5.1 Wikipedia Document Parsing and Indexing

The web documents are full of noises mixed with the original content. In that case it is very difficult to identify and separate the noises from the actual content. INEX 2012 corpus, i.e., Wikipedia dump, had some noise in the documents and the documents are

[4] http://nutch.apache.org/
[5] http://lucene.apache.org/

in XML tagged format. So, first of all, the documents had to be preprocessed. The document structure is checked and reformatted according to the system requirements.

XML Parser. The corpus was in XML format. All the XML test data has been parsed before indexing using our XML Parser. The XML Parser extracts the Title of the document along with the paragraphs.

Noise Removal. The corpus has some noise as well as some special symbols that are not necessary for our system. The list of noise symbols and the special symbols is initially developed manually by looking at a number of documents and then the list is used to automatically remove such symbols from the documents. Some examples are """, "&", "'''", multiple spaces etc.

Named Entity Recognizer (NER). After cleaning the corpus, the named entity recognizer identifies all the named entities (NE) in the documents and tags them according to their types, which are indexed during the document indexing.

Document Indexing. After parsing the Wikipedia documents, they are indexed using Lucene, an open source indexer.

5.2 Tweets Parsing

After indexing has been done, the tweets had to be processed to retrieve relevant documents. Each tweet / topic was processed to identify the query words for submission to Lucene. The tweets processing steps are described below:

Stop Word Removal. In this step the tweet words are identified from the tweets. The stop words and question words (what, when, where, which etc.) are removed from each tweet and the words remaining in the tweets after the removal of such words are identified as the query tokens. The stop word list used in the present work can be found at http://members.unine.ch/jacques.savoy/clef/.

Named Entity Recognizer (NER). After removing the stop words, the named entity recognizer identifies all the named entities (NE) in the tweet and tags them according to their types, which are used during the scoring of the sentences of the retrieved document.

Stemming. Query tokens may appear in inflected forms in the tweets. For English, standard Porter Stemming algorithm[6] has been used to stem the query tokens. After stemming all the query tokens, queries are formed with the stemmed query tokens.

[6] http://tartarus.org/~martin/PorterStemmer/java.txt

5.3 Document Retrieval

After searching each query into the Lucene index, a set of retrieved documents in ranked order for each query is received.

First of all, all queries were fired with AND operator. If at least one document is retrieved using the query with AND operator then the query is removed from the query list and need not be searched again. The rest of the queries are fired again with OR operator. OR searching retrieves at least one document for each query. Now, the top ranked relevant document for each query is considered for Passage selection. Document retrieval is the most crucial part of this system. We take only the top ranked relevant document assuming that it is the most relevant document for the query or the tweet from which the query had been generated.

6 Automatic Summarization

6.1 Sentence Extraction

The document text is parsed and the parsed text is used to generate the summary. This module will take the parsed text of the documents as input, filter the input parsed text and extract all the sentences from the parsed text. So this module has two sub modules, Text Filterization and Sentence Extraction.

Text Filterization. The parsed text may content some junk or unrecognized character or symbol. First, these characters or symbols are identified and removed. The text in the query language are identified and extracted from the document using the Unicode character list, which has been collected from Wikipedia[7]. The symbols like dot (.), coma (,), single quote (‘), double quote (“), ‘!’, ‘?’ etc. are common for all languages, so these are also listed as symbols.

Sentence Extraction. In Sentence Extraction module, filtered parsed text has been parsed to identify and extract all sentences in the documents. Sentence identification and extraction is not an easy task for English document. As the sentence marker ‘.’ (dot) is not only used as a sentence marker, it has other uses also like decimal point and in abbreviations like Mr., Prof., U.S.A. etc. So it creates lot of ambiguity. A possible list of abbreviation had to created to minimize the ambiguity. Most of the times the end quotation (”) is placed wrongly at the end of the sentence like .”. These kinds of ambiguities are identified and removed to extract all the sentences from the document.

[7] http://en.wikipedia.org/wiki/List_of_Unicode_characters

6.2 Key Term Extraction

Key Term Extraction module has three sub modules like Query Term, i.e., tweet term extraction, tweet text extraction and Title words extraction. All these three sub modules have been described in the following sections.

Query/Tweet Term Extraction. First the query generated from the tweet, is parsed using the Query Parsing module. In this Query Parsing module, the Named Entities (NE) are identified and tagged in the given query using the Stanford NER[8] engine.

Title Word Extraction. The title of the retrieved document is extracted and forwarded as input given to the Title Word Extraction module. After removing all the stop words from the title, the remaining tile words are extracted and used as the keywords in this system.

6.3 Top Sentence Identification

All the extracted sentences are now searched for the keywords, i.e., query terms, tweet's text keywords and title words. Extracted sentences are given some weight according to search and ranked on the basis of the calculated weight. For this task this module has two sub modules: Weight Assigning and Sentence Ranking, which are described below.

Weight Assigning. This sub module calculates the weights of each sentence in the document. There are three basic components in the sentence weight like query term dependent score, tweet's text keyword dependent score and title word dependent score. These three components are calculated and added to get the final weight of a sentence.

Query/Tweet Term dependent score: Query/Tweet term dependent score is the most important and relevant score for summary. Priority of this query/tweet dependent score is maximum. The query dependent scores are calculated using equation 1.

$$Q_s = \sum_{q=1}^{n_q} F_q \left(20 + \left(n_q - q + 1 \right) \left(\sum_p \left(1 - \frac{f_p^q - 1}{N_s} \right) \right) \times p \right) \quad (1)$$

where, Q_S is the query/tweet term dependent score of the sentence s, q is the no. of the query/tweet term, n_q is the total no. of query terms, f_p^q is the possession of the word which was matched with the query term q in the sentence s, N_s is the total no. of words in sentence s,

[8] http://www-nlp.stanford.edu/ner/

$$F_q = \begin{cases} 0; \textit{if query term } q \textit{ is not found} \\ 1; \textit{if query term } q \textit{ is found} \end{cases} . \quad (2)$$

and

$$p = \begin{cases} 5; \textit{ if query term is NE} \\ 3; \textit{if query term is not NE} \end{cases} \quad (3)$$

At the end of the equation 1, the calculated query term dependent score is multiplied by p to give the priority among all the scores. If the query term is NE and contained in a sentence then the weight of the matched sentence are multiplied by 5 as the value of p is 5, to give the highest priority, other wise it has been multiplied by 3 (as p=3 for non NE query terms).

Title Word dependent score: Title words are extracted from the title field of the top ranked retrieved document. A title word dependent score is also calculated for each sentence. Generally title words are also the much relevant words of the document. So the sentence containing any title words can be a relevant sentence of the main topic of the document. Title word dependent scores are calculated using equation 4.

$$T_s = \sum_{t=0}^{n_t} F_t \left(n_t - t + 1\right) \left(\sum_{p} \left(1 - \frac{f_p^t - 1}{N_s} \right) \right) \quad (4)$$

where, T_S is the title word dependent score of the sentence s, t is the no. of the title word, n_t is the total number of title words, f_p^t is the position of the word which matched with the title word t in the sentence s, N_s is the total number of words in sentence s and

$$F_t = \begin{cases} 0; \textit{if title word } t \textit{ is not found} \\ 1; \textit{if title word } t \textit{ is found} \end{cases} . \quad (5)$$

After calculating all the above three scores the final weight of each sentence is calculated by simply adding all the two scores as mentioned in the equation 6.

$$W_s = Q_s + T_s \quad (6)$$

where, W_S is the final weight of the sentence s.

Sentence Ranking. After calculating weights of all the sentences in the document, sentences are sorted in descending order of their weight. In this process if any two or more than two sentences get equal weight, then they are sorted in the ascending order of their positional value, i.e., the sentence number in the document. So, this Sentence Ranking module provides the ranked sentences.

6.4 Summary Generation

This is the final and most critical module of this system. This module generates the Summary from the ranked sentences. As in [13] using equation 9, the module selects

the ranked sentences subject to maximum length of the summary.

$$\sum_i l_i S_i < L \tag{9}$$

where l_i is the length (in no. of words) of sentence i, S_i is a binary variable representing the selection of sentence i for the summary and L (=500 words) is the maximum length of the summary.

Now, the selected sentences along with their weight are presented as the INEX output format.

7 Evaluation

7.1 Informative Content Evaluation

The organizers did the Informative Content evaluation [1] by selecting relevant passages. 50 topics were evaluated which was the pool of 14 654 sentences, 471 344 tokens, vocabulary of 59 020 words. Among them, 2801 sentences, 103889 tokens, vocabulary of 19037 words, are relevant. There are 8 topics with less than 500 relevant tokens. The evaluation measures of Information content divergences over {1,2,3,4gap}-grams (FRESA package) because it was too sensitive to smoothing on the qa-rels. So simple log difference of equation 10 was used:

$$\sum \log\left(\frac{\max\left(P(t/reference), P(t/summary)\right)}{\min\left(P(t/reference), P(t/summary)\right)}\right) \tag{10}$$

We have submitted three runs (177, 191 and 192). The evaluation scores with the baseline system scores of informativeness by organizers of all topics are shown in the table 4.

Table 4. The evaluation scores of Informativeness by organizers of all topics

Run Id	*unigram*	*bigram*	*Skip*
192	0.9590	0.9947	0.9947
191	0.9590	0.9947	0.9947
177	0.9541	0.9981	0.9984

7.2 Readability Evaluation

For Readability evaluation [1] all passages in a summary have been evaluated according to Syntax (S), Anaphora (A), Redundancy (R) and Trash (T). If a passage contains a syntactic problem (bad segmentation for example) then it has been marked

as Syntax (S) error. If a passage contains an unsolved anaphora then it has been marked as Anaphora (A) error. If a passage contains any redundant information, i.e., an information that have already been given in a previous passage then it has been marked as Redundancy (R) error. If a passage does not make any sense in its context (i.e., after reading the previous passages) then these passages must be considered as trashed, and readability of following passages must be assessed as if these passages were not present, so they were marked as Trash (T). The readability evaluation scores are shown in the table 5.

Table 5. The evaluation scores of Readability

Run Id	*Relevancy*	*Syntax*	*Structure*	*Nb*
192	0.6020	0.6020	0.6020	2
191	0.6173	0.5540	0.5353	3
177	0.5227	0.4680	0.4680	3

8 Conclusion and Future Works

The tweet contextualization system has been developed as part of the participation in the Tweet Contextualization track of the INEX 2012 evaluation campaign. The overall system has been evaluated using the evaluation metrics provided as part of this track of INEX 2012. Considering that this is the second participation in the track, the evaluation results are satisfactory, which will really encourage us to continue work on it and participate in this track in future.

Future works will be motivated towards improving the performance of the system by concentrating on co-reference and anaphora resolution, multi-word identification, para phrasing, feature selection etc. In future, we will also try to use semantic similarity, which will increase our relevance score.

Acknowledgements. We acknowledge the support of the IFCPAR funded Indo-French project “An Advanced Platform for Question Answering Systems” and the DIT, Government of India funded project “Development of Cross Lingual Information Access (CLIA) System Phase II”.

References

1. SanJuan, E., Moriceau, V., Tannier, X., Bellot, P., Mothe, J.: Overview of the INEX 2011 Question Answering Track (QA@INEX). In: Focused Retrieval of Content and Structure, 10th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX), Geva, S., Kamps, J., Schenkel, R. (Eds.). Lecture Notes in Computer Sc., Springer (2011)
2. Jezek, K., Steinberger, J.: Automatic Text summarization. In: Snasel, V. (ed.) Znalosti 2008. ISBN 978-80-227-2827-0, pp.1--12. FIIT STU Brarislava, Ustav Informatiky a softveroveho inzinierstva (2008)
3. Erkan, G., Radev, D.R.: LexRank: Graph-based Centrality as Salience in Text Summarization. In: Journal of Artificial Intelligence Research, vol. 22, pp. 457--479 (2004)

4. Hahn, U., Romacker, M.: The SYNDIKATE text Knowledge base generator. In: the first International conference on Human language technology research**,** Association for Computational Linguistics , ACM, Morristown, NJ, USA (2001)
5. Kyoomarsi, F., Khosravi, H., Eslami, E., Dehkordy, P.K.: Optimizing Text Summarization Based on Fuzzy Logic. In: Seventh IEEE/ACIS International Conference on Computer and Information Science, pp. 347--352. IEEE, University of Shahid Bahonar Kerman, UK (2008)
6. Bhaskar, P., Bandyopadhyay, S.: A Query Focused Multi Document Automatic Summarization. In: the 24th Pacific Asia Conference on Language, Information and Computation (PACLIC 24), Tohoku University, Sendai, Japan (2010)
7. Bhaskar, P., Bandyopadhyay, S.: A Query Focused Automatic Multi Document Summarizer. In: the International Conference on Natural Language Processing (ICON), pp. 241--250. IIT, Kharagpur, India (2010)
8. Rodrigo, A., Iglesias, J.P., Pe˜nas, A., Garrido, G., Araujo, L.: A Question Answering System based on Information Retrieval and Validation, ResPubliQA (2010)
9. Schiffman, B., McKeown, K.R., Grishman, R., Allan, J.: Question Answering using Integrated Information Retrieval and Information Extraction. In: NAACL HLT, pp. 532--539 (2007)
10. Pakray, P., Bhaskar, P., Pal, S., Das, D., Bandyopadhyay, S., Gelbukh, A.: JU_CSE_TE: System Description QA@CLEF 2010 – ResPubliQA. In: Multiple Language Question Answering (MLQA 2010), CLEF-2010, Padua, Italy (2010)
11. Pakray, P., Bhaskar, P., Banerjee, S., Pal, B.C., Bandyopadhyay, S., Gelbukh, A.: A Hybrid Question Answering System based on Information Retrieval and Answer Validation. In: Question Answering for Machine Reading Evaluation (QA4MRE), CLEF-2011, Amsterdam (2011)
12. Tombros, A., Sanderson, M.: Advantages of Query Biased Summaries in Information Retrieval. In: SIGIR (1998)
13. Radev, D.R., Jing, H., Styś, M., Tam, D.: Centroid- based summarization of multiple documents. J. Information Processing and Management. 40, 919–938 (2004)
14. Lin, C.Y., Hovy, E.H.: From Single to Multidocument Summarization: A Prototype System and its Evaluation. In: ACL, pp. 457--464 (2002)
15. Hardy, H., Shimizu, N., Strzalkowski, T., Ting, L., Wise, G. B., Zhang. X.: Cross-document summarization by concept classification. In: SIGIR, pp. 65--69 (2002)
16. Paladhi, S., Bandyopadhyay, S.: A Document Graph Based Query Focused Multi-Document Summarizer. In: the 2nd International Workshop on Cross Lingual Information Access (CLIA), pp. 55-62 (2008)
17. Bhaskar, P., Banerjee, S., Neogi, S., Bandyopadhyay, S.: A Hybrid QA System with Focused IR and Automatic Summarization for INEX 2011. In: Geva, S., Kamps, J., Schenkel, R.(eds.): Focused Retrieval of Content and Structure: 10th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2011. Lecture Notes in Computer Science, vol. 7424. Springer Verlag, Berlin, Heidelberg (2012)

LIA/LINA at the INEX 2012 Tweet Contextualization track

Romain Deveaud[1] and Florian Boudin[2]

[1] LIA - University of Avignon
romain.deveaud@univ-avignon.fr

[2] LINA - University of Nantes
florian.boudin@univ-nantes.fr

Abstract. In this paper we describe our participation in the INEX 2012 Tweet Contextualization track and present our contributions. We combined Information Retrieval, Automatic Summarization and Topic Modeling techniques to provide the context of each tweet. We first formulate a specific query using hashtags and important words in the Tweets to retrieve the most relevant Wikipedia articles. Then, we segment the articles into sentences and compute several measures for each sentence, in order to estimate their contextual relevance to the topics expressed by the Tweets. Finally, the best scored sentences are used to form the context. Official results suggest that our methods performed very well compared to other participants.

1 Introduction

The INEX Tweet Contextualization tracks aims at providing a small bunch of text (less than 500 words) that gives insights or additional information about a given tweet. For example, when reading a tweet about Whitney Houston's funerals, the user might want to know who is this person, why is she famous and so on... One of the strict constraint was to extract this context from a Wikipedia collection provided by the organizers, so there were several challenges to tackle.

First, it was very important to retrieve relevant and important Wikipedia articles that were related to the Tweets, and that were likely to provide some useful context. Second, considering the word limit of the contexts, only very little parts of these articles had to be kept. For this purpose we segmented the top-ranked articles into sentences and used several measures to score them. These measures range from classic word overlap or cosine similarity to conceptual similarity using topic models.

