Proceedings of the SIGIR 2008 Workshop on Focused Retrieval

Held in Singapore,

24 July 2008.

Edited by
Andrew Trotman,
Shlomo Geva,
and
Jaap Kamps.

Proceedings of the
SIGIR 2008 Workshop on Focused Retrieval.
Held in Singapore,
24 July 2008.

Published by:
Department of Computer Science,
University of Otago,
PO Box 56,
Dunedin,
New Zealand.

Editors:
Andrew Trotman
Shlomo Geva
and
Jaap Kamps

Paper Edition: ISBN 978-0-473-13468-6
Electronic Edition: ISBN 978-0-473-13469-3

http://www.cs.otago.ac.nz/sigirfocus2008/

Preface

These proceedings contain the papers of the SIGIR 2008 Workshop on Focused Retrieval held in Singapore on 24th July 2008. Nine papers were selected by the program committee from fourteen submissions (64% acceptance rate). Each paper was reviewed by two members of the program committee.

When reading this volume it is necessary to keep in mind that these papers represent the opinions of the authors (who are trying to stimulate debate). It is the combination of these papers and the debate that is will make the workshop a success.

We would like to thank the ACM and SIGIR for hosing the workshop. Thanks also go to the program committee, the paper authors, and the participants, for without these people there would be no workshop.

Andrew Trotman
Shlomo Geva
Jaap Kamps

Workshop Organization

Programme Chairs

Andrew Trotman
Shlomo Geva
Jaap Kamps

Programme Committee

James Allan
Charles Clarke
Norbert Fuhr
Shlomo Geva
Jaap Kamps
Gabriella Kazai
Alistair Knott
Mounia Lalmas
Maarten de Rijke
Andrew Trotman
Liu Wenyin

Table of Contents

Focused Retrieval over Richly-Annotated Collections

Matthew W. Bilotti, Le Zhao, Jamie Callan and Eric Nyberg
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, Pennsylvania 15235
{ mbilotti, lezhao, callan, ehn }@cs.cmu.edu

ABSTRACT

This paper introduces a theoretical framework for focused retrieval, based on a formalism called the annotation graph. Annotation graph-based retrieval provides a rich retrieval representation that directly supports query-time constraint-checking of arbitrary relations. This representation can support focused retrieval tasks, such as Question Answering systems, which often have information needs containing constraint types that can not be queried easily under many retrieval models. The problem of annotation graph-based retrieval is mapped onto existing XML element retrieval functionality in the Indri search engine. The remainder of the paper serves to identify and discuss the issues that emerged and illustrate by example what in our opinion constitutes the upcoming research challenges facing the focused retrieval community.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval

General Terms

Theory, Design

Keywords

Focused retrieval, question answering, type systems, annotation graphs

1. INTRODUCTION

The fundamental difference between focused retrieval tasks such as Question Answering (QA), XML Element Retrieval (XML-IR) and passage retrieval and the general *ad hoc* retrieval task is that the unit of retrieval is smaller than that of a document. The document retrieval paradigm forces users to read through the retrieved documents to locate the information that satisfies their information needs. Focused retrieval paradigms aim to ease this burden on the user by locating and retrieving the relevant information directly.

Text retrieval systems score by measuring the similarity between some representation of the text and a query, which is expressed in that same representation and encodes the constraints in the user's information need. Modern document retrieval systems use a lean text representation that makes indexing and retrieval convenient, but that is not capable of fully encoding all information needs, or even most of them. All they can represent is a notion of relevance based on ordering, proximity and frequency of terms. Though this representation can approximate many information needs well enough to be successful on the *ad hoc* retrieval task, it is not as well suited to the focused retrieval task.

The job of a focused retrieval system is significantly more difficult. Because it must assign scores to small units of text, it can not afford to ignore any of the constraints in the information need. Focused retrieval systems aim to provide a rich internal representation, both for the query and for the indexed texts, that reduces the mismatch between the information need and the query by directly supporting constraint-checking of more of the component constraints of the information need.

This paper introduces a formalism called the *annotation graph*, and a theoretical retrieval framework based on annotation graph retrieval. The annotation graph-based retrieval framework supports query-time constraint-checking of relations over information elements encoded in the annotation graph. We argue that this kind of rich information representation and powerful retrieval-time constraint-checking is necessary to support focused retrieval tasks, such as QA. This paper chronicles our experience mapping the annotation graph-based retrieval task onto existing XML element retrieval functionality in the Indri search engine. We conclude with a discussion of the issues that emerged, and a set of examples that illustrate what are, in our opinion, important research challenges in focused retrieval.

SIGIR 2008 Workshop on Focused Retrieval
July 24, 2008, Singapore
The copyright of this article remains with the authors.

2. ANNOTATION GRAPH-BASED RETRIEVAL FRAMEWORK

This section presents a theory of focused retrieval based on a formalism called the annotation graph, which serves as a shared representation for not only information needs, but also the information content of texts.

Type System T	$T = \left(Te = \left\{te_1, te_2, ..., te_{\lvert Te\rvert}\right\}, Tr = \left\{tr_1, tr_2, ..., tr_{\lvert Tr\rvert}\right\}\right)$
Information Element Type te_i	$te_i = (name, parent) \wedge (parent \in Te \vee parent = \emptyset)$
Relation Type tr_i	$tr_i = (name, domain, range) \wedge domain, range \in Te$
Annotation Graph G	$G = \left(E = \left\{e_1, e_2, ..., e_{\lvert E\rvert}\right\}, R = \left\{r_1, r_2, ..., r_{\lvert R\rvert}\right\}, ts\right)$
Information Element e_i	$e_i = (type) \wedge type \in Te \wedge ts = (Te, Tr)$
Relation r_i	$r_i = (type, e_d, e_r) \wedge type = (name, domain, range) \in Tr \wedge$ $domain, range \in Te \wedge ts = (Te, Tr) \wedge e_d = (domain) \wedge e_r = (range)$
Collection C	$C = \left\{g_1, g_2, ..., g_{\lvert C\rvert}\right\}$
Query q	$q = f(g, C)$

Figure 1: Formal definitions for the theoretical retrieval framework based on annotation graphs

Type System

$$T_{BoW} = \left(\begin{array}{l} Te = \left\{ \begin{array}{l} (document, \emptyset), \\ (sentence, \emptyset), \\ (word, \emptyset), \\ (John, \emptyset), \\ (loves, \emptyset), \\ (Mary, \emptyset) \end{array} \right\}, \\ Tr = \left\{ \begin{array}{l} (encloses, document, sentence), \\ (encloses, document, word), \\ (encloses, sentence, word) \end{array} \right\} \end{array} \right)$$

Annotation Graph

$$G_{BoW} = \left(\begin{array}{l} E = \left\{ \begin{array}{l} d = (document), \\ s = (sentence), \\ j = (John), \\ l = (loves), \\ m = (Mary) \end{array} \right\}, \\ R = \left\{ \begin{array}{l} e_s = (encloses, d, s) \\ e_{j1} = (encloses, d, j) \\ e_{l1} = (encloses, d, l) \\ e_{m1} = (encloses, d, m) \\ e_{j2} = (encloses, s, j) \\ e_{l2} = (encloses, s, l) \\ e_{m2} = (encloses, s, m) \end{array} \right\}, \\ ts = T_{BoW} \end{array} \right)$$

Figure 2: Example bag-of-words type system and annotation graph for the sentence, *John loves Mary*

2.1 Type Systems

Each annotation graph is defined with respect to a particular type system, which serves as a vocabulary for the types of information elements that can exist in the graph, and the relations that can be defined among those elements. A *type system* T is defined as a tuple consisting of two sets, Te and Tr, which contain all valid types for information elements and relations, respectively. An *information element type* te_i is defined as a name and an optional parent type pointer. A *relation type* tr_i is defined as a name, a domain type and a range type. Both the domain and range types must be defined in the same type system. See Figure 1 for the formal definitions of type systems, information element types and relation types.

2.2 Annotation Graphs

A piece of information of any type and complexity can be represented as an annotation graph consisting of a set of information elements (vertices) and a set of relations (edges) that hold between pairs of information elements. Although it is often convenient to think of the annotation graph as a representation of marked-up text, the formalism is general enough to represent audio, images, or any other type of data that can be viewed as a discrete set of elements with relations defined over them. An *annotation graph* is defined as a tuple consisting of a set of information elements E, a set of relations R and a pointer to the graph's type system ts.

Information element is a general term used to describe not only discrete units of raw content, such as the tokens in a text document or the notes in a musical composition, but also annotations representing higher-level or more complex content obtained through human or automatic analysis and mark-up of the raw content. An *information element* is defined with a single attribute, the name of a type defined within the type system of which the element is an instance.

Relations between pairs of elements are typed and asymmetric, holding between an information element of the domain type and an information element of the range type as declared in the type system. Many relations are overt in the raw content, for example, adjacency and ordering information between tokens in a text document or pixels in an image. Other types of relations come from the annotation process that adds higher-level information element types, such as syntactic information for a text document or musical phrase structure for audio data. See Figure 1 for the formal definitions of annotation graphs, information elements and relations.

An example of a simple bag-of-words type system and an annotation graph for the sentence, *John loves Mary*, can be found in Figure 2. The type system can represent words enclosed by sentences, which are, in turn, enclosed by documents. Words can also be directly enclosed by documents,

$$q\left(g=(E,R,ts),C\right)=$$
$$\frac{1}{|K|}\sum_{i=1}^{|K|}\begin{cases}\frac{tf(k_i,g)}{df(k_i,C)}, (encloses, document, k_i)\in R\\ 0, otherwise\end{cases}$$
$$K=\{k_1,k_2,...,k_{|K|}\}$$
$$tf(k_i,g)=\sum_{i=1}^{|R|}\begin{cases}1, r_i=(encloses, document, k_i)\\ 0, otherwise\end{cases}$$
$$df(k_i,C)=\sum_{i=1}^{|C|}\begin{cases}1, (encloses, document, k_i)\in R_{g_i}\\ 0, otherwise\end{cases}$$

Figure 3: Bag-of-words retrieval as an instance of the annotation graph-based retrieval framework

which provides for not only transitivity of enclosure, but also the case in which words, such as section titles, occur outside the boundaries of sentences identified in the document. Note that the type system can not represent ordering and proximity constraints between pairs of words, and that as such, it is not powerful enough to distinguish between the sentences *John loves Mary* and *Mary loves John*.

2.3 The Retrieval Process

Let a *collection* be defined as a set of annotation graphs sharing the same type system, each of which serves as a representation of a retrievable item that can be scored in response to a query. In this framework, a *query* is defined as an arbitrary function over an annotation graph and the collection. The query can represent any kind of operation, including but not limited to set operations, such as intersection and union; score combination techniques, such as sum or average; probabilistic operations such as Noisy-OR and not; and weighting schemes, such as *tf.idf* and language models. Query evaluation is a recursive procedure in which queries are broken down into sub-queries, the results of which are then combined by the query function. The queries at the leaves of the tree implement the matching of individual constraints against the annotation graph.

2.4 Example: Bag-of-Words Retrieval

This section describes an example instantiation of the above-proposed retrieval framework. Consider a simple bag-of-words retrieval system that scores the annotation graph g corresponding to a document based on whether the document encloses each k_i of a set of specific keyterms K. One possible implementation is shown in Figure 3. The query q scores the annotation graph g by an arithmetic average over the set of keyterms K, such that the value for each k_i is set to a weight if g contains an *encloses* relation between the document and k_i, and zero otherwise. In this case, the weight is a *tf.idf*-style weight, but any other weighting scheme could be used, or Boolean retrieval could be implemented by setting the weight to 1.

2.5 A Note about Implementation

The retrieval framework proposed in this section relies on a graphical representation, which, while convenient from a formal perspective, might seem to be inefficient both computationally at query-time and also in terms of the size of the index footprint on disk. Without getting into implementation-specific details, it bears emphasizing that this theoretical framework is a view of the retrieval problem meant to aid in understanding the process. The discussion of annotation graph-based retrieval should not be taken as an argument for implementing retrieval systems as full-blown graph similarity engines. The actual index structures in a real implementation could be thought of as a compilation or distillation of the annotation graph representation, and could be tuned to minimize space on disk and maximize the efficiency of query evaluation algorithms operating over them.

3. APPLICATION TO THE QA TASK

Question Answering (QA) is a focused retrieval task that aims to retrieve snippets of text satisfying certain constraints. It can be considered similar to a passage retrieval task, except that the passage size is smaller [2]. The constraint in a QA task is that the passages must contain answers to an input question. QA systems are often implemented as a cascade of document retrieval and an optional passage re-ranking step, followed by answer extraction. The core of a QA system can be thought of as focused retrieval application, bookended by language understanding tools, with question analysis at the beginning and answer selection, validation and presentation at the end.

All QA systems perform some kind of linguistic and semantic analysis on the input question as an initial step to determine how to proceed. The result of this analysis constitutes a specification for an answer to the question, phrased in terms of linguistic and semantic constraints that must hold over a piece of text for it to contain an answer to the question. This rich representation of the information need becomes the input to the focused retrieval task at the core of a QA system, and also, along with the retrieved results, to the answer selection, validation and presentation tasks at the end of the QA process.

Many QA systems, however, rely on a text retrieval component that can not handle this representation directly. These systems are forced to map their information needs into the query representation supported by their text retrieval components, potentially weakening the constraints. Recall the bag-of-words retrieval example introduced earlier. What if a QA system, trying to answer the question, *Who does John love?*, formulated a bag-of-words query consisting of the keyterms *John* and *love*. Under the bag-of-words retrieval model, pieces of text containing those two keyterms would be retrieved, but it is difficult to distinguish between relevant text, such as *John loves Mary* and non-relevant text, such as *Mary loves John*. To filter out these false positives, QA systems often perform on-the-fly linguistic and semantic analysis of the retrieved text, comparing it against the information need and discarding those results that do not satisfy the constraints.

Recently, there has been interest in building a text retrieval interface for QA applications that provides a richer query representation that can directly support retrieval-time checking of certain types of information need constraints [1]. Suppose that a *love* event is a primitive element in the semantics of a QA system to which the same question was asked, *Who does John love?* Provided the collection was properly annotated off-line for instances of *love* events, a text retrieval component capable of query-time constraint-checking would retrieve *John loves Mary*, but *Mary loves John* would not match the query, because *John* is not the actor of the *love* event.

Retrieval-time constraint-checking can also apply to QA

[Arg0 *John*] [Target *loves*] [Arg1 *Mary*]

$$g = \left(\begin{array}{l} E = \left\{ \begin{array}{l} s = (sentence), \\ t = (target), \\ a0 = (arg0), \\ a1 = (arg1), \\ l = (loves), \\ j = (John), \\ m = (Mary) \end{array} \right\}, \\ R = \left\{ \begin{array}{l} e_t = (encloses, s, t), \\ e_{a0} = (encloses, s, a0), \\ e_{a1} = (encloses, s, a1), \\ e_l = (encloses, s, l), \\ e_j = (encloses, s, j), \\ e_m = (encloses, s, m), \\ e_{l2} = (encloses, t, l), \\ e_{j2} = (encloses, a0, j), \\ e_{m2} = (encloses, a1, m), \\ a_{a0} = (child, t, a0), \\ a_{a1} = (child, t, a1) \end{array} \right\}, \\ ts = T_{SRL} \end{array} \right)$$

Figure 4: PropBank-style semantic analysis of the sentence, ***John loves Mary*****, with the corresponding annotation graph**

systems that do not use a full-blown semantic representation internally. Consider a QA system that uses grammatical functions such as *subject*, *object* and *oblique* that can be obtained without semantic analysis. For the question, *Who does John love?*, an answer constraint would require that *John* be the subject of the verb *love*. The grammatical function-based representation is sufficient to distinguish between the relevant and non-relevant text in this case.

These examples can be related back to the annotation graph-based retrieval framework outlined in this paper. The internal representation for information needs used by a QA system is equivalent to a type system. As illustrated, the choice of a type system can range from a simple representation derived from syntax to a full-blown semantic representation, in addition to token-based representations widely supported by text retrieval technology. Analysis of the text in the collection would yield annotation graphs containing relations such as $(loves, John, Mary)$, or $(subject, John, loves)$ and $(object, Mary, loves)$, which could be checked at query time. Query functions could implement operators that can combine or weight individual constraints, as well as account for partial matches, such as $(adores, John, Mary)$.

4. IMPLEMENTATION

This section describes features of the freely-available Indri search engine [5], a part of the Lemur toolkit,[1] used to support annotation graph-based retrieval. The current version of Indri provides support for indexing and retrieval of arbitrary, hierarchical, overlapping fields that was originally added to address the needs of an XML-IR task [4].

[1]See: `http://www.lemurproject.org`

```
#combine[sentence]( john loves )

#combine[sentence]( #max( #combine[target]( loves
  #max( #combine[./arg0]( john )))))
```

Figure 5: Bag-of-words (upper) and PropBank-style (lower) Indri queries for the question, *Who does John love?*

Indri's existing fielded retrieval support can be thought of as an implementation of annotation graph retrieval for a subset of type systems, those under which at most one relation type can be defined to hold over any pair of element types. This relation is implemented in the Indri index as field enclosure; if a field instance of the domain type encloses a field instance of the range type, it is said that the relation holds between the elements corresponding to those fields. At retrieval time, these relations can be checked using query operators that enforce enclosure constraints between fields.

For an XML-IR task, the type system would include the elements found in an XML document, but for a QA task, it is the linguistic and semantic annotations on text that the system uses to locate answers that need to be indexed so that constraints can be checked at retrieval time. One example of a type system for QA is the one used in [1], which supports verb predicate-argument structures with semantic role labels in the style of PropBank [3], as well as common named entity types. Some QA systems use this semantic representation internally throughout the system, annotating text retrieved by a bag-of-words retrieval system on-the-fly and comparing it against an analysis of the question to locate answers. The goal of supporting constraint-checking against this representation at the retrieval stage of the QA process is to reduce the occurrence of false positives, or text that scores well on a keyterm match, but that does not unify with the QA system's expectations for an answer.

Figure 4 shows the example sentence *John loves Mary* with its PropBank-style semantic analysis and the equivalent annotation graph. The root of a verb predicate-argument structure is identified as the Target. In the figure, the bracketed arguments are labeled as Arg0 and Arg1, which correspond to the agent or doer of the action, and the patient or the person for whom the action is done, respectively. Though all argument roles are verb-dependent, users of PropBank have found that Arg0 and Arg1 can be relied upon to have been consistently labeled across verbs.

Figure 4 introduces a new type of relation called *child* that relates the target verb to its arguments. Because the verb does not actually enclose its arguments, Indri supports an additional representation for relations between field instances. In [1], an extension that has subsequently been integrated into the main trunk was made allowing Indri to store an arbitrary pointer to another field instance, called the *parent* field, in the posting for a particular field instance. Representing a relation type in the index in this way is called a *parent-child* relation, as opposed to an *enclosure* relation.

When Indri indexes a corpus annotated off-line with PropBank-style predicate-argument structures, target verbs are represented as field instances of type Target in the index. There are also separate field types for each type of argument,

s_1	John loves Mary.	[Arg0 [Person *John*]] [Target *loves*] [Arg1 [Person *Mary*]]
s_2	John loves Mary.	[Arg0 [Person *John*]] [Target *loves*] [Arg1 *Mary*]
s_3	John loves Mary.	[Arg0 [Person *John*]] [Target *loves*] [Arg2 [Person *Mary*]]
s_4	John says he loves Mary.	[Arg0 [Person *John*]] [Target *says*] [Arg1 [Arg0 *he*] [Target *loves*] [Arg1 [Person *Mary*]]]
s_5	John adores Mary.	[Arg0 [Person *John*]] [Target *adores*] [Arg1 [Person *Mary*]]
s_6	Bill loves Jane.	[Arg0 [Person *Bill*]] [Target *loves*] [Arg1 [Person *Jane*]]
s_7	John gave Jane's book to Mary.	[Arg0 *John*] [Target *gave*] [Arg1 [Person *Jane's*] *book*] [to [Person *Mary*]]

Figure 6: Text collection referred to by the examples in Section 5. The verb predicate-argument structure with PropBank-style semantic role labels and named entity recognition is shown below each sentence.

including numbered complement arguments such as Arg0, Arg1 and Arg2, as well as adjunctive arguments such as Argm-Tmp and Argm-Loc, which represent temporal and locative modifiers, respectively. Arguments are related to their respective targets by use of the parent-child strategy, and enclosure relationships exist between sentences and target verbs, sentences and arguments, sentences and named entities, and arguments and any nested fields, which can include nested named entities as well as targets and arguments.

At query time, Indri provides two different pieces of query syntax to support checking of enclosure and parent-child constraints. Figure 5 shows two queries that might be formulated for the question, *Who does John love?* In the bag-of-words (upper) query, the `#combine[sentence]` operator is used to enforce that the terms *john* and *loves* must be enclosed by the same Sentence field instance. In the PropBank-style (lower) query, the nested `#combine` operators tell Indri to look for a Sentence enclosing a Target enclosing *loves*. In `#combine[./arg0]`, the dot-slash syntax is used to require that the target verb have a child Arg0 field instance enclosing *john*. Throughout, the `#max` operators are used to select the best field instance in the event that more than one match.

Both enclosure and parent-child constraints map to structured query operators in Indri's underlying inference network retrieval model [6]. The query operators restrict keyterm matches to occurring field instances of the specified type. When scoring an field instance, the score contribution of specific keyterm is the number of occurrences of that keyterm over the size of the field instance, smoothed by linear interpolation with a background model based on the document containing the field extent, and also with another model based on the collection as a whole. If the field instance contains no occurrences of the requested keyterm, its score defaults to the background score made up of the document and collection smoothing components.

5. CHALLENGES

The process of building and testing Indri's support for annotation graph-based retrieval revealed some non-trivial issues inherent in implementing this kind of retrieval solution. This section asks the questions as to why the obvious approach of mapping the task of retrieval for QA onto a field-based XML-IR approach did not work well, and what are the requirements a retrieval engine would have to satisfy to be able to support annotation graph-based retrieval. The narrative in this section is centered around a series of examples that illustrate the emergent issues and motivate the discussion. For convenience, our examples involve matching sentences in the sample text collection shown in Figure 6.

5.1 Partial Matching of Structures

Indri's current constraint-checking implementation penalizes missing constraints harshly. Consider the query shown in Figure 7, which describes a *love* event between *John* and some other person. Sentence s_1 is a complete match for this query, and is ranked first. If a requested argument role does not appear in a predicate-argument structure, a background score, which can be quite low, is combined into the overall score.

```
#combine[sentence](
  #max( #combine[target]( loves
   #max( #combine[./arg0](
    #max( #combine[person]( john ))))
   #max( #combine[./arg1]( #any:person )))))
```

Figure 7: This query requests *love* events in which *John* is the agent, and any person is the patient.

Linguistic and semantic analysis tools occasionally make mistakes, sometimes as a result of legitimate ambiguities in the language, such as prepositional phrase attachment, and other times because of a lack of coverage in a rule base or in a training data set. In sentence s_2, the named entity recognition tool failed to identify *Mary* as a person, giving the sentence a structure similar to that of, *John loves his dog.* When scoring s_2, the enclosed Person field instance is not found, so the score defaults to a background score made up of document-specific and collection-wide scores for *Mary* occurring inside field instances of type Person. In the current implementation, it is not possible to directly smooth with the enclosing ARG1 instance.

Sentence s_3 represents a role-labeling error in which the argument corresponding to *Mary* is labeled ARG2, which generally represents a recipient or beneficiary, as opposed to the correct ARG1, which indicates the patient. There is, in fact, no prescription for an ARG2 in the *love* frame in PropBank, which means that the training data for the semantic role labeling tool does not contain this example in its entirety, but errors like this can happen when the semantic role labeling process is decomposed into argument identification, attachment and labeling as separate steps to maximize use of training data. As with sentence s_2, the missing field instance causes a background score to be combined into the overall score. The current implementation will only score field instances of the requested type; it is not able to propose the ARG2 as a match for the `#combine[./arg1]` query clause with some discount factor despite the fact that the ARG2 field instances satisfies the `#any:person` constraint.

As partial matches, sentences s_2 and s_3 are ranked behind s_1. The relative order in which sentences s_2 and s_3 are ranked depends primarily on the document models used for smoothing; in this case, if the documents contain more occurrences of *Mary* tagged as a PERSON than as an ARG2, then s_2 would come first.

5.2 Combining Evidence from Partial Matches

Sentence s_4 is semantically similar, but not equivalent, to sentence s_1. Setting aside the issue of whether the source, in this case *John*, is to be believed when he asserts that he loves Mary, this sentence is clearly relevant to a QA system faced with determining an answer to *Does John love Mary?* or *Who does John love?*

Sentence s_4 is a partial match for the query shown in Figure 7 because it contains two distinct predicate-argument structures, and the query's constraints are distributed between the two structures. The outer structure satisfies the constraint that *John* occur inside a PERSON nested inside an ARG0, and the inner structure satisfies the constraint on the target verb and the attached ARG1 containing any field instance of type PERSON.

The query uses a `#max` operator to score a sentence based on the single best-matching predicate-argument structure it contains, because the current implementation has no way to aggregate belief across multiple structures. Therefore, s_4 is scored on the basis of the inner predicate-argument structure, because it satisfies more of the constraints. In fact, s_4 would get the same score even if it were *Jack said that he loves Mary*, because the `#max` operator hides the effect on the score corresponding to the query's *John* constraint.

5.2.1 Balancing Structural Constraints

The query shown in Figure 7 contains three unweighted constraints. Intuition would suggest that the person or QA system formulating the query intended that the constraints be equally important. The current Indri implementation of Jelinek-Mercer smoothing leads to an interesting phenomenon where s_5, which is a relevant partial match, is ranked behind s_6, which is not relevant at all, but satisfies the target verb constraint. This ranking suggests that, for some reason, Indri is considering the target verb constraint much more important than the other constraints in the query.

The reason for this behavior is that the vast majority of target verb field instances are of length one. The score contribution of the target verb constraint is computed by taking the smoothed count of the number of occurrences of the keyterm divided by the length of the field instance being scored. This means that if the target verb does match, a very high belief is combined into the overall score, and if the target verb does not match, the portion of the score corresponding to the target field instance is zero, resulting in a very low background score based on smoothing with the document and the collection. As a result of this scoring method, the target verb mismatch on s_6 pushes it below s_5, which has mismatches on two constraints.

It may be that certain smoothing methods that are appropriate to fielded retrieval in general may not be optimized for tasks in which assumptions can be made about the nature of the fields. A recent proposal to address this problem is two-level Dirichlet smoothing, an extension of Indri's existing Dirichlet smoothing method to include a smoothing component for the document. This method is less sensitive to the length of the field instance being scored, so the variance of the scores produced by the query operator corresponding to the target verb constraint is reduced. This results in a more sensible ranking that relaxes all constraints on partial matches simultaneously after all partial matches have been retrieved. Under two-level Dirichlet smoothing, sentence s_5 is correctly ranked ahead of sentence s_6.

5.3 Multiple Potentially Relevant Structures

In process of the question analysis, a QA system uses deep linguistic and semantic processing to build a fairly rich representation of the answer it is looking for. Sometimes, a system is able to posit multiple structures that can potentially contain answers. The queries shown in Figures 8, 9 and 10 attempt to retrieve more of the relevant sentences in a single pass.

5.3.1 Combining over Keyterms

The query shown in Figure 8 maintains the same predicate-argument structure as the one shown in Figure 7, but uses a controlled synonymy to specify target verb alternatives that are semantically related. The `#combine[target]` operator is intended to be able to match sentences such as s_1 and s_5 by being flexible about the target verb constraint.

```
#combine[sentence](
 #max( #combine[target]( loves adores
  #max( #combine[./arg0](
   #max( #combine[person]( john ))))
  #max( #combine[./arg1]( #any:person )))))
```

Figure 8: This query is the same as the one shown in Figure 7, except that it requests *adore* events in addition to *love* events.

The implementation of the `#combine` operator essentially performs an arithmetic average in log space over the two score components, each of which is computed as the smoothed count of matching term occurrences over the length of the field instance. As written, the query clause prefers *loves* and *adores* to any other verbs, but it also prefers both verbs to just one of them alone. Knowing how the text collection was annotated, it would be impossible for a TARGET field instance to contain both of those terms. In fact, the only times when a TARGET is longer than length one is the case referred to as the phrasal verb, where a verb and a particle occur in the TARGET together. The query clause will not perform as intended because every TARGET field instance that matches one of the verbs will have a background score combined in from the other verb that does not match.

One potential mitigation for this phenomenon is to wrap the alternate keyterms in a `#syn` operator, which treats occurrences of each term equivalently. One side effect of this choice of operator is that the smoothing values are skewed, particularly for `#syn` operators with large numbers of arguments. Document frequency is computed as the union of the arguments of the `#syn` clause, which can have an affect on the ranking. Another operator choice is `#or`, which implements the probabilistic Noisy-OR. Rankings are a bit easier to understand with `#or`, but it does not really capture the intent of query. Some kind of exclusive-OR operator may be more appropriate, but the question as to how to build such an operator for this task is still open.

5.3.2 Combining over Full Structures

Figure 9 shows a query that wraps an outer `#combine` operator around two full predicate-argument structures in an attempt to match both sentences s_1 and s_4. This query will not perform as expected, because the scores coming out of the two inner `#combine[sentence]` clauses are not, in general, directly comparable. The variance in the document scores for a particular query operator is inversely related to the number of arguments that operator has. The more complex constraints translate to query operators with more arguments, which provide scores on a smaller scale that vary over a more narrow range than do the scores corresponding to simpler constraints.

```
#combine[sentence]( #max(
 #combine[sentence](
  #max( #combine[target](
   #max( #combine[./arg0](
    #max( #combine[person]( john ))))
   #max( #combine[./arg1](
    #max( #combine[target]( loves
     #max( #combine[./arg1]( #any:person )))))))))
 #combine[sentence](
  #max( #combine[target](
   #max( #combine[./arg0](
    #max( #combine[person]( john ))))
   #max( #combine[./arg1]( #any:person )))))))
```

Figure 9: This query requests two potentially relevant structures, the simpler structure contained in sentence s_1, below, and the nested version shown in sentence s_4, above.

This type of query can be difficult to reason about. Consider sentence s_1, which is a complete match for the second `#combine[sentence]` clause in the query. It is also a partial match for the first such clause in the query, as it satisfies the ARG0 constraint on *John*. Although s_1 has an ARG1, that field instance does not enclose any TARGET instances, and so a background score will be combined into the overall score. Even though smoothing with the document will yield at least one match for *loves* inside a TARGET field instance, the background score is low enough in comparison to the matching scores produced by the query clauses corresponding to the constraints that are satisfied to significantly affect the overall score.

5.3.3 Combining over Partial Structures

Given the difficulty in scoring disjunctions over full structures, not to mention the linguistic and semantic analysis task inherent in positing those structures in the first place, a QA system might decide to try specifying a single structure, but allow variations to mitigate annotation error, improve recall or simply to be able to control the relaxation of the constraints. It seems natural to take this approach, because if one constraint fails to match, a background score could be avoided if there is an alternate path for belief in the query.

```
#combine[sentence](
 #max( #combine[target]( loves
  #max( #combine[./arg0](
   #max( #combine[person]( john ))))
  #max( #combine[./arg1]( #any:person ))
  #max( #combine[./arg2]( #any:person )))))
```

Figure 10: This query is the same as the one shown in Figure 7, except that it tries to compensate for annotation error by allowing the patient of the *love* event to occur in the ARG1 or ARG2 positions.

Figure 10 shows an example of a query that matches *love* events in which *John* is the agent, having another PERSON in an argument labeled ARG1 or ARG2. As written, however, the query does not express the system's intent. Because the ARG1 and ARG2 constraints are written in two separate `#max` clauses that are children of the `#combine[target]` clause, this query will rank any sentence satisfying both constraints, such as sentence s_7, ahead of any sentence satisfying only one of them, such as sentences s_1 and s_3.

The current implementation of Indri makes it difficult to encode the notion of alternative constraints. The background score coming from the non-matching query branch will drag the overall score down. An alternative query formulation would but the ARG1 and ARG2 constraints into a single `#max` clause within the `#combine[target]` clause. The `#max` operator has the benefit that the low-scoring query branch is pruned, but it must be used with caution. The query operators inside a `#max` operator must produce scores that are comparable; otherwise, one element may consistently win out in a way that does not necessarily reflect the quality of the match, but instead, an artifact of the scoring model. One way to address this would be to build in the notion of a discount factor, allowing the user to not only specify precedence over the alternatives in a `#max` operator, but also to potentially compensate for a mismatch in the scale of the scores produced by one of the elements of the `#max`.

6. CONTRIBUTIONS

This paper proposed a theory of focused retrieval based on a formalism called the annotation graph. We motivated the potential of the approach using a Question Answering example, arguing that the annotation graph representation addresses the requirements of focused retrieval applications by providing for rich retrieval-time constraint-checking. We tested Indri's support for annotation graph-based retrieval, and discovered that the obvious approach of mapping the problem onto existing XML-IR machinery presented a number of challenges. The identification of these challenges, along with the discussion of specific examples, serves to bring attention to the fact that the focused retrieval problem is not solved and fertile research ground lies ahead. We share our experiences with other interested researchers in the hopes that they will prove useful to those grappling with similar problems.

7. REFERENCES

[1] M. Bilotti, P. Ogilvie, J. Callan, and E. Nyberg. Structured retrieval for question answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2007.

[2] L. Hirschman and R. Gaizauskas. Natural language question answering: The view from here. *Journal of Natural Language Engineering, Special Issue on Question Answering*, Fall–Winter 2001.

[3] P. Kingsbury, M. Palmer, and M. Marcus. Adding semantic annotation to the penn treebank. In *Proceedings of the 2nd International Conference on Human Language Technology Research (HLT 2002)*, 2002.

[4] P. Ogilvie and J. Callan. Hierarchical language models for retrieval of xml components. In *Proceedings of the Initiative for the Evaluation of XML Retrieval Workshop (INEX 2004)*, 2004.

[5] 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 Intelligence Analysis*, 2005.

[6] H. Turtle and W. B. Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222, 1991.

Let's Phrase It: INEX Topics Need Keyphrases

Antoine Doucet
GREYC CNRS UMR 6072
University of Caen Lower Normandy
F-14032 Caen Cedex
France
Antoine.Doucet @info.unicaen.fr

Miro Lehtonen
Department of Computer Science
P.O. Box 68
FI-00014 University of Helsinki
Finland
Miro.Lehtonen@cs.Helsinki.Fi

ABSTRACT

In this paper, we study and discuss the usage of phrases in the INEX evaluation of XML retrieval as well as in related research. We find that the INEX framework could easily become a unique testbed for researchers interested in the exploitation of complex terms in IR, while triggering interest from others. Unfortunately, our analysis of the use of keyphrases in INEX topics shows a downwards trend over the years that impacts on the attention of participants. While NEXI, the official query format of INEX, does indeed support keyphrases, its full potential does not materialize, as topic contents show a lack of consistency in their markup. In 2007, 87% of the INEX queries contained keyphrases, but only 11% of those were marked up. We present simple and low-cost solutions to let the INEX collections deliver their full potential in keyphrase retrieval.

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

General Terms: Theory

Keywords: XML Information Retrieval, Phrase, Keyphrase

1. INTRODUCTION

Keyphrases are important for IR tasks. When used in queries they are matched with similar phrases in the documents. Advanced approaches are based on a keyphrase index which contains the most important if not all phrases of the document collection. INEX has provided excellent testbeds for keyphrase search. The document collections with marked up phrases and queries with explicitly marked keyphrases have inspired researchers to develop methods for parsing, indexing, and matching the phrases in order to improve from the basic keyword search. However, the annual evaluation is subject to criticism regarding keyphrase search. Even if a system with support for keyphrases outperforms the same system without keyphrases, we cannot conclude that the keyphrases actually improve the results as the number of queries involved is too low to make a statistically significant difference. Moreover, systems that can make the most out of keyphrases may not excel in the overall evaluation as they are not given any keyphrases for most of the queries.

Our analysis of the INEX data shows that keyphrases have been seriously neglected in the topic development of the recent INEX evaluations. Only 8% of the INEX 2007 topics define keyphrases, whereas the corresponding numbers were around 70% for the INEX 2003–2004 topics. In order to prevent keyphrases from going into complete oblivion, we suggest that the future topic authors of INEX be encouraged to mark keyphrases explicitly in the topic statements.

This paper is organized as follows. Essential previous research and statistics of phrase searching are presented in Section 2. Keyphrases in the INEX topics over the years are analysed in Section 3. Other connections between INEX and keyphrases are summarized in Section 4 including a brief overview on the document collections and methodology. In Section 5, we discuss the simple ways in which INEX could offer a unique testbed for the use of keyphrases in IR. We conclude the paper in Section 6.

SIGIR 2008 Workshop on Focused Retrieval
July 24, 2008, Singapore.
ACM .

2. KEYPHRASES AND IR TASKS

Numerous information retrieval systems rely on a "bag of words" representation of documents, and thus ignore the relative position of words. It is intuitively clear that taking phrases and collocations into account improves text understanding (for computer-based systems as well as for human readers). Zhai et al. [25] mention many subsequent problems. The biggest one is that of complex lexical units: The meaning of a word association is different from that of the "sum" of the meanings of the individual words they compose of. For instance, the expression "hot dog" is seldom used to refer to a dog. Another example is "to kick the bucket", an expression that means "to die", while "to kick" means "to strike out with the foot", and a "bucket" is a "cylindrical vessel used for holding or carrying liquids or solids". In such cases, it is clear that it is crucial to grasp the meaning of the expressions rather than solely that of the word components. Naturally, there have been numerous attempts to exploit lexical cohesion in IR.

2.1 Definition of a keyphrase

In this paper, we define a keyphrase as a set of adjacent terms, that are intended as a single lexical unit by the user. A keyphrase may be *explicit* (that is, clearly delimited with quotation marks or commas, for instance), or *implicit*, with no way to know, a priori, which sets of words the end user meant as phrases. Sample implicit and explicit keyphrases are shown in Table 1.

2.2 Previous research

Work on the use of phrases in IR has been undergone for over 30 years with mitigated success. Early results were very promising. Unexpectedly, however, the constant growth of test collections caused a drastic fall in the quality of the results. In 1975, Salton et al. [16] showed an improvement in average precision over 10 recall points between 17% and 39% over a keyword-only baseline. In 1989, Fagan [3] reiterated the exact same experiments with a 10 Mb collection and obtained improvements from 11% to 20%. This negative impact of the collection size was lately confirmed by Mitra et al. [13] over a 655 Mb collection, improving the average precison by only one percent ! Turpin and Moffat [19] revisited and extended this work to obtain improvements between 4% and 6%.

A conclusion from this historical work is that keyphrases improve results at low levels of recall, but are globally inefficient for the *n* first ranked documents. According to Mitra et al. [13], such modest benefit from phrases to the best answers is explained by the fact that phrases promote documents that deal with only one aspect of possibly multifaceted queries. For example, one of the TREC-4 topics is about "problems associated with pension plans, such as fraud, skimming, tapping or raiding". Several top-ranked documents discuss pension plans, but not any related problem. Mitra et. al call this problem *inadequate query coverage*.

More recently, Vechtomova [20] started to investigate user-selected keyphrases, which puts aside the problem of phrase recognition so that we can focus on the usage of keyphrases in search tasks. The results show a consistent performance improvement in terms of average precision, and confirm the observation that the improvement is mainly built at low recall levels, while the impact is negative at higher levels.

2.3 Keyphrases in web search

An analysis of the query logs of the Excite search engine by Williams et al. [22] indicated that 5–10% of the web queries were phrase queries and that 41% of the rest also matched a phrase. In our terms, 5–10% of the queries are explicit keyphrases, and 41% of the rest may be implicit keyphrases. Unfortunately, commercial search engines are seldom using keyphrases per se, rather, they rely on proximity search or n-grams.

The generally recognized reason is that taking phrases into account is useful *in some cases*, while in others it only worsens retrieval performance. The key problem is that no way has yet been found to distinguish, *a priori*, the cases where keyphrases are useful from those where they are not. Therefore, it has been considered safer to rely on keywords and proximity search. From the current state of the art, it is straightforward to draw the conclusion that there is still plenty of room for future research on keyphrases in both web search and other search tasks.

3. KEYPHRASES IN INEX TOPICS

In this section, we study the usage of keyphrases in INEX topics over the 6 editions of the evaluation initiative (from 2002 to 2007). Two sample topics, respectively from INEX 2002 and INEX 2007 are presented in Figure 1 and Figure 2.

We have analysed all the accepted topics since 2002, and counted how many of them contain at least one implicit keyphrase and how many of them contain at least one explicit keyphrase. For this, we relied solely on the content of that XML element of the topic that was the most similar to a short, web-like, query. For the 2002–2004 campaigns, we used the `<keywords>` element, while for 2005 to 2007, we looked at the content of the `<title>` element.

Year	Topics	explicit KP	implicit KP	no KP
2002	60	20 (33%)	30 (75% *)	10 (17%)
2003	66	47 (71%)	13 (68% *)	6 (9%)
2004	74	51 (69%)	18 (78% *)	5 (7%)
2005	87	29 (33%)	52 (90% *)	6 (7%)
2006	125	43 (34%)	70 (85% *)	12 (10%)
2007	130	11 (8%)	102 (86% *)	17 (13%)
Total	542	201 (37%)	285 (84% *)	56 (10%)

Table 2: Number of accepted topics with explicit and implicit keyphrases. *) The percentage of implicit keyphrases is relative to the total number of topics without explicit keyphrases.

If the sequence of words was separated by some kind of delimiters (e.g., commas or quotations), and several words were found between those markers, we considered that the topic contained an explicit keyphrase. If the sequence of words contained no delimiters at all, and some adjacent words were clearly meant to be components of a complex lexical unit, the topic was considered to contain an implicit keyphrase. Typical examples are word pairs such as "information retrieval" or "firstname lastname".

In Table 1, we describe how we judge the first 5 topics of INEX 2002. In topic 1, we can easily understand that "description logic" is meant to be a phrase, since the corresponding acronym, "DL" is also included. In topic 3, it is clear that "visualizing large information hierarchies" is an entity. We get a hint at this fact from the repetition of the word "information", hence, "information spaces" is also intended as a phrase. In some cases, deciding whether or not a sequence of words was meant as an implicit keyphrase requires more consideration, but then, the topic description and narrative help us find the correct interpretation.

The per year statistics on the number of INEX topics containing explicit and implicit keyphrases are presented in Table 2.

We immediately notice that the proportion of topics containing keyphrases is much higher than that reported by Williams et al. [22] for web queries (37% of explicit keyphrases versus 5 to 10%, and 84% of implicit keyphrases amongst the rest instead of 41%). This is natural, as the INEX topics are much longer than web queries, and, unlike them, they were carefully thought up, reviewed, and selected by the organizers of the forum. This is actually one reason why it would take little additional effort to formalize keyphrase markup.

In 2003, the use of comma to separate entities in the `<keywords>` element was systematic. This certainly explains the surge in the ratio of explicit keyphrases. But the element name "keywords" was perhaps sometimes taken too literally, as in "keyword *versus* keyphrase". Much to our surprise, several topics are using commas to separate words that are clear phrases. One of many examples is topic 101 shown in Figure 1, where it is very clear that "information retrieval" is a central concept, but the words "information" and "retrieval" are comma-separated in the `<keywords>` element!

The same phenomenon occurred in 2004, with numerous explicit keyphrases on one hand, and comma-separated keyphrase components on the other. The consistency of the comma-separation markup is additionally fading, as several

Topic	Word Sequence	explicit KP	implicit KP
1	description logic DL ABox TBox reasoning	no	yes
2	funding america DARPA US	no	no
3	visualizing large information hierarchies information spaces text multidimensional data datamining databases	no	yes
4	’extreme programming’ experiences results	yes	no
5	QBIC, IBM, image, video, content query, retrieval system	yes	no

Table 1: The “web-like queries” of the first 5 topics of INEX and our interpretation of whether a keyphrase is present or not.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd">
<inex_topic topic_id="101" query_type="CO" ct_no="37">
<title>+"t test" +information </title>
<description>use of the t-test in information retrieval </description>
<narrative>Information retrieval experimenters are advised to compare their new mean-average-precision
results with the baseline using a t test. We have reason to believe that this may be bad, even very bad,
advice, and are interested in papers that apply the t-tests to information retrieval (and possibly other
software engineering tasks). We are also interested in papers that mention alternative statistical
techniques to determine significance. </narrative>
<keywords>t-test, information, retrieval </keywords>
</inex_topic>
```

Figure 1: Topic 101, from INEX 2003.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd">
<inex_topic topic_id="522"  ct_no="190">
<title>April 19th revolution peaceful revolution velvet revolution quiet revolution</title>
<castitle>//*[about(.,"April 19th revolution" "peaceful revolution" "velvet revolution" "quiet revolution")]
</castitle>
<description>Find information about how the April 19th revolution differs from the peaceful,
velvet and quiet revolutions.</description>
<narrative>As a history buff, you have heard of the quiet revolution, the peaceful revolution and the
velvet revolution. For a skill-testing question to win an iPod you have been asked how they differ from
the April 19th revolution.</narrative>
</inex_topic>
```

Figure 2: Topic 522, from INEX 2007.

explicit phrases are double-marked with both commas and quotations marks, as in topic 158, where the `<keywords>` element contains:

```
  turing, test, consciousness, intelligence,
"imitation game".
```

In 2005, there was no more element dedicated to keywords, which caused the beginning of a severe downwards trend on the number of explicit keyphrases. In 2006, an `<ontopic_keywords>` element was inluded in the topic format. Its influence is visible as the downwards trend was temporarily interrupted. Several participants have indeed included keyphrase delimiters in the `<ontopic_keywords>` element, that they then carried over to the `<title>` element. In 2007, unfortunately, the `<ontopic_keywords>` element disappeared, leading to a sharp fallout in terms of explicit keyphrases.

In conclusion, we make two major observations: 1) year after year, the number of explicit keyphrases has been decreasing, 2) the number of implicit keyphrases has been steady during the 6 INEX campaigns, notably, regardless of the topic format and guidelines.

Although it is harder to quantify, a third point should be made about the growing inconsistency of the phrase markup, whenever it was present. Some explicit keyphrases are marked with semi-columns, others with commas, and the rest with single or double quotation marks. Several keyphrases are double-marked (commas plus quotes).

It even seems that participants are confused about whether they are actually **allowed** to mark keyphrases. This impression is supported by examples such as topic 101, which we already mentioned (Figure 1), where two clear keyphrase components are comma-separated. Topic 522 is even more confusing (Figure 2), because the same keyword sequence is used in the `<castitle>` and in the `<title>` elements, with the exception that keyphrase delimitors are used in the `<castitle>` and removed in the `<title>` element. This is very hard to explain. Did the topic author have any reason to doubt that quotation marks were allowed inside `<title>` elements?

4. KEYPHRASES IN INEX RESEARCH

Keyphrases have been taken into account in various ways in the systems developed by INEX participants in the past years of INEX, e.g. when indexing and ranking the documents, or parsing the queries, or even generating formal queries from queries in a natural language. Examples of the different roles of keyphrases in INEX-related research are presented in this section.

4.1 XML documents

Any document with text in a natural language contains enough phrases to make it suitable for research on keyphrase search. From the keyphrase perspective, hypertext documents add an interesting feature to plain text documents: many phrases — the anchor texts — are now marked up with designated markers. HTML documents go even further down the road as they allow titles, emphasized content, list items etc. to be marked up with designated tags in addition to the anchor texts. However, HTML documents are computationally challenging to process for someone searching for phrases as the quality of the HTML code varies. In this sense, XML is the perfect document format for keyphrase search as the markup is highly regular and always well-formed[1].

The collections of XML documents provided by INEX 2002–2007 contain plenty of marked up phrases. In the collection of articles from IEEE journals, phrases are typically marked up because of an intended emphasis on the phrase, whereas the most common marked up phrases in the Wikipedia XML articles are anchor texts of hyperlinks. Both collections provide an interesting playground for methods on indexing and searching keyphrases. The methods that did not convince with plain text documents might have lots of unmaterialized potential with XML documents — if given a second chance.

4.2 Adhoc retrieval

Although some explicitly split the keyphrases of INEX queries into unordered sets of individual keywords [1], many others parse them and match them with similar phrases in the documents. Both Raja et al. and Lehtonen and Doucet compute a separate score for keyphrase similarity in the vector space model which is combined with a keyword similarity score into a single Retrieval Status Value [15, 12]. A less strict interpretation of the concept of a keyphrase is defined in the TRIX system where the words of the keyphrase are only required to appear in a mutual XML element [8].

Approaches where the documents are stored in an XML database [14, 21] inherently support constraints on the order and distance of the keywords. Other database systems supporting keyphrase queries include Cheshire II with proximity indexes [11] as well as TopX with term offsets stored in an auxiliary database table [17].

Phrases are also part of the document representation in several systems for XML IR, e.g. Maximal Frequent Sequences of EXTIRP [2], Rich Document Representation of the extended PLIR [10], and bigram language models of TIJAH [9]. Despite the quite widespread interest in parsing and matching keyphrases, most of the efforts originate in the early years of INEX. The obvious explanation for the recent lack of interest lies in the diminishing proportion of explicitly marked keyphrases in the INEX topics.

4.3 The NLP Track prospect

Besides the adhoc track queries, keyphrases have had an important role in the experiments of the NLP track of INEX. The main challenge of the track has been to convert a natural language query into a formal NEXI query which is the official query format of INEX [18]. One of the earliest systems participating in the track was NLPX which segments sentences into disjoint chunks that eventually converge into NEXI keyphrases [23, 24]. Zargayouna et al. go along the same lines as they *"prefer complex terms to simple ones"* as they generate NEXI queries [7]. However unfortunate it may seem that the official NEXI queries created by INEX participants come with so few explicitly marked keyphrases, the NLP track may regard this as an opportunity to assist by applying various chunking methods to the topic titles. The other option is to have the keyphrases marked by topic developers, but few participants are currently doing it.

5. DISCUSSION

In information retrieval, the use of phrases includes two problems: the first one is that of the detection of phrases

[1]Other than well-formed XML is not defined.

in the document collection, while the second one is to find ways to improve retrieval effectiveness, given good phrases.

As we have seen in Section 4.1, XML mark-up has potential to ease the extraction of phrases, which is a way to isolate the problem of phrase exploitation, that no other evaluation framework provides, even when they show specific consideration for phrase usage. For instance, the topics of NTCIR collections[2], have systematically included a `<CONC>` element, with a list of keywords in which keyphrases are clearly and consistently delimited, but the documents themselves have no comparable phrase-related markup.

As opposed to most IR evaluation forums, we must point out a specificity of INEX, which is that topics are collaboratively contributed by participants. The plentiful of authors makes it naturally harder to keep consistent notation across the topic set, especially when keyphrase markup has barely ever been mentioned in the topic creation guidelines.

Unfortunately, while the XML mark-up of the INEX collections permits to ease phrase extraction from documents, keyphrases have seemingly been abandoned in the topics. This has naturally led to a drop in their usage by INEX participants. As detailed in Section 4, while several made use of complex terms in the first years, when more explicit phrases were available, the interest in keyphrases has lately diminished drastically: Most new participants are not taking phrases into account.

An easy way to solve the issue mentioned here is to define a strict way to mark keyphrases, and request topics to conform to it. The goal would be to replace all the implicit keyphrases with explicit ones, which could even be corrected by the organizers, although leaving it to the participants is certainly preferable. Clearly, the additional amount of work is small, nearly negligible.

Having a large set of clearly marked up phrases, together with the implicit phrases contained in the document collection should make INEX a unique testbed for researchers interested in the use of complex terms in IR. At the same time, it makes no harm to others, who can very easily ignore the keyphrase delimitors.

However, we must point out that our goal is not only to please participants interested in the use of keyphrases, but to avoid causing any inconvenience to others. We hope that the generalization of explicit keyphrases will encourage all participants to get involved, which should come naturally when statistically siginficant performance improvements are achieved (retrieval performance has reportedly been improved, but the small number of keyphrases does not leave a chance to statistically significant results without a massive performance surge).

6. CONCLUSION

The topic definitions of the past INEX evaluations have had several shortcomings regarding the formulation of keyphrases. An INEX evaluation of keyphrase queries is currently a lost opportunity despite its potential. Nonetheless, we are not far from solving the major problems and seeing a brighter future for the research on searching XML documents for keyphrases. The key facts that we have shown in this paper are the following:

- Keyphrases are common in real world queries, but the methodology for processing them is not yet mature.
- INEX document collections are rich in phrases, but regarding the INEX topics, keyphrases are about to become extinct.
- Several systems with result submissions to the past INEX evaluations support keyphrase queries.

The only thing needed for a decent INEX evaluation of keyphrase search are more queries — INEX topics — with explicitly marked keyphrases. The additional effort required is minimal but invaluable as it has the potential to revive the research activity in the area of keyphrase search. The first step entails writing more detailed guidelines for topic development, and the second step, collective topic authoring by the INEX participants.

[2]"NII Test Collection for IR systems", http://research.nii.ac.jp/~ntcadm/index-en.html

7. REFERENCES

[1] C. L. Clarke and P. L. Tilker. Multitext experiments for inex 2004. In Fuhr et al. [5], pages 85–87.

[2] A. Doucet, L. Aunimo, M. Lehtonen, and R. Petit. Accurate Retrieval of XML Document Fragments using EXTIRP. In *INEX 2003 Workshop Proceedings*, pages 73–80, Schloss Dagstuhl, Germany, 2003.

[3] J. L. Fagan. The effectiveness of a nonsyntactic approach to automatic phrase indexing for document retrieval. *Journal of the American Society for Information Science*, 40:115–132, 1989.

[4] N. Fuhr, M. Lalmas, S. Malik, and G. Kazai, editors. *Advances in XML Information Retrieval and Evaluation, 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, November 2005, Revised Selected Papers*, volume 3977 of *Lecture Notes in Computer Science*. Springer, 2006.

[5] N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors. *Advances in XML Information Retrieval, Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 2004*, volume 3493 of *Lecture Notes in Computer Science*. Springer, 2005.

[6] N. Fuhr, M. Lalmas, and A. Trotman, editors. *Comparative Evaluation of XML Information Retrieval Systems, 5th International workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006, Dagstuhl Castle, Germany, December 2006*, volume 4518 of *Lecture Notes in Computer Science*. Springer, 2007.

[7] Haïfa Zargayouna and Victor Rosas and Sylvie Salotti. Shallow Parsing of INEX Queries. In Fuhr et al. [6], pages 284–293.

[8] Jaana Kekäläinen and Marko Junkkari and Paavo Arvola and Timo Aalto. TRIX 2004 — struggling with the overlap. In Fuhr et al. [5], pages 127–137.

[9] Johan List and Vojkan Mihajlović and Georgina Ramírez and Arjen de Vries and Djoerd Hiemstra and Henk Ernst Blok. TIJAH: Embracing IR Methods in XML Databases. *Information Retrieval*, 8(4):547–570, 2005.

[10] M. Karimzadegan, J. Habibi, and F. Oroumchian. Logic-based XML information retrieval for determining the best element to retrieve. In Fuhr et al. [5], pages 88–99.

[11] R. R. Larson. Cheshire II at INEX '04: Fusion and Feedback for the Adhoc and Heterogeneous Tracks. In Fuhr et al. [5], pages 322–336.

[12] M. Lehtonen and A. Doucet. Extirp: Baseline retrieval from wikipedia. In Fuhr et al. [6], pages 119–124.

[13] M. Mitra, C. Buckley, S. A., and C. Cardie. An analysis of statistical and syntactic phrases. In *Proceedings of RIAO97, Computer-Assisted Information Searching on the Internet*, pages 200–214, 1997.

[14] J. Pehcevski, J. Thom, and A.-M. Vercoustre. Combining information retrieval and a native XML database. *Information Retrieval*, 8(4):571–600, 2005.

[15] F. Raja, M. Keikha, M. Rahgozar, and F. Oroumchian. Using rich document representation in XML information retrieval. In Fuhr et al. [6], pages 294–301.

[16] G. Salton, C. Yang, and C. Yu. A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science*, 26:33–44, 1975.

[17] M. Theobald, R. Schenkel, and G. Weikum. TopX and XXL at INEX 2005. In Fuhr et al. [4], pages 282–295.

[18] Trotman,Andrew and Sigurbjörnsson,Börkur. Narrowed extended xpath i (nexi). In Fuhr et al. [5], pages 16–40.

[19] A. Turpin and A. Moffat. Statistical phrases for vector-space information retrieval. In *Proceedings of the 22nd ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 309–310, 1999.

[20] O. Vechtomova. The role of multi-word units in interactive information retrieval. In *Proceedings of the 27th European Conference on Information Retrieval, Santiago de Compostela, Spain*, pages 403–420, 2005.

[21] F. Weigel, K. U. Schulaz, and H. Meuss. Ranked retrieval of structured documents with the S-Term vector space model. In Fuhr et al. [5], pages 238–252.

[22] H. E. Williams, J. Zobel, and D. Bahle. Fast phrase querying with combined indexes. *ACM Transactions on Information Systems (TOIS)*, 22(4):573–594, 2004.

[23] A. Woodley and S. Geva. NLPX at INEX 2005. In Fuhr et al. [4], pages 358–372.

[24] A. Woodley and S. Geva. NLPX at INEX 2006. In Fuhr et al. [6], pages 302–311.

[25] Zhai, Chengxiang, X. Tong, N. Milic Frayling, and E. D.A. Evaluation of syntactic phrase indexing. In *Proceedings of the 5th Text Retrieval Conference, TREC-5*, pages 347–358, 1997.

Sound ranking algorithms for XML search

Djoerd Hiemstra[1], Stefan Klinger[2], Henning Rode[3], Jan Flokstra[1], and Peter Apers[1]

[1]University of Twente, [2]University of Konstanz, and [3]CWI
hiemstra@cs.utwente.nl, klinger@inf.unikonstanz.de,
henning@cwi.nl, flokstra@cs.utwente.nl, apers@cs.utwente.nl

ABSTRACT

We argue that ranking algorithms for XML should reflect the actual combined content and structure constraints of queries, while at the same time producing equal rankings for queries that are semantically equal. Ranking algorithms that produce different rankings for queries that are semantically equal are easily detected by tests on large databases: We call such algorithms *not sound*. We report the behaviour of different approaches to ranking content-and-structure queries on pairs of queries for which we expect equal ranking results from the query semantics. We show that most of these approaches are not sound. Of the remaining approaches, only 3 adhere to the W3C XQuery Full-Text standard.

1. INTRODUCTION

Models for ranked retrieval of XML data should comprise four parts: 1) a model of the text, 2) a model of the structure, 3) a query language, and 4) a ranking algorithm. Ranking is of the utmost importance if an effective XML search system is needed. Some queries might match millions of elements from the text database, but users will only be able to inspect a few. Many of the early structured text retrieval models do not consider ranked retrieval results, or if they do only as an afterthought, i.e., by ranking the retrieval results using a text-only query disregarding the structural conditions in the query [6]. A simple but powerfull way to take the structure of the results into account is to apply a standard information retrieval model to the retrieved content, and then propagate or aggregate the scores based on the structure [7, 13]. In several of these approaches to ranking, the propagation is guided by weighting paths to elements differently by so-called augmentation weights [8, 9], to model for instance that a title element is more likely to contain important information than a bibliography item. Instead of propagating or aggregating the scores from the leaf nodes, *algebraic* approaches include the ranking functionality inside each operator of the query language [2, 16]. Ranking might also include relaxation of the queries' structural conditions, for instance by rewriting complex queries step-wise to simpler queries [4]. The development of effective ranking algorithms for XML information retrieval is studied in the workshops of the Initiative for the Evaluation of XML retrieval (INEX) [14].

This paper studies mathematical properties of ranking algorithms. While we pursue *effective* algorithms as described above, in this paper we additional pursue *sound* ranking algorithms. Ranking algorithms for structured information retrieval are sound if the following two conditions are met:

1. Ranking should reflect the actual, combined content and structure constraints;
2. Two queries that are semantically equal (from a standard –unranked– XPath or XQuery perspective) should produce the same ranked results.

An example of a system that violates Condition 1 would be a system that first runs the query as a Boolean selection, and then ranks the resulting elements using a standard text retrieval model, i.e., a ranking algorithm that ignores the structure. Suppose we are looking for articles that talk about *ranked xml retrieval* which were supported in one way or another by *John Doe*. This might be formulated as follows using the NEXI query language [18] (Similar examples will be provided for XQuery Full-text [1] below).

```
//article[about(.//p, ranked xml retrieval) and
            about(.//ack, john doe)]
```
(1)

NEXI stands for Narrowed Extended XPath I, a version of XPath that only supports the descendant and self steps, but that is extended by a special `about()` function. The results of a NEXI query are not in document order, but ranked by estimated relevance to the `about()` parts. If the system first performs a Boolean selection, then it suffers from the well-known disadvantages of Boolean systems: if we interpret the `about()` function as a conjunctive query for which all three words *ranked*, *xml* and *retrieval* should occur in the document, then it is for long queries unlikely that any article matches the query (not because there are no relevant articles, but because they discuss for instance *probabilistic xml retrieval*, or *ranked structured retrieval*, or *ranked xml search*, etc. In that case the result would be empty. If we however interpret the `about()` function as a disjunctive query for which the matching of a single word suffices, then the ranking (i.e., a ranking that ignore the structure) would ignore the paragraphs and acknowledgments. In this case, the top document might very well discuss the holiday diary of John Doe, in which he acknowledges the top ranked XML systems for retrieval (i.e., it might be the paragraph the matches *john doe* and the acknowledgments that match *ranked xml retrieval*).

We believe a true XML retrieval system should meet Condition 1 above. Suppose such a system executes the following query.

```
//article[about(.//p1, xml)]|//article[about(.//p, xml)]  (2)
```

If Condition 1 is met then the system's ranking reflects the actual, combined content and structure constraints, so the ranking will reflect a match in the *p1* elements or a match in the *p* elements. These queries occur a lot in complex documents, such as the IEEE journal data used in the Initiative for the Evaluation of XML Retrieval (INEX) from 2002 to 2005 [14]. In this collection the elements *p1* and *p* both refer to types of paragraphs, as do the elements *p2*, *ip1*, *ip2*, etc. In queries, the user usually does not want to distinguish these different kinds of paragraphs, hence the query above. In fact, such cases were that frequent in INEX that the organization introduced tag equivalence classes [14], and additional query syntax to ease the formulation of such queries (which is also allowed in XPath 2.0). The following NEXI query is equivalent to the query above:

```
//article[about(.//(p1|p), xml)]          (3)
```

Suppose the system ranks the returned articles for the second query differently than for the first query, resulting in 8 articles in the top 10 that were previously not in the top 10. In that case, the system violates Condition 2: Because the queries are semantically equal, they should result in the same ranking. In order for a ranking algorithm to be *sound* it should meet Condition 1 and Condition 2. We will show in this paper that for systems that meet Condition 1, it is not trivial to meet Condition 2 as well. In fact, we believe it might be impossible to come up with a ranking approach that meets Condition 2 in all cases, especially in the case of XQuery full-text for which there are many ways of formulating the same query.

In this paper, we will investigate the soundness of ranking algorithms by systematically comparing the retrieval results of ranking algorithms that meet Condition 1 for two queries that are semantically equal. As a starting point of our study, we hypothesize that all ranking algorithms meet Condition 2 as well. Only if we find an example that violates Condition 2 we will drop the hypothesis. We will show that for almost all reasonable ranking algorithms, there are examples of two semantically equal queries and a data set for which the two queries produce a different ranking.

The paper is organized as follows. Section 2 describes the queries used for analysing the soundness of ranking algorithms, and how they are executed. Section 3 presents the test data used. Section 4 presents the ranking approaches we evaluated. In Section 5 the experimental results are presented. Finally, Section 6 concludes this paper.

2. THE TEST QUERIES

Our analysis of the problem follows that of Mihajlovic [15, Chapter 3], who identifies three requirements for scoring in structured retrieval models. XML ranking algorithms should provide:

Score computation: Given a text query and a set of nodes, compute the score of each node. This is provided by traditional information retrieval models.

Score propagation: This is needed for all XPath axis steps. To do an axis step from a node for which a score was computed, the scores need to propagate to the result nodes. For some axis steps, for instance the parent step, the score of several children needs to propagate to a single result node.

Score combination: Score combination is needed if the same set of nodes is scored multiple times and the final score should reflect the scores of the nodes in both sets.

As an example, consider the following XQuery Full-Text query, that ranks the acknowledgments elements (*ack*) that thank *John Doe* in articles about *XML*, similar to the query in Example 1 above:

```
for $d score $s in doc("test.xml")//ack
where ../article ftcontains "xml"
  and . ftcontains ("John", "Doe")              (4)
order by $s desc
return $d
```

Here, score *computation* is needed for the `article` elements (scored by the similarity to *xml*) and for the `ack` elements (scored by the the similarity to *John Doe*). The query should rank `ack` elements, so the scores of the `article` elements needs to be *propagated* to their child `ack` elements. Finally, the two scores for each `ack` element need to be *combined* in a final score.

If the user wants to rank the acknowledgments from articles about *XML* that thank *John Doe*, he/she might as well pose the following query.

```
for $d score $s in
  doc("test.xml")//article[. ftcontains "xml"]/ack
where . ftcontains ("John", "Doe")              (5)
order by $s desc
return $d
```

In fact, several queries are possible that are semantically equal as meant by Condition 2 above. We define *semantic equality* as follows: The XPath representation of a NEXI query is defined as the query produced by replacing every NEXI function `about(n, s)` with `fn:contains(fn:string(n), "s")`. Two NEXI queries are *semantically equal* if and only if their XPath representations are equivalent, *i.e.*, return the same result when evaluated. Similarly, the XQuery representation of an XQuery Full-Text query is defined as the query produced by replacing every XQuery Full-Text function `n ftcontains "s"` by `fn:contains(fn:string(n), "s")`, and two Full-Text queries are *semantically equal* if and only if their XQuery representations are equivalent. However, because of the properties of score computation, propagation, and combination, two semantically equal queries might produce different rankings, and might therefore return different (top) elements to the user, i.e., the ranking algorithm is not *sound.*

Case 1: The semantics of score computation

In our first case we look at the semantics of *score computation.* Score computation is the most important of Mihajlovic's requirements. Here, we only consider simple scoring, i.e., scoring of queries without using proximity or phrases (in

NEXI, phrases can be marked as with double quotes), i.e., the following query.

`//article[. ftcontains ("xml", "ir", "db")]` (6)

In XQuery Full-text, this query retrieves articles that match any of the terms, i.e., the standard behavior is that of a Boolean OR query. Given these semantics, we expect scoring to be compositional, that is, if we select article containing *"xml"* and union those with articles containing *"ir"*, *"db"* as shown in the following query, then we expect the same results as the query above.

```
//article[. ftcontains "xml"]|//article[.
            ftcontains ("ir", "db")]
```
(7)

Several alternative formulations are possible in XQuery Full-Text, for instance `//article[. ftcontains "xml" ftor ("ir", "db")]` or `//article[. ftcontains "xml" or . ftcontains("ir", "db")]`. The alternative formulations select the exact same articles, and if simple scoring behaves as a Boolean OR query, then we expect a sound ranking approach to produce the same rankings for all these formulations.

However, from the user's point of view, we might argue that scoring should have the semantics of the Boolean AND: the best documents are the ones containing *all* three query terms, not just many occurrences of either query term. If simple scoring behaves as a Boolean AND query, then we expect a sound ranking of Query 6 approach to produce the same rankings as the following query:

```
//article[. ftcontains "xml"][.
            ftcontains("ir", "db")]
```
(8)

or alternative formulations: `//article[. ftcontains "xml" ftand ("ir", "db")]`, or `//article[. ftcontains "xml" and . ftcontains ("ir", "db")]`. These queries (and similar NEXI queries) correspond to different query plans in the PF/Tijah XML search system [11]. These plans use so-called *score region algebra* to process these queries. Figure 1 contains the query trees for the three plans. Instead of putting the region algebra operators in the trees (the exact definition of algebraic operators in outside the scope of this paper), the figure contains the partial queries that represent the intermediate results at that stage of the query plan.