The rest of the paper is organized as follows. Section 2 describes the process we followed to extract candidate sentences, which includes Tweet formatting and document retrieval on Wikipedia. Then, we describe in Section 3 the various sentence scoring methods we used in this work.

2 Candidate Sentence Extraction

Considering that the task is to provide context from Wikipedia text, one crucial step was to retrieve Wikipedia articles that are highly relevant to the Tweet. Hopefully, relevant articles contain important sentences that give lots of contextual information.

2.1 #HashtagSplitting and Tweet formatting

Hashtags in Tweets are very important pieces of information, since they are tags that were generated by the user. Making a parallel with TREC-like topics, we can view the hashtags as the title while the Tweet itself is the description.

However the main problem with hashtags is that they often are composed of several words concatenated together (e.g. #WhitneyHouston). We used an algorithm based on Peter Novig's chapter on "Natural Language Corpus Data" in [5] to split the hashtags. For each Tweet, all the hashtags we converted into a short keyword query.

We also removed all the retweet mentions (RT), user mentions (@somebody) and stopwords (based on the standard INQUERY stoplist) from the Tweets. The final output of this Tweet formatting process is a clean Tweet without stopwords or useless mentions, as well as a very short and user-generated representation of this Tweet.

2.2 Retrieving Wikipedia articles

Retrieving relevant Wikipedia articles is the first crucial part for finding contextually relevant sentences. For this purpose we use the well-known Markov Random Field model [3] to represent dependencies between query words. It has indeed performed consistently well on several variety of ad-hoc search tasks across the years.

Given an initial Tweet $\mathcal{T}$, the output of the method described in the previous section is a set of hashtags $H_{\mathcal{T}}$ and a set of terms $Q_{\mathcal{T}}$. We then score Wikipedia articles D according to the following function:

$$s(H_{\mathcal{T}}, Q_{\mathcal{T}}, D) = \lambda \times score_{MRF}(H_{\mathcal{T}}, D) + (1 - \lambda) score_{MRF}(Q_{\mathcal{T}}, D)$$

where λ is a free smoothing parameter which was empirically set to 0.8 for all our experiments. We used the Sequential Dependance Model instantiation of MRF, which is defined as follows:

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

where the features weights are set according to the author's recommendation ($\lambda_T = 0.85$, $\lambda_O = 0.1$, $\lambda_U = 0.05$). f_T, f_O and f_U are the log maximum likelihood estimates of query terms in document D, computed over the target collection with a Dirichlet smoothing ($\mu = 2500$).

From the ranked list of Wikipedia articles, we only consider the 5 top articles as relevant. The underlying assumption is that a Tweet may discuss only a very limited amount of topics, due to the 140 characters limit. Since encyclopedic topics are very well delimited between Wikipedia articles, we thought 5 articles would treat roughly 4-5 to 10 different topics.

3 Sentence scoring

After selecting the 5 best ranked Wikipedia articles with respect to a Tweet $\mathcal{T}$, the next step is sentence segmentation. Each article is split into sentences which are the context candidates. We describe in this section the various scoring methods we used to estimate their importance with respect to the Tweet context.

3.1 Automatic summarization

First, we used some NLP scores that are widely used in the field of automatic summarization. For each candidate sentence S we computed:

- the word overlap between S and $Q_\mathcal{T}$, and between S and $H_\mathcal{T}$,
- the cosine similarity between S and $Q_\mathcal{T}$, and between S and $H_\mathcal{T}$,
- the TextRank [4] score of S in the context of the article from which it belongs.

3.2 Conceptual similarity

We the conceptual similarity measure, we wanted to estimate at which point a candidate sentence is close to a thematic or a topic that may be related to the Tweet. We used two sources from which we extracted the concepts: Wikipedia and the Web.

The Wikipedia source is a dump from July 2011 of the online encyclopedia that contains 3,214,014 documents[1]. For the Web source, we removed the spammed documents from the category B of the ClueWeb09 according to a standard list of spams for this collection[2]. We followed authors recommendations [2] and set the "spamminess" threshold parameter to 70. The resulting corpus is composed of 29,038,220 web pages.

We model the concepts using Latent Dirichlet Allocation [1], a generative probabilistic topic model. We want to model topics that are highly related to the Tweet, hence we perform LDA on the top-ranked Wikipedia or Web documents originally retrieved using the scoring function defined in 2.2. The documents of the collection are modeled as mixtures over K topics each of which

[1] `http://dumps.wikimedia.org/enwiki/20110722/`

[2] `http://plg.uwaterloo.ca/~gvcormac/clueweb09spam/`

is a multinomial distribution over the vocabulary W. Each topic multinomial distribution ϕ_k is generated by a conjugate Dirichlet prior with parameter β, while each document multinomial distribution θ_d is generated by a conjugate Dirichlet prior with parameter α. Thus, the topic proportions for document d are θ_d, and the word distributions for topic k are ϕ_k. In other words, $\theta_{d,k}$ is the probability of topic k occurring in document d (i.e. $P(k|d)$). Respectively, $\phi_{k,w}$ is the probability of word w belonging to topic k (i.e. $P(w|k)$).

In our sense, a concept is a topic generated by LDA from these top-ranked and supposedly highly relevant documents. Given a sentence S, a Tweet $\mathcal{T}$ and the learned topics $K_{\mathcal{T}}$, the conceptual score of S is given by:

$$\sigma(S) = \frac{1}{|K_{\mathcal{T}}|} \sum_{k \in K_{\mathcal{T}}} \left(\sum_{d} P(k|d) P(d|\mathcal{T}) \sum_{w \in W} p(w|k) \log \frac{N}{df_w} \right)$$

where N is the total number of documents in the collection, and df_w is the document frequency of w.

3.3 Tweeted URLs as context

A large part of the Tweets of the collection come along with an URL. This URL is the most important piece of context available, however the organizers judged to label as "manual" all the runs that used this information. We were not aware of this limitation and computed measures that are similar to the automatic summarization ones.

When a URL is present in the Tweet, we download the page and extract its title as well as the content of the body. For each candidate sentence S we computed:

- the word overlap between S and the title of the web page, and between S and the body content of the web page,
- the cosine similarity between S and the title of the web page, and between S and the body content of the web page.

3.4 Forming context

Our three runs follow the three types of measures we described above. After every sentence have been attributed a score, they are ordered and the top-ranked sentences are selected to form context (within the limit of 500 words).

4 Conclusions

In this paper we presented our contributions to the INEX 2012 Tweet Contextualization Track. We used various techniques involving automatic summarization and topic modeling algorithms to score the candidate sentences.

References

1. David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.
2. Gordon Cormack, Mark Smucker, and Charles Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Information Retrieval*, 2011.
3. 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.
4. Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, EMNLP '04, pages 404–411, 2004.
5. Toby Segaran and Jeff Hammerbacher. *Beautiful Data: The Stories Behind Elegant Data Solutions.* O'Reilly Media, 2009.

IRIT at INEX 2012: Tweet Contextualization

Liana Ermakova, Josiane Mothe

Institut de Recherche en Informatique de Toulouse
118 Route de Narbonne, 31062 Toulouse Cedex 9, France
liana.ermakova.87@gmail.com, josiane.mothe@irit.fr

Abstract. In this paper, we describe an approach for tweet contextualization developed in the context of the INEX 2012. The task was to provide a context up to 500 words to a tweet from the Wikipedia. As a baseline system, we used TF-IDF cosine similarity measure enriched by smoothing from local context, named entity recognition and part-of-speech weighting presented at INEX 2011. We modified this method by adding bigram similarity, anaphora resolution, hashtag processing and sentence reordering. Sentence ordering task was modeled as a sequential ordering problem, where vertices corresponded to sentences and sequential constraints were represented by sentence time stamps.

Keywords: Information retrieval, tweet contextualization, summarization, sentence extraction, sequential ordering problem, hashtag, anaphora resolution.

1 Introduction

In 2012 at the tweet contextualization INEX task, systems should provide a context about the subject of the tweet. The context should be a readable summary up to 500 words composed of passages from the English Wikipedia corpus from November 2011 [1]. INEX organizers selected about 1000 non-personal tweets in English.

Twitter is "a microblogging service that enables users to post messages ("tweets") of up to 140 characters - supports a variety of communicative practices; participants use Twitter to converse with individuals, groups, and the public at large, so when conversations emerge, they are often experienced by broader audiences than just the interlocutors" [2]. Twitter's data flow is examined in order to measure public sentiment, follow political activity and news [3]. However, tweets may contain information that is not understandable to user without some context. User may be not familiar with mentioned named entities like persons, organizations or places. Searching for them on a mobile device is time consuming and expensive. Therefore providing concise coherent context seems to be helpful. Contextualization as a summary on a specific topic may be used at libraries, editorial boards, publishers, Universities and Schools, cellular providers. The last ones can include it in a package of services for their clients, e.g. to clarify information about news tweet on a mobile device without web searching. In the summary, a customer will find relevant context for names, people, places and events from the news tweet.

adfa, p. 1, 2011.

Though the idea to contextualize tweets is quite recent [4], there are several works on summarization [5] as well as on sentence retrieval [6]. Saggion and Lapalme (2002) provide the following definition of a summary:

A summary is "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" [7].

Summaries may be either "extracts" (the most important sentences extracted from the original text), or "abstracts" (if these sentences are paraphrased) [8]. Anyway abstract generation is based on extracting components [9] and that is why sentence retrieval module seems to be the most valuable with regard to summary informativeness.

This year we modified the extraction component developed for INEX 2011 [10] which showed the best results according to relevance evaluation [11]. However, there were several drawbacks in readability: unresolved anaphora and sentence ordering. Apparently, anaphora resolution should result on not only readability, but also informativeness of a text. Thus, we added anaphora resolution and we reordered the extracted sentences with regard to a graph model. So, the task was reduced to travelling salesman problem which was solved by greedy nearest neighbor algorithm. Sentences were modeled as graph vertex and the similarity measure between them corresponded to edges. Moreover, we improved our approach by using linear combination of bigram and unigram similarity measure instead of unigram cosine. Last year two sentences from a New York Times article were considered as a query. This year approximately 1000 real tweets were collected by the organizers [1]. The tweets contained hashtags and @replies. Hashtags seems to provide very important information and therefore we assigned to them additional weight.

The paper is organized as follows. Firstly, we describe the modifications we made relative to previous year. Then we discuss evaluation results. Future development description concludes the paper.

2 Method Description

2.1 Searching for Relevant Sentences

The baseline system is based on TF-IDF cosine similarity measure enriched by smoothing from local context, named entity recognition and part-of-speech weighting [10].

First changes we made concern bigrams. Bigrams provide more specific information than unigrams and they are frequent enough in comparison with trigrams. Bigrams treating does not imply any syntactic analysis. The number of shared bigrams are often used to evaluate summaries [11][12]. Therefore, for each query and each sentence we computed the linear combination of the unigram and bigram cosine. We assigned the weight 0.3 and 0.7 to unigram and bigram similarity measure respectively.

In order to resolve pronoun anaphora we added the mention from the previous context. A mention is added in a summary only if other mentions excluding pronouns do

not occur in the same sentence as the pronoun anaphora. Anaphora was also resolved at the stage of sentence extraction. Since all mentions of the same notion may be considered as contextual synonyms, we included them into vector representation of a sentence, i.e. we expanded the original sentence by the contextual synonyms of all concepts occurring within this sentence. Anaphora resolution was performed by Stanford CoreNLP[1].

One of the features frequently used in the Twitter is the hashtag symbol #, which "is used to mark keywords or topics in a Tweet. It was created organically by Twitter users as a way to categorize messages" and facilitate the search [13]. Hashtags are inserted before relevant keywords or phrases anywhere in tweets – at the beginning, middle, or end. Popular hashtags often represents trending topics. Bearing it in mind, we put higher weight to words occurring in hashtags. Usually key phrases are marked as a single hashtag. Thus, we split hashtags by capitalized letters.

Moreover, important information may be found in @replies, e.g. when a user reply to the post of a politician. "An @reply is any update posted by clicking the "Reply" button on a Tweet" [14]. Sometimes people use their names as Twitter usernames. Therefore, we split these usernames in the way we did it with hashtags.

2.2 Sentence reordering

As Barzilay et al. showed in 2002 sentence ordering is crucial for readability [15]. In single document summarization the sentence order may be the same as the initial relative order in the original text. However, this technique is not applicable to multi-document summarization. Therefore, we propose an approach to increase global coherence of text on the basis of its graph model, where vertices represents sentences and the same TF-IDF cosine similarity measure as in searching for relevant sentences. If two relevant sentences are neighbors in the original text, they are considered as a single vertex. The hypothesis is that neighboring sentences should be somehow similar to each other and the total distance between them should be minimal. Firstly, we computed the similarity between sentences and reduced sentence ordering task to travelling salesman problem.

The travelling salesman problem (TSP) is an NP-hard problem in combinatorial optimization. Given a list of cities and their pairwise distances, the task is to find the shortest possible route that visits each city exactly once and returns to the origin city. In the symmetric case, TSP may be formulated as searching for the minimal Hamiltonian cycle in an undirected graph. Asymmetric TSP implies a directed graph [16]. The obvious solution is to use brute force search, i.e. find the best solution among all possible permutations. The complexity of this approach is $O(n!)$ while other exact algorithms are exponential. Therefore, we chose the greedy nearest neighbor algorithm with minor changes.

Since sentence ordering does not request to return to the start vertex and the start vertex is arbitrary, we tried every vertex as the start one and chose the best result.

[1] http://nlp.stanford.edu/software/corenlp.shtml

However, this method does not consider chronological constraints. So, we modified the task and it gave us the sequential ordering problem (SOP).

SOP "is a version of the asymmetric traveling salesman problem (ATSP) where precedence constraints on the vertices must also be observed" [17]. SOP is stated as follows. Given a directed graph, find a Hamiltonian path of the minimal length from the start vertex to the terminal vertex observing precedence constraints.

Usually SOP is solved by the means of integer programming. Integer programming is NP-hard and these methods achieved only limited success [17]. Therefore, we solved the problem as follows. Firstly, we ordered sentences with time stamps $s_1 - s_2 - \cdots - s_n$. Sentences without time stamp were added to the set $P = \{p_j\}_{j=\overline{1,m}}$. For each pair $s_i - s_{i+1}$ we searched for the shortest path passing through vertices from P. These vertices were removed from P and $i = i + 1$. If $i = n$, we searched for the shortest path passing through all vertices in P and the edge with the maximal weight was removed.

The major disadvantage of this approach is that a text with the same repeated sentence would be falsely overscored. The naïve approach to avoid it is to use a threshold value. However, it cannot deal with sentences of almost the same sense but different length (e.g. with more adjectives). We adopted the idea of H. G. Silber and K. F. Mccoy that nouns provide the most valuable information [18] and that is why we propose to introduce coefficients to distinguish the impact of nouns, other significant words and stop-words. A sentence was mapped into a noun set. These sets were compared pairwise and if the normalized intersection was greater than a predefined threshold the sentences were rejected.

3 Evaluation

Summaries were evaluated according to their informativeness and readability [1].

Informativeness was estimated as the overlap of a summary with the pool of relevant passages (number of relevant passages, vocabulary overlap and the number of bigrams included or missing). For each tweet, all passages were merged and sorted in alphabetical order. Only 15 passages with the highest score from each run were added in the pool. Assessors had to provide a binary judgment on whether the passage is relevant to a tweet or not.

We submitted three runs. The first run A considered only the unigram cosine between a query and a sentence. The second run C took into account the linear combination of the unigram and bigram similarity measures but did not imply anaphora resolution. The third one B differed from C by resolved anaphora.

Informativeness results for the submitted runs are presented in Table 1. Column *Run* corresponds to the run id, *Unigrams, Bigrams and Skip bigrams* represents the proportion of shared unigrams, bigrams and bigrams with gaps of two tokens respectively. According to informativeness evaluation, the impact of the linear combination of the unigram and bigram similarity measures is smaller than the impact of anaphora resolution.

Table 1. Informativeness evaluation

Run	Unigrams	Bigrams	Skip bigrams	Average
B	0.8484	0.9294	0.9324	0.9034
C	0.8513	0.9305	0.9332	0.9050
A	0.8502	0.9316	0.9345	0.9054

Readability was measured as an average score of proportion of text that makes sense in context (relevance), proportion of text without syntactical errors (syntax) and proportion of text without unresolved anaphora and redundant information (structure). Readability evaluation also provides evidence that anaphora resolution has a stronger influence on average score than the use of bigram cosine. It increases dramatically the structure score.