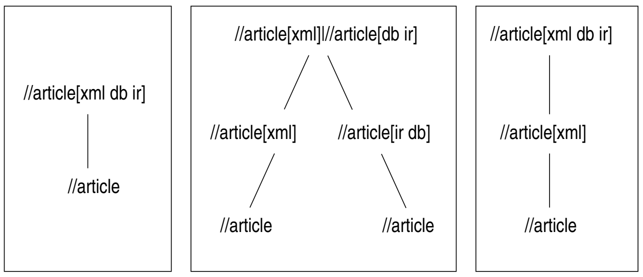


Figure 1: Query plans 1a, 1b and 1c for Case 1

Following the line of reasoning of sound ranking algorithms presented above, the simple scoring plan shown in Figure 1a should either produce the same ranking as the disjunctive plan shown in Figure 1b, or it should produce the same ranking as the conjunctive plan shown in Figure 1c. If Plan 1a produces a result different from both Plan 1b and 1c, then the score computation is not sound: Users of retrieval systems should be able to understand the difference between OR-queries and AND-queries, however, it is hard to anticipate on semantics that is different from these two.

Case 2: Score propagation – downwards

In the second case we look at *downwards score propagation*: Suppose the user is interested in sections about *"databases"* from articles about *"xml"*. In this case, the scores of the article elements have to be propagated to the section elements. Such a query can be processed in two ways. Either first score all articles, propagate the scores to the contained sections, and score those, as shown in the following query:

```
//article[. ftcontains "xml"]//section[.
       ftcontains "db"]
```
(9)

... or, first score all sections, and then score the articles that contain these sections, as follows:

```
//section[. ftcontains "db"][./ancestor::article
       ftcontains "db"]
```
(10)

The query trees of the actual query plans are shown in Figure 2.

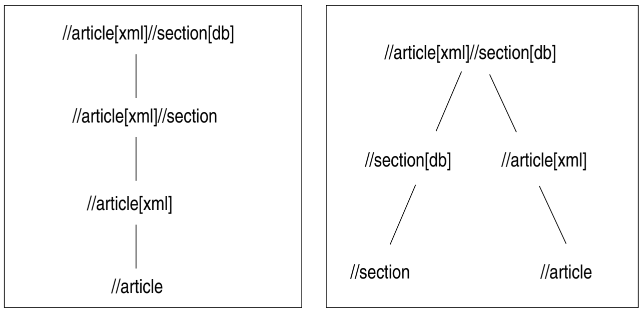


Figure 2: Query plans for Case 2

Following the line of reasoning of sound ranking algorithms presented above, there is no reason why both queries and both query plans above should not produce the exact same ranking of section elements.

Case 3: Score propagation – upwards

In the second case we look at *upwards score propagation*: Suppose the user that was interested in articles about *"xml"* with sections about *"databases"* now wants to retrieve the articles. In this case, the scores of the section elements have to be propagated upwards to the article elements. Again, the query can be processed in two ways. Either first score all articles, and then propagate the scores to the contained sections upwards as shown in the following query:

```
//article[. ftcontains "xml"][.//section
       ftcontains "db"]
```
(11)

... or, first score all sections and propagate the score to articles that contain these sections, as follows:

```
//section[. ftcontains "db"]/ancestor::article[.
       ftcontains "xml"]
```
(12)

The query trees of the actual query plans are shown in Figure 3. Again, a sound ranking approach would produce the exact same ranking for both queries.

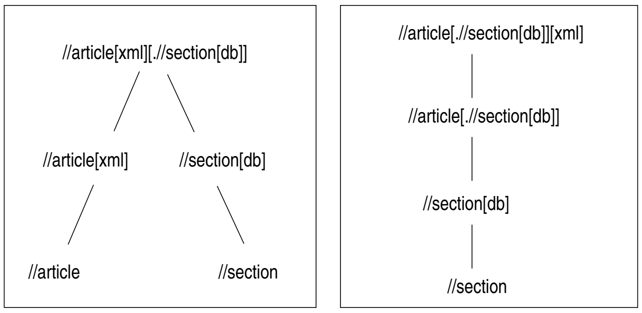

//article[xml][.//section[db]]
//article[xml]
//section[db]
//article
//section
//article[.//section[db]][xml]
//article[.//section[db]]
//section[db]
//section

Figure 3: Query plans for Case 3

Case 4: Score Combination – union

Case 4 looks at *score combination*, more specifically at score combination when the union of two node sets is taken. Suppose the user wants articles that mention *"xml"* in a section, *or* that mention *"db"* in the title:

```
//article[.//section ftcontains "xml" or .//title
          ftcontains "db"]                              (13)
```

As discussed above, both XQuery/XPath Full-Text and NEXI support a union operator "|" that might be used as well. For instance, an alternative formulation of the query above would union two sets of article nodes, one of which the sections contain *"xml"* and another set of article nodes which titles contain *"db"*.

```
//article[.//section ftcontains "xml"]|//article[
          .//title ftcontains "db"]                     (14)
```

For article nodes that are in both sets, the union operator should somehow combine the scores. The query trees of the actual query plans are shown in Figure 4. A sound ranking approach produces the exact same ranking for both queries.

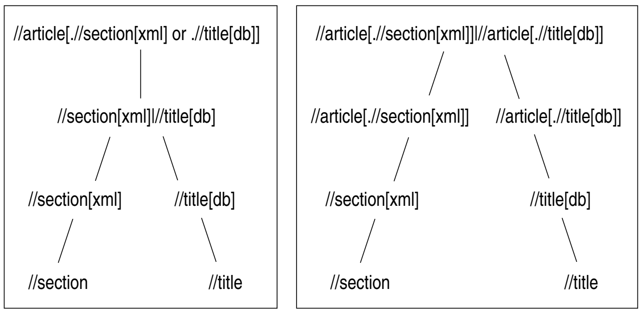

//article[.//section[xml] or .//title[db]]
//section[xml]|//title[db]
//section[xml]
//title[db]
//section
//title
//article[.//section[xml]]|//article[.//title[db]]
//article[.//section[xml]]
//article[.//title[db]]
//section[xml]
//title[db]
//section
//title

Figure 4: Query plans for Case 4

Mihajlovic' score region algebra [15] supports an intersection operator similar to the union operator above. Such an operator is not supported by the XQuery/XPath Full-Text and NEXI, since it is unnecessary in practice. Therefore, we will not consider score combination in the case of intersecting two node sets.

Case 5: XQuery Full-text scoring properties

The XQuery Full-text standard imposes very few restrictions on scoring: The numeric score computed by queries is *implementation-dependent*, i.e, scoring may differ between implementations; scoring is not specified by the W3C specification, and scoring is not required to be specified by the implementor for any particular implementation. The standard however imposes the following two restrictions on full-text contains expressions [1]:

A full-text contains expression returns a Boolean value.

So, a full-text contains expression always distinguishes the matching nodes from the non-matching nodes. In Mihajlovic's score region algebra [15], operators compute scores for *all* nodes, that is, all nodes always match the expression. One might argue that this follows the XQuery Full-text standard (each full-text expression always returns *true*), but at least it is not in the spirit of XQuery Full-text. We will call the semantics of the score region algebra operators *ranking semantics* and the semantics suggested by XQuery Full-text *matching semantics*. In practice, matching semantics is often required in practical systems. The current PF/Tijah implementation has default matching semantics. We investigate both matching semantics and ranking semantics to see which of the two is more likely to produce sound ranking.

Score values are of type xs:double in the range [0, 1].

This restriction is not imposed by score region algebra [15]. In fact, many well-performing ranking functions – for instance Okapi's BM25 [17] – produce scores greater than 1 in some cases, and even if they do not, the approach might produce scores greater than 1 after score propagation and score combination.

Case 5 does not define an extra set of queries. The soundness of rankings produced by the queries in Case 1 to 4 above are checked when using matching semantics and ranking semantics. Furthermore, we check if the scores of the results of the queries in Case 1 to 4 are in the range [0, 1].

3. THE TEST DATA

We will evaluate the test cases on artificial data to ensure that we control the circumstances in which queries fail. Some ranking algorithms might be sound in most cases, for instance, they might be sound, unless the elements that are ranked are nested. In the examples above, we would expect every article to have only one acknowledgments section; which might actually be defined in a DTD or XML schema. The schema might also contain elements that have a many-to-one relation to `article` elements, such as `paragraph` elements, or elements that might be nested inside themselves, such as `section` elements (i.e., sections, subsection, and subsubsections might all be ambiguously referred to as `section`. When element scores are propagated through the document structure, they might have to be aggregated in case of many-to-one relations, or divided in case of one-to-many relations, or both aggregated and divided in case of nested relations. The following DTD contains several of such cases:

```
<!ELEMENT root (article | report)* >
<!ELEMENT article (title, section+) >
<!ELEMENT report (title, section+) >
<!ELEMENT section (heading, (section+ | paragraph+)) >
<!ELEMENT title (#PCDATA) >
<!ELEMENT heading (#PCDATA) >
<!ELEMENT paragraph (#PCDATA | list)* >
<!ELEMENT list (item | list)* >
<!ELEMENT item (#PCDATA) >
```

We distinguish the following 5 cases in which elements from two node sets can be nested: 1:1, 1:n, 1:n where the elements of the second node set are nested, n:m where the elements of first node set are nested, and n:m where the elements of both sets are nested.

article vs. title	$1 : 1$
article vs. paragraph	$1 : n$
article vs. section	$1 : n$ nested
section vs. paragraph	n nested : m
section vs. list	n nested : m nested

The queries presented in the cases above are all examples of the "1:n nested" type, i.e., the queries refer to `article` elements and `section` elements. Each query is run as in one of the above five types; so, `article` and `section` are replaced by: 1) `article` elements and `title`, 2) `article` elements and `paragraph`, 3) `article` elements and `section`, 4) `section` elements and `paragraph`, 5) `section` elements and `list`.

4. THE RANKING APPROACHES

We test a total number of 200 ranking approaches for XML search. These approaches are for an important part the same as the approaches Mihajlovic [15] evaluated in oder to find effective ranking approach for XML search. So, our ranking approaches are motivated mostly by testing those approaches for which we expect good recall and precision values in standard information retrieval evaluations, such as those provided by INEX. The number of ranking approaches that can be defined for XML search is endless however, and we do not attempt in anyway to test a complete (sub-)set of all possible ranking functions.

The choices of our ranking approaches are restricted by PF/Tijah's algebraic approach to XML search. Each operator in score region algebra follows the same pattern: It operates on a context node set (context region set), and a target node set (target region set). Both the context nodes and the target nodes might have scores already from previous operations. Each operator has to combine the score of target node with the scores of (possibly) multiple matching context nodes. For each target node, we distinguish three situations: 1) the target node does not match any context node: In this case the target node is not returned (matching semantics) or it is returned with a default score (ranking semantics); 2) the target node matches one and only one context node: In this case, the target node is returned with the *combined score* of the target node and the matching context node; 3) the target node matches more than one context node: In this case, scores of the matching context nodes are first *aggregated*, and then *combined* with the score of the target node. *Aggregation* and *combination* define two of the dimensions along which we define ranking approaches, the other dimensions are the *retrieval model*, and *ranking/matching semantics*:

Score combination

We test 5 different ways to combine the scores of the context node and the target node: adding, multiplying, maximum, minimum, and average.

Score aggregation

We test the same 5 different ways to aggregate the scores of the matching context nodes: sum, product, maximum, minimum, and average. This brings the total number of approaches to 25.

The retrieval model

We test four different retrieval models, bringing the total number of approaches to 100: LMS, a standard language model using linear interpolation smoothing [10]; LM, a standard language model without smoothing; NLLR: normalized log-likelihood ratio (a simple derivation of LMS that produces log-linear scores) [12]; BM25: Okapi's BM25 ranking formula [17].

Ranking semantics vs. matching semantics

We test each approach with ranking semantics (each operator returns all nodes with a score), and matching semantics (each operator returns a selection of the nodes), so 200 approaches in total over all four dimensions.

5. INVESTIGATING THE SOUNDNESS IN PRACTICE

Using the DTD described in Section 3, a 100kB artificial test collection with articles and reports was generated. The text nodes were generated from a simple language model of three words (*"ir", "db", and "xml")*, where each word is generated by some probability. This way, almost every element will match a query to some extent, with scores similar to other elements, possibly resulting in different rankings. Each of the 200 ranking approaches from Section 4 was tested on a pair of queries from one of the four cases from Section 2, where each query followed one of the five ways in which data can be nested described in Section 3, defining in total 5000 queries. We summarize the results by reporting the most important lessons learned.

Lessons for Case 1: Score computation

In all cases that used the NLLR retrieval model, we detected unsound ranking behavior on the test data. The NLLR retrieval model differs from the LMS retrieval model mainly because it uses query length normalization. Apparently, retrieval models that use some form of query length normalization are not sound. Other examples of retrieval models that uses query length normalization are vector space models that use the cosine similarity.

For almost all ranking approaches that use *matching semantics*, we detected unsound ranking behavior when comparing Plan 1a to 1c. Apparently, matching semantics excludes the possibility to follow the semantics of AND-queries, which seems logical because most models retrieve elements even if they do not contain all query terms. An exception is the LM retrieval model, i.e., the language model without smoothing: this is the only model that does not produce unsound rankings behavior when comparing Plan 1a to 1c.

Unsound ranking behavior was occassionally detected when comparing Plan 1a to 1b. The ranking approaches tested are more likely to follow the semantics of OR-queries.

Lessons for Case 2: Score propagation – down

In most cases that used the BM25 retrieval model, we detected unsound ranking behavior, except when score combination uses the product or the maximum. We believe the unsound behavior can be explained by the fact that the BM25 retrieval model uses the number of documents as one of its parameters. This was implemented in the system as the size of the target node set (i.e., the size of the set that needs to be ranked). We believe implementing BM25's N (the number of documents) by taking the size of the set to be ranked is the only sensible thing to do. We cannot take a predefined N, because the set might be the result of a complex selection query, possible combining for instance article elements and report elements and then restricting them on some other criterion.

The size of the sets, however, differs depending on the query plan used. This is even the case in more simple queries such as `//article//section[. ftcontains "xml"]`: If the system first selects the sections that are contained by articles (excluding the sections contained by reports in our test data) then the size of the set to be ranked is obviously smaller than the size of the complete set of sections as represented by the query `//section[. ftcontains "xml"][./ancestor::article]` Interestingly, all *tf.idf* term weighting algorithms use the number of documents to be ranked in their definition. The results indicate that all such approaches would produce unsound rankings.

We did not detect unsound ranking for BM25 if score combination uses the product or the maximum. If the maximum is used for score combination, then the approach would often ignore the BM25 score, so this approach is useless in practice. We are unable to explain the behavior when score combination uses the product: It might be an artefact of the data. This needs to be analyzed in the future.

Lessons for Case 3: Score propagation – up

In almost all cases that used the BM25 retrieval model, we detected unsound ranking behavior of the queries on the test data, except when score combination uses the maximum. As above, we have strong indication that this is due to the use of the size of the set that needs to be ranked in the definition of the model, which differs depending on the query plan used.

Lessons for Case 4: Score Combination – union

Unsound ranking was detected if score aggregation uses the average score of all matching context nodes. This might be due to the fact that taking the average function is not associative, and produces different values depending on the order in which it is evaluated.

Lessons for Case 5: XQuery FT properties

By design, the ranking approaches using ranking semantics (half of the approaches) do not adhere to the XQuery Full-text standard, or at least they are not in the spirit of the standard. A bigger problem might be the restriction that scores should be between 0 and 1. The retrieval models NLLR and BM25 produced scores greater than 1 in all cases, the score aggregation that uses the sum of scores also produced scores greater than 1 in most cases. The score combination that uses the sum of scores produced scores greater than 1 in some cases.

Overall lessons learned

If we only consider ranking approaches that: 1) did not produce unsound rankings; 2) did never produce scores greater than 1; and 3) use matching semantics, then only 3 approaches remain: These three approaches use the language model without smoothing (LM), multiply for score combination and either product, minimum or maximum for score aggregation. Whereas approaches based on language models without smoothing *might* be sound, it is likely that the search quality of the systems is below average: It is well-known, that smoothing is important for getting high quality retrieval results.

If we drop the requirement that scores should never by greater than 1, then 4 ranking approaches remain. Again, all of them use the LM retrieval model.

If we however drop the requirement that scoring should uses matching semantics, then 13 ranking approaches remain, among which several approaches use the language model with smoothing (LMS).

6. CONCLUSIONS

We report the behavior of 200 ranking approaches to ranking content-and-structure queries on pairs of queries for which we expect equal ranking results from the query semantics. We show that most of these approaches are not sound, i.e., they fail to produce equal rankings in the cases studied. Of the remaining approaches, only 3 adhere to the W3C XQuery Full-Text standard, which requires so-called matching semantics, and which requires retrieval scores to be smaller or equal than 1 at all times. The difficulties in implementing effective and sound ranking for XQuery Full-Text might affect its acceptance as a standard in the future.

Is ranking really necessary?

The XQuery Full-Text standard was largely motivated by complex retrieval queries. The XQuery Full-Text Use Cases [3] discuss for instance querying across element boundaries, wild cards, stop words, stemming, sensitivity to diacritics, cardinalities, existential quantification, proximity, implicit sentences and paragraphs, the use of thesauri, etc. Only a small part of the Use Cases consider scoring. So, maybe scoring is not really necessary in XML search?

Table 1 presents the average precision at 10 elements retrieved over 114 NEXI queries provided by the INEX 2006 evaluation. If we treat each `about()`-clause in NEXI as a Boolean OR (this is the default behavior of XQuery Full-text), then 97 out of 114 queries do not find any relevant

Approach	P@10	queries failed?
Boolean OR	0.053	97 (85 %)
Boolean AND	0.200	59 (52 %)
LMS/MULT/MAX/Matching	0.361	22 (19 %)
BM25/ADD/MAX/Matching	0.379	18 (16 %)
LMS/MULT/MAX/Matching/prior	0.439	15 (13 %)

Table 1: Retrieval quality on 114 INEX 2006 content-and-structure queries

element in their top 10, the average precision at 10 being 0.053. The best language model and BM25 approaches tested in this paper reduce the number of failed queries to respectively 22 and 18 (about five times less errors) and respectively 0.361 and 0.379 average precision at 10 (about 500% performance increase). If we add an element length prior to the language modeling approach (longer elements are more likely to be relevant; this was not tested for soundness in this paper), then the number of failed queries is down to 15 and the average precision is up to 0.439 (more than 700% improvement in performance). Clearly, a system without ranking is useless compared to the system that includes high quality ranking algorithms.

Other effective ranking techniques

We ignored many effective ranking techniques in this paper, for instance techniques using spans of words to handle the proximity queries defined in the XQuery Full-Text standard, but also the element length priors mentioned in the previous paragraph which are not covered by the standard. Our current definition of soundness (two Full-Text queries are semantically equal if and only if their XQuery representations produce the same results) does not directly provide ways to reason about the soundness of ranking given these techniques and/or options. For future research, we hope to provide a definition of soundness that does not refer to a non-Full-Text version of the query, but instead allows us to reason about the soundness of ranking in XQuery Full-Text directly.

Acknowledgments

This work was funded in part by the Dutch BSIK research program MultimediaN: Semantic Multimedia Access. Many thanks to participants of the Dagstuhl Seminar "Ranked XML Querying" [5] for helpful comments and discussions.

7. REFERENCES

[1] S. Amer-Yahia, C. Botev, S. Buxton, P. Case, J. Doerre, M. Holstege, J. Melton, M. Rys, and J. Shanmugasundaram. XQuery 1.0 and XPath 2.0 full-text 1.0. Technical report, World Wide Web Consortium, May 2008. http://www.w3.org/TR/xpath-full-text-10/

[2] S. Amer-Yahia, C. Botev, and J. Shanmugasundaram. Texquery: a full-text search extension to XQuery. In *Proceedings of the 13th international World Wide Web conference*, 2004.

[3] S. Amer-Yahia and P. Case. XQuery and XPath Full Text 1.0 use cases. Technical report, World Wide Web Consortium, May 2008. http://www.w3.org/TR/xpath-full-text-10-use-cases/

[4] S. Amer-Yahia, L.V.S. Lakshmanan, and S. Pandit. Flexpath: flexible structure and full-text querying for XML. In *Proceedings of the 2004 ACM SIGMOD international conference*, 2004.

[5] S. Amer-Yahia, D. Hiemstra, T. Roelleke, D. Srivastava, and G. Weikum. DB & IR Integration: Report on the Dagstuhl Seminar "Ranked XML Querying". In *SIGMOD Record 37*, 2008. (to appear)

[6] F.J. Burkowski. Retrieval activities in a database consisting of heterogeneous collections of structured text. In *Proceedings of the 15th ACM Conference on Research and Development in Information Retrieval (SIGIR'92)*, pages 112–124, 1992.

[7] D. Carmel, Y.S. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer. Searching XML documents via XML fragments. In *Proceedings of the 26th ACM Conference on Research and Development in Information Retrieval (SIGIR'03)*, pages 151 – 158, 2003.

[8] N. Fuhr and K. Großjohann. XIRQL: A query language for information retrieval in XML. In *Proceedings of the 24th ACM Conference on Research and Development in Information Retrieval (SIGIR'01)*, pages 172–180, 2001.

[9] T. Grabs. Generating vector spaces on-the-fly for flexible XML retrieval. In *Proceedings of the SIGIR workshop on XML and Information Retrieval*, pages 4–13, 2002.

[10] D. Hiemstra and W. Kraaij. Twenty-One at TREC-7: Ad-hoc and cross-language track. In *Proceedings of the seventh Text Retrieval Conference TREC-7*, pages 227–238. NIST Special Publication 500-242, 1998.

[11] D. Hiemstra, H. Rode, R .van Os, and J. Flokstra. PF/Tijah: Text search in an XML database system. In *Proceedings of the 2nd International Workshop on Open Source Information Retrieval*, 2006.

[12] W. Kraaij. *Variations on Language Modeling for Information Retrieval*. PhD thesis, University of Twente, 2004.

[13] M. Lalmas. Dempster-Shafer's theory of evidence applied to structured documents: modelling uncertainty. In *Proceedings of the 20th ACM Conference on Research and Development in Information Retrieval (SIGIR'97)*, pages 110 – 118, 1997.

[14] M. Lalmas and A. Tombros. Evaluating XML retrieval effectiveness at INEX. *SIGIR Forum 41*(1):40–57, 2007.

[15] V. Mihajlovic. *Score Region Algebra: A flexible framework for structured information retrieval*. PhD thesis, University of Twente, 2006.

[16] V. Mihajlovic, H.E. Blok, D. Hiemstra, and P.M.G. Apers. Score region algebra: Building a transparent XML-IR database. In *Proceedings of the 14th International Conference on Information and Knowledge Management (CIKM)*, pages 12–19, 2005.

[17] S.E. Robertson, S. Walker, M. Beaulieu, M. Gatford, and A. Payne. Okapi at TREC-4. In *Proceedings of the 4th Text Retrieval Conference (TREC-4)*, pages 73–97. NIST Special Publication 500-236, 1995.

[18] Andrew Trotman and Börkur Sigurbjörnsson. Narrowed Extended XPath I (NEXI). In *Advances in XML Information Retrieval, Third International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, pages 16–40, 2004.

Experiments and Evaluation of Link Discovery in the Wikipedia

Wei Che (Darren) Huang
Faculty of IT
Queensland University of Technology
Brisbane, Australia
w2.huang@student.qut.edu.au

Andrew Trotman
Department of Computer Science
University of Otago
Dunedin, New Zealand
andrew@cs.otago.ac.nz

Shlomo Geva
Faculty of IT
Queensland University of Technology
Brisbane, Australia
s.geva@qut.edu.au

ABSTRACT

Collaborative knowledge management systems such as the Wikipedia are becoming ever more popular – and these systems typically contain hypertext links between documents. The Wikipedia offers both manual and automated link creation. In fact several different systems providing links for Wikipedia documents now exit. Problematically the quality of automatically generated links has never been quantified. An evaluation method for Wikipedia link discovery approaches is essential.

We introduce the Link-the-Wiki task launched at INEX in 2007. 90 documents were orphaned from the collection and participants were required to build systems that identified the missing links. The different automated link discovery techniques used by participants are outlined. Details of two successful techniques are given, one using the titles of pre-existing documents to identify anchors and destinations, the other using pre-existing links between documents to identify possible links in new documents. In this paper, we mainly focus on the analysis and assessment of Wikipedia link discovery and discuss possible future evaluation techniques.

We examine one system in further detail and conduct a scalability experiment in which 1% of all Wikipedia documents were used and the performance studied in detail – link discovery in this system is shown to be scalable.

Finally, potential research directions for link discovery, assessment and evaluation are discussed.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms: Measurement, Experimentation

Keywords: Wikipedia, Link-the-Wiki, INEX, Assessment, Evaluation, Information Retrieval, XML IR

1. INTRODUCTION

1.1 Motivation

The goal of collaborative hypertext knowledge management (as seen, for example, in the Wikipedia) is to interlink all related knowledge. This helps users realize their particular information need, regardless of their level of understanding, by allowing them to click between text of entries expressing different and related concepts at different and related depths of coverage. Without hypertext links a user must search and browse (or otherwise navigate) into requisite content in order to expand their understanding. It is utterly inconvenient for the user to repeatedly search the collection simply to find content related to their core need. Worse, sometimes the content is not easily reachable using navigational facilities provided by the knowledge management system.

Links between pages are essential for navigation, but most systems require authors to manually identify each link. Authors must identify both the anchors and the target page in order to place a link. This creates a heavy and often unnecessary burden on content providers [1] who should focus on the content and not on the relationship between their content and content already in the collection. As the size of the collection increases the task of manually identifying links can become unmanageable. The maintenance cost of keeping all links up to date is huge – and the Wikipedia has seen faster than linear growth for many years. Authors are typically unaware of all pre-existing content to which they might links, and even if they are they are unlikely to be aware of content created concurrently with their page. Page maintenance, in particular, linking to content added after a page is created is a burden to content providers who often do not maintain their content (hence the collaborative nature of these information resources). Worse, Ellis et al. [2] have shown significant differences in the links assigned by different people.

Several systems (such as the Wikipedia) support simple text-search facilities to help content providers identify anchors and links. External search engines such as Google, Qwika, Lycos and Yahoo can also be used to search well established knowledge management sites [3].

There are further problems! Linking is still typically performed between documents even though some documents are long and a better destination might be an anchor within a document. Link discovery methods have not yet been integrated into even the most successful systems (such as the Wikipedia). Links outside the closed system (i.e. to the web) are also manually added. There are many inaccurate and unnecessary links added to documents. And link spam is beginning to surface.

To eliminate the human effort required to build a highly accurate hyperlink-link network, to reduce the chance of erroneous links, and to keep links up-to-date, automatic link discovery mechanisms are needed.

Herein we concentrate on the Wikipedia because of its success and because of the availability of the INEX Wikipedia document collection. In particular we discuss the Link-the-Wiki track held at INEX 2007 in which automated link discovery systems were

SIGIR 2008 workshop on Focused Retrieval
July 24, 2008, Singapore

solicited from participants and judged against human created hypertext links for 90 documents.

The techniques used by each participant are discussed and contrasted, and then the results of the top two performing groups are analyzed in detail. We find that document-to-document link discovery systems are very good at exhibiting high precision levels at most points of recall, systems are scalable and that several different techniques might be used. This result motivates us to examine (and outline future work in) anchor to Best-Entry-Point (BEP) identification. We discuss assessment and evaluation of this new focused retrieval task is in detail.

1.2 Related Work

As suggested by Wilkinson & Smeaton [1], navigation between linked documents is a great deal more than simply navigating multiple results of a single search query, linking between digital resources is becoming an ever more important way to find information. Through hypertext navigation, users can easily understand context and realize the relationships of related information. However, since digital resources are distributed it has become difficult for users to maintain the quality and the consistency of links. Automatic techniques to detect the semantic structure (e.g. hierarchy) of the document collection and the relatedness and relationships of digital objects have been studied and developed [4]. Early works, in the 1990s, determined whether and when to insert links between documents by computing document similarity. Approaches such as term repetition, lexical chains, keyword weighting and so on were used to calculate the similarity between documents [5, 6, 7]. These approaches were based on a document-to-document linking scenario, rather than identifying which parts of which documents were interrelated.

Several conferences and workshops (in particular at SIGIR and LinkKDD) focused on link analysis and discovery. Most recently the Link-the-Wiki track at INEX required participants to build systems that discover potential anchors (representing the content of topics) and relevant destinations (Best Entry Points within a document) for each anchor [8, 9]. The details of this track are briefly described in Section 2.

The link-network within Wikipedia is only valuable if it is maintained and all links are up-to-date, this is especially a problem in the case of a newly created article that should be linked to by pre-existing pages. Links in each document can be within the Wikipedia or other web resources outside the Wikipedia [10] so the document collection can never be closed. Although there are many methods in modern IR that can be applied to facilitate search, few experiments have been done in collaborative semantic linking [11].

Adafre & de Rijke [12] identify most links in the Wikipedia as conceptual. The Wikipedia link-network offers hierarchical information and links aim to expand the concepts in their anchors. The anchors imply the concept while the links are complementary to the concept. Since there is no strict standard of editing there are problems with *over linking* and *missing links*. Adafre & de Rijke proposed a method of discovering *missing links* in Wikipedia pages by clustering topically related pages using LTRank and identified link candidates by matching anchor texts. Page ranks using the LTRank method are based on the co-citation and page title information. Experimental results showed a reasonable outcome.

Jenkins (2007) developed a link suggestion tool, *Can We Link It*. This tool identifies anchors within a document that have not been linked and that might be linked to other pages [13]**.** Using this tool, the user can accept, reject, or *"don't know"* to leave a link as undecided. This tool also lets the user add links back to Wikipedia document.

A collaborative knowledge management system, called *PlanetMath*, based on the Noosphere system has been developed for mathematics [14]. It is encyclopedic, (like the Wikipedia), but mainly used for the sharing of mathematical knowledge. Since the content is considered to be a semantic network, entries should be cross-referenced (linked). An automatic linking system provided by Noosphere employs the concept of conceptual dependency to identify each entry for linking. A classification hierarchy used in online encyclopedias is used to improve the precision of automatic linking. In practice, the system looks for common anchors that are defined in multiple entries and creates links between them, once the page metadata is identified as related. Based on the Noosphere system, NNexus (Noosphere Networked Entry eXtension and Unification System) was developed to automate the process of the automatic linking [15]. This was the first automatic linking system to eliminate the linking efforts required by page authors. Declarative linking priorities and clauses are specified to enhance linking precision. An approach, called *invalidation index*, was developed to invalidate entries belonging to those concepts where there are new entries. Reputation based collaborative filtering techniques could be used to provide personalized links.

Research on the Wikipedia has been undertaken in recent years. In order to find cultural biases, network analysis algorithms such as HITS and PageRank have been used [16]. Based on Markov Chains [17], a set of experiments for finding related pages within the Wikipedia collection was undertaken using two Green-based methods [18], *Green* and *SymGreen*, and three classical approaches, *PageRankOfLinks*, *Cosine with tf-idf* and *Co-citations*. The results show the Green method has better performance at finding similar nodes than only relying on the graph structure. Although page titles and category structure can be used to classify documents, properties such as the internal text of the articles, the hierarchical category, and the linking structure should be used [19]. *Wikirelate* proposed by Strube & Ponzetto [20] uses *Path*, *Information content* and *Text overlap* measures to compute the semantic relatedness of words. These measures mainly rely on either the texts of the articles or the category hierarchy. Gabrilövich & Markövitch [21] introduce a new approach called Explicit Semantic Analysis (ESA), which computes relatedness by comparing two weighted vectors of Wikipedia concepts that represent words appearing within the content. Common to this research is the use of the *existing* linking structure and content (category, etc.); we are interested in developing approaches to generate *new* links.

Various link-based techniques based on the correlation between the link density and content have been developed for a diverse set of research problems including link discovery and relevance ranking [12]. Moreover, communities can be identified by analyzing the link graph [22]. Beside co-citation used by Kumar et al. [23] to measure similarity, bibliographic coupling and

SimRank based on citation patterns, and the similarity of structural context (respectively), have also been used to identify the similarity of web objects [24]. The companion algorithm derived from HITS has also been proposed for finding related pages (by exploiting links and their order on a page) [25, 26].

The assessment of results has been a challenge in IR experiments for many years because there is no standard procedure, relevance is hard to define and cross-assessor agreement levels are often low (so individual judgments come under dispute). Worse, it is difficult to compare IR methods which are able to retrieve highly relevant documents with those that retrieve less relevant documents because assessments are usually binary. The use of Precision-Recall curves is typical in IR; however, Schamber [27] argues that traditional P-R based comparison using binary relevance cannot adequately capture the variability and complexity of relevance. Relevance is a multilevel circumstance where, for a user, the degree of relevance may vary from document to document.

Several studies have examined components that influence judgments and the criteria of relevance (including graded relevance) in information seeking and retrieval [28]. Kekalainen and Jarvelin [29] argue that evaluation methods should be flexible enough to handle different degrees of judgment scales. They proposed generalized precision and recall that can incorporate a continuous relevance scale into the traditional precision and recall measures. Their experiments demonstrate that the evaluation approach can distinguish between retrieval methods fetching highly relevant documents from those retrieving partially relevant documents.

2. INEX LINK-THE-WIKI TRACK

The Wikipedia is composed of millions of interlinked articles in numerous languages and offers many attractive features for retrieval tasks [30]. The current INEX Wikipedia collection contains a snapshot of the Wikipedia English collection from 2006 and contains 660,000 documents and is about 4GB in size. In INEX 2007 the linking task used 90 documents (topics), nominated from the existing collection by participants [31]. Topic nomination was preferred over random selection because some documents contain very few links (because, for example, they are very short). The topic-documents were removed from the collection (as were links to and from the documents) and treated as if new. The task was to identify a set of incoming and outgoing links to and from these orphaned documents together with the corresponding anchor text within the orphaned documents.

2.1 Assessment and Evaluation

There are two challenges in the LTW track at INEX. The first is to identify a set of text anchors that may semantically be linked to other pre-existing documents – these are candidate outgoing links. The second is the identification of candidate incoming links from other Wikipedia pages into the new document. Several natural language use issues such as synonymy and multiple meanings may cause anchor text inaccuracies and deficiencies. For example, the term *IR* can mean *Information Retrieval* or *Information Registry*, depending on context. Should *Modern Information Retrieval* or just *Information Retrieval* be highlighted as a term when both articles exist in the collection? As an aside, of course both could be linked, but unfortunately the Wikipedia interface (the web) does not currently (easily) support multiple links per anchor.

It is important to rank the discovered links for a user's selection, but it is not immediately clear how this should be done. A typical scenario might involve a user who wishes to inspect and then accept or reject recommended links. The user is unlikely to go through hundreds of potential anchors. Therefore, the most likely anchors should be presented first. Furthermore, even with automated linking the system must balance extensive linking against link quality. Ranking is necessary in order to determine a cut off point in link recommendation.