Table 2. Readability evaluation

Run	Relevance	Syntax	Structure	Average
B	**0.4964**	**0.4705**	**0.4204**	**0.4624**
153	0.4984	0.4576	0.3784	0.4448
164	0.4759	0.4317	0.3772	0.4283
162	0.4582	0.4335	0.3726	0.4214
197	0.5487	0.4264	0.3477	0.4409
C	**0.449**	**0.4203**	**0.3441**	**0.4045**
A	**0.4911**	**0.3813**	**0.3134**	**0.3953**

4 Conclusion

In this article, we describe a method to tweet contextualization based on the local Wikipedia dump. As a baseline system, we used TF-IDF cosine similarity measure enriched by smoothing from local context, named entity recognition and part-of-speech weighting presented at INEX 2011. We modified this method by adding bi-gram similarity, anaphora resolution, hashtag processing and sentence reordering. Sentence ordering task was modeled as a sequential ordering problem, where vertices corresponded to sentences and sentence time stamps represented sequential con-straints. We proposed the greedy algorithm to solve the sequential ordering problem based on chronological constraints. However, the organizers did not evaluate sentence order. In order to deal with redundant information we mapped each sentence into a noun set. These sets were compared pairwise and if the normalized intersection was greater than a predefined threshold, the sentences were rejected.

According to informativeness evaluation, the impact of the linear combination of the unigram and bigram similarity measures is smaller than the impact of anaphora resolution. Readability evaluation also provides evidence that anaphora resolution has a stronger influence on average score than the use of bigram cosine.

In future, we plan to work further with anaphora resolution and sentence ordering. It seems to be useful to find additional features special for the Twitter and to expand queries by synonyms and relations from WordNet. This should increase relevance as well as readability.

5 References

[1] "INEX 2012 Tweet Contextualization Track." [Online]. Available: https://inex.mmci.uni-saarland.de/tracks/qa/. [Accessed: 02-Aug-2012].

[2] D. Boyd, S. Golder, and G. Lotan, "Tweet, Tweet, Retweet: Conversational Aspects of Retweeting on Twitter," in *Proceedings of the 2010 43rd Hawaii International Conference on System Sciences*, 2010, pp. 1–10.

[3] N. Savage, "Twitter as medium and message," *Commun. ACM*, vol. 54, pp. 18–20, 2011.

[4] E. Meij, W. Weerkamp, and M. de Rijke, "Adding Semantics to Microblog Posts," *Pro-ceed-ings of the fifth ACM inter-na-tional con-fer-ence on Web search and data min-ing*, 2012.

[5] S. Gholamrezazadeh, M. A. Salehi, and B. Gholamzadeh, "A Comprehensive Survey on Text Summarization Systems," *Computer Science and its Applications*, pp. 1–6, 2009.

[6] V. G. Murdock, "Aspects of Sentence Retrieval," *Dissertation*, 2006.

[7] H. Saggion and G. Lapalme, "Generating Indicative-Informative Summaries with SumUM," *Association for Computational Linguistics*, vol. 28, no. 4, pp. 497–526, 2002.

[8] J. Vivaldi, I. da Cunha, and J. Ramırez, "The REG summarization system at QA@INEX track 2010," *INEX 2010. Workshop Preproceedings*, pp. 238–242, 2010.

[9] G. Erkan and D. R. Radev, "LexRank: Graph-based Lexical Centrality as Salience in Text Summarization," *Journal Of Artificial Intelligence Research*, vol. 22, pp. 457–479, 2004.

[10] L. Ermakova and J. Mothe, "IRIT at INEX: Question Answering Task," *Focused Retrieval of Content and Structure, 10th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2011), Geva, S., Kamps, J., Schenkel, R. (Eds.). Lecture Notes in Computer Science, Springer*, pp. 219–227, 2012.

[11] E. SanJuan, V. Moriceau, X. Tannier, P. Bellot, and J. Mothe, "Overview of the INEX 2011 Question Answering Track (QA@INEX)," *Focused Retrieval of Content and Structure, 10th International Workshop of the Initiative for the*

Evaluation of XML Retrieval (INEX 2011), Geva, S., Kamps, J., Schenkel, R. (Eds.), 2012.

[12] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pp. 74–81, 2004.

[13] "Twitter Help Center | What Are Hashtags ("#" Symbols)?" [Online]. Available: https://support.twitter.com/articles/49309-what-are-hashtags-symbols. [Accessed: 02-Aug-2012].

[14] "Twitter Help Center | What are @Replies and Mentions?" [Online]. Available: https://support.twitter.com/groups/31-twitter-basics/topics/109-tweets-messages/articles/14023-what-are-replies-and-mentions. [Accessed: 02-Aug-2012].

[15] R. Barzilay, N. Elhadad, and K. R. McKeown, "Inferring Strategies for Sentence Ordering in Multidocument News Summarization," *Journal of Artificial Intelligence Research*, no. 17, pp. 35–55, 2002.

[16] В. В. Морозенко, *Дискретная математика: учеб. пособие*. Пермь: ПГУ, 2008.

[17] I. T. Hernádvölgyi, "Solving the sequential ordering problem with automatically generated lower bounds," *Proceedings of Operations Research 2003*, pp. 355–362, 2003.

[18] H. G. Silber and K. F. Mccoy, "Efficiently computed lexical chains as an intermediate representation for automatic text summarization," *Computational Linguistics - Summarization*, vol. 28, no. 4, pp. 1–11, 2002.

DCU@INEX-2012: Exploring Sentence Retrieval for Tweet Contextualization

Debasis Ganguly, Johannes Leveling, and Gareth J. F. Jones

CNGL, School of Computing, Dublin City University, Dublin 9, Ireland
{dganguly, jleveling, gjones}@computing.dcu.ie

Abstract. For the participation of Dublin City University (DCU) in the INEX-2012 tweet contextualization task, we investigated sentence retrieval methodologies. The task requires providing the context to an ad-hoc real-life tweet. This context is to be constructed from Wikipedia articles. Our approach involves indexing the passages in Wikipedia articles as separate retrievable units, extracting sentences from the top ranked passages, computing the sentence selection score for each such sentence with respect to the query, and then returning the top most similar ones. The simple sentence selection strategy performed quite well in the task. Our best run has ranked first from the *readability* perspective and ranked eighth as ordered by *informativeness* out of 33 official runs.

1 Introduction

The tweet contextualization task was first introduced at INEX in 2011. The task requires construction of a short summary so as to explain the context associated with a given tweet. This context information has to be constructed from Wikipedia articles. As an example, for the CNN tweet "RT @CNNLive: View stake-out camera at funeral home where #WhitneyHouston body is expected to arrive in New Jersey. Watch live: `http://t.co/nyqT4PUa`", the system is expected to provide such expository information as who is Whitney Houston, what is she famous for, how did she die etc.

The task being different from standard ad-hoc IR poses with its own set of challenges. Firstly, the tweet text is very different from keyword based queries of ad-hoc search or web search. This necessiatates applying pre-processing steps on the tweet texts to get an appropriate query string. For example, the tweet hash-tags do not exist in Wikipedia articles and needs to be appropriately processed to get a useful query term. Secondly, a standard passage retrieval may not be suitable for the task because of the restriction on the length in the reported summary. The text in a passage itself may surpass the length theshold requirement of the summary. It thus makes sense to decompose passages into smaller units, i.e. sentences, and collate them together.

Previous approaches to INEX-QA have mostly used passage retrieval coupled with a summarizer. Sentence retrieval on the other hand has widely been employed in TREC-QA tasks for both factoid and definition question answering [1].

Sentence retrieval has the potential to perform well for tweet contextualization because sentences being short contain more focussed information than the relatively larger passages which may contain digressory content. Furthermore, the effect of sentence retrieval on tweet contextualization has still been unexplored. This motivated us to apply various sentence retrieval strategies on the tweet contextualization task.

2 System Description

In this section, we describe our system details. After describing the document and query processing, and retrieval, we focus on to our working methodologies for sentence selection.

2.1 Document Indexing

We used a modified version of the SMART[1] system for the experiments at INEX 2012. Each paragraph from the Wikipedia corpus[2] was indexed as a retrievable document unit. The beginning of a passage is marked by the XML tag <*p*>. This resulted in a total of over 26M passages to retrieve from. Extracted portions of documents, namely text under the <*title*>, <*p*>, <*h*>, <*t*> tags, were indexed using single terms and a controlled vocabulary (or pre-defined set) of statistical phrases following Salton's blueprint for automatic indexing [2]. Stopwords that occur in the standard stop-word list included within SMART were removed. Words were stemmed using a variation of the Lovins' stemmer implemented within SMART. Frequently occurring word bi-grams (loosely referred to as phrases) were also used as indexing units. We used the N-gram Statistics Package (NSP)[3] on the English Wikipedia text corpus from INEX 2006 and selected the 100,000 most frequent word bi-grams as the list of candidate phrases.

2.2 Query Processing

The tweet texts were pre-processed to produce queries to retrieve against the indexed collection. The pre-processing steps are described as follows. The URLs from the tweets were removed employing a regular expression based pattern matcher. *Medial capital* words, i.e. words with inner uppercase letters, were split into separate words e.g. the word "WhitneyHouston" was decomposed into "Whitney" and "Houston". Tweet hash-tags were split up into the prefix # character followed by the word, e.g. "#Whitney" was decomposed into # and "Whitney".

The following word breaking rules were applied to split hashtags starting with "#" and usernames starting with "@": A break between the last and current character is employed if:

[1] `ftp://ftp.cs.cornell.edu/pub/smart/`

[2] `http://dev.termwatch.es/esj/Term2IR/2012/data/tweetcontext2012corpus.xml.gz`

[3] `http://www.d.umn.edu/~tpederse/nsp.html`

i) the last character is lower case and the current character is upper case or digit (e.g. "OccupyWallStreet" ->"Occupy Wall Street");
ii) the last character is upper case and the last character of a valid acronym, the current character is also upper case or a digit (e.g. "CNNNews" -> "CNN News";
iii) the last character and the current character have different case and the resulting word would be longer than 3 characters.

2.3 Retrieval

The context for each tweet was constructed in two passes as follows. In the first pass, we retrieved N passages using language modelling (LM) [3] similarity with Jelinek-Mercer smoothing. The smoothing parameter λ was set to 0.6. In the second pass, we score senteneces based on three different methodologies, explained later in details. We then concatenate the top M sentences until the length of the concatenated summary string exceeds the threshold of 500 characters limit. The concatenation step ensures that we do not add duplicate sentences in the summary.

2.4 Sentence Retrieval Methodologies

Language Modelling Similarity. The most simple sentence scoring technique is that of scoring a sentence S by its LM score computed with respect to the query i.e. the pre-processed tweet text. This is done as shown in Equation 1.

$$P(S|Q) \propto \prod_{q \in Q} \mu P(q|S) + (1 - \mu) P(q) \tag{1}$$

Note that the smooting parameter μ used in Equation 1 is different from λ which was used for retrieving the passages as discussed in Section 2.3.

Relevance Model Similarity. The second sentence selection strategy which we use is derived from relevance model (RLM) term scores [4]. The key idea in RLM-based retrieval is that relevant documents and query terms are assumed to be sampled from an underlying hypothetical model of relevance R pertaining to the information need expressed in the query. In the absence of training data for the relevant set of documents, the only observable variables are the query terms and the top-ranked R pseudo-relevant documents assumed to be generated from the relevance model. Thus, the estimation of the probability of a word w being generated from the relevance model is approximated by the conditional probability of observing w given the observed query terms. Thus higher a word w co-occurs with a query tern q, higher is the likelihood of w to be sampled from the relevance model, i.e. higher is $P(w|\mathrm{R})$. This is shown in Equation 2.

$$P(w|q_i) \propto \sum_{j=1}^{R} P(w|D_j) P(q_i|D_j) \tag{2}$$

We can easily extend this notion of relevance model weighting of terms to whole sentences by simply aggregating over the constituent words of a sentence. This is shown in Equation 3 which we use to score every sentence and select the top-scoring ones in the returned summary.

$$P(S|\mathrm{R}) = \prod_{w \in S} P(w|\mathrm{R}) \tag{3}$$

Topical Relevance Model Similarity. This sentence selection score is based on an extended version of relevance model (RLM) similarity. In our extended relevance model, we compute the probabilities $P(w|D)$s by marginalizing them over a set of latent topics. Firstly, we estimate the topic distribution over the set of top ranked passages retrieved in the initial step by latent Dirichlet allocation (LDA) [5]. LDA outputs two distribution vectors θ (from document to topic) and ϕ (from topic to word). Modified smoothed document models are obtained by using these two distributions as shown in Equation 4, where K is the number of topics used in the LDA estimation.

$$P(w|D) = \sum_{k=1}^{K} P(w|z_k, \phi) P(z_k|D, \theta) \tag{4}$$

We then use the topic smoothed document models in the estimation of RLM i.e. we use the definition of $P(w|D)$ as obtained from Equation 4 in 2 to obtain an extended RLM sentence selection methodology which we name topical relevance model (TRLM) similarity.

$$P(w|q_i) \propto \sum_{j=1}^{R} \big(\sum_{k=1}^{K} P(w|z_k, \phi) P(z_k|D_j, \theta) \big) P(q_i|D_j) \tag{5}$$

3 Run Description

We submitted three official runs (run ids: 185, 186 and 187) for the INEX-2012 Tweet contextualization task. The first pass passage retrieval for each of the three runs is identical and follows the description of Sections 2.1, 2.2 and 2.3. The sentence retrieval strategies of each of these runs is different. Run 185 used simple language modelling (LM) similarity, run 186 used RLM similarity, whereas run 187 used TRLM similarity to score sentences. The number of top documents used for the (T)RLM estimation was set to 20. For TRLM, the additional parameter K, i.e. the number of topics, was set to 5. Our submissions did not use any automatic summarization techniques for sentence selection. We rather relied on pure IR-based approaches to generate the twweet contexts.

4 Evaluation

The tweet contexts were evaluated with two measures: a) *informativeness*, which measures the *closeness* of the answer string with a golden reference with the

Table 1. Official results for INEX-2012 Tweet contextualization task

Run Id	Run Description	Rank	Informativeness Metrics		
			Uni-gram	Bi-gram	Skip-gram
185	LM sentence retrieval	8	0.8265	0.9129	0.9135
186	RLM sentence retrieval	10	0.8347	0.9210	0.9208
187	TRLM sentence retrieval	11	0.8360	0.9235	0.9237
178	Official best	1	**0.7734**	**0.8616**	**0.8623**
194	Organizers' baseline	4	0.7864	0.8868	0.8887

Run Id	Run Description	Rank	Readability Metrics		
			Relevance	Syntax	Structure
185	LM sentence retrieval	1	**0.7728**	**0.7452**	**0.6446**
186	RLM sentence retrieval	5	0.7008	0.6676	0.5636
187	TRLM sentence retrieval	14	0.6093	0.5252	0.4847
194	Organizers' baseline	4	0.6975	0.6342	0.5703

help of KL divergence between the two; and b) *readability*, which measures the syntactic coherence of the text such as whether it has grammatical errors, has unresolved anaphora or is redundant etc [6].

Table 1 reports the official results of our three submitted runs. Along with our runs, the table shows the offcial best run as measured by informativeness and also the run submitted by the organizers as the baseline. Informativeness evaluation involves computation of three metrics: the KL divergence between the golden summary and the returned summary for uni-grams, bi-grams, and bi-grams with two allowable gaps in between [6]. Note that KL divergence being a distance measure implies that a lower value of this metric is indicative of a better result. The readability metric on the other hand reports the proportion of text which has correct syntax, structure and is relevant in the context. As a result, a higher value of these metrics indicates a better result.

It can be seen that the most simple sentence retrieval technique using LM similarity fairly well, achieving rank eight, as measured by informativeness. This run in fact achieves the best readability result.

Our other runs, i.e. the (T)RLM based sentence selection strategies, have not performed well in the official evaluation. The release of official relevance assessments namely the reference summary context for each tweet would enable us to tune the parameters of the two other sentence selection strategies in order to achieve an improved performance.