It is essential to define a standard methodology and procedures to assess link quality and to quantitatively evaluate and compare different approaches.

INEX 2007 used a variation of the Cranfield methodology. We have already discussed topic selection. From the topics we automatically generated the assessments (the ground-truth). Because the topic-documents were extracted from the existing Wikipedia collection, links both into the collection and from the collection already exist. These were used as the ground-truth, and were then eradicated from both the topic and the collection before the topics were distributed. This ground truth was not ideal (as we shall discuss later) but nonetheless reasonable, as it was what that was in the Wikipedia at the time.

Constructing the assessments in this way resulted in no manual assessment effort, and facilitated the evaluation of systems with a very large number of topics. Participating search engines were explicitly forbidden from using the existing links to and from the topics (the whole collection was used in the INEX ad hoc track complete with links) although links within the collection that were unrelated to the topics could be used. Participant's search engines returned ranked lists of possible incoming and outgoing links for each topic. Evaluation was carried out using MAP, R-Prec and P@R. Incoming and outgoing links were evaluated separately.

This kind of automatic generation of link-assessments is applicable only to document-to-document link discovery because these are the only kinds of links that exist within the collection. Because of this INEX 2007 limited link discovery to document-to-document linking.

The goal of the task is to perform focused retrieval. That is, to link anchors in one document to focused units (e.g. *sections*, *images*, *elements*, or *passages*) in another. An anchor link click should ideally lead a user not only to a relevant document, but also to the best entry point within that document with respect to the anchor context. This requires far more elaborate assessment and evaluation and is discussed later in this article.

2.2 The Quality of Wikipedia Links

Although we treat the Wikipedia links as the ground-truth, they are obviously not perfect. Some links in the Wikipedia are already automatically generated and the validity is questionable. *Year* links, for example, are very often unrelated to the content of the document, but are easy to discover. Problematically they may also lead to optimistic evaluation results when identified by link-discovery systems using automatic assessment generation techniques such as we describe. Many potentially good links that have not been identified by Wikipedia users are amenable to automatic discovery – but will not be scored using automatic assessment generation. Such useful returned links which are missing from the ground truth could result in poor evaluation scores for highly effective link discovery systems, leading to pessimistic evaluation results. So although it is not possible to quantify the absolute performance using automated assessment, the procedure we used provides a trade-off between assessment effort (essentially none) and absolute accuracy of measurement.

It is a reasonable to conjecture that *comparative evaluation* of methods and systems is still informative. Through *comparative* analysis of automated linking systems, it should remain possible to improve link discovery methods.

3. WIKIPEDIA LINKS

Links in the Wikipedia can be classified into several types. Crudely, they can be divided into linking within Wikipedia and outside web links. Less crudely:

- Linking to an article which has the exact same name as an anchor.
- Linking to an article which has a different name from the anchor, we identify the following kinds:
  - **Synonyms**. Linking to a page whose name has the same meaning as the anchor but different spelling. For example, the word "gods" in the following sentence, *The elves were originally imagined as a race of minor nature and fertility gods*, is linked to the page named *Deity*.
  - **Tense**. The past tense of a word may be linked to a page name as its present tense or its noun form. For example, the "pluralized" in the sentence, *Elf can be pluralised as both elves and elfs*, is linked to the page name *plural*.
  - **Presenter**. A name of an entity may link to its related presenter such as the singer of a song or the director of a film.
  - **Language**. Some old language characters (e.g. Latin and Old Norse) may be linked to related English words. For example, the word "*Ljósálfar*" in the sentence, *he also based them on the god-like and human-sized Ljósálfar of Norse mythology*, is linked to the page *Light elf* which is in turn redirected by Wikipedia to a page titled *Light elves*.
  - **Definition**. Some anchors are linked to the related page names that may express the meaning of the anchor. For example, the word "good" in the following sentence, *They are great smiths and fierce warriors on the side of good*, is linked to the page "*Goodness and value theory*" with the title "*value theory*".
  - **Disambiguation**. Some links are redirected to a page that lists possible linking candidates. For example, the anchor "Moving Pictures" is linked to the page *Moving Pictures* that lists a serious of related pages (e.g. *Moving Pictures (album)*, *Moving Pictures (novel)*, *Moving Pictures (song)* and *Moving Pictures (band)*).
- An anchor may be linked to a page that integrates several similar pages. Anchors that link to these "similar" pages are later redirected to the new integrated page.
- Anchors may be in the *references* section: these are anchors that link to destinations either inside or outside Wikipedia.
- Anchors in the *See Also* section: this is a list of related topics that link to Wikipedia pages.
- *External Link* Anchors: there is also a list of related topics that link to pages outside Wikipedia (that is, to the web).

Problematically, if most page names exactly match an anchor text, we can produce a simple method that systematically matches potential anchor strings with page names to identify most links – and achieve a recall of near 1.

We examined the 90 LTW topics from INEX 2007 and found that in 81 of the 90 topics at least 50% of the links match an existing page name (see Table 1). This could be because the links were generated through careful construction by a user, or automatically by matching page names, either way such links are relatively easy to find. Although this implies that we can expect high recall from simple page-name matching strategies, it does not necessarily mean that we can expect high precision – many matching links are not relevant (for example, polyvalent terms). As the Wikipedia is a huge repository of definitions it is relatively easy to find matching page names which are not relevant.

Ratio of Match	Number of Topics
90% ~ 100%	1
80% ~ 90%	8
70% ~ 80%	26
60% ~ 70%	35
50% ~ 60%	16
40% ~ 50%	2
30% ~ 40%	2

Table 1: Ratio of matching names between anchors and links

4. APPROACHES TO LINK-THE-WIKI

In this section we briefly describe the approaches that were taken by the Link-the-Wiki participants.

The University of Amsterdam system assumed that Wikipedia pages link to each other when articles are similar or related in content. For each of the 90 topics, the system queried the index of the entire collection, (excluding the topics). This was done by using the full topic as the query, but excluding stop words, and with important terms derived from a language model. The top 100 files (anchors) were selected for each topic. They experimented with line matching from the orphans to the anchor files. For the outgoing links, the system matched each line of a topic with the lines of the anchors until a matching line was found. For the incoming links, the system iterated over all lines of each anchor for each line of the topic. The generated runs were based on the names of the pages, exact lines, and longest common substrings (LCSS) expanded with WordNet synonyms. The results show that the run based on restricting the line matching to the names of pages performed best.

The University of Otago system identified terms within the document that were over represented by comparing term frequency in the document with the expected term frequency (computed as the collection frequency divided by document frequency). From the top few over-represented terms they generated queries of different lengths. A BM25 ranking search engine was used to identify potentially relevant documents. Links from the source document to the potentially relevant documents (and back) were constructed. They showed that using 4 terms per query was more effective than fewer or more. The Otago system was effective at early recall but not overall.

The University of Waterloo system found the first 250 documents (in document collection order) that contain the topic titles and then generated article-to-article Incoming links. For outgoing links, they performed link analysis. The system computed the probabilities that each candidate anchor would be linked to a destination file. The probability that a candidate anchor would be linked was computed (essentially) as the ratio of the number of times that the anchor text was actually linked in the collection, to the number of times that the anchor text appeared in the collection.

The Queensland University of Technology (QUT) system identified incoming links using a ranked search for documents that were about the new document title. Outgoing links were identified by running a window over the new document text and looking for matching document titles in the collection. The window size varied from 12 words down to 1 word, and included stop words. Longer page names were ranked higher than shorter page names, motivated by the observation that the system was less likely to hit on a longer page name by accident.

The best performing approaches were those that used either existing anchors to predict suitable anchors (Waterloo), or matching document titles to predict suitable anchors. The performance of these 2 approaches[1] is depicted in Figure 1. Both approaches produce a very good result with high precision over a wide range of recall levels. This is precisely the kind of performance needed to satisfy a user.

5. EVALUATION RESULTS

In this section we concentrate on the two most successful approaches at INEX 2007 [31, 32], those of Waterloo and QUT.

5.1 Anchor vs. Page Title Link Discovery

There are considerable differences between the two approaches. The Waterloo approach relies on the availability of an extensive pre-existing web of anchor to document links in the collection. This pre-requisite may not always be satisfied, particularly when a new cluster of documents in a new domain is added to the collection in bulk, or when a new Wikipedia-like resource is created. However, the approach can discover links that are not solely based on a match between anchor text and a document title. If an anchor is frequently linked to a document with a different title, it will become a highly probable link. For instance, the Waterloo system was able to link *Educational Philosophy* to a document titled *The Philosophy of Education*. By contrast, the QUT approach only discovered matching document titles. Although the performance of QUT is somewhat lower, the approach is applicable to any collection, regardless of the pre-existing link structure. It could immediately be applied to any document collection, new or pre-existing.

[1] The graphs shown in this paper for the participating systems were generated after INEX 2007 and after the participants had fixed bugs and implemented corrections. The results we present will, therefore, not match those reported at INEX.

Figure 1 presents the precision-recall curves for the two systems. "Anchors 90" is the Waterloo system and "Page Titles 90" is the QUT approach. Both are shown for the 90 INEX topics. The anchor-based approach is better at almost all recall points.

5.2 Scalability of Link Discovery

To test the scalability of automated link discovery we additionally ran an extensive experiment on the collection. We randomly extracted 1% of the 660,000 documents and re-ran the experiment. So-far only QUT have provided results.

The QUT experiment was run on a PC with 2GB memory and 1.6GHz clock speed. It took 6 minutes to complete the process, processing in excess of 1,100 documents per minute. Figure 1 also presents the recall-precision curve for that run. It can be seen that performance over a very large number of topics selected at random is similar to the performance achieved over the INEX set, suggesting that 90 topics is sufficient to measure the performance of such systems. This result suggests that the manual choice of topics for INEX 2007 was not biased – which further suggests that topics can be randomly chosen in future years (thus further reducing the cost of assessing such systems for a document-to-document linking scenario).

Importantly, it is feasible to manually assess 90 topics whereas it is not be feasible to assess 6,600 using the resources available to INEX. Manual assessment would allow us to study more deeply the nature of link discovery – to identify those links returned by automatic systems that have not been identified by Wikipedia authors. It would also allow us to identify links that are already in the Wikipedia but which are not useful (e.g. *year* links are common, yet often of little use).

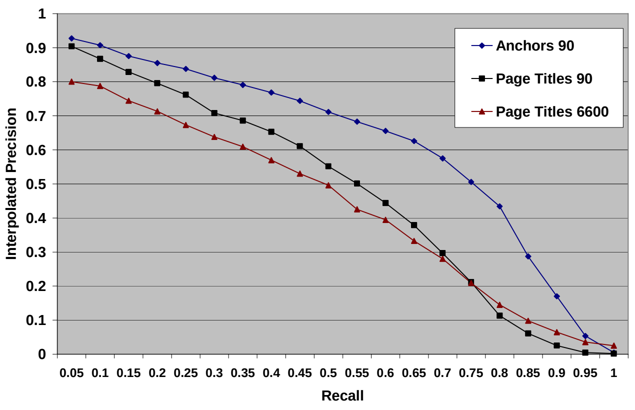


Figure 1. Scalability test: Differences in performance topic sets is likely to be human bias in topic choice

5.3 Page Name Based Link Discovery

It is straightforward to obtain candidate anchors by systematic comparison of substrings (of various lengths) against exiting page titles in the collection. Numerous matches arise, not all useful, so a pruning strategy is needed. QUT adopted the following (effective) strategy:

- Identify all candidate phrase-anchors of length 12 words down to 2, in that order.
- Append candidate year anchors
- Append all single term anchors

No ordering was performed other than the above. Phrases were ordered by length, followed by years, followed by single terms. Within these groups the ordering was in the sequence in which the anchors were encountered.

QUT found that it is possible to improve their result by re-ordering the combined single-term and year anchors by the probability of the word being an anchor. This probability is estimated as the ratio of the number of times that the named page had been linked to, to the number of times that the page name appears in the collection (the collection frequency). Alternatively they used the number of documents in which an anchor text appears (document frequency). The performance degraded when phrase anchors were used in re-ordering – it appears that the phrase match heuristic is more useful than the estimated phrase anchor probability. Only short single term candidate anchors were ordered by the probability of being linked.

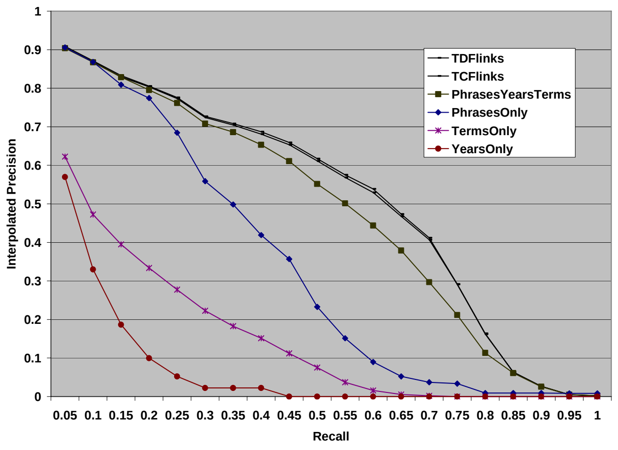


Figure 2: Linking strategy comparison

In order to assess the contribution of each component (phrase, year, and term), QUT created separate submissions for each component. Figure 2 presents the recall-precision curves. Most surprisingly the contribution of the year links is small; they are ubiquitous throughout Wikipedia and were expected to contribute considerably to performance. Single terms contribute more than years over all because there are many more terms that could be linked. However, it is difficult to avoid irrelevant links using only single terms. The phrase links achieve higher precision and recall than terms and years. This is because phrases (long phrases in particular) that match a page name are highly unlikely to occur in a document without also being related to the page of the same name. Both years and single terms frequently match a page name, but not in the correct context. The combination of phrases, years, and terms is very effective as can be seen from the combined curve. The ranking of single terms by probability of being an anchor provides further improvements. The top 2 curves in Figure 2 correspond to these variations. The improvement is only marginally greater when using the document frequency in place of collection frequency.

6. FOCUSED LINK DISCOVERY

The INEX Link-the-Wiki track in 2007 called for document-to-document linking. The goal of the task is to find links that point not only to a relevant document, but also to a location within that document from which a user should start reading in order to satisfy their information need. This location is called a Best Entry Point, or BEP [33]**.**

Furthermore, it is reasonable to expect to see anchors that could point to multiple locations. For instance, there may be numerous pages about *Education Theory*, not just a single overview page by that name. It may be necessary to impose some limit on the number of links per document, (and the number of links per anchor) to avoid linking every word of every document to another document. Just because the Wikipedia (or rather, a standard web browser) currently does not support the presentation and handling of multiple links per anchor it does not mean that we cannot or should not explore this scenario.

In future INEX evaluations the task will be defined as anchor to BEP link discovery, and allow multiple links per anchor (actually, the latter is essential for manual evaluation purposed where two systems might link the same anchor to different document, both of which are relevant). Traditional performance measures such as MAP (Mean Average Precision) will be adapted to address the performance differences of link-discovery methods in this new scenario.

Automated generation of assessments (the INEX 2007 model) produced an incomplete and biased ground truth, biased on what users did, not what they might (or should) have done. This bias is not dissimilar from that seem with relevance feedback experiments in which a user can only improve on results they have seen, and is not able to identify better and more relevant documents they have not seen. This problem was already explored in Section 2.2. Furthermore, the Wikipedia links do not have anchor-to-BEP functionality, nor do they have multiple links per anchor. Therefore it will not be possible to automatically generate assessments for evaluating anchor-to-BEP runs. For this it is necessary to employ a manual assessment procedure as well as a revised evaluation strategy. We identify this as a necessary future direction for research, and currently study it. An assessment tool for facilitating effective inspection and binary relevance judgment of individual anchor-to-BEP links is outside the scope of this paper, however we also currently study this. In what follows we assume that such a tool exists and that such relevance judgments of links will be available.

6.1 Proposed Evaluation Procedure

In automated link discovery there are two simultaneous ranking requirements: first a candidate list of anchors, second a candidate list of target documents for those anchors. In order to derive a single performance score over all proposed anchors and targets, the performance of each must be combined.

A suitable form for a document score may be:

$$A = MAP(links) \qquad (1)$$

Where *A* is the single anchor score, defined as the mean average precision. For evaluation purposes runs will be of finite length so MAP will be computed up to that point of recall.

We must also allow for anchors to be matched with some flexibility. An anchor may be defined in several slightly different ways. For instance, *The Theory of Relativity*, *Theory of Relativity*, and *Relativity* may well be conceptually identical anchors. Furthermore, if the anchor text occurs several times in a document we would expect only one instance to be anchored (as, for example, is seen in the Wikipedia) and so the location of the anchor may vary without being logically incorrect (we leave for yet-further work the question of which occurrence of an anchor is best to choose). In deriving a relevance score for an anchor a match has to be defined as conceptual, requiring only some minimal term overlap with an anchor in the assessments. The same kind of problem is seen in Question Answering where templates have been used to match correct answers, rather than document locations.

Similarly, a BEP cannot be defined with absolute accuracy. Some reasonable proximity to a designated BEP in the assessments should be allowed. So a BEP might be considered relevant if, when viewed on a screen, it is no more than some distance (*N* words) away from a point chosen by an assessor (INEX uses a similar scheme for scoring BEPs).

So in summary, an anchor-to-BEP link can be assessed as relevant on the basis of approximately matching both the anchor and the BEP of a relevant link in the assessments.

Having computed individual anchor-to-BEP link score the document score can be derived:

$$D = \sum_{anchors} f(P)_i A_i \qquad (2)$$

Where A_i is the score assigned to a particular anchor, and $f(P_i)$ is a monotonically decreasing function of the position of the BEP in the target document. The score can then be averaged over all topics in a run to provide the final run score.

7. CONCLUSIONS AND FUTURE WORK

As far as we are aware, the Link-the-Wiki task at INEX is the first to offer extensive reusable independent evaluation resources for link discovery. We have described this new evaluation task and then compared and contrasted the two most successful approaches submitted to Link-the-Wiki at INEX 2007. We further provided results of extensive linking experimentation with a very large set of documents (1% of the collection) and found that linking is feasible, effective, and scalable.

A fully automated procedure for document-to-document link analysis that costs virtually nothing to administer is described. The procedure was used at INEX 2007 and allowed us to create a fast evaluation procedure with a turnaround time of days and not months because no manual assessment was required. The procedure allows for a very large number of documents to be used in experiments, and we demonstrate this by using 6,600 documents for assessment. For link-discovery we have overcome the assessment bottleneck which is encountered in most other tasks in collaborative evaluation forums such as INEX and TREC.

We further proposed to extend the task to anchor-to-BEP link discovery, and to multiple links per anchor. We describe the requirements for evaluating such a task and propose an evaluation procedure that is derived from standard well established IR methodology of measuring MAP.

There is still much to explore in link discovery in Wikipedia. For document-to-document link discovery there was no demonstrated successful use of document similarity metrics to determine the appropriateness of a link. Sub-document similarity measures (as seen in ad hoc focused-retrieval experiments) are expected to be successful for BEP identification. That is, the similarity between the immediate anchor's context and the immediate BEP context. It is not necessary for whole documents to be highly related for valuable links to exist. For instance, a document on Information Retrieval which briefly refers to Latent Semantic Analysis may well link to a document which discusses Dimensionality Reduction. Although the two *documents* may not seem related, a *section* on latent semantic analysis in one document may link to a *section* on singular value decomposition in the other. There is ample scope for natural language processing technology to explore ways by which context similarity can be used to improve the accuracy of link analysis at a granularity well below whole document.

We believe we have solved the evaluation problem for document-to-document linking and currently explore the evaluation of anchor-to-BEP linking in the context of focused-retrieval.

8. REFERENCES

[1] Wilkinson, R. and Smeaton, A. F. (1999) Automatic Link Generation, *ACM Computing Surveys*, 31(4), December 1999.

[2] Ellis, D., Furner-Hines, J. and Willett, P. (1994) On the Measurement of Inter-Linker Consistency and Retrieval Effectiveness in Hypertext Database, *In Proceedings of the 17th Annual International Conference on Research and Development in Information Retrieval*, Dublin, Ireland, July 1994, 51-60.

[3] Wikipedia: Searching, 2007. http://en.wikipedia.org/wiki/ Wikipedia: Searching.

[4] Green, S. J. (1999) Building Hypertext Links By Computing Semantic Similarity, *IEEE Transactions on Knowledge and Data Engineering*, September/October 1999, 11(5), 713-730.

[5] Allan, J. (1997) Building Hypertext using Information Retrieval, *Information Processing and Management*, 33(2) 145-159.

[6] Green, S. J. (1998) Automated Link Generation: Can We Do Better than Term Repetition?, *In Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, 75-84.

[7] Zeng, J. and Bloniarz, O. A. (2004) From Keywords to Links: an Automatic Approach, *In Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)*, 5-7 April 2004, 283-286.

[8] Denoyer, L. and Gallinari, P. (2006) The Wikipedia XML Corpus, *ACM SIGIR Forum*, 40(1), June 2006, 64-69.

[9] Link the Wiki Track, *INitiative for the Evaluation of XML retrieval (INEX)*, 2007. http://inex.is.informatik.uni-duisburg .de/2007/linkwiki.html.

[10] Kolbitsch, J. and Maurer, H. (2006) Community building around encyclopedic knowledge, *Journal of Computing and Information Technology*, 14.

[11] Baeza-Yates, R. A. and Ribeiro-Neto, B. (1999) Modern Information Retrieval, *ACM Press Addison-Wesley*, 1999.

[12] Adafre, S. F. and de Rijke, M. (2005) Discovering missing links in Wikipedia, *In Proceedings of the SIGIR 2005 Workshop on Link Discovery: Issues, Approaches and Applications*, Chicago, IL, USA, 21-24 August 2005.

[13] Jenkins, N. (2007) *Can We Link It*, http://en.wikipedia.org /wiki/User:Nickj/Can_We_ Link_It.

[14] Krowne, A (2003) An Architecture for Collaborative Math and Science Digital Libraries, *Thesis for Master of Science Virginia Polytechnic Institute and State University,* 19 July 2003.

[15] Gardner, J., Krowne, A. and Xiong, L. (2006) NNexus: Towards an Automatic Linker for a Massively-Distributed Collaborative Corpus, *In Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing,* 17-20 November 2006, 1-3.

[16] Bellomi, F. and Bonato, R. (2005) Network Analysis for Wikipedia, *In Proceedings of the 1st International Wikipedia Conference (Wikimania'05),* Frankfurt am Main, Germany, 4-8 August 2005.

[17] Norris, J. R. (1997) Markov chains, *Cambridge Series in Statistical and Probabilistic Mathematics*, 2, Cambridge: Cambridge University Press.

[18] Ollivier Y. and Senellart P. (2007) Finding Related Pages Using Green Measures: An Illustration with Wikipedia, *In Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI'07),* Vancouver, Canada, 22-26 July 2007.

[19] Schönhofen P. (2006) Identifying decument topics using the Wikipedia category network, *In Proceedings of the 2006 IEEE/EIC/ACM International Conference on Web Intelligence (WI'06),* Hong Kong, 18-22 December 2006.

[20] Strube, M. and Ponzetto, S. P. (2006) WikiRelate! Computing Semantic Relatedness Using Wikipedia, *In Proceedings of the 21th National Conference on Artificial Intelligence (AAAI'06),* Boston, Massachusetts, USA, July 2006, 16-20.

[21] Gabrilovich, E. and Markovitch, S. (2007) Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis, *In Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07),* Hyderabad, India, 6-12 January 2007.

[22] Kumar, R., Raghavan, P., Rajagopalan, S. and Tomkins, A. (1999) Trawling the Web for emerging cyber-communities, *Computer Networks*, 31(11-16), 1481-1493.

[23] Jeh, G. and Widom, J. (2002) SimRank: a measure of structural-context similarity, *In Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'02)*, Edmonton, Canada, 23-26 July 2002, 538-543.

[24] Kessler, M. M. (1963) Bibliographic coupling between scientific papers, *American Documentation*, 14(10-25), 1963.

[25] Dean, J. and Henzinger, M. R. (1999) Finding related pages in the World Wide Web, *Computer Networks*, 1999, 31(11-16), 1467-1479.

[26] Kleinberg, J. (1998) Authoritative sources in a hyperlinked environment, *In Proceedings of the 9th Annual ACM–SIAM Symposium on Discrete Algorithms*, San Francisco, CA, USA, 25-27 January 1998, 668-677.

[27] Schamber, L. (1994) Relevance and Information Behavior, *Annual review of information science and technology 29,* Medford, NJ: Information Today, 3-48.

[28] Vakkari, P. and Hakala, N. (2000) Changes in relevance criteria and problem stages in task performance, *Journal of Documentation*, 56(5), 540-562.

[29] Kekäläinen, J. and Järvelin, K. (2002) Using graded relevance assessments in IR evaluation, *Journal of the American Society for Information Science and Technology*, November 2002, 53(13), 1120-1129.

[30] Wikipedia, the free encyclopedia, 2007. http://wikipedia.org/.

[31] Huang, W. C., Xu, Y., Trotman, A. and Geva, S. (2008) Overview of INEX 2007 Link the Wiki track, In Proceeding of INEX 2008 Workshop, Dagstuhl, Germany.

[32] Itakura, K. Y. and Clarke, C. L. A. (2007) University of Waterloo at INEX2007: Ad Hoc and Link-the-Wiki Tracks, *In Pre-proceedings of the INEX 2007 Conference,* Dagstuhl, Germany, 380-387.

[33] Reid, J., Lalmas, M., Finesilver, K. and Hertzum, M. (2005) Best Entry Points for Structured Document Retrieval – Part II: Types, Usage and Effectiveness, *Information Processing & Management*, 42(1), 89-105.

[34] D. Jenkinson, A. Trotman, Wikipedia Ad Hoc Passage Retrieval and Wikipedia Document Linking, *In Pre-proceedings of the INEX 2007 Conference,* Dagstuhl, Germany, 365-379.

[35] K. N. Fachry, J. Kamps, M. Koolen, J. Zhang, The University of Amsterdam at INEX 2007 *In Pre-proceedings of the INEX 2007 Conference,* Dagstuhl, Germany, 388-402.

An Analysis of Multi-Focus Questions

Chuan-Jie Lin
Department of Computer Science and Engineering
National Taiwan Ocean University
No 2, Pei-Ning Road, Keelung 202, Taiwan R.O.C.
+886-2-24622192 ext. 6610
cjlin@ntou.edu.tw

Ren-Rui Liu
Department of Computer Science and Engineering
National Taiwan Ocean University
No 2, Pei-Ning Road, Keelung 202, Taiwan R.O.C.
+886-2-24622192 ext. 6610
freesia@cyber.cs.ntou.edu.tw

ABSTRACT

This paper proposes a new definition of question classification based on question focus. Multi-focus questions are defined as questions containing multiple foci. It would be better to present their answers in a table. Besides defining the focus degree, relations between question foci are also discussed. A multi-focus question can be decomposed into several subquestions. The characteristics of the subquestions determine the representation of the final answers. Several simulation experiments have been carried out to verify the importance of focus analysis for multi-focus questions in a QA system.

Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: Question-answering (fact retrieval) systems

General Terms

Theory, Design, Experimentation

Keywords

Multi-focus questions, focus degree, question analysis, question decomposition

1. INTRODUCTION

Question answering has been a hot research topic in recent years. Large scale QA evaluation projects such as TREC QA-Track [1], QA@CLEF [2], and NTCIR QAC and CLQA Tracks [3][4] greatly support the developments of QA techniques.

In recent QA research, three classes of questions, factoid, list, and complex questions, are generally studied. Factoid questions are commonly studied in famous QA evaluation tasks. They are often "fact-based, short-answer" questions as described in the overview of TREC QA Tracks [5]. For a factoid question, the required answer is often a short string, i.e. a name of an entity, a temporal expression, or a noun phrase, *etc*. For example, the answer to the factoid question "*in what year did the Titanic sink*" is a short temporal expression "*1912*".

List questions created in recent QA evaluation forums are also factoid questions. The difference is that a list question often expresses the multiplicity of its answers explicitly (e.g. "*what countries have been struck by Tsunamis*") or directly requests an amount of various answers (such as "*name 20 countries that produce coffee*"). Answers to a list question are often presented in a list. For example, the list "*Papua New Guinea, Indonesia, New Guinea, Japan*" is an answer to the list question "*what countries have been struck by Tsunamis*".

Complex questions, on the other hand, often refer to long-answer questions. It means that an answer to a complex question is often a longer passage, say, a set of sentences, a paragraph, or even an article. Questions asking about reason ("*why does the moon turn orange*"), definition ("*what is an atom*"), method ("*how to sort data in Excel*"), or biography ("*who was Galileo*") are complex questions.

But during our studying on question answering, we found that some questions seem not be able to be classified into the above three classes. An example of such questions is:

Q_1: What are the populations of the countries in the world?

Its answer is not simply a number (as an answer to the factoid question "*what is the population of Japan*") nor a list of country names (as the answers to the list question "*what are the countries in the world*"), but rather a set of <*country_name*, *population*> tuples. It is also suitable to present the answers by a table. For question Q_1, we can give answers in a two-column table like Table 1 whose columns represent "*country name*" and "*population*", respectively.

Table 1. The answers[1] to Q_1
"What are the populations of the countries in the world?"

Country	Population
Brazil	178,470
Czech	10,236
Egypt	71,931
Iceland	290
Malaysia	24,425
Yemen	20,010
...	...

A main difference between the new question class and the known three question classes is that: a question of the new class has two or more question foci, while a question of the known classes often has only one question focus. To our best knowledge, there is no research discussing on this topic.

Question decomposition is important for a QA system to answer multi-focus questions. However, the previous research on question decomposition rarely discussed the multiplicity of question foci. Saquete et al. [6] aimed on complex temporal questions expecting only single answers (single-focus questions

[1] Data in Table 1 come from the following UN webpage: http://www.un.org/Pubs/CyberSchoolBus/infonation/e_infonation.htm.

as defined later in this paper). The purpose of question decomposition in their work was to resolve temporal information carried in a question. Harabagiu et al. [7] worked on complex questions in LREC which were all single-focus questions. Their decomposition process was more likely a procedure to produce lots of queries which were semantically related to the original question in order to gather more information of answers. Besides, the responses of their system were summaries from multiple documents, not a set of tuples. Lin and Cho [8] proposed a method to do question segmentation, but the main purpose of their work was to identify different questions occurring in a post, not inside one question sentence.

The question decomposition in the START system [9] is most relevant to this topic. If START cannot find answers to a question, it will try to decompose the question and solve the subquestions one by one. For an example selected from their paper, in order to answer the question "*when was the 20th president of the U.S. born*", START first seeks the answer to the first subquestion "*who was the 20th president of the U.S.*", whose answer is "*James Abram Garfield*", and then finds the answer to the second subquestion "*when was James Abram Garfield born*", which is also the final answer, "*Nov. 19, 1831*". We can see that the answer to the original question is still a single string. START does not yet manage to answer in table fashion.

Katz et al. [9] referred to a question needed to be decomposed as a "complex question". In our definition, such a question is classified as a multi-focus single-response question.

This paper proposes a new principle to classify questions which have more than one question focus. Section 2 introduces the idea of multi-focus questions by defining the focus degree and the relations among the foci. Section 3 gives some simulation experiments to show the importance of handling multi-focus questions. Section 4 concludes this paper.

2. Definition of Multi-Focus Questions

The definitions of question focus are not quite the same in many papers. Lehnert [10] defined it as "*the question concept that embodies the information expectations expressed by the question*", while recent QA research groups [11][12] often refer to it as "*a word or sequence of words which indicate what information is being asked for in the question*" which means a question focus is a substring of a question.

To extend the definition to multi-focus questions, we first consider the necessity of question decomposition, and then define the question focus degree.

2.1 Question Decomposition

As we have seen in Section 1, some questions need to be decomposed before looking for answers. The decomposition process divides a question into a set of subquestions.

In our observation, a subquestion may depend on another question, i.e. a subquestion cannot be answered before another subquestion is answered. The dependencies of the subquestions form a dependency chain. Take Q_1 as an example:

Q_1: What are the populations of the countries in the world?
$Q'_{1,1}$: What are the countries in the world?
$Q'_{1,2}$: What is the population of <ansQ'$_{1,1}$>?
Dependency chain: $Q'_{1,1} \rightarrow Q'_{1,2}$

Question Q_1 is decomposed into two subquestions and its second subquestion depends on its first subquestion, which means we have to find the names of all the countries in advance so that we can answer the population of each country.

In the next sections of this paper, we call the rightmost subquestion in a dependency chain the *final subquestion*, while other subquestions are *preceding subquestions*. Note that there can be more than one final subquestion in a question, as we will see later.

2.2 Question Focus and Focus Degree

Our definition of question focus of a subquestion follows the convention. The **question focus of a question** can be re-defined as the set of the question foci of its subquestions.

Now we define the **focus degree** (FD) as the cardinality of the question focus set. For example:

Q_2: Give me the capitals, national flowers, and national trees of all the countries.

The question foci of Q_2 are "*capital*", "*national flower*", "*national tree*", and "*country*". Therefore, the focus degree of question Q_2 is 4.

A **multi-focus question** is a question whose focus degree is larger than 1. On the other hand, a question with a focus degree as 1 is called a **single-focus question**.

2.3 Relations between Question Foci

In order to learn the characteristics of multi-focus questions, we visited three QA websites, PTT[2], Yahoo Knowledge+[3], and Baidu Zhidao[4], to collect multi-focus questions. Although the questions are written in Chinese, we believe that the nature of multi-focus questions should be language-independent, so the Chinese question set can reveal the characteristics as the questions written in other languages.

Because the question collections in those QA websites are too large to browse, we submitted some particular queries which might often be realized by multi-focus questions. Examples of the queries are "country + caption" and "city + population".

After browsing 2,650 questions, we collected a set of 222 multi-focus questions. The number does not reveal the true ratio of multi-focus questions in human languages. A larger scale of investigation should be conducted in the future.

In our investigation, we find that the relation between two question foci is helpful to define the format of required answers. Note that when discussing the relation between two question foci, only their head nouns or the main constituents are considered.

The following four relations are explored from the collected multi-focus question set:

[2] http://www.ptt.cc/
[3] http://tw.knowledge.yahoo.com/
[4] http://zhidao.baidu.com/

(1) Coordinate Relation

The original question is a combination (mostly coordinate conjunction) of its subquestions. For example:

Q3: I want to know the biggest city and the largest port in Japan.
Q'3,1: What is the biggest city in Japan?
$\underbrace{\text{the biggest city}}_{QF_1}$
Q'3,2: What is the largest port in Japan?
$\underbrace{\text{the largest port}}_{QF_2}$
Dependency chain: {Q'3,1, Q'3,2}

Q_3 is in fact a conjunction of its two subquestions, $Q'_{3,1}$ and $Q'_{3,2}$. In this case, two subquestions can be answered independently.

(2) Entity-Attribute Relation

One question focus represents an entity type while the other focus represents an attribute of the entity type. For example:

Q4: What are the 5 largest cities and their populations?
Q'4,1: What are the 5 largest cities?
$\underbrace{\text{the 5 largest cities}}_{QF_1}$
Q'4,2: What is the population of <ansQ'4,1>?
$\underbrace{\text{the population}}_{QF_2}$
Dependency chain: Q'4,1 → Q'4,2

QF_2 "*population*" in Q_4 is an attribute of QF_1 "*city*". To answer this question, we should know the 5 largest cities before we can find the population of each city.

(3) Entity-Relationship Relation

Both question foci are entity types. One focus is explicitly expressed but the other focus is hidden. Instead, the surface text of the other focus represents the relationship between the two entity types. For example,

Q5: Names of capitals of Spanish speaking countries?
Q'5,1: What are Spanish speaking countries?
$\underbrace{\text{Spanish speaking countries}}_{QF_1}$
Q'5,2: Name of capital of <ansQ'5,1>?
$\underbrace{\text{capital}}_{QF_2}$
Dependency chain: Q'5,1 → Q'5,2

The two question foci in Q_5 are actually "*city*" and "*country*". The phrase QF_2 is the relationship "*capital-of*" between the explicitly stated focus QF_1 "*country*" and the hidden focus "*city*".

Someone may argue that a question like Q_5 does not need a table to present its answers, because the question only requests a list of capital names. But it seems to us that a list of <*country*, *capital*> tuples is more expressive and welcome.

Because there are also relationships relating to three or more entities, naturally there will be questions asking all the entities involved in a specific relationship at the same time. This definition can be extended to a relation among three or more entities.

(4) Thematic Relation

A more complicated case is that the two question foci are involved in a predicate-argument structure. "Argument" here refers to semantic argument. The two foci are often two arguments related to a predicate. For example:

Q6: When are the proper times to take vitamins?
Q'6,1: List all the vitamins.
$\underbrace{\text{the vitamins}}_{QF_1}$
Q'6,2: When is the proper time to take <ansQ'6,1>?
$\underbrace{\text{the proper time}}_{QF_2}$
Dependency chain: Q'6,1 → Q'6,2

This question asks for a list of vitamins and the best time to take each kind of vitamin. Take the predicate "take" into account, QF_1 "*vitamin*" is the patient of the predicate and QF_2 "*proper time*" is the time of the predicate. Both are the semantic arguments of the predicate "take".

Here is another example:

Q7: How to take care of fruit trees?
Q'7,1: List all the fruit trees.
$\underbrace{\text{the fruit trees}}_{QF_1}$
Q'7,2: How to take care of <ansQ'7,1>?
$\underbrace{\text{How to take care}}_{QF_2\text{: the procedure to take care}}$
Dependency chain: Q'7,1 → Q'7,2

The surface text of QF_2 is "*how to take care*" which asks for a procedure to take care of something. Considering the action of "*taking care*", QF_2 corresponds to its method and QF_1 "*fruit trees*" is the beneficiary. Different fruit trees may need different care procedures, so the answers could be presented by multiple columns.

This definition can also be extended to a relation among two or more entities, because there can be many arguments related to a predicate.

It also seems possible to define a relation that one focus is a predicate and the other focus is one of its arguments, just like in the question "who has done what to whom". But we cannot find a good example to illustrate this case, or perhaps it is because people rarely ask questions without predicates.

According to the definitions given above, relations of all pairs of question foci in a question can be determined. Take Q_2 as an example:

Q2: Give me the capitals, national flowers, and national trees of all the countries.
Q'2,1: List all the countries.
$\underbrace{\text{the countries}}_{QF_1}$
Q'2,2: What is the capital of <ansQ'2,1>?
$\underbrace{\text{the capital}}_{QF_2}$
Q'2,3: What is the national flower of <ansQ'2,1>?
$\underbrace{\text{the national flower}}_{QF_3}$
Q'2,4: What is the national tree of <ansQ'2,1>?
$\underbrace{\text{the national tree}}_{QF_4}$
Dependency chain: Q'2,1 → {Q'2,2, Q'2,3, Q'2,4}

The relations between QF_1/QF_2, QF_1/QF_3, and QF_1/QF_4 are all entity-relationship relations (relationships between "*country*" and "*city*", "*flower*", and "*tree*", respectively). The relations between QF_2/QF_3, QF_2/QF_4, and QF_3/QF_4 are coordinate relations.

Note that not all pairs of foci in a question have relations defined above. For example:

- Q_8: List the capitals of the countries in the world and their populations.
- $Q'_{8,1}$: List all the countries. (QF_1: the countries)
- $Q'_{8,2}$: What is the capital of <ansQ'$_{8,1}$>? (QF_2: the capital)
- $Q'_{8,3}$: What is the population of <ansQ'$_{8,2}$>? (QF_3: the population)
- Dependency chain: $Q'_{8,1} \rightarrow Q'_{8,2} \rightarrow Q'_{8,3}$

The foci QF_1 "*country*" and QF_2 "*capital*" have entity-relationship relation. QF_2 "*capital*" and QF_3 "*population*" have entity-attribute relation. But QF_1 and QF_3 do not have any direct relation.

2.4 Answer Format

The new definition of question classes is neither incompatible nor exclusive to the conventional definition. In fact, they define questions in different aspects.

A multi-focus question is defined by the number of question foci. A factoid or complex question is defined by the complexity of its answer types. A list question is defined by the multiplicity of its requested answers. We call a question a **single-response question** if it does not request a list of answers.

If a question can be decomposed into several subquestions, the complexity and multiplicity of each subquestion's answer type determines the format of the answers.

Q: What are the names of all the countries in Europe along with their capitals?
A: I don't know the answer.
Q: What are all the countries in Europe?
A: I have information about the following countries in Europe:
Albania, Andorra, Austria, Belarus, Belgium, Bosnia and Herzegovina, the UK, ..., Switzerland, and Ukraine
Q: What is the capital of Albania?
A: Tirane
Q: What is the capital of Andorra?
A: Andorra La Vella
...
Q: What is the capital of Ukraine?
A: Kiev

Final Answers:

Country	Capital
Albania	Tirane
Andorra	Andorra La Vella
Austria	Vienna
Belarus	Minsk
Belgium	Brussels
...	...
Switzerland	Bern
Ukraine	Kiev

Figure 1. Answer Responses to TQ_1 "*What are the names of all the countries in Europe along with their capitals?*"

Considering a question Q which can be decomposed into subquestions Q_1, Q_2 ... Q_n, and the dependency chain is $Q_1 \rightarrow Q_2 \rightarrow ... \rightarrow Q_n$. The answers to the question Q can be presented in an n-column table where the ith column gives the answers to the subquestion Q_i and the data in this column match the answer type of Q_i.

If Q_1 is a list question, the number of possible answers shown in the first column will be larger than 1, so the table will contain multiple rows. But if all the preceding subquestions Q_i (where $1 \leq i \leq n\text{-}1$) are single-response questions, there is only one possible combination of data in the first n-1 columns, so the original question Q becomes a conventional single-response question or a list question according to the type of the final subquestion Q_n.

3. Simulation Experiments

In order to show the necessity of handling multi-focus questions, three simulation experiments have been carried out and the results confirm our assertion.

Because the structures (in syntax level or semantic level) of multi-focus questions have not been fully studied, we have no modules to decompose multi-focus questions yet. In order to see the effect of multi-focus handling in a QA system, we manually performed the question decomposition and submitted the subquestions to an online QA system. We chose the START system [9] to do the experiment.

Because START is good at answering geographical questions, three multi-focus questions asking information of countries are selected as the testing questions. They are:

- TQ_1: What are the names of all the countries in Europe along with their capitals?
- TQ_2: What are the climates of the countries in North America?
- TQ_3: Can anyone give me a list of the largest 8 countries in Asia with their flags?

All their first subquestions are factoid list questions. All of the final subquestions are single-response questions, but their answer types are different. The final subquestion of TQ_1 is a factoid question, while the one of TQ_2 is more like a complex question. The final subquestion of TQ_3 can be considered as a factoid question only that its answer is a picture. The simulation experiments are described in the following subsections.

3.1 Simulation Experiment 1

Figure 1 illustrates the procedure of answering the multi-focus question TQ_1 by using START. Question TQ_1 was manually decomposed as follows:

- TQ_1: What are the names of all the countries in Europe along with their capitals?
- $TQ'_{1,1}$: What are all the countries in Europe?
- $TQ'_{1,2}$: What is the capital of <ansTQ'$_{1,1}$>?

Answers to the subquestion $TQ'_{1,1}$ were successfully collected from START system. Each answer was then used to replace <ansTQ'$_{1,1}$> to create its corresponding subquestion $TQ'_{1,2}$. For example, "*Albania*" was one of the answers to the first subquestion. It created a second subquestion "*what is the capital of Albania*" and then submitted to START. The answer to the second subquestion was paired with the answer to the first subquestion and formed a tuple in the table of final answers.

As we can see, when the original question TQ_1 is submitted to START without decomposition, the response of START is "*I don't know the answer.*" By detecting question foci and decomposing the question properly, answers to TQ_1 can be wonderfully created.

3.2 Simulation Experiment 2

Question TQ_2 was manually decomposed as follows:

TQ_2:	What are the climates of the countries in North America?
$TQ'_{2,1}$:	What are the countries in North America?
$TQ'_{2,2}$:	What is the climate of <ansTQ'$_{2,1}$>?

Q: What are the climates of the countries in North America?
A: Unfortunately, I don't know what the climates of countries are in North America.
Q: What are the countries in North America?
A: I have information about the following countries in North America:
the Bahamas, Barbados, Bermuda, Canada, Clipperton Island, Cuba, the Dominican Republic, Greenland, Grenada, Haiti, Jamaica, Mexico, Puerto Rico, Saint Pierre and Miquelon, and the USA
Q: What is the climate of the Bahamas?
A: tropical marine; moderated by warm waters of Gulf Stream
Q: What is the climate of Barbados?
A: tropical; rainy season (June to October)
...
Q: What is the climate of the USA?
A: mostly temperate, but tropical in Hawaii and Florida, arctic in Alaska, semiarid in the great plains west of the Mississippi River, ...

Final Answers:

Country	Climate
the Bahamas	tropical marine; moderated by warm waters of Gulf Stream
Barbados	tropical; rainy season (June to October)
Bermuda	subtropical; mild, humid; gales, strong winds common in winter
Canada	varies from temperate in south to subarctic and arctic in north
Clipperton Island	tropical; humid, average temperature 20-32 degrees C, wet season (May to October)
...	...
Saint Pierre and Miquelon	cold and wet, with much mist and fog; spring and autumn are windy
the USA	mostly temperate, but tropical in Hawaii and Florida, arctic in Alaska, semiarid in the great plains west of the Mississippi

Figure 2. Answer Responses to TQ_2 "*What are the climates of the countries in North America*?"

Figure 2 illustrates the procedure of answering the multi-focus question TQ_2 by using START. Again, START failed to answer the original multi-focus question, but then successfully provided the final answers following our answering procedure.

3.3 Simulation Experiment 3

TQ_3 is a more demonstrative example, as its answers are pictures, not text. It was manually decomposed as follows:

TQ_3:	Can anyone give me a list of the largest 8 countries in Asia with their flags?
$TQ'_{3,1}$:	What are the largest 8 countries in Asia?
$TQ'_{3,2}$:	What is the flag of <ansTQ'$_{3,1}$>?

Figure 3 illustrates the procedure of answering the multi-focus TQ_3 by using START. After submitting TQ3, START tried hard to find information to resolve the anaphora but failed in the end. Our proposed procedure successfully created the final answers once more.

4. Conclusion and Future Works

This paper proposes a new analyzing method to define multi-focus questions. The focus degree is used to describe the number of question foci of a question. Four relations between question foci are defined and explained. They can be used to analyze a multi-focus question.

Three simulation experiments were conducted to verify the necessity of handling multi-focus questions. Without multi-focus question processing, none of the testing questions could be answered by an online QA system. But if the relations between the answers of the subquestions were correctly identified and the subquestions were correctly created, wonderful responses could be constructed.

The most important skills to be developed in the future are relation identification and question decomposition for multi-focus questions. A large set of multi-focus questions is needed to learn the structures or patterns in multi-focus questions. A real QA system designed to answer multi-focus questions is needed to be developed to see the actual effect.

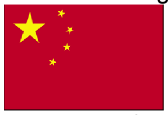
Furthermore, whether a question contains multiple foci is also ambiguous. Take question Q_6 as an example. It is possible that Q_6 requests different answers for different vitamins, or simply asks for one proper time for all kinds of vitamins. It remains a future work to find a method to solve the ambiguity.

5. References

[1] Dang, H. T., Lin, J., and Kelly, D. 2006. Overview of the TREC question answering track. In Proceedings of the 15th Text REtrieval Conference (TREC 2006).

[2] Giampiccolo, D., Forner, P., Peñas, A., Ayache, C., Cristea, Jijkoun, D. V., Osenova, P., Rocha, P., Sacaleanu, B., and Sutcliffe R. 2007. Overview of the CLEF 2007 multilingual question answering track. In Working Notes for the CLEF 2007 Workshop (Budapest, Hungary, September 19-21, 2007).

Q: Can anyone give me a list of the largest 8 countries in Asia with their flags?
A: I was unable to find the referent for the pronoun THEIR. Sorry.
Q: What are the largest 8 countries in Asia?
A: The following are the countries ranked in the top eight places for area among countries in Asia.
- China
- India
- Kazakhstan
- Saudi Arabia
- Indonesia
- Iran
- Mongolia
- Pakistan

Q: What is the flag of China?
A: 

Q: What is the flag of India?
A:

...
Q: What is the flag of Pakistan?
A:

Final Answers:

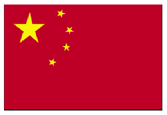
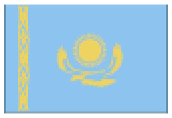
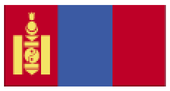
Country	Flag
China	
India	
Kazakhstan	
Saudi Arabia	
Indonesia	
Iran	
Mongolia	
Pakistan	

Figure 3. Answer Responses to TQ$_3$ "*Can anyone give me a list of the largest 8 countries in Asia with their flags*?"

[3] Fukumoto, J., Kato, T., Masui, F., and Mori, T. 2007. An overview of the 4th question answering challenge (QAC-4) at NTCIR Workshop 6. In Proceedings of the 6th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access (Tokyo, Japan, May 15-18, 2007).

[4] Sasaki, Y., Lin, C-J., Chen, K-H., and Chen, H-H 2007. Overview of the NTCIR-6 cross-lingual question answering (CLQA) task. In Proceedings of the 6th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access (Tokyo, Japan, May 15-18, 2007).

[5] Voorhees, E. 2001. Overview of the TREC 2001 question answering track. In Proceedings of the Tenth Text REtrieval Conference (TREC 2001).

[6] Saquete, E., Martinez-Barco, P., Munoz, R., and Gonzalez, J. L. V. 2004. Splitting complex temporal questions for question answering systems. In Proceedings of ACL 2004, 566-573.

[7] Harabagiu, S., Lacatusu, F., and Hickl, A. 2006. Answering complex questions with random walk models. In Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 220-227.

[8] Lin, C-J. and Cho, C-H. 2006. Question pre-processing in a QA system on Internet discussion groups. In Proceedings of COLING-ACL 2006, Workshop on Task-Focused Summarization and Question Answering, 16-23.

[9] Katz, B., Borchardt, G., and Felshin, S. 2005. Syntactic and semantic decomposition strategies for question answering from multiple resources. In Proceedings of the AAAI 2005 Workshop on Inference for Textual Question Answering, 35–41.

[10] Lehnert, W. 1978. The Process of Question Answering - A Computer Simulation of Cognition, Lawrence Erlbaum Associates, Hillsdale, N.J.

[11] Moldovan, D., Harabagiu, S., Pasca, M., Mihalcea, R., Girju, R., Goodrum, R., Rus, V. 2000. The structure and performance of an open-domain question answering system. In Proceedings of the Conference of the Association for Computational Linguistics (ACL-2000), 563-570.

[12] Ferret, O., Grau, B., Hurault-Plantet, M., Illouz, G., Monceaux, L., Robba, I., and Vilnat, A. 2001. Finding an answer based on the recognition of the question focus. In Proceedings of TREC-10, 362-370.

A study of statistical query expansion strategies for sentence retrieval

David E. Losada
Grupo de Sistemas Inteligentes
Departamento de Electrónica y Computación
Universidad de Santiago de Compostela
Campus sur s/n, Santiago de Compostela, Spain
dlosada@dec.usc.es

ABSTRACT
The retrieval of sentences that are relevant to a given information need is a challenging passage retrieval task. In this context, the well-known vocabulary mismatch problem, present in most Information Retrieval processes, arises severely because of the fine granularity of the task. Short queries, which are usually the rule rather than the exception, come to aggravate the problem. Consequently, effective sentence retrieval methods tend to apply some form of query expansion, usually based on pseudo-relevance feedback. Nevertheless, there are no extensive studies comparing different expansion strategies for sentence retrieval problems. In this work we aim to fill this gap. We start from a set of retrieved documents in which relevant sentences have to be found. In our experiments we test different term selection strategies and we also check whether expansion before sentence retrieval can yield reasonable performance. This is particularly novel because expansion techniques for sentence retrieval are often applied after a first retrieval of sentences and there are no comparative results available between expansion before and after sentence retrieval. This comparison is valuable not only for testing distinct expansion-based methods but also because there are important implications in time efficiency.

Keywords
Sentence retrieval, Query expansion, Information Retrieval

1. INTRODUCTION
The availability of effective sentence retrieval methods is potentially beneficial to many IR systems. There are many tasks whose performance is affected by the effectiveness of a sentence retrieval module. In web IR, information access can be facilitated provided that a good ranking of sentences, ordered by estimated relevance to the user, is supplied [18]. Question answering systems usually require some form of passage retrieval to isolate the document pieces in which the answer is likely to be found. This step is often done at the sentence level [11]. One of the main areas in text summarization is centered on building summaries by extracting important sentences from the target document(s). If the summaries are query-biased then effective techniques to measure query-sentence similarities are needed [16]. Information Extraction methods involve often some sentence retrieval algorithm to support their processes [12]. Sentence retrieval mechanisms have also been found important in Machine Translation [6].

SIGIR 2008 Workshop on Focused Retrieval.
July 24, 2008, Singapore.

Given a set of documents, our work focuses on a retrieval task based on selecting sentences relevant to a given information need, which is expressed as a textual query. This sentence retrieval problem is delimited to work with documents highly related to the query. This simulates a working environment in which an initial document retrieval is run and, next, the top ranked documents are input to a sentence retrieval module that filters out the irrelevant sentences and supplies the user with a rank of sentences. As argued in [18], a sentence retrieval interface of this kind would be very valuable, especially for searches in which the user does not have a clear idea about the topics involved and the sentences supplied can help her/him to clarify the purpose of the search.

Query expansion strategies, which have played a major role in document retrieval, are not sufficiently tested for sentence retrieval problems. Although some works have reported improvements using classical expansion techniques via pseudo-relevance feedback [8], there are no comparisons available testing extensively different term selection methods and studying the effect of the number of sentences and terms used for expansion. Expansion strategies developed for document retrieval might be ineffective for sentence retrieval because the number of matching terms is much smaller and, thus, performance might be harmed. Due to the importance of query expansion in sentence retrieval, we feel strongly that a complete study on this subject is required. The vocabulary mismatch problem is a severe obstacle to yield effective retrieval at the sentence level and the role of query expansion as an alleviation tool needs to be carefully analyzed. Furthermore, there are no comparative results between expansion before sentence retrieval and expansion after sentence retrieval. Expansion before sentence retrieval has been particularly neglected. Since we start from a set of top ranked documents, it makes sense to study blind feedback methods working directly with the initial ranking of documents and compare them with regular pseudo-relevance feedback applied after running a first sentence retrieval process. Note also that this has important implications for efficiency that should not be disregarded.

Our study will be primarily focused on two standard automatic expansion methods that have worked well in document retrieval problems: pseudo-relevance feedback (PRF) [4] and Local Context Analysis (LCA) [19]. These techniques are general enough to be applied across different domains and collections. Although some works have managed to get effective expansion with linguistic resources, we are concerned here only with purely statistical methods, which are simpler and applicable under very distinct scenarios.

The rest of the paper is organized as follows. Section 2 reviews some papers related to our research. Section 3 presents the sentence retrieval method and the expansion techniques tested. The experiments are reported and analyzed in section 4. The paper ends with some conclusions.

2. RELATED WORK

Sentence retrieval is a challenging area. Many researchers have proposed different solutions based on a wide range of models and techniques such as query expansion (either via pseudo-relevance feedback or with the aid of a lexical resource), part-of-speech tagging, clustering of sentences, named entities, supervised learning, and language modeling. Despite the variety of the approaches investigated, simple adaptations of regular tf/idf measures (sometimes aided with some form of pseudo-relevance feedback) can be labeled as state of the art sentence retrieval methods [2, 9].

Many studies have examined the use of expanded queries either via pseudo-relevance feedback [5] or with the assistance of a terminological resource, such as Wordnet [20]. The effect of pseudo-relevance feedback is known to be very sensitive to the quality of the initial ranks. Motivated by this, some researchers have applied selective feedback [1], which is more stable but requires training data. In [11] some experiments investigating the effects of pseudo-relevance feedback on sentence retrieval were reported. Query expansion produced negative results but a single expansion technique, based on Relevance Models, was tested. On the other hand, expansion with synonyms or related terms from a lexical resource is problematic because noisy terms can be easily introduced into the new query. Moreover, a large terminological resource, with good coverage, is not always available. As a matter of fact, lexical expansion is usually equal or inferior to purely statistical expansion methods in sentence retrieval [7, 15, 14].

Expansion approaches based on co-occurrence data have been also proposed [20] but there is not much evidence that these approaches can outperform the standard pseudo-feedback methods.

Rather than expanding queries with new terms, other studies have focused on improving the matching process by analyzing carefully the nature of the sentence components. For example, in [9], patterns such as phrases, combinations of query terms and named entities were identified into sentences and the sentence retrieval process was driven by such artifacts. Although this technique was very effective for detecting redundant sentences, it was not significantly better than a regular tf/idf baseline for finding relevant sentences.

In [10], an effective sentence retrieval method, based on extracting highly frequent terms from top ranked documents, was designed. This method actually represents a form to exploit the information from top retrieved documents before sentence retrieval. It was successfully compared against query expansion using pseudo-relevance feedback from top retrieved sentences (i.e. expansion after sentence retrieval). Nevertheless, expansion before sentence retrieval (i.e. expanding directly from the top retrieved documents) was not properly tested. For instance, sophisticated expansion techniques, such as LCA, were not considered in the experimental design.

There is therefore the general feeling in the sentence retrieval community that some form of expansion is needed to achieve reasonably good performance. However, expansion methods have not been adequately compared and, actually, we can find in the literature conflicting outcomes depending on the collection, baseline method tested, etc. In the present work we aim to clarify the role of expansion strategies in sentence retrieval by testing some standard methods against three different datasets and applying a very competitive baseline.

In the literature of sentence retrieval, the peculiarities of the sentence retrieval task are often ignored. Most expansion studies do not make full use of the information available but simply apply expansion methods that worked well in document retrieval. We argue that the ranked set of documents contains valuable information on the importance of terms that should not be disregarded. In this respect, we believe that it is important to check the effectiveness of query expansion methods when applied before sentence retrieval (i.e. working directly on the top retrieved documents available). There are at least two reasons that support this claim. First, sentence retrieval is very sensitive to the quality of the query and, hence, we might be safer working on the initial set of documents rather than on a subsequent ranking of sentences. Second, it would avoid retrieving an initial ranking of sentences and therefore would bring about a benefit in terms of efficiency.

3. SENTENCE RETRIEVAL METHOD

To study properly different query expansion strategies we need first to decide which sentence retrieval method is appropriate for our purposes. Since we want to evaluate the ability of expansion techniques to improve the state-of-the-art in sentence retrieval, we have to set a competitive sentence retrieval technique. In [2], the results of some sentence retrieval experiments are discussed. A simple vector space retrieval technique is shown to perform at least as well as any other method and, actually, its performance is the most robust. This method, which we will refer to as tf/isf[1], applies a weighting scheme that is a variant of tf/idf applied at the sentence level. Although other effective methods, such

[1]isf stands for inverse sentence frequency

as those based on clusters of sentences, can be found in the literature [7, 15, 14], we skip them deliberately because the tf/isf method is simpler and we therefore avoid possible biases and complications coming from evolved approaches (e.g. the effect of the quality of the clusters). We believe strongly that the simplicity of this method is a good feature, making the results presented here potentially applicable in very different scenarios. The relevance of a sentence s given a query q is estimated in [2] as:

$$tf\_isf(s,q) = \sum_{t \in q} log(tf_{t,q}+1) log(tf_{t,s}+1) log(\frac{n+1}{0.5+sf_t}) \quad (1)$$

where sf_t is the number of sentences in which t appears, n is the number of sentences in the collection and $tf_{t,q}$ ($tf_{t,s}$) is the number of occurrences of t in q (s).

To further check that tf/isf was competitive we designed some preliminary experiments whose results are reported in section 4.1. This included experiments using alternative sentence retrieval methods (OKAPI BM25 and Language Modeling with KLD), different combinations of the pre-processing strategies and even additional tests using idf statistics (instead of isf). This evaluation demonstrated clearly that tf/isf is a consistent sentence retrieval method whose performance is comparable or superior to the best performance attainable by other effective methods.

3.1 Expansion after sentence retrieval

By query expansion *after* sentence retrieval (ASR) we refer to the regular pseudo-relevance feedback process adapted to the sentence retrieval case. First, the query is run against the sentences in the top retrieved documents and, next, the top retrieved sentences are used to mine expansion terms. Two main strategies are considered to select new terms: PRF and LCA.

Pseudo-relevance feedback (also called local or blind feedback) is a traditional concept in IR [3], which basically consists of selecting the terms with more counts in the top retrieved sentences. Although it did not work well with small (pre-TREC) collections, its merits for large-scale document retrieval have been apparent in many TREC experiments [17]. Nevertheless, the effects this method has on sentence retrieval have not been studied in detail. Actually, some papers have reported improvements after expansion via pseudo-relevance feedback but other studies are sceptical about local feedback improving sentence retrieval [11]. We therefore expect that the experiments reported here help to shed light on this issue. Note also that there are some parameters needed for success, such as the number of top sentences and the number of expansion terms. Sentences are very small pieces of text and retrieval performance may be very sensitive to the parameter configuration here.

LCA is a successful expansion method proposed by Xu and Croft [19]. It has been adopted by other research groups in several large-scale experiments in document retrieval [17]. Nevertheless, the effects of LCA in sentence retrieval are barely discussed in the literature and there are no experimental results available comparing LCA and local feedback. The main motivation to propose LCA was that local feedback fails if there is a large number of non-relevant items in the top ranked set. The LCA method tries to be less erratic and is designed to work on document passages. We take here an instance of the LCA proposal where passages are simply document sentences. The main hypothesis behind LCA is that common terms from relevant documents (sentences, in our case) will tend to co-occur with query terms within the top-ranked documents (sentences). In this way, a term selection metric is defined, yielding an expansion method that is more robust than local feedback. Although LCA works for *concept* selection, where concepts can be single terms or phrases, we are only concerned here with selecting single terms for expansion. Let us consider a query q, whose query terms are $qt_1, \cdots, qt_m$, and a set of top ranked sentences $S = \{s_1, s_2, \cdots, s_n\}$. The terms appearing in S are ranked according to the formula:

$$\begin{aligned} f(t,q) &= \prod_{qt_i \in q} (\delta + co\_degree(t, qt_i))^{idf(qt_i)} \\ co\_degree(t, qt_i) &= log_{10}(1 + co(t, qt_i)) \cdot idf(qt_i)/log_{10}(n) \\ co(t, qt_i) &= \sum_{s_j \in S} tf(t, s_j) \cdot tf(qt_i, s_j) \\ idf(t) &= min(1.0, log_{10}(N/N_t)/5.0) \end{aligned}$$

where N is number of sentences in the collection, N_t is number of sentences in the collection containing t, $tf(w, s_j)$ is the number of occurrences of w in sentence s_j and δ is a constant set to 0.1 to avoid zero value. This term ranking function is a variant of the regular tf/idf measure utilized popularly in IR. Most often preferred terms will be those rare terms (idf effect) that co-occur frequently with many query terms.

Given this measure, the terms in the retrieved sentences can be ordered in decreasing order of $f(t,q)$ and the top ranked terms are selected to expand the query.

For simplicity, we do not consider here any parameterized re-weighting strategy (e.g. based on Rocchio's formula). In both methods (pseudo-relevance feedback and LCA), the selected terms are simply incorporated as new terms in the query. Note that this involves expansion (new terms that were not present in the original query) but also basic re-weighting (the query term frequency is increased for terms belonging to the old query that are also selected in the expansion phase).

3.2 Expansion before sentence retrieval

One strategy that has not received much attention is to run query expansion before retrieving any sentence (BSR). This alternative was not explored in the past but it could become very valuable. First, for efficiency reasons: we can skip the initial sentence retrieval process (no sentence retrieval is required for doing expansion). Second, query expansion may be more robust if we work directly from the top ranked documents. Observe that poor queries will likely introduce a great deal of noise if we use them again to retrieve some sentences to feed the term selection module. The initial ranking of documents is arguably weak for such queries but, still, a second usage of the original topic for query expansion purposes might be not advisable. It is therefore interesting to evaluate empirically these issues and compare expansion BSR and expansion ASR.

Some experiments were designed to evaluate expansion BSR. The term selection methods were the same as those explained in the previous section but the sentences used to mine the expansion terms are taken directly from the initial ranking of documents available for the task. More specifically, the top X documents (X is a parameter) are used for term mining. To maintain consistency with the ASR experiments, term selection works also at the sentence level. The BSR-version of LCA extracts new terms analyzing the co-occurrences in the sentences of the top X documents. Similarly, expansion BSR with pseudo-relevance feedback incorporates into the query the terms with more counts in the sentences of the top X documents. However, note that there is no sentence retrieval here (e.g. if $X = 1$ then all sentences from the top document are considered in the term selection process).

3.3 Complexity issues

Expansion ASR introduces an important time penalty because it requires a sentence retrieval process for term selection. In contrast, expansion BSR works directly from the sentences in the top retrieved documents. This is a considerable saving.

Given a set of sentences (either a set of sentences ranked in decreasing order of similarity to a given query -expansion ASR- or a set of sentences appearing in top ranked documents -expansion BSR-), it is interesting to compare the steps needed to compute the ranks of terms with pseudo-relevance feedback or LCA. Pseudo-relevance feedback simply requires to traverse the sentences and accumulate the term counts in a proper data structure (e.g. a term-count structure ordered by count). LCA requires also to go on every sentence and accumulate the $co(t, q_i)$ counts (for each q_i). The time complexity of this process across retrieved sentences is equivalent to the time complexity needed by pseudo-relevance feedback (although the space complexity is higher with LCA because we need to store independent statistics for each query term). Anyway, LCA incorporates an additional time penalty to compute the final $f(t, q)$ values (product across query terms). This cost, which is linear with respect to the number of query terms, could be assumed to be negligible, especially if queries are short.

4. EXPERIMENTS

We designed a complete pool of experiments to test the expansion configurations. The experiments were run against three different collections of data (the ones supplied in the context of the TREC-2002, TREC-2003 and TREC-2004 novelty tracks [7, 15, 14]). There are no newer TREC collections suitable for our experiments because we need relevance judgments at the sentence level. This sort of judgments is only available in the novelty track, whose last edition took place in 2004. The novelty track data was constructed as follows. Every year there were 50 topics available. In TREC-2002, the topics were taken from TRECs 6, 7 and 8[2]. In 2003 and 2004, the topics were created by assessors designated specifically for the task [15, 14] (topics N1-N100). For each topic, a rank of documents was obtained by NIST using an effective retrieval engine. In 2002 and 2003 the task aimed at finding relevant sentences in relevant documents and, therefore, the ranks included only relevant documents (i.e. given a topic the set of relevant documents to the topic were collected and ranked using a document retrieval engine). On the contrary, the TREC-2004 ranks contained also non-relevant documents (i.e. the initial search for documents was done against a regular document base, with relevant and non-relevant documents). Note that this means that the non-relevant documents are close matches to the relevant documents, and not random non-relevant documents [14]. In any case, the ranks of documents contained at most 25 relevant documents for each query. The documents were segmented into sentences, the participants were given these ranks of sentence-tagged documents and they were asked to locate the relevant sentences. The relevance judgments in this task are complete because the assessors reviewed carefully the ranked documents and marked every sentence as relevant or non-relevant to the topic. In TREC-2002, very few sentences were judged as relevant (approximately 2% of the sentences in the documents). In TREC-2003 and TREC-2004 the average percentage of relevant sentences was much higher than in 2002 (approximately 40% in 2003 and 20% in 2004).

We consider here two different evaluation measures: the F measure, which was the official measure utilized in the TREC novelty tracks, and precision at ten sentences retrieved (P@10). The F measure is meaningful even when the number of relevant sentences varies widely across topics [7]. The F values reported here are obtained by retrieving 5% of the sentences in TREC 2002, and 50% of the sentences in TREC 2003 and TREC 2004. These thresholds, which have been applied in the past, are reasonable given the amount of relevant sentences in every collection. Additionally, P@10 ratios are included in our reports. P@10 is important in many applications, such as web sentence retrieval [18], which require a good distribution of relevant material in the top rank positions.

We focus our interest on short queries (constructed from the title tags of the TREC topics) because handling properly this type of queries is challenging in sentence retrieval. These queries are good candidates for expansion because they are often ambiguous.