5 Conclusions and Future work

In our first participation at the INEX Tweet contextualization task, we applied sentence retrieval to construct answer fragments for each tweet. Three different

sentence selection methodologies were used: i) language modelling (LM) score, ii) relevance modelling (RLM) scoring of a sentence by accumulating over the RLM scores of its constituent terms, and iii) topical relevance modelling (TRLM) scoring of a sentence by accumulating over the topic smoothed RLM scores of its constituent terms.

The results confirm that simple IR-based sentence selection techniques can perform fairly well on both the informativeness and the readability metrics, without the application of any complex NLP techniques. The main advantage of the sentence retrieval methodologies is that these are very fast in contrast to computationally intensive NLP methods.

Acknowledgments

This research is supported by the Science Foundation Ireland (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (CNGL) project.

References

1. Voorhees, E.M.: Overview of the TREC 2003 question answering track. (2003) 54–68
2. Salton, G.: A Blueprint for Automatic Indexing. ACM SIGIR Forum **16**(2) (Fall 1981) 22–38
3. Hiemstra, D.: Using Language Models for Information Retrieval. PhD thesis, Center of Telematics and Information Technology, AE Enschede (2000)
4. Lavrenko, V., Croft, B.W.: Relevance based language models. In: Proceedings of the SIGIR '01, ACM (2001) 120–127
5. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3** (2003) 993–1022
6. SanJuan, E., Moriceau, V., Tannier, X., Bellot, P., Mothe, J.: Overview of the INEX 2011 Question Answering Track (QA@INEX). In: Pre-proceedings of the INitiative for the Evaluation of XML retrieval workshop (INEX 2011), Saarbrcken (Germany) (December 2011) 145–153

Testing a Statistical Word Stemmer based on Affixality Measurements in INEX 2012 Tweet Contextualization Track

Carlos-Francisco Méndez-Cruz[1], Edmundo-Pavel Soriano-Morales[1], and Alfonso Medina-Urrea[2]

[1] GIL-Instituto de Ingeniería UNAM, México
cmendezc@ii.unam.mx, sorianopavel@gmail.com
[2] El Colegio de México A.C., México
amedinau@colmex.mx

Abstract. This paper presents an experiment of statistical word stemming based on affixality measurements. These measurements quantify three characteristics of language. In this experiment we tested one strategy of stemming with three different sizes of training data. The developed stemmer was used by the automatic summarization system CORTEX to preprocess input texts and produce readable summaries. All summaries were evaluated as part of the INEX 2012 Tweet Contextualization Track. We present the results of evaluation and a discussion about our stemming strategy.

Key words: INEX, Automatic summarization system, Affixality Measurements, Morphological Segmentation, Statistical Stemming, CORTEX, Tweet Contextualization.

1 Introduction

The task proposed in the INEX 2012 Tweet Contextualization Track consists in obtaining some textual context from the English Wikipedia about the subject of a tweet. The final contextualization of the tweet should take the form of a readable summary of 500 words. An amount of 1133 documents, contextualized tweets with text from Wikipedia from November 2011, were processed in order to obtained summaries. Bibliographic references an empty Wikipedia pages were omitted.

The evaluation of summaries was done by the INEX organizers taking into account informativeness and readability. The former was obtained using Kullback-Leibler divergence with Dirichlet smoothing by comparing n-gram distributions. The latter was accomplished by the participants in the track; they evaluated the summaries taking into account syntax, anaphoric resolution and redundancy. More details of the system of evaluation and the INEX 2012 Tweet Contextualization Track could be found in [1].

For this track we developed a stemmer based on morphological segmentation. The stemmer was coupled with CORTEX, an automatic summarization system, in order to generate the summaries. We tested three sizes of training corpora to determine the best option for statistical stemming for English.

The organization of this paper is as follows: in Section 2 we review some approaches of morphological segmentation; in Section 3 we present word stemming; in Section 4 we describe the affixality measurements; Section 5 presents the stemming strategy; evaluation obtained in INEX track is expose in Section 6 and finally, in Section 7, we briefly present our conclusions and future work.

2 Morphological Segmentation

The first work for unsupervised discovery of morphological units of language is due to Zellig Harris [2]. His method, commonly known as *frequent successor*, consists in counting different letters or symbols before and after a possible morphological boundary. As more different symbols, the probability of a true morphological cut increases. This approach shown, among other things, that uncertainty is a well clue for morphological segmentation.

Now a day, one of the most utilized methods for unsupervised learning of morphology is based on Minimum Description Length (MDL) approach. This has been developed as a computational system called *Linguistica* [3, 4].[1] This method tries to obtain a lexicon of morphs inferred from a corpus. The best lexicon is the one that has the less redundancy, i.e. when the description length of the data is the lowest. Also, this utilizes some combinatorial structures called *signatures* in order to improve segmentation. This method has been employed for stemming work in [5]. In that paper the developed stemmer was utilized for an information retrieval task instead of summarization.

The mission of preprocessing documents for tasks of NLP, such as Question Answering, Information Retrieval or Automatic Text Summarization, in agglutinative languages is more complex. This is due to the fact that agglutinative languages have numerous combinations of morphs rather than a simple *prefix-stem-suffix* combination. A method of unsupervised morphological segmentation for these kinds of languages is called *Morfessor* [6–9]. [2] This approach uses MDL by Maximum a Posteriori framework. Also, it integrates a morphotactic analysis to represent each word by a Hidden Markov Model (HMM). We are not sure if this method has been used for word stemming.

3 Word stemming

The majority of NLP systems preprocesses documents in order to decrease the Vector Space Model representation. This is the case of CORTEX, which will be explained below. A well-known strategy for that purpose is word stemming, i.e.

[1] http://linguistica.uchicago.edu
[2] http://www.cis.hut.fi/projects/morpho/

truncating words by eliminating the inflection. Also, it is possible to remove derivational affixes.

The methods most widely used for word stemming are created by means of hand-made rules, like [10, 11]. These kinds of stemmers have been successfully applied for European languages. However, languages with more complex morphology than English, such as agglutinative ones, need unsupervised morphological strategies in order to deal with language complexity.

In [12] a review of stemming methods is presented. The variety of stemming approaches includes: distance function to measure an orthographical similarity [13], directed graphs [14, 5], and frequency of n-grams of letters [15]. Moreover, there are some works about stemming evaluation in information retrieval tasks, for example [16, 17].

4 Affixality Measurements

The affixality measurements used to morphological segmentation were proposed for Spanish in [18, 19]. These measurements have been also applied to Czech [20], and to the Amerindian Languages Chuj and Tarahumara [21]. This approach lies on the linguistic idea that there is a *force* between segments of a word (morphs) called affixality. If we can quantify this affixality, we can expect some peaks where morphological cuts are possible. In next sections we present the way to calculate these measurements.

4.1 Entropy

As we said above, Harris's approach revealed that uncertainty helps to morphological segmentation. This uncertainty could be seen as the Shannon's concept of information content (entropy) [22]. To calculate the entropy of a possible segmentation, given $a_{i,j}::b_{i,j}$ as a word segmentation, and $B_{i,j}$ as a set of all segments combined with $a_{i,j}$, we can used the formula:

$$H\left(a_{i,j} :: B_{i,j}\right) = -\sum p\left(b_{k,j}\right) \times log_2\left(p\left(b_{k,j}\right)\right) \tag{1}$$

where $k = 1, 2, 3, \ldots |B_{i,j}|$ and each $b_{k,j} \in B_{i,j}$. For our purpose we tested peaks of entropy from right to left in order to discover suffixes.

4.2 Economy Principle

The Economy Principle could be understood as follows: fewer units at one level of language are combined in order to create a great number of other units at the next level. Taking advantage of this principle, we can define a stem as a word segment that belong to a big set of relatively infrequent units, and affixes as word segments that belong to a small set of frequent ones. In [23] a quantification of this economy was suggested, however, we present a reformulation. Given a word

segmentation $a_{i,j}::b_{i,j}$, the economy of a segmentation is calculated depending on type of morph hypothesized:

$$K^p_{i,j} = 1 - \frac{|A_{i,j}| - |A^p_{i,j}|}{|B^s_{i,j}|}; \qquad K^s_{i,j} = 1 - \frac{|B_{i,j}| - |B^s_{i,j}|}{|A^p_{i,j}|} \tag{2}$$

where $A_{i,j}$ is the set of segments which alternate with $b_{i,j}$ ($a_{i,j} \in A_{i,j}$), and $B_{i,j}$ a set of segments which alternate with $a_{i,j}$ ($b_{i,j} \in B_{i,j}$). Also, let $A^p_{i,j}$ be the set of segments which are likely prefixes, and $B^s_{i,j}$ the set of segments which are likely suffixes.

4.3 Numbers of Squares

Joseph Greenberg [24] proposed the concept of square when four expressions of language, let say A, B, C, D, are combined to form AC, BC, AD, and BD. Hence, we set $c_{i,j}$ as a number of squares found in segment j of the word i.

5 Stemming Strategy

The affixality of all possible segmentations within a word is estimated by an average of normalized values of the three explained measurements:

$$AF^n(s_x) = \frac{c_x/\max c_i + k_x/\max k_i + h_x/\max h_i}{3} \tag{3}$$

To calculate this affixality, a training corpus of raw text is required. In this track we use three different sizes of 100k, 200k, and 500k word tokens. With an index of affixality calculated for each possible word segment, it is possible to choose a strategy for morphological segmentation; for example [19] propounded four strategies.

In this experiment we use a peak-valley strategy for segmentation. Given a set of affixality indexes inside a word af^k_i, let $af^k_{i-1} < af^k_i > af^k_{i+1}$ be a peak of affixality from left to right, where k is the length of the word plus one (the ending of the word). The main disadvantage of this approach is that small peaks are taking into account generating oversegmentation.

Regarding stemming, we truncate words at most left peak of affixality. For a language with scare morphology like English, we can imagine that a most right peak of affixality could be sufficient for stemming. However, in order to improve CORTEX summarization, we decide to strongly conflate words by a left-peak strategy. Next section explains CORTEX's approach.

5.1 CORTEX Summarizer

As we mentioned before, CORTEX is an automatic text summarizer system. A wide explanation of this summarizer could be found in [25–29]. Here, we briefly

describe some relevant aspects. First, CORTEX represents input documents in Vector Space Model. To do that, the documents should be preprocessed. Actually, we incorporate our stemmer in this step.

After preprocessing, a frequency matrix γ is generated representing the presence and absence of words (terms) in a sentence:

$$\gamma = \begin{bmatrix} \gamma_1^1 & \gamma_2^1 & \dots & \gamma_i^1 & \dots & \gamma_M^1 \\ \gamma_1^2 & \gamma_2^2 & \dots & \gamma_i^2 & \dots & \gamma_M^2 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \gamma_1^P & \gamma_2^P & \dots & \gamma_i^P & \dots & \gamma_M^P \end{bmatrix}, \quad \gamma_i^\mu \in \{0, 1, 2, \dots\} \tag{4}$$

each element γ_i^μ of this matrix represents the number of occurrences of the word$i$ in the sentence μ; $1 \leq i \leq M$ words, $1 \leq \mu \leq P$ sentences.

Then, statistical information is extracted from the matrix by calculating some metrics. More information about these metrics could be found in [30]. A summary of this metrics is offered here; they are based on frequencies, entropy, measures of Hamming and hybrid values.

1. **Frequency measures.**
   (a) Term Frequency: $F^\mu = \sum_{i=1}^{M} \gamma_i^\mu$
   (b) Interactivity of segments: $I^\mu = \sum_{\substack{i=1 \\ \xi_i^\mu \neq 0}}^{M} \sum_{\substack{j=1 \\ j \neq \mu}}^{P} \xi_i^j$
   (c) Sum of probability frequencies: $\Delta^\mu = \sum_{i=1}^{M} p_i \gamma_i^\mu$; p_i = word's i probability
2. **Entropy.** $E^\mu = -\sum_{\substack{i=1 \\ \xi_i^\mu \neq 0}}^{M} p_i \log_2 p_i$
3. **Measures of Hamming.** These metrics use a Hamming matrix H, a square matrix $M \times M$:

$$H_n^m = \sum_{j=1}^{P} \left\{ \begin{matrix} 1 & \text{if } \xi_m^j \neq \xi_n^j \\ 0 & \text{elsewhere} \end{matrix} \right\} \quad \text{for} \begin{matrix} m \in [2, M] \\ n \in [1, m] \end{matrix} \tag{5}$$

   (a) Hamming distances: $\Psi^\mu = \sum_{\substack{m=2 \\ \xi_m^\mu \neq 0}}^{M} \sum_{\substack{n=1 \\ \xi_n^\mu \neq 0}}^{m} H_n^m$
   (b) Hamming weight of segments: $\phi^\mu = \sum_{i=1}^{M} \xi_i^\mu$
   (c) Sum of Hamming weight of words per segment: $\Theta^\mu = \sum_{\substack{i=1 \\ \xi_i^\mu \neq 0}}^{M} \psi_i$; every word. $\psi_i = \sum_{\mu=1}^{P} \xi_i^\mu$
   (d) Hamming heavy weight: $\Pi^\mu = \phi^\mu \Theta^\mu$
   (e) Sum of Hamming weights of words by frequency: $\Omega^\mu = \sum_{i=1}^{M} \psi_i \gamma_i^\mu$
4. **Titles.** $\theta^\mu = \cos\left(\frac{\sum_{i=1}^{M} \gamma_i^\mu \text{Title}}{\|\gamma^\mu\| \|\text{Title}\|} \right)$

Finally, a decision algorithm combines those metrics to score sentences. Two averages are calculated, $\lambda_\mu > 0.5$, and $\lambda_\mu < 0.5$ ($\lambda_\mu = 0.5$ is ignored):

$$\sum^{\mu} \alpha = \sum_{\substack{\nu=1 \\ \|\lambda_\mu^\nu\| > 0.5}}^{\Gamma} \left(\|\lambda_\mu^\nu\| - 0.5 \right); \sum^{\mu} \beta = \sum_{\substack{\nu=1 \\ \|\lambda_\mu^\nu\| < 0.5}}^{\Gamma} \left(0.5 - \|\lambda_\mu^\nu\| \right) \tag{6}$$

The next expression is used to calculate the score of each sentence:

$$\text{If } \left(\sum^{\mu} \alpha > \sum^{\mu} \beta \right)$$
$$\text{then } \Lambda^{\mu} = 0.5 + \frac{\sum^{\mu} \alpha}{\Gamma} \text{ else } \Lambda^{\mu} = 0.5 - \frac{\sum^{\mu} \beta}{\Gamma}$$

CORTEX sorts final sentences by using $\Lambda^{\mu}; \mu = 1, \cdots, P$. Additionally, CORTEX let us delimit a compression rate, which was fixed at 500 words.

6 Experiments and Results

6.1 Design of Experiments

We made use of three sizes of training corpora, 100K, 200K, and 500K word tokens, to test our stemmer. With these sizes we performed the three runs for INEX track. The assigned numbers of runs were 153 (100K), 154 (200K), and 155 (500K). The corpus for evaluation was the 1133 contextualized tweets with text from Wikipedia from November 2011. About training corpora, we selected 24 documents from the same contextualized tweets.

6.2 Results

For informativeness, CORTEX, coupled with our stemmer, obtained rank 12, 14, and 15. Average scores of informativeness are shown in Table 1. The best run in this evaluation was run 154 (200K).

Table 1. Average scores of informativeness

Rank	Run	Unigrams	Bigrams	Skip
12	**154**	**0.8233**	**0.9254**	**0.9251**
14	155	0.8253	0.9280	0.9274
15	153	0.8266	0.9291	0.9290

Those scores were computed by organizers using a Perl script (inexqa-eval.pl); for details about this script check [1].

On the other hand, the best results for readability evaluation were obtained by run 155 (500K), see Table 2. Comparing our results with other runs, run 155 (500K) obtained rank 4 in relevance, rank 6 in syntax, and rank 9 in structure. The worst run in our experiment was the run 153 (100K) in both evaluations.

Table 2. Scores of readability

Run	Relevance	Syntax	Structure
155	**0.6968**	**0.6161**	**0.5315**
154	0.5352	0.5305	0.4748
153	0.4984	0.4576	0.3784

7 Conclusions and Future Work

In this paper we reported an experiment using a stemmer based on morphological segmentation. We used affixality measurements in order to segment words. This stemmer was coupled with CORTEX, an automatic summarization system.

We suggested the next stemming strategy: given some peaks of affixality of a word, we truncated at most left peak. Also, we tested three training corpus sizes to obtain statistical information for the affixality indexes: 100K, 200K, and 500K word tokens. Our two goals were to know if our stemming strategy can produce readable summaries, and if different sizes of training corpora can improve the CORTEX performance.