4.1 Evaluating the baseline

To ensure that the baseline (tf/isf, eq. 1) is capable of yielding state of the art performance, we ran some preliminary experiments comparing it against Okapi BM25 [13] and a Language Modeling approach based on Kullback-Leibler Divergence (KLD) as described in [8] (with Dirichlet smoothing). The performance of BM25 is influenced by some parameters: $k1$ controls the term frequency effect, b controls a length-based correction and $k3$ is related to query term frequency. We tested exhaustively different parameter con-

[2]The complete list of topics chosen for the novelty track can be found in [7]

	TREC-2002		
	tf/isf	BM25 (best run) $k1 = .4, b = 0, k3 = 1$	KLD (best run) $\mu = 3000$
P@10	.19	.19	.16
F	.19	.19	.17*
	TREC-2003		
	tf/isf	BM25 (best run) $k1 = .6, b = 0, k3 = 1$	KLD (best run) $\mu = 1000$
P@10	.74	.76	.73
F	.51	.51	.50*
	TREC-2004		
	tf/isf	BM25 (best run) $k1 = .2, b = 0, k3 = 1$	KLD (best run) $\mu = 500$
P@10	.43	.44	.41
F	.37	.37	.37

Table 1: Comparing different sentence retrieval baselines: tf/isf, BM25 and KLD (with Dirichlet Smoothing).

figurations ($k1$ between 0 and 2 in steps of 0.2, b between 0 and 1 in steps of 0.1 and different values of $k3$ between 1 and 1000). Similarly, we experimented with the KLD model for different values of the μ constant, which determines the amount of smoothing applied (μ = 10, 100, 500, $1k$, $3k$, $5k$). Results are reported in Table 1. A run marked with an asterisk means that the difference in performance between the run and tf/isf is statistically significant[3]. In all collections, there was no statistically significant difference between the tf/isf run and the best BM25 run. We also observed that BM25 was very sensitive to the parameter setting (many BM25 runs performed significantly worse than tf/isf). On the other hand, KLD was inferior to both tf/isf and BM25. These results reinforced previous findings about the robustness of the tf/isf method [2, 9] and demonstrated that this method is a very solid baseline. Note also that tf/isf does not have any free parameter whereas the results reported for BM25 and KLD are the best ones obtained across the configurations tested.

We also experimented with different combinations of the standard preprocessing strategies (stopwords vs no stopwords, stemming vs no stemming). Although there was not much overall difference, the runs with stopword processing and no stemming were slightly more consistent.

The tf/isf method takes the isf statistics from the sentences in the documents available for the task (which is a small set of sentences). A term that is very common within the retrieved documents would therefore receive a low isf weight. This might be problematic because content-bearing terms that are frequent in a given set of documents are assigned small weights. We were therefore wondering whether better performance might be obtained using data from a larger collection. To check this, we indexed a large collection of documents (the collection used in the TREC-8 adhoc experiments) and ran some experiments with regular idf statistics obtained from this index (i.e. in eq. 1 sf_t was replaced by n_t, which is the document frequency of t in TREC-8). The original tf/isf method computed at the sentence level over the small document base was slightly superior. It appears that the small index of sentences is good enough for sentence retrieval (at least for these short queries). We therefore set the basic sentence retrieval method to be the original tf/isf approach with stopword and no stemming.

Note that we use short queries, while the groups participating in the TREC novelty tracks were allowed to use the whole topic. This means that the results presented here are not comparable to any of the results reported in the novelty tracks. Actually, we expect that the results obtained here are worse than the ones achieved in TREC because of our experimental conditions. Nevertheless, short queries are the rule rather than the exception in many applications and it is therefore important to study in depth the sentence retrieval performance with such queries. Moreover, query expansion methods are especially important when the user supplies few search terms.

4.2 Evaluating query expansion strategies

Let us now pay attention to the effects of query expansion on sentence retrieval performance. With expansion ASR, we first ran the tf/isf sentence retrieval method on the ranked set of documents associated to each query. This produced a ranked set of sentences from which some expansion terms were selected using PRF or LCA. These new terms were included into the query and the tf/isf sentence retrieval model was run again with the expanded query. We tested different configurations of the number of expansion terms (5, 10, 20 and 50) and the number of top retrieved sentences in which terms are selected (5, 10, 25, 50 and 100). On the other hand, expansion BSR selects terms directly from the ranked set of documents. We planned experiments using the top 1, 5, 10, 15 or 25 documents with varying number of expansion terms.

The experimental results are reported in Tables 2 (expansion ASR - P@10), 3 (expansion BSR - P@10), 4 (expansion ASR - F measure) and 5 (expansion BSR - F measure). The tables include also the performance of the baseline tf/isf with no expansion (underlined after the collection's name). For each collection and type of expansion the best parameter configurations are marked in bold. Expansion runs whose improvement over the baseline is statistically significant are marked with an asterisk.

First of all, it is interesting to observe the effect of expansion on the TREC-2002 collection. There are some expansion configurations that show P@10 and F ratios that are higher than the baseline's ratios. Anyway, nearly all improvements are not statistically significant. Observe that this collection contains very few relevant sentences ($\approx 2\%$) and, therefore, any expansion strategy is likely incorporating unrelated terms into the new queries. PRF is particularly problematic here because it often performs worse than the baseline (32 out of the 80 TREC-2002 PRF expansion runs perform worse than the baseline). In contrast, LCA does not improve significantly over the baseline but, at least, there are fewer LCA runs yielding performance that is poorer than the baseline's performance (14 runs out of 80).

On the other hand, most expansion methods produce statistical significant improvements in TREC-2003 and TREC-

[3]Along this work, we applied two different significance tests, the t-test and the Wilcoxon test, and we show only an asterisk when both tests agree on the significance of the difference (95% confidence level).

# exp terms	PRF					LCA				
	# top sentences					# top sentences				
	5	10	25	50	100	5	10	25	50	100
TREC-2002 (basel: .19)										
5	.19	.19	.22	**.24***	.23	.20	.19	.19	.21	.21
10	.21	.20	.21	**.24**	**.24***	.19	.18	.19	.21	**.23**
20	.19	.17	.20	.21	.23	.18	.16	.19	.22	.22
50	.19	.20	.20	.20	.22	.18	.18	.19	.21	.22
TREC-2003 (basel: .74)										
5	.78*	.78*	**.81***	.79	.79*	.75	.75	.75	.75	.75
10	.78*	.79*	.80*	**.81***	.80*	.79*	.78*	.80*	.81*	.80
20	.80*	.78	.80*	.80	.79	.80*	.78*	.80*	**.82***	.79
50	.78*	.77	.79	.75	.76	.79*	.77	.79*	.79	.81*
TREC-2004 (basel: .43)										
5	.46	.48*	.49*	.50*	.48*	.45	.47	.51*	.51*	.47
10	.47	.48*	.51*	.54*	.54*	.47	.50*	.49*	.49*	.50*
20	.47	.49*	.50*	**.55***	**.55***	.46	.47	.48	.49*	.49*
50	.44	.46	.50*	.52*	.54*	.45	.46	.49*	.50*	**.53***

Table 2: Expansion ASR - Precision at 10 sentences

# exp terms	PRF					LCA				
	# docs					# docs				
	1	5	10	15	25	1	5	10	15	25
TREC-2002 (basel: .19)										
5	.19	.20	**.22**	.19	.19	.20	.22	.21	.21	.21
10	.16	.19	.21	.21	.17	.20	.22	.23	.23	.23
20	.16	**.22**	.19	.19	.17	.18	.22	.22	.21	.23
50	.17	**.22**	.20	.18	.17	.17	**.24**	.23	.22	.22
TREC-2003 (basel: .74)										
5	.75	.77	.76	.75	.77	.74	.77	.78	.77	.74
10	.74	.78	.76	.77	**.79**	.72	.79	.78	.76	.79
20	.75	**.79**	.77	.77	.76	.71	.77	.79	**.81***	.80
50	.71	.75	**.79**	.78	.76	.71	.78	.79	**.81***	.77
TREC-2004 (basel: .43)										
5	.42	.49	.47	.50*	.48*	.42	.46	.46	.47	.49
10	.39	.50*	.50*	.49*	.51*	.43	.47	.48	.50*	.48
20	.38	.49	.50*	.50*	.54*	.43	.48	.49	.49	.50*
50	.37	.46	.51*	.54*	**.55***	.40	.43	.49	.49	**.52***

Table 3: Expansion BSR - Precision at 10 sentences

# exp terms	PRF					LCA				
	# top sentences					# top sentences				
	5	10	25	50	100	5	10	25	50	100
TREC-2002 (5% sens ret.) (basel: .19)										
5	.19	.19	.19	.19	.19	.20	.19	.19	.19	.19
10	.19	.18	.19	.20	.20	.19	.18	.18	.19	**.20**
20	.18	.18	.19	.20	**.21**	.18	.18	.18	**.20**	**.20**
50	.18	.18	.19	.20	**.21**	.18	.18	.19	**.20**	**.20**
TREC-2003 (50% sens ret.) (basel: .51)										
5	.53	.54*	.54*	.55*	.55*	.52	.52*	.53*	.53*	.53*
10	.55*	.55*	.55*	**.57***	.56*	.53*	.53*	.55*	.55*	.55*
20	.55*	.56*	.56*	.56*	**.57***	.55*	.55*	.56*	.56*	.56*
50	.56*	.56*	**.57***	**.57***	**.57***	.56*	.56*	.56*	**.57***	**.57***
TREC-2004 (50% sens ret.) (basel: .37)										
5	.38	.38	.38	.38	.39*	.37	.37	.37	.38	.38
10	.38	.38	.39*	.39*	.39*	.38	.38*	.38*	.39*	.39*
20	.39*	.39*	.39*	.40*	.40*	.38*	.39*	.39*	.39*	.39*
50	.39*	.39*	.40*	**.41***	.40*	.39*	.39*	**.40***	**.40***	**.40***

Table 4: Expansion ASR - F measure

# exp terms	PRF					LCA				
	# docs					# docs				
	1	5	10	15	25	1	5	10	15	25
TREC-2002 (5% sens ret.) (basel: .19)										
5	.17	.18	.18	.16	.17	.19	.19	.20	.20	.19
10	.17	**.19**	.18	.17	.16	.18	.20	.20	.20	.20
20	.17	.18	.18	.17	.16	.18	.19	**.21**	.20	.20
50	.16	.18	.18	.17	.15	.19	.20	**.21**	**.21**	**.21**
TREC-2003 (50% sens ret.) (basel: .51)										
5	.55*	.55*	.55	.55*	.55*	.52*	.53*	.53*	.53*	.54*
10	.55*	**.56***	.55	**.56**	**.56***	.52*	.55*	.54*	.54*	.56*
20	.56*	.55*	.55	**.56**	**.56***	.55*	.55*	.56*	.56*	**.57***
50	.55	**.56***	**.56***	**.56***	**.56***	.56*	.56*	**.57***	**.57***	**.57***
TREC-2004 (50% sens ret.) (basel: .37)										
5	.37	.38	.38	.38	.38	.37	.38	.38	.38	.38
10	.38	.38	.38	.38	.38	.37	.38	.39*	.39*	.39*
20	.38	.38	.39	.39	.39*	.38	.39*	.39*	.39*	.39*
50	.38	.39	.39*	**.40***	**.40***	.38	.39*	.40*	.40*	**.41***

Table 5: Expansion BSR - F measure

2004 (for both P@10 and F). These results show that statistical query expansion is beneficial in sentence retrieval provided that the amount of relevance sentences in the ranked set of documents is not extremely low.

Next, we analyze the trends with respect to the number of expansion terms and the number of top sentences/documents.

Expansion ASR, P@10 (Table 2). The standard PRF expansion tends to achieve the highest P@10 performance when a few expansion terms are selected from many sentences. A safe configuration would be 10 expansion terms selected from 50 or 100 sentences. With LCA-based expansion, the ideal number of sentences is also high but this expansion method tolerates slightly better expansions with more terms.

Expansion BSR, P@10 (Table 3). When expanding queries before sentence retrieval, PRF looks much more sensitive to the parameter setting. It is quite difficult to identify a good configuration because the optimal performance is found at very different places depending on the collection. Only in TREC-2004 the improvements over the baseline are statistical significant. On the other hand, LCA looks less erratic. A high number of terms (50) extracted from a large number of documents (15-25) seems to be a good configuration.

Expansion ASR, F-measure (Table 4). With both expansion methods, PRF and LCA, the highest performance tends to be found when a large number of expansion terms are selected from a large number of top sentences. P@10 is a high precision measure but the F measure is influenced by both precision and recall. This explains why the optimal F performance is found with expansions involving many new terms, whilst the optimal P@10 performance is achieved usually with fewer expansion terms.

Expansion BSR, F-measure (Table 5). With LCA, there is also a clear tendency to prefer many expansion terms extracted from many documents. However, with PRF, the optimal configuration varies significantly depending on the collection. These results agree with those found for P@10 with BSR.

Given this report, it is clear that LCA performs the best with many expansion terms extracted either from many sentences (ASR) or from many documents (BSR). PRF is much more erratic and its optimal expansion configuration is much more difficult to assess.

Let us now analyze the best and average performance attainable by each expansion method. For a clearer picture of the experimental outcome, these results are summarized in Table 6. The difference between the best ASR and BSR runs has been tested for statistical significance and the BSR run is marked with the symbol † when the difference between the run and the respective ASR run is significant. In terms of P@10, there is no significant difference between the best runs. This means that any configuration (ASR/BSR + PRF/LCA) can lead to optimal performance provided that the parameters (number of expansion terms and number of top sentences/documents) are set adequately. Looking at the average P@10 values, we found some interesting trends. With expansion ASR, PRF is more solid than LCA. On the contrary, with expansion BSR, LCA tends to be more reliable (especially when the conditions are difficult -few relevant sentences-). This makes sense because the sentences feeding the ASR term selection module are potentially closer to the query than the sentences feeding the BSR term selection module. Recall that expansion ASR runs an initial sentence retrieval from the query and the retrieved sentences are used for term selection purposes. In contrast, expansion BSR works directly with the initial ranked set of documents, where the on-topic sentences might be scattered across the documents. This means that a rough term selection metric (such as local feedback) is good enough with expansion ASR but it is less consistent when there is not an initial sentence retrieval process.

In terms of the F measure, the results are basically the same as the ones found with P@10. PRF tends to work better with expansion ASR while LCA tends to be more solid with expansion BSR. In two collections the best run of PRF with expansion ASR performs statistically significantly better than the best run of PRF with expansion BSR. It is interesting to note that the single collection where the difference is not significant is TREC-2002, where there are few relevant sentences. This makes sense because expansion ASR is very sensitive to the quality of the initial sentence retrieval process. If these ranked sentences contain many non-relevant items then expansion ASR will hardly improve on expansion BSR.

In terms of effectiveness, expansion ASR with PRF and expansion BSR with LCA are the most robust expansion methods for sentence retrieval. Both approaches lead to good P@10 and F performance ratios. Since expansion BSR is less expensive than expansion ASR (because we do not need an initial sentence retrieval process), expansion BSR with LCA looks the most suitable choice. One can rightly argue that LCA is more costly than PRF but, as argued in section 3.3, the additional complexity requirements are acceptable. This means that we can achieve state-of-the-art sentence retrieval performance with significant savings in terms of efficiency. This is a novel result because the studies conducted in the literature have been mostly focused on the standard expansion methods (ASR). Furthermore, if the aim of the retrieval application is to retrieve ten good sentences (i.e. recall is not a major issue) then expansion BSR with PRF is a good choice. As shown in Table 6, this retrieval technique, which is the most efficient method, does not perform significantly worse than the other expansion methods (in terms of P@10).

		TREC-2002			
		ASR	BSR	ASR	BSR
		best		*avg*	
P@10	PRF	.24	.22	.21	.18
(basel: .19)	LCA	.23	.24	.20	.22
F	PRF	.21	.19	.19	.17
(basel: .19)	LCA	.20	.21	.19	.20
		TREC-2003			
P@10	PRF	.81	.79	.79	.76
(basel: .74)	LCA	.82	.81	.78	.77
F	PRF	.57	.56†	.56	.55
(basel: .51)	LCA	.57	.57	.55	.55
		TREC-2004			
P@10	PRF	.55	.55	.50	.48
(basel: .43)	LCA	.53	.52	.48	.47
F	PRF	.41	.40†	.39	.38
(basel: .37)	LCA	.40	.41	.39	.39

Table 6: Comparing the best and average performance of expansion ASR and expansion BSR with PRF and LCA

5. CONCLUSIONS

In this paper we have presented a thorough study on the effects of query expansion strategies for sentence retrieval. We have worked with an standard sentence retrieval method, proved that it is competitive against other robust techniques and supplied a complete study of query expansion under this framework.

The results of our study can be summarized as follows. In terms of effectiveness, expansion ASR with PRF and expansion BSR with LCA are the most robust expansion methods for sentence retrieval. Both approaches lead to good P@10 and F performance ratios. Since expansion BSR is less expensive than expansion ASR (because we do not need an initial sentence retrieval process), expansion BSR with LCA looks to be the most suitable choice. This means that we can achieve state-of-the-art sentence retrieval performance with significant savings in terms of efficiency. This is a novel result because the studies conducted in the literature have been mostly focused on the standard expansion methods (ASR).

Regarding the number of expansion terms and the number of top documents/sentences from which terms are mined, we found that the more documents/sentences fed into the expansion modules the better performance. This is not very surprising. On the other hand, LCA shows a slight tendency to achieve its highest performance with expansions involving many terms while PRF is more erratic with respect to the ideal number of expansion terms. In general, PRF is very sensitive to the parameter setting. Although the top performance attainable by PRF tends to be similar to LCA's top performance, the parameter settings are more problematic with PRF.

Summing up, although some past studies have been skeptical on the role of query expansion for sentence retrieval, our report shows that it is a consistent technique to improve sentence retrieval performance provided that the retrieved documents contain a reasonable amount of relevant sentences. The two methods tested, PRF and LCA, can produce significant benefits when parameters are set appropriately. Even with an extremely low population of relevant sentences (TREC-2002), a proper query expansion configuration (e.g. BSR+LCA for high precision purposes and ASR+PRF otherwise) hardly damages performance.

ACKNOWLEDGMENTS

I thank the support obtained from projects TIN2005-08521-C02-01 (*Ministerio de Educación y Ciencia*) and PGIDIT07-SIN005206PR (*Xunta de Galicia*).

6. REFERENCES

[1] N. Abdul-Jaleel, J. Allan, B. Croft, F. Diaz, L. Larkey, X. Li, M. Smucker, and C. Wade. UMass at TREC 2004: Novelty and hard. In *Proceedings of the 13th Text Retrieval Conference (TREC 2004)*, 2004.

[2] J. Allan, C. Wade, and A. Bolivar. Retrieval and novelty detection at the sentence level. In *Proc. SIGIR-03, the 26th ACM Conference on Research and Development in Information Retrieval*, pages 314–321, Toronto, Canada, 2003. ACM press.

[3] R. Attar and A. Fraenkel. Local feedback in full-text retrieval systems. *Journal of the Association for Computing Machinery*, 24(3):397–417, 1977.

[4] C. Buckley, A. Singhal, M. Mitra, and G. Salton. New retrieval approaches using SMART: TREC 4. In D.Harman, editor, *Proc. TREC-4*, pages 25–48, 1996.

[5] K. Collins-Thompson, P. Ogilvie, Y. Zhang, and J. Callan. Information filtering, novelty detection, and named-page finding. In *Proc. TREC-2002, the 11th text retrieval conference*, 2002.

[6] T. Doi, H. Yamamoto, and E. Sumita. Example-based machine translation using efficient sentence retrieval based on edit-distance. *ACM Transactions on Asian Language Information Processing*, 4(4):377–399, 2005.

[7] D. Harman. Overview of the TREC 2002 novelty track. In *Proc. TREC-2002, the 11th text retrieval conference*, 2002.

[8] L. Larkey, J. Allan, M. Connell, A. Bolivar, and C. Wade. UMass at TREC 2002:cross language and novelty tracks. In *Proc. TREC-2002, the 11th text retrieval conference*, 2002.

[9] X. Li and B. Croft. Novelty detection based on sentence level patterns. In *Proc. of CIKM-2005, The ACM Conference on Information and Knowledge Management*, 2005.

[10] D. Losada and R. T. Fernández. Highly frequent terms and sentence retrieval. In *Proc. 14th String Processing and Information Retrieval Symposium, SPIRE'07*, Santiago de Chile, October 2007.

[11] V. Murdock. *Aspects of sentence retrieval.* PhD thesis, University of Massachussetts, 2006.

[12] C. Nobata and S. Sekine. Towards automatic acquisition of patterns for information extraction. In *Proc. of International Conference of Computer Processing of Oriental Languages*, 1999.

[13] S. Robertson, S. Walker, S. Jones, M. HancockBeaulieu, and M. Gatford. Okapi at TREC-3. In D.Harman, editor, *Proc. TREC-3, the 3rd Text Retrieval Conference*, pages 109–127. NIST, 1995.

[14] I. Soboroff. Overview of the TREC 2004 novelty track. In *Proc. TREC-2004, the 13th text retrieval conference*, 2004.

[15] I. Soboroff and D. Harman. Overview of the TREC 2003 novelty track. In *Proc. TREC-2003, the 12th text retrieval conference*, 2003.

[16] A. Tombros and M. Sanderson. Advantages of query biased summaries in information retrieval. In *Proc. of SIGIR-98, the 21st ACM International Conference on Research and Development in Information Retrieval*, pages 2–10. ACM press, August 1998.

[17] E. Voorhees and D. Harman, editors. *TREC:Experiment and Evaluation in Information Retrieval*, chapter The TREC AdHoc Experiments, pages 79–97. The MIT press, 2005.

[18] R. White, J. Jose, and I. Ruthven. Using top-ranking sentences to facilitate effective information access. *Journal of the American Society for Information Science and Technology (JASIST)*, 56(10):1113–1125, 2005.

[19] J. Xu and B. Croft. Query expansion using local and global document analysis. In *Proc. SIGIR-96, the 19th ACM Conference on Research and Development in Information Retrieval*, pages 4–11, Zurich, Switzerland, July 1996.

[20] H. Zhang, H. Xu, S. Bai, B. Wang, and X. Cheng. Experiments in TREC 2004 novelty track at CAS-ICT. In *Proc. TREC-2004, the 13th text retrieval conference*, 2004.

Text Snippets from the DomGraph

Jacob Ratkiewicz[1,2]
jpr@cs.indiana.edu

Filippo Menczer[1,2]
fil@indiana.edu

[1]Complex Networks Lagrange Laboratory, Institute for Scientific Interchange Foundation, Torino, Italy
[2]Dept. of Computer Science, Indiana University, Bloomington, Indiana, USA

ABSTRACT

We explore the idea that the Document-Object Model tree of an HTML page — absent any semantic or heuristic interpretations of the tags and their positions — provides cues about the importance of the information it contains. This hypothesis is evaluated by constructing a DomGraph, i.e., a network of the DOM trees of pages connected by their hyperlinks, and using it in a snippet-extraction technique. In this process, we also address technical issues related to the processing of the resultant very large graph. Snippets produced by this technique are compared in a user study to those extracted by a reasonable and simple baseline method, and found to be clearly preferred by users.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Information networks, Performance evaluation (effectiveness), Question-answering (fact retrieval) systems*; H.3.7 [**Information Storage and Retrieval**]: Digital Libraries—*User issues*; H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia—*User issues*

General Terms

Algorithms, Design, Experimentation, Human Factors

Keywords

Document object model, graph topology, DomGraph, snippet extraction.

1. INTRODUCTION

Ranking a chunk of information by its usefulness in answering a query is a critical component of an information retrieval system. Various definitions of what a "chunk" is, and methods for ranking these chunks, have been proposed for use on the Web. The most widely-used concept of a unit of information on the Web is that of a single Web page, and the leading ranking algorithm for determining the prestige of these pages is PageRank [3], usually augmented by commercial search engines with proprietary improvements.

Users often interact with the results of a text information retrieval system by viewing a list of "snippets" of text from each retrieved document, in order to judge each document's usefulness in answering their query without actually visiting it. These snippets are classically derived from the text of the document, ignoring HTML markup if it is present. Snippet extraction algorithms use natural-language processing cues in an attempt to find coherent bits of text that contain the query terms and give the user some contextual information about the page when seen in isolation.

This paper attempts to unify the above two approaches. Rather than using PageRank at the page level to rank entire pages, we explore its use at the level of each page's Document-Object Model (DOM) tree, driving the ranking of relevant snippets of text from the page. In doing so, we explore the hypothesis that the structure of a page's DOM tree might lend cues about the relative importance of the different bits of information it contains — even absent any heuristic or semantic information about the HTML tags composing the tree.

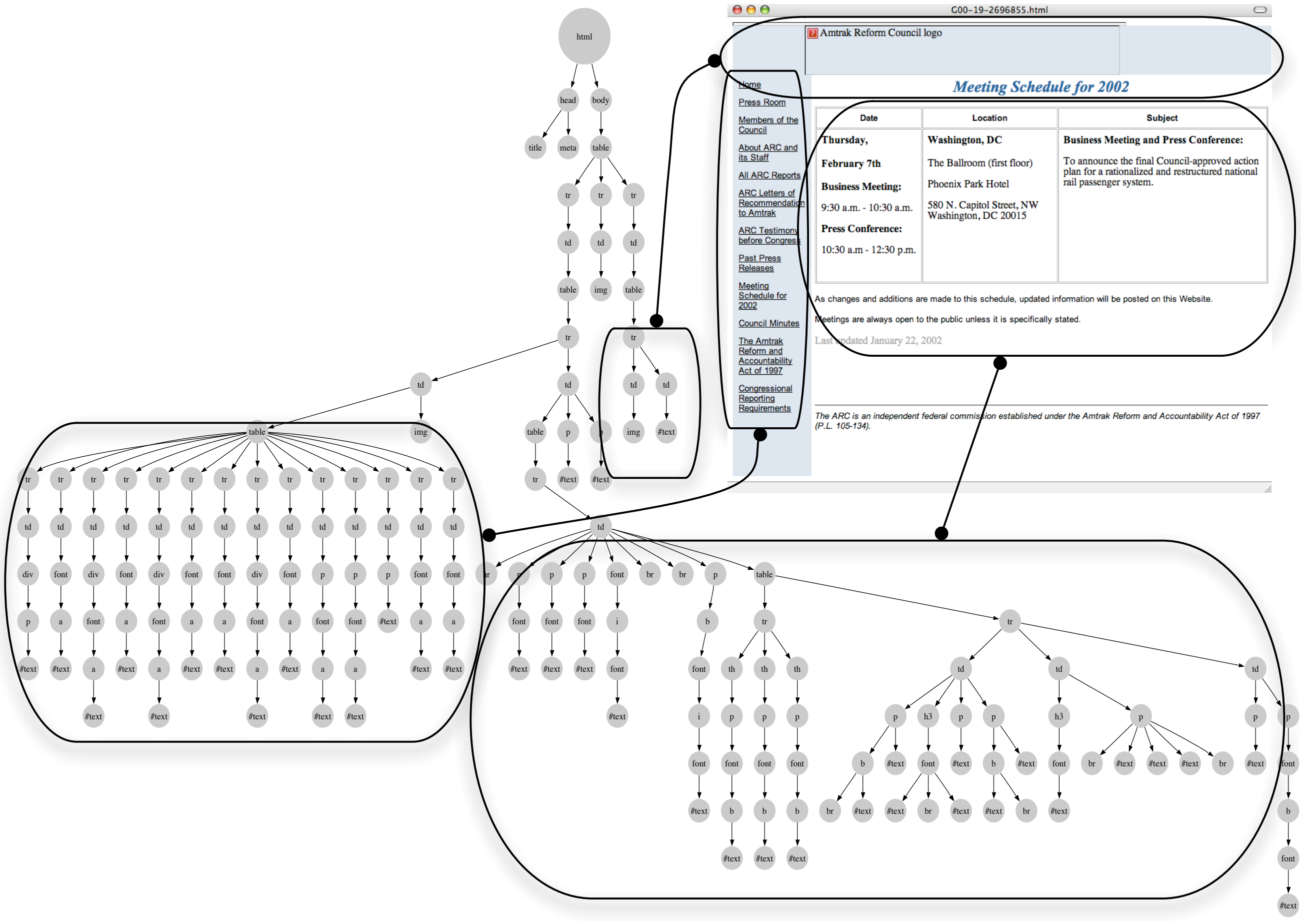
The main contributions of this paper are as follows:

- We describe the construction of the DomGraph, a hybrid graph comprised of the page link graph in conjunction with the DOM trees of individual pages.
- We discuss the technical difficulties in processing the large graph that results, in the process creating Webgraph++, a more scalable translation of the Webgraph library [2] to C++.
- We evaluate by a user study the use of this graph for selecting query-biased snippets from pages.

2. RELATED WORK

Many current algorithms for computing the prestige of nodes in a Web graph are based on PageRank [3], which assigns prestige in a manner proportional to indegree to a first approximation [7]. Ranking of Web objects at a level higher than that of pages has also been explored, such as in HostGraph [1] or SiteRank [11, 12]. One of this paper's contributions is the novel idea of ranking Web objects at a gran-

SIGIR 2008 Workshop on Focused Retrieval July 24, 2008, Singapore


G00-19-2696855.html

Amtrak Reform Council logo

Meeting Schedule for 2002

Home
Press Room
Members of the Council
About ARC and its Staff
All ARC Reports
ARC Letters of Recommendation to Amtrak
ARC Testimony before Congress
Past Press Releases
Meeting Schedule for 2002
Council Minutes
The Amtrak Reform and Accountability Act of 1997
Congressional Reporting Requirements

Date	Location	Subject
Thursday, **February 7th** **Business Meeting:** 9:30 a.m. - 10:30 a.m. **Press Conference:** 10:30 a.m - 12:30 p.m.	**Washington, DC** The Ballroom (first floor) Phoenix Park Hotel 580 N. Capitol Street, NW Washington, DC 20015	**Business Meeting and Press Conference:** To announce the final Council-approved action plan for a rationalized and restructured national rail passenger system.

As changes and additions are made to this schedule, updated information will be posted on this Website.

Meetings are always open to the public unless it is specifically stated.

Last updated January 22, 2002

The ARC is an independent federal commission established under the Amtrak Reform and Accountability Act of 1997 (P.L. 105-134).

Figure 1: A sample page from the .GOV dataset, along with its DOM representation. Page elements such as tables and navigation bars can greatly increase the complexity of a page's DOM without affecting its apparent length. Corresponding areas of the page and the DOM representation are highlighted.

ularity lower than that of a page, namely individual DOM elements.

The Document-Object Model view of HTML was codified by a W3C standard,[1] and is widely used. Use of the DOM for data extraction has been discussed by several previous works. One of the contexts in which it has been explored is that of accessibility issues. Gupta *et al.* [9] outline a method for condensing a page to its most essential elements to facilitate viewing of the page on mobile devices, or interpretation of the page by accessibility software such as screen readers. Their method is based on the DOM and involves a number of heuristic rules for selecting the important parts of the page, based on its genre. Buyukkokten *et al.* [4] describe "accordion summarization," a similar technique which uses the DOM in an attempt to identify standalone "semantic textual units" of the page which can be viewed as summaries of parts of the page. The user can then drill down to the part of the page which most interests her. The use of the DOM to guide computerized data mining efforts is explored by Can *et al.* [5], who use DOM cues and heuristics to identify and extract postal addresses from HTML pages. The use of the DOM for topical crawlers has been proposed by Pant and Menczer [13] to examine the textual context of a link and evaluate whether to follow it.

In the domain of segmenting unstructured text using only endogenous cues, relevant works include LexRank [6] and TextTiling [10]. In the former, the authors propose a method for determining the prestige (or *salience*) of sentences in a document by a method which involves computing similarities between pairs of sentences, building a network of these pairs in which two sentences are connected if their similarity is greater than a given threshold, and then running a modified PageRank on this network. This method is proposed to drive an extractive summarization algorithm. TextTiling is a technique for breaking a long stream of text (a *discourse*) into a series of smaller *topics*. The method works by computing similarities between adjacent "blocks" (3-5 sentence chunks) of text, and setting topic boundaries in places where the similarity value is relatively low.

[1]`http://www.w3.org/DOM/`

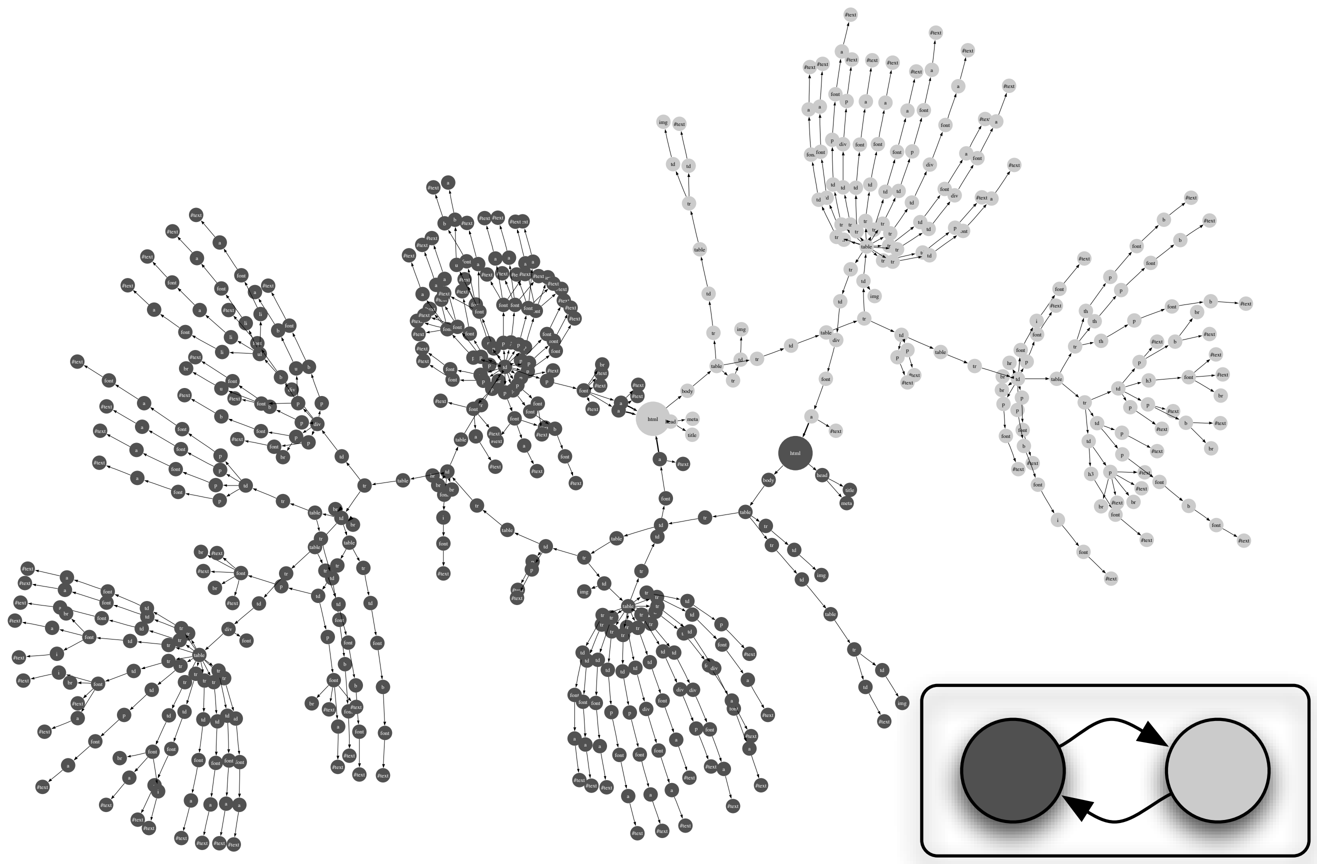


Figure 2: A subset of the DomGraph containing nodes belonging to two neighboring pages, one of them the page in Figure 1. For ease of visual identification, the root (`<html>` tag) of each DOM is displayed as a larger circle. In the Page graph (inset), the nodes representing each of these pages link to each other; however, in the DomGraph, distinct anchor tags each contribute their inlink to the target page's `<html>` tag.

Turpin *et al.* [14] outline a method for fast snippet generation in the search engine context, but focus more on the speed of the production than the quality of what is produced. A recent patent by Google, Inc. [8] describes a method for producing snippets, but no claim is made that this method is actually what is used by Google, and their true method is likely more complex.

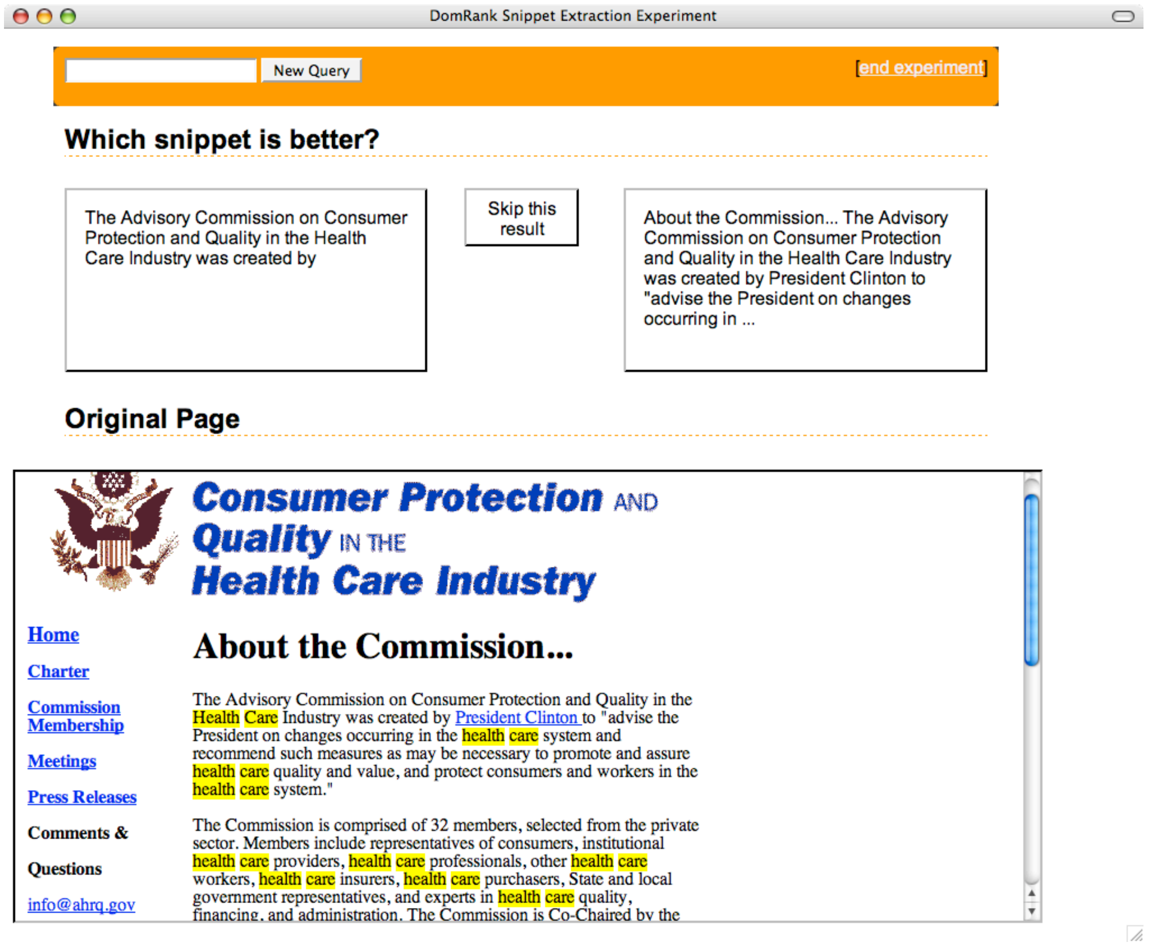
3. THE DOM GRAPH

The DOM is in essence a way of viewing an HTML page as a tree — a fully-connected, directed acyclic graph with the node representing the `<html>` tag at the root. This simplifies access to the contents of the page by programming languages such as JavaScript, which can modify the DOM on the client side in order to implement interactive Web pages. Figure 1 illustrates a sample page and its DOM representation.

We wish to use this graph representation of each page as a stand-in for that page in the "standard" link graph of all pages. That is, we build a graph in which the nodes are elements of each page's DOM tree, and the links are induced both by parent-child connections in the DOM, and by hyperlinks between pages. Hyperlinks are treated as edges between the tag defining the hyperlink and the root of the target page's DOM tree. In the case of anchored hyperlinks (i.e. links of the form `<a href="a.com#anchor">...`), we connect the tag defining the hyperlink and the parent tag of the anchor in the target document. This expansion of a page into its DOM nodes typically increases the size of the graph by two orders of magnitude. Figure 2 illustrates a subset of the DomGraph. The two larger nodes in this graph are the `<html>` tags of the pages represented; the smaller nodes are nodes representing DOM elements created by the expansion.

Data is drawn from the `.gov` corpus.[2] This corpus contains about a million documents comprising approximately 20 GB of text. Not all of the documents are HTML; plain text documents and documents in other formats are also included. From this collection we extracted two graphs. The first is a "standard" graph, which we call S, in which the nodes are pages and the links between nodes are induced by page hyperlinks. The second is the DomGraph, D, which is described above. This latter graph is comprised of around 220 million nodes connected by around 230 million edges.

[2]`http://es.csiro.au/TRECWeb/govinfo.html`


DomRank Snippet Extraction Experiment

New Query

[end experiment]

Which snippet is better?

The Advisory Commission on Consumer Protection and Quality in the Health Care Industry was created by

Skip this result

About the Commission... The Advisory Commission on Consumer Protection and Quality in the Health Care Industry was created by President Clinton to "advise the President on changes occurring in ...

Original Page

Consumer Protection AND Quality IN THE Health Care Industry

Home

Charter

Commission Membership

Meetings

Press Releases

Comments &

Questions

info@ahrq.gov

About the Commission...

The Advisory Commission on Consumer Protection and Quality in the Health Care Industry was created by President Clinton to "advise the President on changes occurring in the health care system and recommend such measures as may be necessary to promote and assure health care quality and value, and protect consumers and workers in the health care system."

The Commission is comprised of 32 members, selected from the private sector. Members include representatives of consumers, institutional health care providers, health care professionals, other health care workers, health care insurers, health care purchasers, State and local government representatives, and experts in health care quality, financing, and administration. The Commission is Co-Chaired by the

Figure 3: A screenshot of the user study experiment in action, with the query "health care." The candidate snippets are shown on the left and right, with the original page beneath. The position of the DOM snippet and baseline snippet was randomized at each instance. If at any time the user wished to enter a new query he or she could do so with the form at the top of the screen.

A graph of such size was a challenge to analyze with the computing hardware we had available. The library, Webgraph [2], that we chose to perform the initial analysis was written in Java. This language limits the sizes of certain data structures in a way which made our analysis impossible. To surmount this difficulty we undertook to translate this library into C++, and the resulting Webgraph++ library is available online.[3]

In order to leverage the S and D graphs for information retrieval, we compute the PageRank of the nodes in each. To distinguish the PageRank computed on the page graph from the results of that algorithm run on the DomGraph, we refer to the latter as DomRank. Further, we built two text indices using the Lucene[4] index and search platform. The first, for the standard graph, indexed the raw text of each page (i.e. with HTML markup stripped out). The second indexed the text of all textual DOM nodes. We refer to the first of these indices as the "page index," and the second as the "DOM index."

[3]`http://homer.informatics.indiana.edu/~nan/webgraph/`
[4]`http://lucene.apache.org/`

4. EXTRACTING TEXT SNIPPETS

The goal of our experiment was to measure the effectiveness of DOM cues for segmenting text in a meaningful way. To this end, we evaluated snippets extracted from the DOM against snippets extracted by a baseline method. We describe each of these methods in turn. Each method begins with a 'bag' of query terms, with stopwords removed, and an integer n which signifies which page in the result list should be considered for snippet extraction. It then performs a query against the page index to arrive at a list of page identifiers. The methods differ in what they do with these identifiers, as follows.

Baseline method. As a baseline method, we attempted to mimic the behavior of commercial search engines. This method takes the n-th page identifier from the list derived as above, and retrieves the plain text of the associated page

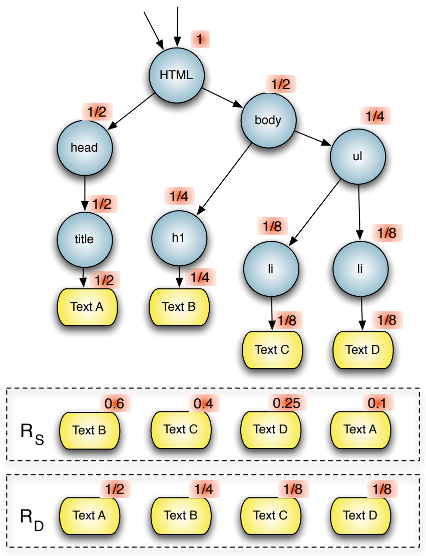


Figure 4: The DOM-based method for snippet extraction. Given a fixed page, the method extracts all text nodes and computes two rankings — one based on each node's similarity to the query as determined by Lucene, called R_s, and another by its DomRank, R_d. The method then returns the node that has the highest aggregate score, determined by $\alpha R_s + (1-\alpha)R_d$ for a constant α. This illustration assumes an incoming DomRank flow of 1.0 to the `html` node, and neglects the effect of the damping factor. Query similarity values are hypothetical. The winning node in this case would be `Text B` — it is ranked 4 (highest) in R_s, and 3 in R_d; thus in its case, $\alpha R_s + (1-\alpha)R_d = 0.7 \cdot 4 + 0.3 \cdot 3 = 3.7$.

(with HTML markup removed). It segments the text into sentences using the OpenNLP software package, trained on a corpus of text from the Wall Street Journal.[5] The method then attempts to return one of the following, in decreasing order of preference:

1. A single sentence containing all of the query terms.
2. A pair of sentences which together contain all the query terms
3. A "window" of text, possibly containing several sentences, which contains as many query terms as possible.

The size of the "window" in (3) is fixed at 30 words; further, if a snippet picked by (1) or (2) above is longer than 30 words, it is trimmed around the query terms with preference given to removing words at the end of the sentence. In our best assessment, this method seems to produce snippets similar to those returned by commercial search engines.

[5]Software package and trained model: `http://opennlp.sourceforge.net/`

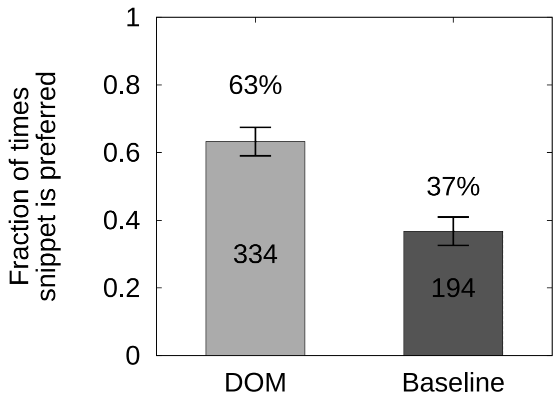


Figure 5: Results of the user study. The error bars are for a 95% confidence interval. The difference is statistically significant with $p < 0.01$ from a two-tailed t-test.

DOM method. The DOM method takes the n-th page identifier from the list just as the baseline method does; however, it derives its snippet in a different way. It performs a second query against the index of DOM nodes, limiting its search to the nodes which comprise the page used by the baseline method. This yields a list of text nodes ranked by their similarity to the query. It then creates a new list in which the rank of each item is determined by a linear combination of the each item's rank in the text similarity list and its rank in a list ordered by DomRank. Thus, if R_s is the node's rank when ordered by similarity and R_d is its rank by DomRank, its rank in the result list is determined by a sort on the quantity

$$\alpha R_s + (1-\alpha)R_d.$$

The value $\alpha = 0.7$ was chosen empirically. The DOM text at the head of the resulting list is what is returned as a snippet. If it is longer than 30 words, the 30-word-long window which contains as many query terms as possible is what is returned. Figure 4 illustrates this process.

5. EVALUATION

To evaluate the use of the DomGraph and DomRank for extracting important page cues, we performed a Web-based user study[6] comparing the two snippet-extraction algorithms. Users were asked to enter a query of their choice. They were then shown a pair of snippets, derived as above with $n = 1$ initially, and asked to click the one they thought was more helpful, or skip the result if neither was the clear winner. When the user rated either of the two snippets or clicked "skip," he or she was shown a new pair of snippets with $n \leftarrow n + 1$ (i.e. from the next page in the result set). Pages for which both methods produced the same snippet were skipped. At any point the user wished, he or she could enter a new query or end the experiment. A screenshot of the experiment in action is shown in Figure 3. The average length of a snippet returned from the baseline method was 27 ± 3 words, or 149 ± 10 characters; the DOM-based

[6]Indiana University IRB #07-11684

method's snippets were significantly shorter, at 12 ± 2 words or 63 ± 9 characters.

The experiment was active for approximately 1 week, and garnered around 40 participants who rated 528 snippets all together. The number of times a snippet created by each method was chosen over the other is shown in Figure 5.

Thus, the snippets extracted from the DOM were preferred approximately 50% more than those extracted by the baseline method. We are encouraged by these results, pointing to the DomGraph as a useful tool in Web applications.

6. DISCUSSION AND FUTURE WORK

The method presented here for extracting snippets directly from the DOM tree of Web documents does not rely on any heuristics or semantic information about the text, but solely on graph topological information. Our experimental evaluation shows it to outperform a simple and reasonable method that relies on textual cues alone. We do not present this as an argument that DOM-based snippet extraction would clearly outperform a state-of-the-art method, but rather to support the conclusion that the topology of the DOM contains semantic information that should be exploited for information retrieval.

Future work should deal with optimizing the computation of DomRank for the large graph which results from expansion of each page's DOM, as well as the size of the index involved. Among the techniques that might be explored are trimming each page's DOM to remove irrelevant nodes (i.e. nodes that do not affect the DomRank of any text nodes), and the possibility that the DomRank may be computed in some optimized way from the PageRank of the corresponding page graph.

An obvious candidate for application of techniques of this type is Web page summarization. This has been previously explored [4], but our method is unique in that it does not rely on page- or page-genre specific heuristics. Question answering applications could also benefit from this method's ability to select important bits of text from a list of candidates, based on the structure of the Web page from which the snippets are derived. More in general, applications such as clustering and classification could be improved by this technique's ability to select important items (and thus remove noise) from Web page data. DomRank even applies to non-textual data appearing in HTML pages, such as images and other media.

7. ACKNOWLEDGMENTS

The authors wish to thank the members of the NaN group at Indiana University for helpful discussions. They would also like to thank the participants in the user study, including the WEBIR and SIGIR mailing list members. This work is supported in part by the Indiana University School of Informatics in Bloomington, Indiana, and the Institute for Scientific Interchange Foundation in Turin, Italy.

8. REFERENCES

[1] Bharat, K., Chang, B.-W., Henzinger, M. R., and Ruhl, M. Who links to whom: Mining linkage between web sites. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 51–58.

[2] Boldi, P., and Vigna, S. The webgraph framework i: compression techniques. In *WWW '04: Proceedings of the 13th international conference on World Wide Web* (New York, NY, USA, 2004), ACM, pp. 595–602.

[3] Brin, S., and Page, L. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems 30*, 1–7 (1998), 107–117.

[4] Buyukkokten, O., Garcia-Molina, H., and Paepcke, A. Accordion summarization for end-game browsing on PDAs and cellular phones. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2001), ACM, pp. 213–220.

[5] Can, L., Qian, Z., Xiaofeng, M., and Wenyin, L. Postal address detection from Web documents. In *WIRI '05: Proceedings of the International Workshop on Challenges in Web Information Retrieval and Integration* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 40–45.

[6] Erkan, G., and Radev, D. R. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research 22* (2004), 457–479.

[7] Fortunato, S., Boguñá, M., Flammini, A., and Menczer, F. On local estimations of PageRank: A mean field approach. *Internet Mathematics*. (to appear).

[8] Google Inc. Detecting query-specific duplicate documents. US Patent no. 6615209, 2003.

[9] Gupta, S., Kaiser, G., Neistadt, D., and Grimm, P. DOM-based content extraction of HTML documents. In *WWW '03: Proceedings of the 12th international conference on World Wide Web* (New York, NY, USA, 2003), ACM, pp. 207–214.

[10] Hearst, M. A. TextTiling: segmenting text into multi-paragraph subtopic passages. *Comput. Linguist. 23*, 1 (1997), 33–64.

[11] Kleinberg, J. M. Authoritative sources in a hyperlinked environment. *Journal of the ACM 46*, 5 (1999), 604–632.

[12] Meiss, M. R., Menczer, F., Fortunato, S., Flammini, A., and Vespignani, A. Ranking web sites with real user traffic. In *WSDM '08: Proceedings of the international conference on Web search and Web data mining* (New York, NY, USA, 2008), ACM, pp. 65–76.

[13] Pant, G., and Menczer, F. Topical crawling for business intelligence. In *ECDL '03: Proc. 7th European Conference on Research and Advanced Technology for Digital Libraries* (Trondheim, Norway, 2003).

[14] Turpin, A., Tsegay, Y., Hawking, D., and Williams, H. E. Fast generation of result snippets in web search. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval* (New York, NY, USA, 2007), ACM, pp. 127–134.

Modelling Anchor Text Retrieval in Book Search based on Back-of-Book Index

Hengzhi Wu
Queen Mary, University of London, UK
hzwoo@dcs.qmul.ac.uk

Gabriella Kazai
Microsoft Research
Cambridge, UK
gabkaz@microsoft.com

Thomas Roelleke
Queen Mary, University of London, UK
thor@dcs.qmul.ac.uk

ABSTRACT

This paper proposes a probabilistic logic abstraction for modelling *tf*-boosting approaches to anchor text retrieval, adapted for the task of page-search in books. The underlying idea is to view the back-of-book index (BoBI) as a list of anchors pointing to pages in the book. First, we model the direct application of hypertext-based *tf*-boosting to books and show that this naive method of propagating anchor-text from the BoBI does not deliver the desired *tf*-boosting effect. To address this, we then propose a revised anchor-text retrieval model based on a novel voter approach. In this approach, each page of the book, where a given term occurs, acts as a virtual voter to the pages referenced by the BoBI for that term. The *tf*-boosting effect is achieved by propagating term weights from the voter pages to the pages in the BoBI. We use probabilistic Datalog for the high-level abstract modelling of retrieval strategies, which allows for the evolution and transfer of successful techniques from one domain, such as anchor-text retrieval in Web IR, to a similar domain, such as book search.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval Models

General Terms

Algorithms

1. INTRODUCTION

As a result of ongoing digitization and mass-digitization projects around the world, searching over a large collection of digitized books has become a new area of interest in the field of IR. Reflecting this, the INitiative for the Evaluation of XML IR (INEX) has launched a Book Search track[1] in 2007. The track runs two core tasks: 1.) The Book Retrieval task, where the units of retrieval are whole books; and 2.) The Page in Context task, which studies the application of passage and XML element retrieval methods to books. Both tasks have been shown to benefit from the use of structural information extracted from digitized books [11, 15, 14, 20]. In this paper, we focus on the Page in Context task, and view the challenge of locating relevant pages inside books as a task of finding the best entry points into the book content.

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

We build on the observation that books are typically highly structured and, in particular, we leverage the potential presented by a book's back-of-book index (BoBI) as a searching and browsing tool. Unlike the table of contents, a BoBI provides specific information on relevant sections of text by pointing to key concepts discussed in the book [1, 8]. Its significance is that it distinguishes important topics and concepts, as identified by the author, from other keywords that may appear in the book.

In contrast to the traditional full-text index used in IR, which reflects the global distribution of terms (among pages), but provides limited evidence for selecting best entry points (e.g., best pages), an author generated BoBI may hold the key for identifying the best entry points. Although some term statistics may also be derived from a BoBI (e.g., number of linked pages), the full-text index is still likely to provide a good source of information for estimating relevance. Thus, in this paper, we propose a strategy for combining the traditional full-text index with the BoBI by integrating anchor text retrieval in the context of the page finding task.

The underlying idea is to view the BoBI as a set of anchors pointing to respective parts of the text in the book. Given this view, the question of how to best utilise this source of information for the identification and retrieval of relevant pages inside a book arises.

Term propagation mechanisms in the form of anchor text or hypertext retrieval have been successfully applied in Web IR [2], and have also been found to be beneficial for enterprise [9], and XML retrieval [19]. The underlying principle of term propagation (also referred to as context augmentation) is the use of terms from linked source documents in the representation of a destination document. A link may be a structural link between components in a document hierarchy, or a hyperlink connecting two web pages, for example. A consequence of term propagation is that the destination document receives a term frequency boost, which can then lead to a higher retrieval score being assigned to the document by a search system.

In this paper, we investigate the modelling of term propagation strategies adapted to the task of book search, exploiting the BoBI as a list of anchor texts linking to pages in the book. We employ probabilistic Datalog (PD) [4], an expressive high level language, for modelling information retrieval taking into account the specifics of book search.

SIGIR 2008 Workshop on Focus Retrieval
July 24, 2008, Singapore

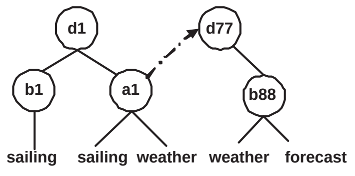


Figure 1: Structured documents with hyperlink

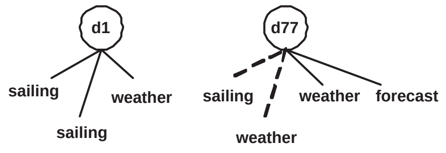


Figure 2: Augmentation through hierarchy and hyper-link

The two main contributions of this paper are: 1.) The modelling of *tf*-boosting strategies in PD, and 2.) The proposal of the voter model for *tf*-boosting.

The paper is structured as follows. Section 2 introduces anchor text retrieval and modelling in PD. Section 3 discusses why the classic anchor text retrieval strategy fails in book search, and introduces a voter model for the task. Section 4 formally defines anchor text retrieval in book search, modelling the strategies in PD. Finally, section 5 summarises the paper and outlines future work.

2. BACKGROUND

In this section, we present two examples of anchor text retrieval applied to structured and hyperlinked documents, and review the basic principles of probabilistic modelling in PD.

2.1 Context Augmentation

Term propagation in anchor text retrieval is an application of context augmentation [13, 16], where the occurrence of a term in the anchor text (of the source document) is propagated to the destination document via a structural or hyperlink that connects the two documents. As a result, the content of the destination document is augmented and the term frequencies (*tf*) of terms in the destination document are increased.

We present two context augmentation mechanisms, depending on the type of linkage between documents. We illustrate these using the small sample collection of two documents, d_1 and d_{77}, shown in Figure 1. Each document in the collection is a tree with nodes and edges. The nodes of a tree may be composite elements, which contain other nested nodes, and leaf nodes, which contain the text of the document. Nodes are labelled based on the following convention: “d” represent the root node of a document, “b” is short for body, and “a” indicates an anchor. A solid line between nodes represents a structural link and a dash-dot arrow from an anchor element implies a hyperlink.

2.1.1 Hierarchical augmentation

Hierarchical augmentation is typically applied to structured documents (e.g., in XML IR), where the occurrence of a term in a child-node is propagated to the representation of its parent node. In this case, the term frequency is propagated upwards along the structure of the document tree. One reason to do this is to support the retrieval of composite nodes, which may provide a better match than their child nodes alone (e.g., when one query term is contained in one child node and another query term in another child node) [5].

For example, in Figure 1, we obtain the augmented representation of d_1 by propagating the *tf*'s of the terms that occur in the child nodes: b_1 and a_1. We use the notation $n_L(t,c)$ to denote the *tf* of a term t in a context c, where n_L stands for number of locations. The term frequencies in the child nodes are thus $n_L(sailing, b_1) = 1$ and $n_L(sailing, a_1) = 1$, and $n_L(weather, a_1) = 1$. After augmentation, we obtain the following *tf*'s in d_1: $n_L(sailing, d_1) = 2$ and $n_L(weather, d_1) = 1$. Augmentation, thus, allows for d_1 to act as a combined representation of its child nodes. This is illustrated in the left side of Figure 2.

One typical question in hierarchical augmentation concerns whether to propagate raw term frequencies or apply some kind of normalization or weighting [17]. In this paper, we model both raw and weighted term frequency propagations.

2.1.2 Anchor-text-based augmentation

Applying augmentation in a hyperlinked environment allows to combine evidence from external documents for the representation and retrieval of a hyperlinked document.

Tf-boosting is one such strategy, which directly propagates and combines term frequencies into the representation of the destination document. Anchor text retrieval using *tf*-boosting have in fact been shown to improve retrieval performance in Web IR [2]. Furthermore, *tf*-boosting was found to be more effective than other link-based strategies in Web IR [10].

We illustrate *tf*-boosting using the sample collection of Figure 1. Here, the anchor element a_1 has a hyperlink pointing to document d_{77}. Assuming that the representation of d_{77} already contains the terms “weather” and “forecast” as a result of hierarchical augmentation from b_{88}, we start with the following *tf*'s in d_{77}: $n_L(weather, d_{77}) = 1$ and $n_L(forecast, d_{77}) = 1$. The terms “sailing” and “weather” in a_1 are terms of the anchor text, which is often interpreted as a description of the destination document. By propagating the terms of the anchor text to the destination document, we can obtain a *tf*-boosted representation of d_{77}, where the *tf* of “weather” is boosted from 1 to 2. In addition, the term “sailing” which did not occur in d_{77} originally, is now also represented. After propagation, the term frequencies for d_{77} are: $n_L(sailing, d_{77}) = 1$, $n_L(weather, d_{77}) = 2$, and $n_L(forecast, d_{77}) = 1$. This is illustrated in the right hand side of Figure 2.

2.2 Modelling in Probabilistic Datalog

We model retrieval strategies in probabilistic Datalog (PD) [4]. PD is a combination of deterministic Datalog (a query language used in deductive databases) and probability theory. It is a highly expressive and flexible platform for modelling and prototyping different retrieval strategies.

To introduce PD, we implement the example term propagation strategies discussed in Section 2.1. First, we create the following relational structures to store the necessary information about the collection (see Figure 4): The `term(Term,Context)` relation is used for storing term occurrences along with their corresponding location information, e.g., the term “sailing” occurs in the anchor element a_1. The location information is stored as

```
pdRule := {assumption} goal ':-' body
assumption :=
    'DISJOINT' | 'INDEPENDENT' | 'SUBSUMED' |
    'SUM' | 'MAX'
goal ::= NAME {assumption} '(' varList ')'
body ::= subgoalList
varList := var {',' varList}
var := VARIABLE
subgoalList := subgoal {'&' subgoalList}
subgoal := NAME '(' argList ')' {'|' evidenceKey}
argList := arg {',' argList}
arg := VARIABLE | constant
evidenceKey := '(' varList ')'
```

Figure 3: Extract of PD Syntax

term	
Term	Context
sailing	d1/b1
sailing	d1/a1
weather	d1/a1
weather	d77/b88
forecast	d77/b88

part_of	
Child	Parent
d1/a1	d1
d1/b1	d1
d77/b88	d77

link	
Context	URL
d1/a1	www.bbc.co.uk

site	
Context	URL
d77	www.bbc.co.uk

Figure 4: Relations of link-based retrieval

a path, expressed in XPath [2], however, for readability we use a simplified syntax throughout this paper, e.g., "d1/a1". The hierarchical structure of documents in the collection is represented in the `part_of(Child,Parent)` relation, e.g., "d1/a1" is a part of "d1". The hyperlink structure is modelled through two relations: `link(Context,URL)` and `site(Context,URL)`. The `link` relation associates the XPath of an anchor node with the URL of the destination, while the `site` relation maps the URL to the destination document.

The syntax of our PD is given in Figure 3. In this syntax, everything between a pair of curly brackets, i.e. '{' and '}', is optional; the assumption 'SUM' is an alias of 'DISJOINT' hence the two are interchangeable.

A PD rule consists of a goal, a body and arguments. A rule is evaluated such that the goal is true if and only if the body is true. For example, the following rule demonstrates a common term matching strategy in IR, stating that if T occurs in a query and T is a term in a document D, then D is retrieved.

```
retrieve(D) :- query(T) & term(T, D).
```

The weights (probabilities) associated with the results of a rule are obtained through probability computations, which involve probability estimation and probability aggregation. Probability estimation assigns initial probabilities to the rows of a relation; the probabilities may be term-based, i.e. *tf*, or document-based (or element-based in XML), i.e. *df*. Probability aggregation, on the other hand, combines probabilities of one or multiple relations, e.g., by summation or multiplication.

[2]http://www.w3.org/TR/xpath

term(T,S) & part_of(S,D)			
Term	Context	Child	Parent
sailing	d1/b1	d1/b1	d1
sailing	d1/a1	d1/a1	d1
weather	d1/a1	d1/a1	d1
weather	d77/b88	d77/b88	d77
forecast	d77/b88	d77/b88	d77

(a) intermediate result of rule body

augTerm	
Term	Parent
sailing	d1
sailing	d1
weather	d1
weather	d77
forecast	d77

(b) hierarchical augmentation

augTerm_d		
$1/N_L(d)$	Term	Parent
1/3	sailing	d1
1/3	sailing	d1
1/3	weather	d1
1/2	weather	d77
1/2	forecast	d77

(c) probability estimation

w_augTerm		
$P(t\|d)$	Term	Parent
2/3	sailing	d1
1/3	weather	d1
1/2	weather	d77
1/2	forecast	d77

(d) probability aggregation

Figure 5: Intermediate results of hierarchical augmentation

For instance, to estimate the within-document *tf*'s of a term T given that it occurs in document D, we have the following rule:

```
w_term(T, D) :- term(T, D) | (D).
```

The rule contains a special form of our probabilistic Datalog variant: The "| (D)" is the so-called evidence key, so that the tuples in the goal "w_term" (stands for weighted term) have the probabilistic semantics $P(T|D)$. We will return to this in more detail in sections 4.2 and 4.3. Further information about PD can be found in [4, 7, 18].

2.2.1 Hierarchical augmentation in PD

Let us now present a model of augmentation in PD. First, we give the PD rules for the hierarchical augmentation described in Section 2.1.1:

```
augTerm(T, D) :- term(T, S) & part_of(S, D).
augTerm_d(T, D) :- augTerm(T, D) | (D).
w_augTerm SUM(T, D) :- augTerm_d(T, D).
```

The first rule yields the augmented term frequency for the term T in document D by evaluating the predicates that T occurs in a context S, and that the context S is part of a document D. The second rule estimates the *tf* for `augTerm`. Finally, the third rule aggregates the probabilities and yields the final results.

Figure 5 illustrates the steps for processing this hierarchical augmentation rule. Figures 5(a) and 5(b) show the intermediate and final results of the first rule; Figure 5(c) shows the result of the probability estimation, i.e. the second rule; and Figure 5(d) shows the results of the probability aggregation, i.e. the third rule.

2.2.2 Anchor-text-based augmentation in PD

The PD rules for anchor text retrieval with *tf*-boosting presented in section 2.1.2 are given as follows:

```
augTerm(T, D) :- term(T, S) & part_of(S, D).
augTerm(T, D) :- term(T, A) & link(A, U) & site(D, U).
augTerm_d(T, D) :- augTerm(T, D) | (D).
w_augTerm SUM(T, D) :- augTerm_d(T, D).
```

term(T,A) & link(A,U) & site(D,U)					
Term	Context	Context	URL	Context	URL
sailing	d1/a1	d1/a1	www.bbc.co.uk	d77	www.bbc.co.uk
weather	d1/a1	d1/a1	www.bbc.co.uk	d77	www.bbc.co.uk

(a) intermediate result of rule body

augTerm	
Term	Parent
sailing	d77
weather	d77

(b) anchor-text-based augmentation

augTerm	
Term	Parent
sailing	d1
sailing	d1
weather	d1
weather	d77
forecast	d77
sailing	d77
weather	d77

(c) Union of hierarchical and anchor-text-based augmentation

augTerm_d		
$1/N_L(d)$	Term	Parent
1/3	sailing	d1
1/3	sailing	d1
1/3	weather	d1
1/4	weather	d77
1/4	forecast	d77
1/4	sailing	d77
1/4	weather	d77

(d) probability estimation

w_augTerm		
$P(t\|d)$	Parent	Term
2/3	sailing	d1
1/3	weather	d1
1/2	weather	d77
1/4	forecast	d77
1/4	sailing	d77

(e) probability aggregation

Figure 6: Intermediate results of tf-boosting

The first rule performs hierarchical augmentation to obtain the original within-document *tf* of terms. The anchor text based *tf*-boosting is performed by the second rule in two steps: 1.) The propagated term frequency for the term T in document D is obtained by evaluating the predicates that the term T appears in a context A, where A has a hyperlink to a URL U, which is associated to a document D; and 2.) The propagated *tf* is combined with the original *tf* of term T in document D. The third and fourth rules estimate and aggregate probabilities, respectively, to yield the final probability scores.