According to results of evaluation, our stemming strategy produces not only readable summaries but also competitive ones. That is, from an average of relevance, syntax, and structure (0.6148), run 155 obtained a rank 7 among 27 runs. What is more, concerning informativeness, run 154 obtained rank 12 among 33 participants.

Regarding corpus sizes, it is not clear what size is the best for English, between 200K and 500K word tokens. However, it is clear that increasing corpus size is a good strategy because 100K obtained the worst results. Additionally, a greater training corpus gives better position in the ranking, for example, from an average of relevance, syntax, and structure, run 155 (500K) obtained rank 7 and run 153 (100K) obtained rank 15.

In future experiments we will test different strategies for morphological segmentation and stemming. Additionally, we can test different stemming approaches, such as Porter's stemmer.

References

1. SanJuan, E., Moriceau, V., Tannier, X., Bellot, P., Mothe, J.: Overview of the INEX 2011 Question Answering Track (QA@INEX). In: INEX 2011 Workshop Pree-Proceedings, IR Publications, Hofgut Imsbach, Saarbrücken, Germany (2011) 145–153
2. Harris, Z.S.: From Phoneme to Morpheme. Language **31** (1955) 190–222
3. Goldsmith, J.: Unsupervised Learning of the Morphology of a Natural Language. Computational Linguistics **27** (2001) 153–198
4. Goldsmith, J.: An Algorithm for the Unsupervised Learning of Morphology. Natural Language Engineering **12** (2006) 353–371

5. Paik, J.H., Mitra, M., Parui, S.K., Jarvelin, K.: GRAS: An effective and efficient stemming algorithm for information retrieval. ACM Trans. Inf. Syst. **29** (2011)
6. Creutz, M., Lagus, K.: Unsupervised Discovery of Morphemes. In: Proc. of the Workshop on Morphological and Phonological Learning of ACL-02, Philadelphia, SIGPHON-ACL (2002) 21–30
7. Creutz, M.: Unsupervised segmentation of words using prior distributions of morph length and frequency. In Hinrichs, E., Roth, D., eds.: 41st Annual Meeting of the ACL, Sapporo, Japan. (2003) 280–287
8. Creutz, M., Lagus, K.: Induction of a Simple Morphology for Highly-Inflecting Languages. In: Proc. of 7th Meeting of the ACL Special Interest Group in Computational Phonology SIGPHON-ACL. (2004) 43–51
9. Creutz, M., Lagus, K.: Inducing the Morphological Lexicon of a Natural Language from Unannotated Text. In: Int. and Interdisciplinary Conf. on Adaptive Knowledge Representation and Reasoning (AKRR05). (2005) 106–113
10. Lovins, J.B.: Development of a Stemming Algorithm. Mechanical Translation and Computational Linguistics **11** (1968) 23–31
11. Porter, M.F.: An algorithm for Suffix Stripping. Program **14** (1980) 130–137
12. Lennon, M., Pierce, D., Tarry, B., Willet, P.: An evaluation of some conflation algorithms for information retrieval. J. of Information Science **3** (1981) 177–183
13. Majumder, P., Mitra, M., Pal, D.: Bulgarian, Hungarian and Czech stemming using YASS. In: Proceedings of Advances in Multilingual and Multimodal Information Retrieval, Springer-Verlag, Berlin (2008) 49–56
14. Bacchin, M., Ferro, N., Melucci, M.: A probabilistic model for stemmer generation. Mechanical Translation and Computational Linguistics **41** (2005) 121–137
15. McNamee, P., Mayfield, J.: Character n-gram tokenization for European language text retrieval. Information Retrieval **7** (2004) 73–97
16. Krovetz, R.: Viewing Morphology as an Inference Process. In: Proccedings of the 16th ACM/SICIR Conference. (1993) 191–202
17. Hull, D.A.: Stemming algorithms - A case study for detailed evaluation. Journal of the American Society for Information Science **47** (1996) 70–84
18. Medina-Urrea, A.: Investigación cuantitativa de afijos y clíticos del español de México. Glutinometría en el Corpus del Español Mexicano Contemporáneo. PhD thesis, El Colegio de México, México (2003)
19. Medina-Urrea, A.: Automatic Discovery of Affixes by means of Corpus: A Catalog of Spanish Affixes. Journal of Quantitative Linguistics **7** (2000) 97–114
20. Medina-Urrea, A., Hlaváčová, J.: Automatic Recognition of Czech Derivational Prefixes. In: Proceedings of CICLing 2005. Volume 3406., LNCS, Springer, Berlin/Heidelberg/New York (2005) 189–197
21. Medina-Urrea, A.: Affix Discovery based on Entropy and Economy Measurements. Texas Linguistics Society **10** (2008) 99–112
22. Shannon, C., Weaver, W.: The Mathematical Theory of Communication. University of Illinois Press, Urbana (1949)
23. de Kock, J., Bossaert, W.: Introducción a la lingüística automática en las lenguas románicas. Gredos, Madrid (1974)
24. Greenberg, J.H.: Essays in Linguistics. The Univ. of Chicago Press, Chicago (1957)
25. Torres-Moreno, J.M.: Résume automatique de documents. Lavoisier, Paris (2011)
26. Torres-Moreno, J.M., Saggion, H., da Cunha, I., SanJuan, E., Velázquez-Morales, P.: Summary Evaluation with and without References. Polibits **42** (2010) 13–19
27. Saggion, H., Torres-Moreno, J.M., da Cunha, I., SanJuan, E.: Multilingual summarization evaluation without human models. In: 23rd Int. Conf. on Computational Linguistics. COLING '10, Beijing, China, ACL (2010) 1059–1067

28. Torres-Moreno, J.M., Velazquez-Moralez, P., Meunier, J.: CORTEX, un algorithme pour la condensation automatique de textes. In: ARCo. Volume 2. (2005) 365
29. Torres-Moreno, J.M., Velazquez-Morales, P., Meunier, J.: Condensés de textes par des méthodes numériques. JADT **2** (2002) 723–734
30. Torres-Moreno, J.M., St-Onge, P.L., Gagnon, M., El-Bèze, M., Bellot, P.: Automatic Summarization System coupled with a Question-Answering System (QAAS). CoRR **abs/0905.2990** (2009)

INEX 2012 Benchmark
A semantic space for tweets contextualization

Mohamed Morchid and Georges Linares*

Univérsite d'Avignon, Laboratoire d'Informatique d'Avignon,
339 chemin des Meinajaries, Agroparc BP 1228, 84911 Avignon cedex 9, France
{mohamed.morchid,georges.linares}@univ-avignon.fr
http://www.lia.univ-avignon.fr

Abstract. In this paper, we present a method of tweet contextualization by using a semantic space to extend the tweet vocabulary. This method is evaluated on the tweet contextualization benchmark. Contextualization is build with the sentences from English Wikipedia. The context is obtained by querying a baseline system of summary. The query is made with words from a semantic space that is estimated via a latent dirichlet allocation (LDA) algorithm. Our experiment demonstrate the effectiveness of the proposal.

Keywords: LDA, tweet, contextualization, INEX, benchmark, 2012

1 Introduction

Microblogging, provided by several services as Twitter[1] or Jaiku[2], is a new phenomenon. This form of communication enables users to broadcast their daily activities or opinions. This new communication vector, describe Internet users status in short posts disseminated in the Web. Twitter is the most popular microblogging tool. This study deals with the tweet contextualization with Wikipedia sentences. This task met two main problems: The vocabulary style and size.
Note that it is difficult to contextualize a tweet, since on at following features: a tweet has few words and the vocabulary used is quit different that the vocabulary used in Wikipedia articles.
These difficulties increase with the Web size, the dispersion and the fragmentation of the Web information. We evaluate the proposed method in the INEX2012 benchmark [2].
Different aspects of Twitter have been studied recently, as a case study [4] or as compact swap highly reactive space which can extract some descriptors of opinions or public cares [5].
We propose an approach based on the mapping of source documents in a reduced semantic space in which some words could be found by a LDA analysis

* This work was funded by the ANR project SuMACC (ANR-10-CORD-007) in CONTINT 2010 program.
[1] http://www.twitter.com
[2] http://www.jaiku.com

[1]. Other approaches like LSI/LSA [6, 7] or [8] are based on statistical models that demonstrated their efficiency on various speech processing tasks. [9] uses the LSA (Latent Semantic Analysis) technique to extract the most relevant phrases from a spoken document. In [10], the authors apply LSA to an encyclopedic database for keyword extraction. We hope this method will permit to extend tweet vocabulary with others relevant words.
The remainder of the paper is organized as follows: the proposed approach is formulated in Section 2; the experimental protocol is described in Section 3; and concluding remarks are given in Section 4.

2 Tweet contextualization system

The tweet contextualization system can be decomposed as two steps. The first one is to build the query of a tweet, then, send this query to the summary system to receive the tweet context.
Concretely, the proposed method proceeds with 5 successive steps:

1. estimate off-line an LDA model on a large corpus of document D; this step produces a topic space T_{spc} of size $n^{T_{spc}}$ with a vocabulary $v^{T_{spc}}$
2. use Gibbs sampling to infer a topic distribution for a tweet t with T_{spc} to obtain a features vector V^z of the LDA classes distribution (each of these classes being implicitly associated to a topic)
3. map V^z and $v^{T_{spc}}$ to obtain a score $s(w)$ of popularity for each word w. Then, a subset S^w is composed with the words that have obtained the best score.
4. create a query q with the words of t and S^w
5. send q to the summary baseline system to receive the context c of t.

Fig. 1. Architecture of the tweet contextualization system

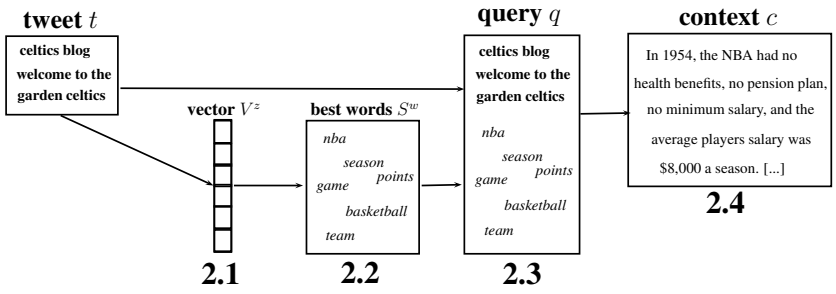


Figure 1 presents the tweet contextualization system. It can be decomposed as follows:

2.1 build a features vector V^z of a tweet by mapping t and T_{spc}
2.2 calculate the score of each word of $v^{T_{spc}}$ and extract a subset S^w of the words with best score
2.3 compose a query q with the words of t and S^w
2.4 send q to the baseline summary system and receive the context c the tweet t.

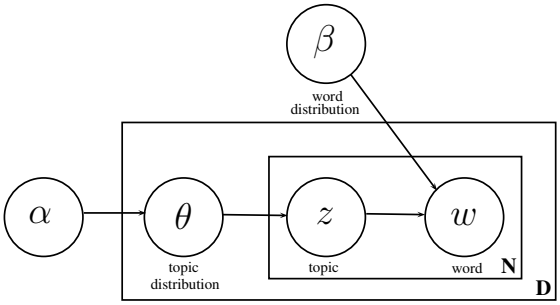
The next sections describe in-depth the main 4 parts of this process.

2.1 Features vector V^z

The Twitter language is quite unusual and sometimes constrained by the limit of the 140 characters. Using the conventional keywords, tweet query q can be affected by these features. We propose to pass through the semantic space T_{spc} from a LDA to increase the robustness of the method. Then, a features vectors V^z is calculated.The next sections describe this process.

Semantic space T_{spc}: LDA model considers a document (viewed as a *bag of words* [11]) as a probabilistic mixture of latent topics. These latent topics are characterized by a probability distribution of words associated with this topic. At the end of LDA analysis, we obtain n_{spc} classes with a set of its characteristic words and their emission probabilities.

Fig. 2. The LDA model



LDA formalism is described in Figure 2. To generate a word w in a document, a hidden topic z is sampled from a multinomial distribution defined by a vector θ of that document. Knowing z, the distribution over words is multinomial with parameters β. The parameter θ is drawn for all document from a common Dirichlet prior parameterized α. θ permit to tie the parameters between different documents. See [1] for more details.

In our experiments LDA is applied on a corpus D composed from Wikipedia articles (about 1GB). This set of documents represents about 1 billion words. A semantic space of 400 topics is obtained. This number of topics is set empirically. For each LDA class, we select the 20 words with the maximum weight.

After the estimate of the background topic model T^{spc}, we have to project the tweet in this semantic space and build a features vector V^z.

topic distribution V^z of t: We use Gibbs sampling to infer a topic distribution for the tweet t [12]. Then, a features vector V^z is obtained where the ith feature V_i^z $(i = 1, 2, \ldots, n^{T_{spc}})$ is the probability of the topic z_i knowing t:

$$V_i^z = P(z_i|t) \ . \tag{1}$$

2.2 Best words from vocabulary $v^{T_{spc}}$

This method allows a simple extraction of a subset S^w of the most representative words of the topic space vocabulary $v^{T_{spc}}$ knowing V^z. The system extracts $|S^w|$ (In our experiments, $|S^w| = 30$) words that obtain the highest score s. This score is the prior probability that a word can be generated by the tweet t:

$$s(w) = P(w|t) \tag{2}$$

$$= \sum_{i=1}^{n^{T_{spc}}} P(w|z_i)P(z_i|t) \tag{3}$$

$$= \sum_{i=1}^{n^{T_{spc}}} P(w|z_i)V_i^z \tag{4}$$

where $P(w|z_i)$ is the probability that the word w $(w \in v^{T_{spc}})$ was generated by the topic z_i. The score s is normalized by the highest that a word have obtained:

$$0 \leq s(w) \leq 1 \ . \tag{5}$$

Table 1 shows that the words of the tweet don't appears necessairly in S^w. That is what motivated this approach: find some others word to extend the tweet vocabulary. For example, the tweet (2) do not contain some relevant words like *army, war,muslim* or *islamic*.

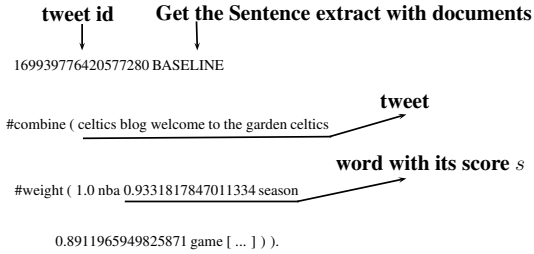
2.3 Query q

The subset S^w is used to compose the query q with the words of the tweet t. This query q is also send to the baseline XML-element retrieval system powered by Indri [13] to receive a context c of t.
The initial query is composed with the words of the tweet only. But tweets are

Table 1. Examples of tweets with the 10 words with the best score. On **bold** some interesting words that do not appear in the tweet vocabulary.

tweets	10 best words of S^w ($\|S^w\| = 30$)
celtics blog welcome to the garden celtics (1)	**nba** season game team points **basketball** games time year played
syrian troops attack residential areas in hama and homs (2)	**battle army** street forces troop troops **war muslim** men **islamic** city
bras for after breast implant surgery 3 tips (3)	blood **heart** surgery **pain** body pressure patient patients muscle **tissue**
did you know that 2012 is the international year of sustainable energy for all you can find out more at our (4)	development international **world environmental global** public human national **policy government**
wow childhood abuse disrupts brain formation study (5)	children **disorder mental** child **therapy syndrome treatment** disorders people symptoms

Fig. 3. Example of a query q



limited by their size of 140 words and by their vocabulary. For these reasons, we extend this Indri query with the words of S^w weighted by their score s as shows in Figure 1. Figure 3 shows the different element of a query q of a tweet t. q is composed by an *id*, *format* and a indri query. This query is the association of the tweet words and the S^w words weighted by their score s.

2.4 Context c

The query q is sent to the baseline XML-element retrieval system. The system return a context c. This context is build with the English Wikipedia sentences [2]. The index of the retrieval system covers all words (no stop list, no stemming) and all XML tags. We query this baseline system in batch mode using the perl APIs [3].

Example of a tweet context c:

tweet t: celtics blog welcome to the garden celtics.
context c: In later life, Cousy was Commissioner of the American Soccer League from 1974 to 1979, and he has been a color analyst on Celtics telecasts since the 1980s. Today, he is a marketing consultant for the Celtics, and occasionally makes broadcast appearances with Mike Gorman and ex-Celtic teammate Tom Heinsohn. In 1954, the NBA had no health benefits, no pension plan, no minimum salary, and the average players salary was $8,000 a season. [...] 147 Boston Celtics season was the 1st season of the Boston Celtics in the Basketball Association of America (BAA/ NBA).

3 Experiments and results

1,142 tweets [2] are used for this task. Each tweets have a *i*d and at most 140 words. The first step is to create a semantic space T^{spc} with LDA. LDA need a large corpus of documents. English Wikipedia articles form this corpus. Then, the topic space T^{spc} is composed with 400 topics of 20 words.

Table 2. Results of the run.

Unigramme	Bigramme	Skip	Relevance	Syntax	Structure
0.7909	0.8920	0.8938	0.6208	0.6115	0.5145

Table 2 presents the results of the INEX 2012 benchmark. The score of unigramme, bigramme and skip are evaluated by INEX 2012 organizers. These measures do not take into account readability. The readability is the measures of relevance, syntax and structure. These evaluations are estimated on the same pool of tweets.

[3] http://qa.termwatch.es/data

3.1 Conclusion

In this paper we present a method to extend tweet vocabulary. This method have been experimented in the INEX 2012 benchmark. To measure the effectiveness of our proposed method, we have to compare this results to the results of a run using just the tweet words.

Acknowledgements. We want to thinks Eric SanJuan for the baseline XML-element retrieval system.

References

1. D.M. Blei and A.Y. Ng and M.I. Jordan Latent dirichlet allocation The Journal of Machine Learning Research,3,993–1022,JMLR. org (2003)
2. Eric SanJuan, Véronique Moriceau, Xavier Tannier, Patrice Bellot and Josiane Mothe, Overview of the INEX 2012 Tweet Contextualization Track, Working Notes for the CLEF 2012 Workshop, Roma, Italy (to appear)
3. Clarke, F., Ekeland, I.: Nonlinear oscillations and boundary-value problems for Hamiltonian systems. Arch. Rat. Mech. Anal. 78, 315–333 (1982)
4. Z. Yang and J. Guo and K. Cai and J. Tang and J. Li and L. Zhang and Z. Su Understanding retweeting behaviors in social networks Proceedings of the 19th ACM international conference on Information and knowledge management,1633–1636, ACM(2010)
5. F. Larceneux Buzz et recommandations sur Internet: quels effets sur le box-office? Recherche et applications en marketing,45–64,JSTOR (2007)
6. S. Dumais Latent semantic indexing (LSI) and TREC-2 NIST SPECIAL PUBLICATION SP,105–105,NATIONAL INSTIUTE OF STANDARDS & TECHNOLOGY (1994)
7. J.R. Bellegarda A latent semantic analysis framework for large-span language modeling bookFifth European Conference on Speech Communication and Technology (1997)
8. T. Hofmann Probabilistic latent semantic indexing bookProceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval,50–57,ACM (1999)
9. J.R. Bellegarda Exploiting latent semantic information in statistical language modeling Proceedings of the IEEE,88,,8,1279–1296,,IEEE (2000)
10. Y. Suzuki and F. Fukumoto and Y. Sekiguchi Keyword extraction using term-domain interdependence for dictation of radio news bookProceedings of the 17th international conference on Computational linguistics-Volume 2,1272–1276,organization = "Association for Computational Linguistics (1998)
11. G. Salton Automatic text processing: the transformation Analysis and Retrieval of Information by Computer,S. l.]: Addison-Wesley Publishing Co (1989)
12. G. Casella and E.I. George Explaining the Gibbs sampler American Statistician,167–174,JSTOR (1992)
13. Strohman, T. and Metzler, D. and Turtle, H. and Croft, W.B. Indri: A language model-based search engine for complex queries Proceedings of the International Conference on Intelligent Analysis (2005)

Two Statistical Summarizers at INEX 2012 Tweet Contextualization Track

Juan-Manuel Torres-Moreno[1,2] and Patricia Velázquez-Morales[3]

[1] École Polytechnique de Montréal - Département de Génie Informatique
CP 6079 Succ. Centre Ville H3C 3A7 Montréal (Québec), Canada.
[2] Université d'Avignon et des Pays de Vaucluse
BP 911228, 84911 Avignon Cedex 9, France
juan-manuel.torres@univ-avignon.fr
[3] VM Labs, 84000 Avignon, France.
patricia_velazquez@yahoo.com
http://www.polymtl.ca

Abstract. According to the organizers, the objective of the 2012 INEX Tweet Contextualization Task is: "*...given a tweet, the system must provide some context about the subject of the tweet, in order to help the reader to understand it. This context should take the form of a readable (and short) summary, composed of passages from [...] Wikipedia.*" We present summarizers Cortex and KL-summ applied to the INEX 2012 task. Cortex summarizer uses several sentence selection metrics and an optimal decision module to score sentences from a document source. KL-summ is a new statistical summarizer based on Kullback-Leibler divergence (the same used by INEX organizers) to score sentences. The results show that Cortex system (using original tweets) outperforms KL-summ on INEX task.

Keywords: INEX, Automatic Summarization System, Tweet contextualization, Cortex, KL Divergence.

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 [2–5]. 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 [8]. CORTEX[4] is an automatic

[4] *CORTEX es Otro Resumidor de TEXtos* (CORTEX is anotheR TEXt summarizer).

summarization system [9] which combines several statistical methods with an optimal decision algorithm, to choose the most relevant sentences.

An open domain Question-Answering system (QA) has to precisely answer a question expressed in natural language. QA systems are confronted with a fine and difficult task because they are expected to supply specific information and not whole documents. At present there exists a strong demand for this kind of text processing systems on the Internet. A QA system comprises, *a priori*, the following stages [10]:

- 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 [11–13]. 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 [14–16]. In the systems described in [14], 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 [15], 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 [16] 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 2012 Tweet Contextualization Track. In Section 3 we explain the methodology of our work. Experimental settings and results obtained with Cortex and KL-summ summarizers are presented in Section 4. Section 5 exposes the conclusions of the paper and the future work.

2 INEX 2012 Tweet Contextualization Track

The Initiative for the Evaluation of XML Retrieval (INEX) is an established evaluation forum for XML information retrieval (IR) [17]. In 2012, tweet contextualization INEX task at CLEF 2012, aims "*given a new tweet, the system must provide some context about the subject of the tweet, in order to help the reader to understand it. This context should take the form of* **a readable summary, not exceeding 500 words, composed of passages from a provided Wikipedia corpus.**"[5]

[5] `https://inex.mmci.uni-saarland.de/tracks/qa/`

Like in Question Answering track of INEX 2011, the present task is about contextualizing tweets, i.e. answering questions of the form "What is this tweet about?" using a recent cleaned dump of the Wikipedia[6]. As organizers claim, the general process involves three steps:

- Tweet analysis.
- Passage and/or XML elements retrieval.
- Construction of the answer.

Then, a relevant passage segment contains:

- Relevant information but
- As few non-relevant information as possible (the result is specific to the question).

2.1 Document Collection

The corpus has been constructed from a dump of the English Wikipedia from November 2011. All notes and bibliographic references were removed to facilite the extraction of plain text answers. (Notes and bibliographic references are difficult to handle). Resulting documents contains a title, an abstract and section. Each section has a sub-title. Abstract end sections are made of paragraphs and each paragraph can have entities that refer to Wikipedia pages.

2.2 Tweets set

The committee of INEX has defined about 1000 tweets for the Track 2012. 1133 tweets in English were collected by the organizers from Twitter[7]. Tweets were selected and checked among informative accounts (for example, @CNN, @TennisTweets, @PeopleMag, @science...), in order to avoid purely personal tweets that could not be contextualized. Information such as the user name, tags or URLs will be provided.

3 The Text Summarizers used

3.1 Cortex Summarization System

Cortex [18, 19] 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 2012 Tweet Contextualization Track 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

[6] See the official INEX 2012 Tweet Contextualization Track Website: `https://inex.mmci.uni-saarland.de/tracks/qa/`.

[7] `www.tweeter.com`

to create a corpus composed of the query and the the multi-document retrieved by a Perl program supplied by INEX organizers[8]. This program is coupled to Indri system[9] 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 [20, 19] 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).
6. The sum of Hamming weigths 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 example, 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 query word set T and the sentence word set S.

$$\text{Overlap}(T, S) = \frac{card(S \cap T)}{card(T)} \tag{1}$$

The system scores each sentence with a decision algorithm that relies on the normalized metrics. Before combining the votes of the metrics, these are partitioned into two sets: one set contains every metric $\lambda^i > 0.5$, while the other set contains every metric $\lambda^i < 0.5$ (values equal to 0.5 are ignored). We then calculate two values α and β, which give the sum of distances (positive for α and negative for β) to the threshold 0.5 (the number of metrics is Γ, which is 11 in our experiment):

$$\alpha = \sum_{i=1}^{\Gamma} (\lambda^i - 0.5); \quad \lambda^i > 0.5 \tag{2}$$

$$\beta = \sum_{i=1}^{\Gamma} (0.5 - \lambda^i); \quad \lambda^i < 0.5 \tag{3}$$

The value given to each sentence s given a query q is calculated with:

[8] See: `http://qa.termwatch.es/data/getINEX2011corpus.pl.gz`

[9] 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/`

$$\begin{aligned} &\text{if}(\alpha > \beta) \\ &\quad \text{then} \;\; \text{Score}(s,q) = 0.5 + \tfrac{\alpha}{T} \\ &\quad \text{else} \;\; \text{Score}(s,q) = 0.5 - \tfrac{\beta}{T} \end{aligned} \tag{4}$$

The Cortex system is applied to each document of a topic and the summary is generated by concatenating higher score sentences.

3.2 The KL-summ summarization system

The main idea of KL-summarizer is to weight the sentences of a document, by minimizing the divergence of each sentence from document source. This idea is quite simple. Several divergence measures can be utilized: Jensen-Shannon, Kullback-Leibler, etc. However, in order to obtain a good summarizer on this specific task, we decide of implement the same measure of evaluation proposed by the INEX' organizers.

FRESA measure [21, 22] is similar to ROUGE evaluation [23] but it does not uses reference summaries. It calculates the divergence of probabilities between the candidate summary and the document source. Among these metrics, Kullback-Leibler (KL) and Jensen-Shannon (JS) divergences have been used [24, 21] to evaluate the informativeness of summaries. In this paper, we use FRESA, based in KL divergence with Dirichlet smoothing, like in the 2010, 2011 and 2012 INEX edition [25], to evaluate the informative content of summaries by comparing their n-gram distributions with those from source documents.

FRESA simply considered absolute log-diff between frequencies. Let T be the set of terms in the source. For every $t \in T$, we denote by C_t^T its occurrences in the source and by C_t^S its occurrences in the summary. The FRESA package computed the divergence between source and summaries as:

$$\mathcal{D}(T||S) = \sum_{t \in T} \left| \log\left(\frac{C_t^T}{|T|} + 1\right) - \log\left(\frac{C_t^S}{|S|} + 1\right) \right| \tag{5}$$

To score each sentence, several automatic measures were computed:

- FRESA_1: Uni-grams of single stems after removing stop-words.
- FRESA_2: Bi-grams of pairs of consecutive stems (in the same sentence).
- FRESA_{SU4}: Bi-grams with 2-gaps also made of pairs of consecutive stems but allowing the insertion between them of a maximum of two stems.

All FRESA scores, $\text{FRESA}_1 = 1 - \mathcal{D}(T||S)$ are normalized between 0 and 1. High values mean a less divergence of summary from source document. In other words, lower divergences (High Fresa scores) shows a more quantity of content of summary.

So, the relevant sentences will be selected as having the less divergence values. The two first modules are based on the Cortex system[10]. Finally, the third

[10] See section 3.1.

module generates summaries by displaying and concatenating of the relevant sentences.

At first, the first 50 documents of the cluster are concatenated into a single multi-document in chronological order. Placing the tweet q (enriched or not) like the title of this long document. The divergence between each sentence among the all others is computed using equation 5.

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)[11], in particular on the INEX 2012 Tweet Contextualization Track.

The strategy of Cortex and KL-summ systems to deal multi-document summary problem is quite simple: first, a long single document D is formed by concatenation of all $i = 1, ..., n$ relevant documents provided by Indri engine: $d_1, d_2, ...d_n$. The first line of this multi-document D is the tweet T. Both summarizers systems extract of D the most relevant sentences following T. Then, this subset of sentences is sorted by the date of documents d_i. The summarizers add sentences into the summary until the word limit is reached. To evaluate the performance of Cortex and KL-summ systems on INEX tweet contextualization track, we used the online package available from INEX website[12].

4.1 INEX Tweets enrichment

Two different strategies were employed to generate 1133 queries from tweets:

1. No pre-processing of tweet.
2. Enrichment of each tweet by semi-automatic synonyms of 100 heavy terms (their weights were calculated using the tf).

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.

2) Enrichment of tweet. The query has been semi-manually enriched as following. Firstly, a list T terms from the set of tweets was extracted then sorted by their term frequency. Liste T was manually inspected to extract the 100 first "relevant" terms. These 100 relevant terms were injected into tweets to enriched them.

Table 1 shows an example of the results obtained by Cortex and KL-summ systems using the 50 first documents retreived by Indri as input. The tweet that the summary should contextualiser in this case was the number 169231181181747200:

```
<topic id="169231181181747200">
<tweet>
```

[11] `https://inex.mmci.uni-saarland.de/`
[12] `http://qa.termwatch.es/data/`

```
CNNLive View stake-out camera funeral home WhitneyHouston body
expected arrive New Jersey Watch live
</tweet>
```

The tweets were enriched automatically using the 100 first terms (sorted by their tf weight) obtained from the words contained in the tweets set.

For example, for the tweet 169231181181747200, strategy 2 produce the following list of synonyms:

```
-- funeral : cremation
-- home : domicile
-- new : recent
-- watch : observe
-- live : exist
```

Then, query 169231181181747200 is enriched as show:

q = "`cnnlive view stakeout camera funeral cremation home domicile whitneyhouston body expected arrive new recent jersey watch observe live exist`"

Compound words are not detected in this phase. Since, "New Jersey" was separated in "New" and "Jersey", then "New" was enriched by "recent". This criterion is simple but it seems well enrich some tweets too shorts.

Table 1 presents Cortex and KL-summ results (queries enriched or not; tweet=167355997915058176 see Appendix) 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. In this particular example, we observe that KL-summ outperforms all summarizers.

Table 1. Example of Summarization results on tweet 167355997915058176.

Summary type	Uni-grams	Bi-grams	SU4 bi-grams	FRESA Averages
Baseline summary	33.36151	41.42226	41.40674	38.73017
Empty baseline	45.13673	53.58049	53.52617	50.74779
Random uni-grams	35.10965	43.58796	43.50809	40.73523
Random 5-grams	31.52959	39.73750	39.76517	37.01075
Cortex (query=Tweet)	32.16833	40.07052	40.14011	*37.45966*
Cortex (query=Tweet+Synonyms)	33.33823	41.26795	41.33727	38.64782
KL-summ (Query=Tweet)	31.79170	39.66283	39.74201	**37.06551**

Figure 1 shows the official results of participants of INEX 2012 contextualization task. The performances (rank) of our summarizers are: Cortex=11/33,

Cortex with enriched tweets=21/33 and KL-summ=16/33. In this figure, rank axes represents the SU4-bigrams values.

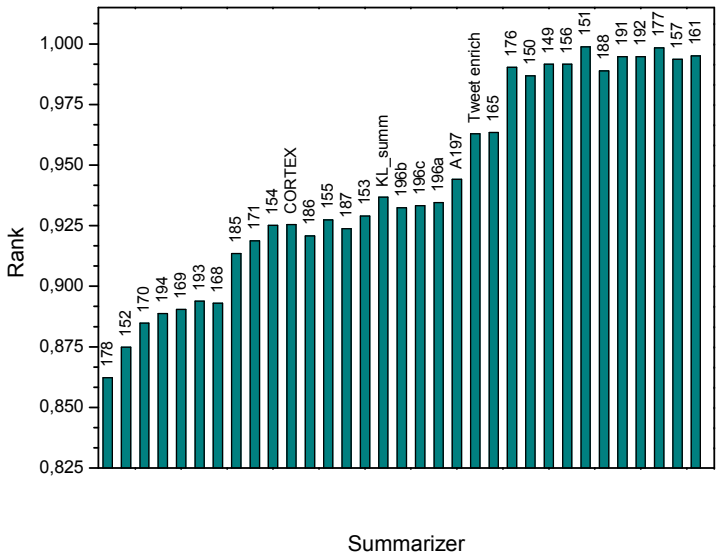


Fig. 1. Official results for systems participants on INEX 2012 contextualization task. Our systems are: CORTEX, KL-summ and CORTEX with enriched tweets.

5 Conclusions

In this paper we have presented the Cortex and KL-summ summarization systems applied on INEX 2012 Tweet Contextualization Track. The first one is based on the fusion process of several different sentence selection metrics. The decision algorithm obtains good scores on the INEX 2012 Tweet Contextualization Track (the decision process is a good strategy without training corpus). The second one is based on the divergence between summary and the source document.

Cortex summarizer using original tweets as inputs has obtained very good results in the automatic FRESA evaluations. In fact, semi-automatic tweet enrichment has disappointed in this task. Cortex using original tweets outperforms Cortex using enriched queries. We think that the strategy of enrichment, without compound words detection, was a very simple process. A module of compound words may improves the performance of this strategy. In other hands, the KL-summ summarizer is slightly less good than Cortex system. In fact, function 5 is

not additive. Since, a simple sort based on Fresa scores (1-divergence calculated in local form) is not enough to maximize the global score over the whole document. A more complex strategy of optimization must be implemented in order to deal this problem. We show that simple statistical summarizers show good performances in this complex task.

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. J.M. 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. I. Da Cunha, S. Fernandez, P. Velázquez Morales, J. Vivaldi, E. SanJuan, and J.M. Torres-Moreno. A new hybrid summarizer based on vector space model, statistical physics and linguistics. In *MICAI 2007: Advances in Artificial Intelligence*, pages 872–882. Springer Berlin/Heidelberg, 2007.
9. J.M. Torres-Moreno, P. Velazquez-Morales, and JG. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.

16. 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.
17. 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.
18. 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.
19. J.M. Torres-Moreno, P.L. St-Onge, M. Gagnon, M. El-Bèze, and P. Bellot. Automatic summarization system coupled with a question-answering system (qaas). *in CoRR*, abs/0905.2990, 2009.
20. J.M. Torres-Moreno, P. Velazquez-Morales, and J.G. Meunier. *Condensés de textes par des méthodes numériques. JADT*, 2:723–734, 2002.
21. J.-M. Torres-Moreno, H. Saggion, I. da Cunha, E. SanJuan, and P. Velázquez-Morales. Summary Evaluation with and without References. *Polibits*, 42(42):13–19, 2010.
22. H. Saggion, J.-M. Torres-Moreno, I. da Cunha, and E. SanJuan. Multilingual summarization evaluation without human models. In *23rd Int. Conf. on Computational Linguistics*, COLING '10, pages 1059–1067, Beijing, China, 2010. ACL.
23. C.-Y. Lin. Rouge: A Package for Automatic Evaluation of Summaries. In *Workshop on Text Summarization Branches Out*, 2004.
24. Annie Louis and Ani Nenkova. Automatically Evaluating Content Selection in Summarization without Human Models. In *Empirical Methods in Natural Language Processing*, pages 306–314, Singapore, August 2009.
25. Eric SanJuan, Patrice Bellot, Véronique Moriceau, and Xavier Tannier. Overview of the inex 2010 question answering track (qa@inex). In Shlomo Geva, Jaap Kamps, Ralf Schenkel, and Andrew Trotman, editors, *Comparative Evaluation of Focused Retrieval*, volume 6932 of *Lecture Notes in Computer Science*, pages 269–281. Springer Berlin / Heidelberg, 2011.

6 Appendix

We present the Cortex summary of tweet 167355997915058176: *"Scientists claim 42000-year-old paintings of seals by Neanderthals found in Spanish cave CIZy-Rotb"*. (Numbers in bold indicate the weight of each sentence).

0.861 Therefore, as human populations slowly increased, the cave bear faced a shrinking pool of suitable caves, and slowly faded away to extinction, as both Neanderthals and anatomically modern humans sought out caves as living quarters, depriving the cave bear of vital habitat. **0.880** The evidence found in archeological sites suggests these early humans were Nomadic, that they lived in caves, and acquired sustenance by hunting wild boar, red deer, mountain goat s, fallow deer, and horse s, competing with other predators like the leopard, the brown bear, and the wolf. **0.902** Late Oldowan/Early Acheulean humans such as "Homo ergaster/Homo erectus" may have been the first people to invent central campsites or home bases and incorporate them into their foraging and hunting strategies

like contemporary hunter-gatherers, possibly as early as 1.7 million years ago; however, the earliest solid evidence for the existence of home bases or central campsites among humans only dates back to 500,000 years ago. **0.859** Christopher Boehm Harvard university press Raymond C. Kelly speculates that the relative peacefulness of Middle and Upper Paleolithic societies resulted from a low population density, cooperative relationships between groups such as reciprocal exchange of commodities and collaboration on hunting expeditions, and because the invention of projectile weapons such as throwing spears provided less incentive for war, because they increased the damage done to the attacker and decreased the relative amount of territory attackers could gain. **1.000** In particular, Emil Bächler suggested that a bear cult was widespread among Middle Paleolithic Neanderthal s. A claim that evidence was found for Middle Paleolithic animal worship c 70,000 BCE originates from the Tsodilo Hills in the African Kalahari desert has been denied by the original investigators of the site. **0.865** However, recent archaeological research done by the anthropologist and archaeologist Steven Kuhn from the University of Arizona reveals that this gender-based division of labor did not exist prior to the Upper Paleolithic in Middle Paleolithic societies and was invented relatively recently in human prehistory. **0.871** One theory holds that behavioral modernity occurred as a sudden event some 50 kya in prehistory, possibly as a result of a major genetic mutation or as a result of a biological reorganization of the brain that led to the emergence of modern human natural language s. Proponents of this theory refer to this event as the "Great Leap Forward" "or the" "Upper Paleolithic Revolution". **0.941** Since genetics does not reject the hypothesis of a Neanderthal-modern admixture, and morphological and archaeological evidence suggest that Neanderthal lineages survived into later Upper Paleolithic populations, Pestera cu Oase findings provide a strong argument in favor of an admixture model between regional Neanderthals and early modern humans. **0.965** In another study, researchers have recently found in Pestera Muierilor, Romania, remains of European humans from years ago who possessed mostly diagnostic modern anatomical features, but "also" had distinct Neanderthal features not present in ancestral modern humans in Africa, including a large bulge at the back of the skull, a more prominent projection around the elbow joint, and a narrow socket at the shoulder joint. **0.910** The distribution of the D allele, high outside Africa but low in sub-Saharan Africa, has been suggested to indicate involvement of an archaic Eurasian population, and current estimates of the divergence time between modern humans and Neanderthals based on mitochondrial DNA are in favor of the Neanderthal lineage as the most likely archaic Homo population from which introgression into the modern human gene pool took place.

INEX Tweet Contextualization Track at CLEF 2012: Query Reformulation using Terminological Patterns and Automatic Summarization

Jorge Vivaldi and Iria da Cunha

Universitat Pompeu Fabra
Institut Universitari de Lingüística Aplicada
Barcelona
{iria.dacunha,jorge.vivaldi}@upf.edu
http://www.iula.upf.edu

Abstract. The tweet contextualization INEX task at CLEF 2012 consists of the developing of a system that, given a tweet, can provide some context about the subject of the tweet, in order to help the reader to understand it. This context should take the form of a readable summary, not exceeding 500 words, composed of passages from a provided Wikipedia corpus. Our general approach to get this objective is the following: we perform some automatic reformulations of the initial tweets provided for the task (obtaining a list of terms related with the main topic of all them using terminological patterns). Then, using these reformulated tweets, we obtain related documents with the search engine Indri. Finally, we use REG, an automatic extractive summarization system based on graphs, to summarize these documents and provide the summary associated to each tweet.

Key words: INEX, CLEF, Tweets, Terms, Named Entities, Wikipedia, Automatic Summarization, REG.

1 Introduction

The tweet contextualization INEX (Initiative for the Evaluation of XML Retrieval) task at CLEF 2012 (Conference and Labs of the Evaluation Forum) consists of the developing of a system that, given a tweet, can provide some context about the subject of the tweet, in order to help the reader to understand it. This context should take the form of a readable summary, not exceeding 500 words, composed of passages from a provided Wikipedia corpus. Like in the Question-Answering (QA) of INEX 2011, the task to be performed by the participating groups is contextualizing tweets, that is 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 elements retrieval

and construction of the answer. Relevant passages would be segments containing relevant information and also containing as little non-relevant information as possible (the result is specific to the question).

The test data are about 1000 tweets in English collected by the organizers of the task from Twitter. They were selected among informative accounts (for example, @CNN, @TennisTweets, @PeopleMag, @science...), in order to avoid purely personal tweets that could not be contextualized. Information such as the user name, tags or URLs is provided. The document collection for all the participants, that is the corpus, has been rebuilt based on a dump of the English Wikipedia from November 2011. Resulting documents are made of a title, an abstract and sections with sub-titles.

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" [1]. Summaries can be divided into "extracts", if they contain the most important sentences extracted from the original text (ex. [2], [3], [4], [5], [6], [7]), and "abstracts", if these sentences are re-written or paraphrased, generating a new text (ex. [8], [9], [10]). Most of the current automatic summarization systems are extractive.

Our general approach is the following: we perform some automatic reformulations of the initial queries provided for the task (obtaining a list of terms related with the main topic of all the tweets using terminological patterns). Then, using these reformulated queries, we obtain related documents with the search engine Indri[1]. Finally, we use REG ([11], [12]), an automatic extractive summarization system based on graphs, to summarize these documents and provide the final summary associated to each query.

This approach is similar to the one used at QA@INEX track 2010 (see [13]) and 2011 (see [14]), since the same summarization system is employed. Nevertheless, in our past participations, the system was semi-automatic, while in this work the system is totally automatic, from the reformulation of the queries using terminological patterns, until the multi-document summarization of all the retrieved documents.

The evaluation of the participant systems involves two aspects: informativeness and readability. Informativeness evaluation is automatic, using the automatic evaluation system FRESA (FRamework for Evaluating Summaries Automatically) ([15], [16], [17]), and readability evaluation is carried out manually (evaluating syntactic incoherence, unsolved anaphora, redundancy, etc.).

Following this introduction, the paper is organized as follows. In Section 2, the summarization system REG is shown. In Section 3, some information about terminology and terminological patterns is given. In Section 4, the methodol-

[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/`

ogy is explained. In Section 5, experimental settings and results are presented. Finally, in Section 6, conclusions are exposed.

2 State-of-the-art and Resources

2.1 Term Extraction

The notion of term that we have adopted in this work is based on the "Communicative Theory of Terminology" [18]: a term is a lexical unit (single/multiple word) that activates a specialized meaning in a thematically restricted domain. Terms detection implies the distinction between domain-specific terms and general vocabulary. Its results are useful for any NLP task containing a domain specific component such as: ontology and (terminological) dictionary building, text indexing, automatic translation and summarization systems, among others. In spite of its large application field, its reliable and practical recognition still constitutes a bottleneck for many applications.

As shown in [19], [20] and [21] among others, there are several methods to obtain the terms from a corpus. On the one hand, there are methods based on linguistic knowledge, like Ecode [22]. On the other hand, there are methods based on single statistical measures, such as ANA [23] or a combination of them, such as EXTERMINATOR [24]. Some tools combine both linguistic knowledge and statistically based methods, such as TermoStat [25], the algorithm shown in [26] or the bilingual extractors by [27] and [28]. However, none of these tools uses any kind of semantic knowledge. Notable exceptions are Metamap [29], Trucks [30] and YATE [31], among others. Also Wikipedia must be considered, since it is a very promising resource that is increasingly being used for both monolingual ([32], [33]) and multilingual term extraction [34].

Most of the tools, in particular those including an important linguistic component, takes into consideration the fact that terms usually follow a small number of POS patterns. In [35] it was shown that three patterns (noun, noun-adjective and noun-preposition-noun) cover more that 90% of the entries found in medical terminological dictionaries. Many of the above mentioned tools make some use of this fact. Nevertheless, some researchers like in [36] dynamically calculate the list of patterns found in terminological resources.

2.2 Named Entities Extraction

Named Entity Recognition (NER) may be defined as the task to identify names referring to persons, organizations and locations in free text; later this task has been expanded to obtain other entities like dates and numeric expressions. This task was originally introduced as possible types of fillers in Information Extraction systems at the 6th Message Understanding Conference [37]. Although initially this task was limited to identify such expressions, later it has been expanded to their labeling with one entity type label ("person", "organization", etc.). Note that an entity (such as "Stanford", the American university at the

U.S.) can be referenced using several surface forms (e.g., "Stanford University" and "Stanford") and a single surface form (e.g., "Stanford") can refer to several entities (the university but also an American financer, several places in the UK or a financial group). See [38] for an interesting review.

NER has proved to be a task useful for a number of NLP tasks as question answering, textual entailment and coreference resolution, among others. The recent interest in emerging areas like bioinformatics allows to expand this recognition task to proteins, drugs and chemical names. While early studies were mostly based on handcrafted rules, most recent ones use supervised machine learning as a way to automatically induce rule-based systems or sequence labeling algorithms starting from a collection of training examples.

Often, corpus processing tools include some text handling facilities to perform simple NER detection for facilitating later processing. Some of them are based in language specific peculiarities such as initial upper case letters together with some heuristics for name entities placed at the beginning of the sentence. This is the case of the tool used for this experiment (see a description in [39]).

2.3 The REG System

REG ([11], [12]) 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 a higher score are 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 [6], a system that carries out an unsupervised extraction of the relevant sentences of a document using several numerical measures and a decision algorithm.

3 Methodology

A main point in this research is to consider that named entities as well as words sequences that agree with the typical terminological patterns (see section 2.1) are representative of the tweets' topic. To test this assertion, we design a methodology to automatically retrieve all significant sequences from the tweets that satisfy the above mentioned criteria.

The first step is to POS tag the tweets file. As a matter of fact, and in order to keep the process fully automatic, a minimal manipulation of the tweets file has been done. It includes only a minor modification to allow the text handling tool to keep the tweet id connected to the tweet itself.

The next step, terminological patterns extraction, has been done using an already existent module of the YATE term extraction tool [31]. This information, together with the POS tagged tweet (to obtain proper nouns info) is used to build the query string for Indri.

Some care has been taken to keep track of multiword sequences as indicated by the Indri query language specification (see examples below).

In order to enrich the queries, we use a local installation of a Wikipedia dump[2] to expand the terms with redirection information from such Wikipedia info. In this way, a query term like "Falklands" may be searched in the Wikipedia to find that it can be also referenced as "Falkland Islands"; therefore, the final query term is rewritten as:

```
#syn(Falklands #1(Falkland Islands))
```

This strategy is also useful to find acronyms expansion as "USGS" and "United States Geological Survey" resulting in the following query:

```
#syn("USGS" #1(United States Geological Survey))
```

Moreover, it allows to find words with different spellings as:

```
#syn(#1(Christine de Pisan) #1(Christine de Pizan))
```

The resulting query has been delivered to Indri, using track organizer's script, to obtain the Wikipedia pages relevant to every query. The following is an example of a full tweet:

```
Increasingly, central banks, especially in emerging markets,
 have been the marginal  buyers of gold http://t.co/9mftD5ju
 via WSJ.
```

and its corresponding query string:

```
#1(marginal buyers of gold),#1(emerging markets),
#1(central banks),#syn("WSJ" #1(The Wall Street Journal))
```

The resulting set of Wikipedia pages has been split in several documents. Each document contains the pages relevant to the query. Such document is the input to the REG summarization system (see section 2.3), which builds a summary with the significant passages.

[2] This resource has been otained using [40].

4 Experiments Settings and Results

As mentioned in section 3, the process is fully automatic. No human intervention has taken place; therefore, errors and/or mistakes in the process may have a multiplicative effect. Most of such issues are exemplified as follows:

1. Tweet itself. The tweets file (including 1000 tweets) prepared by the organization includes several errors like: mispelling, joined words, foreign language, etc. Consider the following examples:
   - 169657757870456833: "Lakers now 17-12 on the season & 12-2 at home. @paugasol 20pts 13rebs 4blks. Bynum 15pts 15rebs. @0goudelock 10pts, two 3 PTers."
   - 169904294642978816: "@ranaoboy @Utcheychy @Jhpiego Thx for the #wiwchat RTs! Great conversation!"
   - 169655717538701312: "METTA. WORLD. PEACE."
   - 170175722449670145: "http://t.co/amQ6IShA"
   - 170207412366745600: "RT @MexicanProblms: #41. When you're eating junk food y tu mom te dice que no comas "chucherias." #MexicanProblems".

   Please note that, in some cases, it results in an empty query string or the resulting sentence is too short, causing POS tagging errors due to lack of context.
2. POS tagging. The output of most of the tools used for tagging (TreeTagger in this case) has some error rate. Unfortunately, errors mentioned above as well as extremely short sentences have a negative influence in the tagger performance.
3. Wikipedia expansion. It may happen that information added through Wikipedia expansion is not fully useful. This may be the case the only added information is the change of the case of some letters of the query term.
4. Indri query system. As shown in [41], this retrieval system has its own limits.
5. REG summarization system. The retrieval system issues a number of Wikipedia pages; therefore, it would be necessary to use a multidocument summarization system. As a matter of fact, REG is a single document summarizer, so some redundance may appear in the summaries.

Some of the above issues may cause unusual results in the terminological patterns extraction tool. Therefore, in such cases, the pages retrieved by Indri may not correspond to the information available in Wikipedia about tweets' topics.

The evaluation of all the participant systems in the tweet contextualization INEX task at CLEF 2012 involves two aspects: informativeness and readability. On the one hand, as mentioned, to evaluate the informativeness the automatic FRESA package is used. This evaluation framework 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 [42], FRESA supports different n-grams and skip n-grams probability distributions. 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.

Table 1 includes the official results of the informativeness evaluation in the the tweet contextualization INEX task at CLEF 2012. This table presents the scores of the 33 participant runs.

Table 1. Final results of informativeness in the tweet contextualization INEX task at CLEF 2012.

Rank	Run	Uni	Bi	Skip
1	178	0.7734	0.8616	0.8623
2	152	0.7827	0.8713	0.8748
3	170	0.7901	0.8825	0.8848
4	194	0.7864	0.8868	0.8887
5	169	0.7959	0.8881	0.8904
6	168	0.7972	0.8917	0.8930
7	193	0.7909	0.8920	0.8938
8	185	0.8265	0.9129	0.9135
9	171	0.8380	0.9168	0.9187
10	186	0.8347	0.9210	0.9208
11	187	0.8360	0.9235	0.9237
12	154	0.8233	0.9254	0.9251
13	162	0.8236	0.9257	0.9254
14	155	0.8253	0.9280	0.9274
15	153	0.8266	0.9291	0.9290
16	196b	0.8484	0.9294	0.9324
17	196c	0.8513	0.9305	0.9332
18	196a	0.8502	0.9316	0.9345
19	164	0.8249	0.9365	0.9368
20	197	0.8565	0.9415	0.9441
21	163	0.8664	0.9628	0.9629
22	165	0.8818	0.9630	0.9634
23	150	0.9052	0.9871	0.9868
24	188	0.9541	0.9882	0.9888
25	176	0.8684	0.9879	0.9903
26	149	0.9059	0.9916	0.9916
27	156	0.9366	0.9913	0.9916
28	157	0.9715	0.9931	0.9937
29	191	0.9590	0.9947	0.9947
30	192	0.9590	0.9947	0.9947
31	161	0.9757	0.9949	0.9950
32	177	0.9541	0.9981	0.9984
33	151	0.9223	0.9985	0.9988

As shown in Table 1, our run (165) obtains the position 22 in the rank. Exactly, it obtains 0.8818 using unigrams, 0.9630 using bigrams and 0.9634 using skip bigrams. The best run in the ranking (178) obtains 0.7734, 0.8616 and 0.8623, respectively.

On the other hand, readability is evaluated manually. Evaluators are asked to evaluate several aspects related to syntactic incoherence, unsolved anaphora, redundancy, etc. The specific orders given to evaluators are:

- Syntax S: "Tick the box is the passage contains a syntactic problem (bad segmentation for example)".
- Anaphora A: "Tick the box if the passage contains an unsolved anaphora".
- Redundancy R: "Tick the box if the passage contains a redundant information, i.e. an information that have already been given in a previous passage".
- Trash T: "Tick the box if the passage does not make any sense in its context (i.e. after reading the previous passages). These passages must then be considered as trashed, and readability of following passages must be assessed as if these passages were not present".

The score is the average normalized number of words in valid passages, and participants are ranked according to this score. Summary word numbers are normalized to 500 words each.

Table 2 includes the final results of readability evaluation in the tweet contextualization INEX task at CLEF 2012. Estimated average scores are available for:

- Relevance: proportion of text that makes sense in context.
- Syntax: proportion of text without syntax problems.
- Structure: proportion of text without broken anaphora and avoiding redundancy.

These measures were estimated on the same pool of tweets as for previously released informativeness evaluation by organizers.

Runs that failed to provide at least 6 consistent summaries in this pool have been kept apart because the estimates were too uncertain for inclusion in the official results. Because of this reason, in Table 2 only 27 runs are shown.

As shown in Table 2, our run (165) obtains the position 7 in the rank. Exactly, it obtains 0.5936 using unigrams, 0.6049 using bigrams and 0.5442 using skip bigrams. The best run in the ranking (185) obtains 0.7728, 0.7452 and 0.6446, respectively.

These results show that the performance of our system is not so good regarding informativeness, but it is much better regarding readability. This difference between informativeness and readability is also shown by other systems (see for example the best runs in both categories, 178 and 185). In our case, we consider that the mentioned mistakes in the tweets and the fact that the terminology extraction is totally automatic can cause that the pages retrieved by Indri are not as relevant as expected. Nevertheless, using an automatic summarization system, we can guarantee that the quality of readability is acceptable.

Table 2. Final results of readability in the tweet contextualization INEX task at CLEF 2012.

Run	Relevance	Syntax	Structure
185	0.7728	0.7452	0.6446
171	0.631	0.606	0.6076
168	0.6927	0.6723	0.5721
194	0.6975	0.6342	0.5703
186	0.7008	0.6676	0.5636
170	0.676	0.6529	0.5611
165	0.5936	0.6049	0.5442
152	0.5966	0.5793	0.5433
155	0.6968	0.6161	0.5315
178	0.6336	0.6087	0.5289
169	0.5369	0.5208	0.5181
193	0.6208	0.6115	0.5145
163	0.5597	0.555	0.4983
187	0.6093	0.5252	0.4847
154	0.5352	0.5305	0.4748
196b	0.4964	0.4705	0.4204
153	0.4984	0.4576	0.3784
164	0.4759	0.4317	0.3772
162	0.4582	0.4335	0.3726
197	0.5487	0.4264	0.3477
196c	0.449	0.4203	0.3441
196a	0.4911	0.3813	0.3134
176	0.2832	0.2623	0.2388
156	0.2933	0.2716	0.2278
188	0.1542	0.1542	0.1502
157	0.1017	0.1045	0.1045
161	0.0867	0.0723	0.0584

5 Conclusions and Future Work

In this paper, our strategy and results for the tweet contextualization INEX task at CLEF 2012 are presented. The task consists of the developing of a system that, given a tweet, can provide some context about the subject of the tweet, in order to help the reader to understand it. This context should take the form of a readable summary, not exceeding 500 words, composed of passages from a provided Wikipedia corpus. The test data are about 1000 tweets in English collected by the organizers of the task from Twitter.

Our system performs some automatic reformulations of the initial tweets provided for the task (obtaining a list of terms related with their main topic using terminological patterns). Then, using these reformulated tweets, we obtain related documents with the search engine Indri. Finally, we use REG to summarize these documents and provide the final summary associated to each tweet.

The results show that, comparing to the other participants, the performance of our system is not so good regarding informativeness (probably due to mistakes in the tweets and problems in the terminology extraction process), but it is much better regarding readability (probably due to the fact of using a summarization system).

In the future we plan to follow several parallel lines: i) to improve term selection and its expansion to refine the queries and therefore to improve the pertinence of the Wikipedia pages retrieved by Indri; ii) to further investigate the actual pertinence of the Wikipedia retrieved pages to the query; and iii) to check the actual weight of summarization process in the full task by testing other summarization systems.

References

1. Saggion, H.; Lapalme, G. (2002). *Generating Indicative-Informative Summaries with SumUM*. Computational Linguistics 28(4). 497-526.
2. Edmunson, H. P. (1969). *New Methods in Automatic Extraction*. Journal of the Association for Computing Machinery 16. 264-285.
3. Nanba, H.; Okumura, M. (2000). *Producing More Readable Extracts by Revising Them*. In Proceedings of the 18th Int. Conference on Computational Linguistics (COLING-2000). Saarbrucken. 1071-1075.
4. 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*. In Proceedings of the Human Language Technology Conference. San Diego. 189-193.
5. 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.
6. Torres-Moreno, J-M.; Velázquez-Morales, P.; Meunier, J. G. (2002). *Condensés de textes par des méthodes numériques*. In Proceedings of the 6th Int. Conference on the Statistical Analysis of Textual Data (JADT). St. Malo. 723-734.
7. da Cunha, I.; Fernández, S.; Velázquez, P.; Vivaldi, J.; SanJuan, E.; Torres-Moreno, J-M. (2007). *A new hybrid summarizer based on Vector Space Model, Statistical Physics and Linguistics*. Lecture Notes in Computer Science 4827. 872-882.
8. Ono, K.; Sumita, K.; Miike, S. (1994). *Abstract generation based on rhetorical structure extraction*. In Proceedings of the Int. Conference on Computational Linguistics. Kyoto. 344-348.
9. Paice, C. D. (1990). *Constructing literature abstracts by computer: Techniques and prospects*. Information Processing and Management 26. 171-186.
10. Radev, D. (1999). Language Reuse and Regeneration: Generating Natural Language Summaries from Multiple On-Line Sources. PhD Thesis. New York, Columbia University.
11. Torres-Moreno, J-M.; Ramírez, J. (2010). *REG : un algorithme glouton appliqué au résumé automatique de texte*. Proceedings of the 10th Int. Conference on the Statistical Analysis of Textual. Roma, Italia.
12. Torres-Moreno, J-M.; Ramírez, J.; da Cunha, I. (2010). *Un resumeur a base de graphes, indépendant de la langue*. In Proceedings of the Int. Workshop African HLT 2010. Djibouti.

13. Vivaldi, J.; da Cunha, I.; Ramírez, J. (2011). *The REG summarization system with question reformulation at QA@INEX track 2010*. In Geva, S. et al. (eds.). INEX 2010, Lecture Notes in Computer Science 6932. 295-302. Berlín: Springer.
14. Vivaldi, J.; da Cunha, I. (2012). *QA@INEX Track 2011: Question Expansion and Reformulation Using the REG Summarization System*. Lecture Notes in Computer Science (LNCS) 7424. 257-268. Berlin: Springer.
15. 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 Int. Conference on Computational Linguistics (COLING 2010). Pekin.
16. Torres-Moreno, J-M.; Saggion, H.; da Cunha, I.; SanJuan, E.; Velázquez-Morales, P. (2010). *Summary Evaluation With and Without References*. Polibitis: Research journal on Computer science and computer engineering with applications 42.
17. Torres-Moreno, J-M.; Saggion, H.; da Cunha, I.; Velázquez-Morales, P.; SanJuan, E. (2010). *Ealuation automatique de résumés avec et sans référence*. In Proceedings of the 17e Conférence sur le Traitement Automatique des Langues Naturelles (TALN). Montreal: Univ. de Montréal et Ecole Polytechnique de Montréal.
18. Cabré, M. T. (1999). *La terminología. Representación y comunicación*. Barcelona: IULA.
19. Cabré, M. T.; Estopà, R.; Vivaldi, J. (2001). *Automatic term detection. A review of current systems*. Recent Advances in Computational Terminology 2. 53-87.
20. Pazienza, M. T.; Pennacchiotti, M.; Zanzotto, F.M. (2005). *Terminology Extraction: An Analysis of Linguistic and Statistical Approaches*. Studies in Fuzziness and Soft Computing 185. 255-279.
21. Ahrenberg, L. (2009). *Term Extraction: A Review*. (Unpublished draft).
22. Alarcón, R.; Sierra, G.; Bach, C. (2008). *ECODE: A Pattern Based Approach for Definitional Knowledge Extraction*. In Proceedings of the XIII EURALEX Int. Congress. Barcelona: IULA, UPF, Documenta Universitaria. 923-928.
23. Enguehard, C.; Pantera, L. (1994). *Automatic Natural Acquisition of a Terminology*. Journal of Quantitative Linguistics 2(1). 27-32.
24. Patry, A.; Langlais, P. (2005). *Corpus-based terminology extraction*. In Proceedings of 7th Int. Conference on Terminology and Knowledge Engineering. Copenhagen.
25. Drouin, P. (2002). *Acquisition automatique des termes: l'utilisation des pivots lexicaux spécialisés*. Ph.D. Thesis. Montreal (Canada): Université de Montréal.
26. Frantzi, K. T.; Ananiadou, S.; Tsujii, J. (2009). Erdmann, M.; Nakayama, K.; Hara, T.; Nishio, S. (2009). *The C-value/NC-value Method of Automatic Recognition for Multi-word Terms*. Lecture Notes in Computer Science 1513. 585-604
27. Vintar, S. (2010). *Bilingual term recognition revisited: The bag-of-equivalents term alignment approach and its evaluation*. Terminology 16(2). 141-158.
28. Gómez Guinovart, X. (2012). *A Hybrid Corpus-Based Approach to Bilingual Terminology Extraction*. In I. Moskowich and B. Crespo (eds.). Encoding the Past, Decoding The Future: Corpora in the 21st Century. Cambridge Scholar Publishing: Newcastle upon Tyne. 147-175.
29. Aronson, A.; Lang, F. (2010). *An overview of MetaMap: historical perspective* and recent advances. Journal of the American Medical Informatics Association 17(3). 229-236.
30. Maynard, D. (1999). *Term Recognition Using Combined Knowledge Sources*. Ph.D. Thesis. Manchester Metropolitan University. Manchester (UK).
31. Vivaldi, J. (2001). *Extracción de candidatos a término mediante combinación de estrategias heterogéneas*. Ph.D. thesis. Universitat Politècnica de Catalunya. Barcelona (Spain).

32. Vivaldi, J.; Rodríguez, H. (2010). *Using Wikipedia for term extraction in the biomedical domain: first experiences*. Procesamiento del Lenguaje Natural 45. 251-254.
33. Cabrera-Diego, L.; Sierra, G.; Vivaldi, J.; Pozzi, M. (2011). *Using Wikipedia to Validate Term Candidates for the Mexican Basic Scientific Vocabulary*. In Proceedings of LaRC 2011: First Int. Conference on Terminology, Languages, and Content Resources. Seoul. 76-85.
34. Erdmann, M.; Nakayama, K.; Hara, T.; Nishio, S. (2009). *Improving the Extraction of Bilingual Terminology from Wikipedia*. ACM Transactions on Multimedia Computing, Communications and Applications 5(4). 31.1-31.16.
35. Estopà, R. (1999). *Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary*. Ph.D. Thesis. Pompeu Fabra University. Barcelona (Spain).
36. Nazar, R.; Cabré, M. T. (2012). *Supervised Learning Algorithms Applied to Terminoloy Extraction*. In 10th Terminology and Knowledge Engineering Conference.
37. Grishman, R.; Sundheim, B. (1996). *Message Understanding Conference - 6: A Brief History*. In Proceedings of the 16th Int. Conference on Computational Linguistics. 466-471.
38. Nadeau, D.; Sekine, S. (2007). *A survey of named entity recognition and classification*. Journal of Linguisticae Investigationes 30(1). 3-26.
39. Martínez, H.; Vivaldi, J.; Villegas, M. (2010). *Text handling as a Web Service for the IULA processing pipeline*. In Proceedings of the 7th conference on International Language Resources and Evaluation (LREC'10). 22-29.
40. Zesch, T.; Müller, C., Gurevych, I. (2008). *Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary*. In 6th LREC Conference Proceedings. 1646-1652.
41. Strohman, Trevor; Metzler, Donald; Turtle, Howard; Croft, Bruce (2005). *Indri: A language-model based search engine for complex queries*. University of Massachusetts Amherst. CIIR Technical Report IR-407.
42. Lin, C-Y. (2004). *ROUGE: A Package for Automatic Evaluation of Summaries*. In Proceedings of Text Summarization Branches Out: ACL-04 Workshop. 74-81.

Author Index