The procedures for processing the *tf*-boosting rules (rules 2-4) are illustrated in Figure 6. Figures 6(a) and 6(b) show the intermediate results for obtaining the propagated term frequencies, i.e. the first step of *tf*-boosting; Figure 6(c) shows the combination of the boosted and the original *tf*'s, i.e. the second step of *tf*-boosting. Figures 6(d) and 6(e) show the results of the probability estimation and aggregation processes.

3. BACK-OF-BOOK INDEX (BOBI)

As mentioned earlier, the rich structure of books is viewed as potentially useful for improving the retrieval effectiveness of book search approaches. Generally, the table of contents (TOC) and the BoBI are regarded as the two main sources of information to aid readers in locating relevant content in books [12]. While the benefits of a hyperlinked TOC's seem obvious, the use of the BoBI in book search tasks is less trivial. In this section, we investigate and discuss the applicability of BoBI for page-level search in books.

3.1 TF-Boosting and the BoBI: Discussion

A BoBI is, in essence, a cross-reference lockup mechanism aimed to support readers of printed books to locate information quickly using keywords as their entry into the content of the book. A BoBI is usually created by the author (or editor)[3] of the book, and as such, it reflects a somewhat personalised view on what the important terms/concepts of a book are. On the other hand, it represents a relevance association in the form of $\langle term, page \rangle$ pairs, similar to an inverted index used in IR. Furthermore, it can also be likened to hyperlink structures of the Web, whereby the anchor text of a BoBI term points to a content page.

[3]Although automatic index generation tools may also be used.

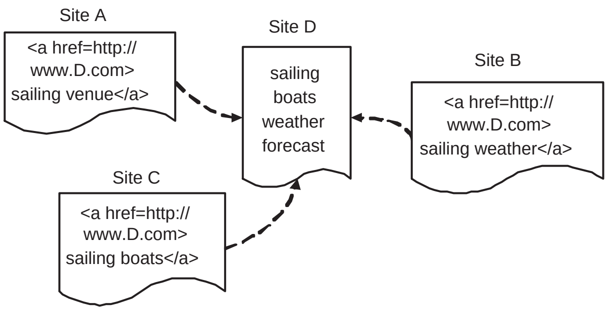


Figure 7: Anchor text topology in Web IR

Inspired by the successful application of anchor text retrieval in Web IR, one may attempt to incorporate the same mechanisms into book search by applying them to the BoBI of books. However, as we demonstrate next, such a direct application of the technology is not suitable for books.

Let us illustrate this with an example, contrasting the link structure of the Web and that of a book. Figure 7 shows three web pages A, B and C, each of which links to a fourth D page. Anchor text retrieval here is based on the principle of exploiting the in-link structure of page D's Web graph and propagating terms from the source pages to D. The more in-links to a page, the more terms are propagated and the stronger the effect of *tf*-boosting.

In a book, the BoBI contains out-links to pages of the book, where index terms occur (see Figure 8). In contrast to the Web scenario, however, the referred pages are likely to receive only a small number of in-links from the BoBI (one in-link per referred page is the most likely case). This means that the boost on referred pages is likely to have little effect. To address this, we propose in the next section a modification to the classic anchor text retrieval strategy and adopt this new voter model approach for the task of page retrieval in books.

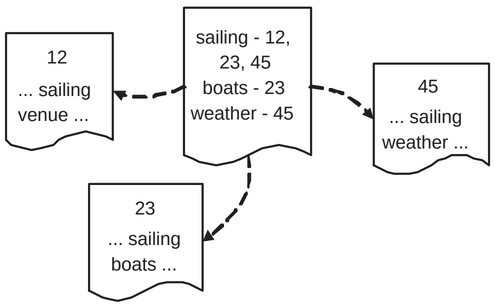


Figure 8: Anchor text topology in book search

3.2 TF-Boosting and the BoBI: A Voter Model

The occurrence of a term in the BoBI of a book implies that it is considered (by the author) of more importance than the rest of the terms on the referenced page that do not appear in the BoBI. Intuitively, the pages 'cited' in the BoBI could be considered as "best entry points" for looking up information in the book. This is further supported by the observation that topics (and alas associated keywords) spanning a number of pages are often only listed in the BoBI once, directing the reader to the first page in a sequence of pages on the topic. To incorporate this characteristic of the BoBI within a retrieval strategy, we employ *tf*-boosting as a method for combining traditional retrieval strategies based on term weights with adapted anchor text retrieval based on the BoBI.

Assuming that the BoBI terms are exact copies of the terms that occur in the referenced pages [4], the BoBI can be considered a subset of a full-text index (FTI) generated at the page-level (i.e., using pages as the document units). Given the intersection of FTI and BoBI, the terms common to both sets could be viewed as providing an association between the pages in the FTI and the pages in the BoBI, each shared term acting as a kind of anchor text link. Based on this view, we can consider each page associated with a given term in the FTI as a virtual voter to the pages referenced by the BoBI: by propagating term weights to the pages in the BoBI, each voter thereby votes for the "best entry points" in the book.

A range of possible options exist for deriving a suitable voting weight, i.e. the propagated term weight. One option is to redistribute the terms in the FTI over the page frequency of terms in the BoBI, i.e. $n_{P,bobi}(t)$ (see section 4.3 for formal definition).

Based on this idea of a voter model, we propose the following *tf*-boosting approach:

Distribute the page frequency of a term t *in the FTI across the occurrences of the term in the BoBI. With regards to voting this means that each entry of the BoBI receives votes from entries of the FTI.*

We detail our model based around the above voting-based *tf*-boosting idea in the next section.

4. MODELLING TF-BOOSTING FOR PAGE SEARCH IN BOOKS

This section defines anchor-text-like retrieval strategies that lead to *tf*-boosting for pages of a book. First, we detail the schema of our book collection. We then present a naive anchor-text propagation approach, and a refined approach expressed as a voter model.

4.1 Schema

A PD engine uses database-like tables for indexing, where a schema is a list of tables and corresponding attributes in the header of tables. We designed our schema based on a collection of books provided by the INEX 2007 Book Search track [5]. A book in this corpus is stored in DjVu XML [6], where the basic structure can be summarised as follows:

```
<DjVuXML>
 <BODY>
  <OBJECT data="file.." ...>
  <PARAM name="PAGE" value=".."/>
  [...]
   <REGION>
    <PARAGRAPH>
     <LINE>
      <WORD coords="..."/>
      <WORD coords="..."/>
      [...]
     </LINE>
     <LINE> [...] </LINE>
     [...]
    </PARAGRAPH>
   </REGION>
  [...]
  </OBJECT>
  [...]
 </BODY>
</DjVuXML>
```

The original source was subsequently converted to an XML syntax, referred to as OCRML, which also contains additional structure mark-ups. Example snippets from the converted XML structure are given below:

```
<page id="54" pageNumber="23">
  <section id="123" label="SEC_BODY">
    hundreds of sailing boats are beached [...]
  </section>
</page>

<page id="224" pageNumber="123">
  <section id="54321" label="SEC_INDEX">
    sailing 12, 23, 45
  </section>
</page>
```

The first segment shows terms in the body of the book's content. This is identified by the label `SEC_BODY`. The second segment shows terms in the BoBI, which is labelled by `SEC_INDEX`. Each element has an unique identifier, reflecting its location in the document tree. The actual page numbers (as printed inside the book) are given by the "pageNumber" attribute.

Based on the OCRML structure and following the conventions introduced in section 3, we define the following schema for our indexes. The `fti_raw` table stores our (non-aggregated) full-text index, which is populated with terms from the body of the book (i.e., content labelled with `SEC_BODY`), while the table

[4]In history books, for example, expanded phrases, such as those of historical events, may be used in the BoBI instead of the actual terms appearing in the text.

[5]http://inex.is.informatik.uni-duisburg.de/2007/bookSearch.html
[6]http://djvu.sourceforge.net/doc/man/djvuxml.html

fti_raw	
Term	PageID
sailing	/book1/page23
sailing	/book1/page23
sailing	/book1/page29
···	···
weather	/book1/page19
weather	/book1/page20
···	···

(a) terms in book pages

bobi	
Term	PN
sailing	13
sailing	20
weather	3
weather	6
weather	100
boats	66

(b) terms in a BoBI

page_map	
PageID	PN
/book1/page23	7
/book1/page29	13
/book1/page31	15
···	···

(c) mapping from page IDs to page numbers

bobiPageFreq		
Probability $P(t\|p)$	Term t	PN p
1/2	sailing	13
1/2	sailing	20
1/3	weather	3
1/3	weather	6
1/3	weather	100
1	boats	66

(d) estimate the page frequencies of terms in the BoBI

fti_all	
$tf_{\text{fti}}(t)$	Term t
8	sailing
15	weather
6	boats

(e) aggregate the total terms weights in the FTI

fti (full-text index)			
Tuple weight $tf_{\text{fti}}(t,p)$	Term t	PageID p	
2	sailing	/book1/page23	(7)
1	sailing	/book1/page29	(13)
2	sailing	/book1/page31	(15)
3	sailing	/book1/page36	(20)
1	weather	/book1/page19	(3)
5	weather	/book1/page20	(4)
1	weather	/book1/page22	(6)
3	weather	/book1/page24	(8)
2	weather	/book1/page26	(10)
3	weather	/book1/page116	(100)
1	boats	/book1/page82	(66)
5	boats	/book1/page83	(67)

(f) tuples with original term weights

nbfti (naive boosted fti)		
Tuple weight $tf_{\text{nbfti}}(t,p)$	Term t	PageID p
$2+0=2$	sailing	··· (7)
$1+1=2$	sailing	··· (13)
$2+0=2$	sailing	··· (15)
$3+1=4$	sailing	··· (20)
$1+1=2$	weather	··· (3)
$5+0=5$	weather	··· (4)
$1+1=2$	weather	··· (6)
$3+0=3$	weather	··· (8)
$2+0=2$	weather	··· (10)
$3+1=4$	weather	··· (100)
$1+1=2$	boats	··· (66)
$5+0=5$	boats	··· (67)

(g) propagates term weights to the pages in the BoBI with naive model

vbfti (voter boosted fti)		
Tuple weight $tf_{\text{vbfti}}(t,p)$	Term t	PageID p
$2+0=2$	sailing	··· (7)
$1+8/2=5$	sailing	··· (13)
$2+0=2$	sailing	··· (15)
$3+8/2=7$	sailing	··· (20)
$1+15/3=6$	weather	··· (3)
$5+0=5$	weather	··· (4)
$1+15/3=6$	weather	··· (6)
$3+0=3$	weather	··· (8)
$2+0=2$	weather	··· (10)
$3+15/3=8$	weather	··· (100)
$1+6/1=7$	boats	··· (66)
$5+0=5$	boats	··· (67)

(h) propagates term weights to the pages in the BoBI with voter model

Figure 9: *Tf*-boosting for book search with a naive model and a voter model

called `bobi` is our back-of-book index, built from content labelled with `SEC_INDEX`. A `page_map` table is used for mapping the 'printed' page numbers of a book to the page IDs.

Page mapping is a practical requirement of processing books in their digitized form, while applications that need to handle such complexity will benefit from the expressiveness and flexibility provided by PD.

The three tables for indexing books are shown in Figures 9(a), 9(b), and 9(c).

With respect to the notation used in Figure 9 in general: The tuple weight corresponds to the number of tuples in which a term-page pair occurs. For example, $tf_{\text{fti}}(t,p) := n_{L,\text{fti\_raw}}(t,p)$, where (t,p) is a tuple, and $n_{L,\text{fti\_raw}}(t,p)$ is the number of tuples in table `fti_raw` (Figures 9(a)) in which (t,p) occurs.

In the next sections, we discuss the modelling of anchor-text *tf*-boosting. First, a naive model using the classic Web IR strategy is defined; and then a tailored voter model for book search is introduced.

4.2 Naive Model

A naive model performs direct term augmentation thus achieving *tf*-boosting to the pages in the BoBI. The idea comes from the Web IR anchor text *tf*-boosting, where terms in anchor texts contribute term frequencies to destinations. In a naive model, the terms in the BoBI are viewed as anchor texts, while their associating pages are viewed as destinations. For instance, in the example OCRML segment labelled by `SEC_INDEX` in section 4.1, the term "sailing" has three referenced pages "12", "23", and "45". Thereby, there are three virtual anchors, and "sailing" is an anchor-text being boosted in the destination pages.

A mathematical definition of the naive model is given by the following formula:

DEFINITION 1. *Boosted* tf*: naive model:*

Let $\text{tf}_{fti}(t,p) := n_L(t,p)$ *be the bare within-page term frequency of term t in page p, i.e. the number of locations in page p at which term t occurs.*

Let bobi be the set of anchors, and let $\text{tf}_{bobi}(t,a_p) := n_L(t,a_p)$ *be the within-anchor term frequency, where anchor* a_p *points to page p. Then, the naive boosted within-page term frequency is the sum of the bare within-page term frequency and the term frequencies of the anchors pointing to the page.*

$$\text{tf}_{nbfti}(t,p) := \text{tf}_{fti}(t,p) + \sum_{a_p} \text{tf}_{bobi}(t,a_p) \qquad (1)$$

In probabilistic Datalog, the following rules model the naive boosting strategy [7]:

```
%1. term frequency in fti
fti SUM(T, X) :- fti_raw(T, X).

%2-3. maps page numbers and page IDs
```

[7]Lines start with '%' are for comment purpose.

```
fti_dist DISTINCT(T, X) :- fti_raw(T, X).
fti_map(T, X, P) :- fti_dist(T, X) & page_map(X, P).

%4-5. augmentation / tf-boosting
nbfti(T, X) :- fti(T, X).
nbfti(T, X) :- bobi(T, P) & fti_map(T, X, P).

%6. estimates probability scores
w_augTerm SUM(T, X) :- nbfti(T, X) | (X).
```

The arguments T, X and P correspond to "Term", "PageID" and "Page Number" (PN) in the schema, respectively. The first rule adds up the term frequencies to obtain the book-wide term frequency. Strictly speaking, this is not a probabilistic operation, since the tuple weights in `fti` (see Figure 9(f)) are total counts rather than probabilities [8]. The `fti` has two columns, i.e. "Term" and "PageID", while for better readability, we also show the corresponding "PN" values (between parentheses) after the page ID, and we will refer to "PN" in our latter discussions. The second and third rules map the page numbers to corresponding page IDs. The fourth and fifth rules perform naive *tf*-boosting. The augmented result is shown in the Figure 9(g). The last rule estimates the probability scores.

The issue here is that the naive modelling of *tf*-boosting does not deliver a reasonable boosting effect; this is because there is at most *ONE* anchor per term. Therefore, the next section describes the idea to relate the indexed (in the BoBI) and non-indexed occurrences of a term, and to propagate the non-indexed occurrences to the indexed occurrences, thus, leading to a boost of the term frequencies in the indexed pages.

4.3 Voter Model

Based on the discussion in section 3.2, a voter is a page occurring in the FTI. Voters vote for the destination pages that occur in the BoBI, and that share the same keywords. In other words, each voter holds anchors to destination pages (there could be more than one destinations, depending on the number of pages in BoBI). The local term frequency of a voter corresponds to the votes it can potentially assign to destinations (candidates). Each destination obtains a portion of the total amount of votes. There are various ways to allocate votes to destinations, and we illustrate here the unbiased voting where the votes are evenly distributed over destinations. Refined methods for vote allocation are to be studied in future work. For example, the vote distribution could consider page properties such as length, location in book, exhaustiveness and/or specificity of the page.

The mathematical definition of the unbiased voter model is given as:

DEFINITION 2. *Boosted* tf*: voter model:*

Let $\mathrm{tf}_{fti}(t) := n_{L,fti\_raw}(t)$ *be the book-wide term (location) frequency. This frequency is equal to the sum over the within-page term frequencies:* $n_{L,fti\_raw}(t) = \sum_p n_{L,fti\_raw}(t,p)$*. In a less formal way, the total (book-wide) term frequency is*

$$\mathrm{tf}_{fti}(t) = \sum_p \mathrm{tf}_{fti}(t,p)$$

where $\mathrm{tf}_{fti}(t,p) := n_{L,fti\_raw}(t,p)$ *is the within-page term frequency.*

The book-wide term frequency of a term t *is distributed over the BoBI entries. Let* $n_{P,bobi}(t)$ *denote the number of pages the respective term entry points to in the BoBI. Note that* $n_{P,bobi}(t) = n_{L,bobi}(t)$*, i.e. the number of pages is equal to the number of locations/tuples, if the BoBI is distinct, i.e. there are no multiple term-page entries of the same term to the same page.*

Then, the boosted term frequency is defined as follows:

$$\mathrm{tf}_{vbfti}(t,p) := \mathrm{tf}_{fti}(t,p) + \frac{\mathrm{tf}_{fti}(t)}{\mathrm{tf}_{bobi}(t)} \tag{2}$$

*The respective term frequencies (*tf*) correspond to total counts of term occurrences in the raw FTI, as the next equation illustrates.*

$$\mathrm{tf}_{fti}(t,p) = n_{L,fti\_raw}(t,p) + \frac{n_{L,fti\_raw}(t)}{n_{L,bobi}(t)} \tag{3}$$

The probabilistic Datalog rules for modelling the voter model are as follows:

```
%1. term's page frequency in bobi
bobiPageFreq(T, P) :- bobi(T, P) | (T).

%2. term frequency in fti
fti SUM(T, X) :- fti_raw(T, X).

%3-4. maps page numbers and page IDs
fti_dist DISTINCT(T, X) :- fti_raw(T, X).
fti_map(T, X, P) :- fti_dist(T, X) & page_map(X, P).

%5-6. obtains statistics for augmentation
bobiIDFreq(T, X) :-
    bobiPageFreq(T, P) & fti_map(T, X, P).
fti_all SUM(T) :- fti(T, X).

%7-8. augmentation / tf-boosting
vbfti(T, X) :- fti(T, X).
vbfti(T, X) :- bobiIDFreq(T, X) & fti_all(T).

%9. estimates probability scores
w_augTerm(T, X) :- vbfti(T, X) | (X).
```

The first rule yields the term's page frequency in the `bobi` (see Figure 9(d)). The tuples in the goal "bobiPageFreq" are conditional probabilities of the form $P(T, P|T)$, i.e. the tuple probabilities are conditioned by the term. The "| (T)" attached to the sub-goal is the evidence key, and this describes the conditional probability. The second, third and fourth rules are similar to the naive model. The fifth rule transfers the page frequency to page IDs, and the sixth rule aggregates the total book-wide *tf* (see Figure 9(e)); these two rules prepare the statistics for the latter *tf*-boosting. The seventh and eighth rules conducts the proposed boosting, the result of `vbfti` is shown in Figure 9(h). Finally, the last rule estimates the probability scores.

Considering the boosting effect of the voter model, we take a closer look at the boosted FTI, the `vbfti` in Figure 9(h). For example, the two occurrences of the term "sailing" which appear in the BoBI (i.e., on page 13 and page 20) receive an equal share of the total *tf* of "sailing" in the FTI, boosting the *tf* by $8/2$ to a total score of 5 in the case of page 13, and 7 for page 20.

[8]The description of a full probabilistic model is part of future research. For the time being, we propose the frequency-based model, since the probabilistic model implies an early aggregation of frequency counts, and this does not deliver the *tf*-boosting effect as applied in hypertext retrieval.

A beneficial effect of the voting-based *tf*-boosting strategy is that since it promotes the pages referenced in the BoBI, it correctly leads to a more user-oriented retrieval paradigm. For example, assuming that the BoBI lists only the first occurrence of a term (at the start of a topic), the system will rank the page referenced in the BoBI ahead of other pages containing the term, and alas directing the user to the page where they should start reading. For instance, page 3 is the first page about "weather", while the adjacent page 4 continues the discussion. Without the voter model or with the naive model, page 4 gets ranked higher, but with the voter model score of page 3 is boosted above page 4's score. Based on this assumption, a best entry point strategy which aims to direct user to those points in the text where they should start reading [3] can be feasibly implemented. In effect, using the voter model, the best entry point selection method of the author, as implemented within the index term selection strategy they applied can be directly reflected in the retrieval and its benefits passed directly onto the user.

5. SUMMARY

This paper proposes a *tf*-boosting model for book/page search. The initial starting point was to apply anchor-text retrieval to the back-of-book index, this idea being based on the fact that a BoBI is a list of anchors pointing to pages. However, the starting point turned out to be "naive" in the sense that the propagation of anchor-text from the BoBI does not deliver the desired *tf*-boosting effect, and, in turn, this observation motivated this paper.

For achieving a *tf*-boosting effect from the BoBI, we needed to revise the hypertext-based modelling. This revision leads to a voter model proposed for *tf*-boosting in book search: the overall idea is to distribute the book-wide term frequency of a term to the pages that are indexed in the BoBI; this corresponds to a voting model in the sense that a term votes for the pages to which its term frequency shall be added.

One of the main contributions of this paper is that the modelling of the voter model demonstrates the benefit of high-level abstract modelling of retrieval strategies. Traditional anchor-text retrieval is modelled in probabilistic Datalog rules; these rules are not directly applicable for book search, and need to be evolved. This evolution demonstrates that through the high-level abstraction of search strategies, successful techniques of one domain (here, anchor-text retrieval in hypertext) can be transferred to another domain (tf-boosting for book/page search).

Future work will include the investigation of different voter models. For example, deciding how many votes an entry page should receive based on proximity (term to page, or page to page). In addition, the evaluation of retrieval effectiveness will be examined with real and large scale collections.

Acknowledgements

We would like to thank Bodin Dresevic and Dejan Lukacevic at the Microsoft Development Centre in Serbia for the use of their tool to mark-up books in OCRML.

6. REFERENCES

[1] N. Abdullah and F. Gibb. Using a task-based approach in evaluating the usability of bobis in an e-book environment. In *ECIR*, pages 246–257, 2008.

[2] N. Craswell, D. Hawking, and S. E. Robertson. Effective site finding using link anchor information. In *SIGIR*, pages 250–257, 2001.

[3] K. Finesilver and J. Reid. User behaviour in the context of structured documents. In *ECIR*, pages 104–119, 2003.

[4] N. Fuhr. Probabilistic Datalog: Implementing logical information retrieval for advanced applications. *Journal of the American Society for Information Science*, 51(2):95–110, 2000.

[5] N. Fuhr and K. Großjohann. XIRQL: An XML query language based on information retrieval concepts. *ACM Trans. Inf. Syst.*, 22(2):313–356, 2004.

[6] N. Fuhr, J. Kamps, M. Lalmas, and A. Trotman, editors. *Focused access to XML documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX), Revised Selected Papers*, Lecture Notes in Computer Science. Springer, 2008.

[7] N. Fuhr and T. Roelleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Transactions on Information Systems (TOIS)*, 14(1):32–66, 1997.

[8] D. J. Harper, I. Koychev, and S. Yixing. Query-based document skimming: A user-centred evaluation of relevance profiling. In *ECIR*, pages 377–392, 2003.

[9] D. Hawking. Challenges in enterprise search. In *ADC*, pages 15–24, 2004.

[10] D. Hawking, E. M. Voorhees, N. Craswell, and P. Bailey. Overview of the trec-8 web track. In *TREC*, 1999.

[11] G. Kazai and A. Doucet. Overview of the INEX 2007 book search track (BookSearch'07). In Fuhr et al. [6].

[12] G. Kazai and A. Trotman. Users' perspectives on the usefulness of structure for XML information retrieval. In *ICTIR*, 2007.

[13] M. Lalmas and T. Roelleke. Four-valued knowledge augmentation for structured document retrieval. In *Proceedings of the 13th International Symposium on Methodologies for Intelligent Systems (ISMIS), Lyon, France*, June 2002.

[14] R. R. Larson. Logistic regression and EVIs for XML books and the heterogeneous track. In Fuhr et al. [6].

[15] W. Magdy and K. Darwish. CMIC at INEX 2007: Book search track. In Fuhr et al. [6].

[16] T. Roelleke. *POOL: Probabilistic Object-Oriented Logical Representation and Retrieval of Complex Objects*. Shaker Verlag, Aachen, 1999. Doktorarbeit (PhD thesis).

[17] T. Roelleke, M. Lalmas, G. Kazai, I. Ruthven, and S. Quicker. The accessibility dimension for structured document retrieval. In *Proceedings of the BCS-IRSG European Conference on Information Retrieval (ECIR), Glasgow*, March 2002.

[18] T. Roelleke, H. Wu, J. Wang, and H. Azzam. Modelling retrieval models in a probabilistic relational algebra with a new operator: the relational Bayes. *VLDB J.*, 17(1):5–37, 2008.

[19] F. Weigel, K. U. Schulz, and H. Meuss. Exploiting native XML indexing techniques for XML retrieval in relational database systems. In *WIDM*, pages 23–30, 2005.

[20] H. Wu, G. Kazai, and M. Taylor. Book search experiments: Investigating IR methods for the indexing and retrieval of books. In *ECIR*, pages 234–245, 2008.

Link Detection in XML Documents: What about repeated links?

Junte Zhang
University of Amsterdam
Amsterdam, the Netherlands
j.zhang@uva.nl

Jaap Kamps
University of Amsterdam
Amsterdam, the Netherlands
kamps@uva.nl

ABSTRACT

Link detection is a special case of focused retrieval where potential links between documents have to be detected automatically. The use case, as studied at INEX's Link the Wiki track, is that of a new, orphaned page (here, a structured XML document) for which we need to detect relevant incoming and outgoing links to other pages (here, the INEX Wikipedia collection). We focus on outgoing links and investigate link density, and especially repeated occurrences of links with the same anchor text and destination.

We provide an extensive analysis of link density and repetition, and look at parameters like the document's length, the distance between anchor text occurrences, and the frequency of the anchor text within an article. We also conduct experiments trying to determine what should be done with links that are repeated. We describe alternative approaches and compare them against two baselines: the first baseline is to link only once, and the second is to link all candidates. The performance is measured with precision and recall in terms of the total set of discovered links. Our main finding is that, although the overall impact of link repetition is modest, performance can increase by taking a informed approach to link repetition.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; H.3.4 Systems and Software; H.3.7 Digital Libraries

General Terms

Measurement, Experimentation

Keywords

Wikipedia, Link Detection, Repeated Links, Evaluation

SIGIR 2008 Workshop on Focused Retrieval
July 24, 2008, Singapore

1. INTRODUCTION

Information Retrieval methods have been employed to automatically construct hypertext on the Web [1, 2, 6], as well for specifically discovering missing links in Wikipedia [4, 3]. These missing links are added manually by users, as well as automatically with scripts. We focus on automatic link-detection. The purpose of detecting missing links is to make navigation within and between pages easier.

To automatically detect whether two nodes, such as two XML files, are implicitly connected, it is necessary to search for some text segments that both nodes share, either explicitly or semantically. Often it is only one specific and extract string [1]. With whole documents, hyperlinks can be generated on the file level. With semi-structured documents, such as HTML or more structured documents in XML, one can *deeplink* by generating hyperlinks on the element level using the structure of the document.

In Wikipedia [14], excessive links make a Wikipedia article difficult to read. Good links in Wikipedia are relevant to the context. A whole document gets more context by adding more links, as extra information is added. However, there is the problem of *link density* in structured XML documents, such as the INEX Wikipedia collection consisting of 660,000 English Wikipedia articles in XML. On the one hand, we may decide to link only once per article to a given destination page. On the other hand, we may decide to link every time that the opportunity presents itself. The issue of link repetition is directly related to link density.

A rule of thumb used at Wikipedia is to "aim for a consistent link density" and "not to link eight words in one sentence and then none in the rest of the article."[1] To further quote the style guideline of Wikipedia:

> For general interest articles, where the links are of the "see also" or "for more information" type, it may be better to not link in the summary, deferring the link until the term is defined later in the article. Numerous links in the summary of an article may cause users to jump elsewhere rather than read the whole summary. For technical articles, where terms in the summary may be uncommon or unusual, and linking is necessary to facilitate understanding, it is permissible and may even be necessary to have a high link density in the introduction.

[1]Wikipedia:Only make links that are relevant to the context, `http://en.wikipedia.org/wiki/Wikipedia:Only_make_links_that_are_relevant_to_the_context`

Specifically, Wikipedia's manual style of links offers hints about repeated links.[2] The issue of overlinking is addressed in this quote:

> A link for any single term is excessively repeated in the same article, as in the example of overlinking that follows: "Excessive" is more than once for the same term, in a line or a paragraph, because in this case one or more duplicate links will almost certainly appear needlessly on the viewer's screen.

However, the inverse could also be true when one anchor term is not linked enough. Anchor terms that are more important should be linked more often. The same Wikipedia manual reads:

> Good places for link duplication are often the first time the term occurs in each article subsection. Thus, if an important technical term appears many times in a long article, but is only linked once at the very beginning of the article, it may actually be underlinked. Indeed, readers who jump directly to a subsection of interest must still be able to find a link.

It has already been pointed out in [7] that the amount of hypertext matters: if you give someone 'too much' hypertext, they will become lost; if you give them 'too little', they will not even be able to get started. The former is also called *overlinking*, while the latter is called *underlinking*. Both cases are seen as poor link structure. Furthermore, automatic or semi-automatic constructed hypertexts with information retrieval techniques can be difficult to use, causing user disorientation and cognitive overload [1].

These guidelines and manual are used for adding manual links in Wikipedia. This leads to our research questions:

- Does link repetition occur, and how often?
- How can we predict when to link in a XML document?
- Will link detection in XML documents improve by taking into account repetitions of links?

The issue link repetition and link density in automatic link detection, especially in XML documents like the INEX Wikipedia collection, is still a conundrum, which we try to address in this paper. The remainder of this paper is structured as follow: we start by embedding our work with related literature in Section 2, then we discuss our experimental setup in Section 3, the results are evaluated and presented in Section 4, and finally we conclude with our conclusion of our research question in Section 5.

2. RELATED WORK

2.1 World Wide Web

On the World Wide Web, automatic hyperlink tools are already available. In [1] an overview of different approaches is given of information retrieval techniques for the automatic construction of hypertext. The idea of global and local similarity is outlined, where the former is related to the whole document, and the latter to text segments in a document. There is a strict correlation between both, and the local similarity is more orientated towards precision to refine results later. Another distinction that was made was between first-order hypertexts which are added by the document author, and second-order hypertexts which are automatically added.

Entirely automatic methods for building second-order hypertext using Information Retrieval (IR) and inspired by relationship visualization techniques and graph simplification is presented in [2]. It is pointed out that document linking is based upon information retrieval similarity measures with adjustable levels of strictness. Using no significant natural language processing, and standard IR indexing techniques, inter-document links can be located and described. The idea of linking using structure is also addressed, like creating a link between parts of a document that are most similar.

A survey of the actual use of hyperlink analysis in web search engine ranking like Google's PageRank and other applications is given in [6]. Such other applications are crawling for high quality pages, search-by-example, computing the reputation of websites, finding "web communities", websites related to same or related topics, and web page categorization. The importance of link evidence for improving the ranking in ad-hoc retrieval using the INEX Wikipedia collection is shown in [11]. These research directions are related to the research in this paper, although our desired result is to improve the quality of automatic link detection for user navigation and serendipitous information seeking in the Wikipedia.

2.2 Link-detection in Wikipedia

For Wikipedia, automatic link construction tools are also available such as the link suggesting tool developed by [9]. In [4] a 2-step approach is presented to automatically discover missing hyperlinks in Wikipedia using the link structure. They compute a cluster a highly similar articles around a given article, and then they identify candidate links from those similar articles that might be missing on that given article. The clusters are computed by using co-citation, i.e. two articles are similar if they are co-cited by a third one.

Since 2007 is there a specific link-detection task at INEX called Link-the-Wiki (LTW). This task basically consists of two sub-tasks: detecting incoming links (from destination to source) and outgoing links (from source to destination).

An approach based on the content of an article is presented in [3], where the whole article is used as a query against the index using the Vector Space Model (VSM). The influence of setting different thresholds for the pool of related articles, and the effect of title matching was checked by measuring the performance using standard IR measures as Mean Average Precision (MAP).

In [5] the incoming links were detected by running a NEXI query [13] with the name nodes (titles) of the topics, or *//article[about(.,name)]*. For detecting outgoing links, all the titles in the collection were stored in an in-memory hashtable and looked up, where the window size varied from 8 words down to 1 word, and included stop words.

A different LTW approach is presented in [8], where the authors detect links by storing an anchor text a if it has a certain ratio and looking it up again during the link detection process. That ratio γ is the ratio between the number of articles that has a link from anchor a to a file d, and the number of files in which the anchor text a occurs only once.

[2]Wikipedia:Manual of Style (links), `http://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style_(links)`

Using this ratio, highly relevant anchor texts can simply be looked up.

The approach adopted in [10] identified terms within the document that were over represented and from the top few generated queries of different lengths. Potentially relevant documents were identified by retrieving and ranking them in BM25/Okapi, where these terms were used as query.

Since the LTW track in 2007 only evaluated unique article-to-article links, the repetitions of links has not been taken into account by this previous and related research. So there was only focus on *what* to link, but not *when* and *how often*. We investigate mainly the latter two directions in this paper. The main challenge of automatic link detection is the actual detection of anchor terms, i.e. substrings that should be made clickable. Once these anchors have been found, IR technologies take over with finding and ranking the most plausible destinations.

3. EXPERIMENTAL SETUP

We start by analyzing the 90 existing topics (*qrels*) that were used at the INEX LTW track by looking at the types of links, repetition of links, and the article length. We continue by defining parameters that could possibly have impact on the detection of repeated links. Finally, we present our link detection approach, and the baselines that we used for our experiments.

3.1 Topics Analysis

3.1.1 Types of links

There are several types of links in the topics. These links have been implemented in the Wikipedia collection using XLink. An overview of the occurrence of these types of links in the un-orphaned (original) topics is presented in Table 1. The top 8 most frequent anchor terms on both the collection and file level are presented in Table 2 and 3.

	All topics		**Link in article**	
Type	**Uniq**	**Total**	**1×**	**Max**
`<collectionlink>`	5,786	8,868	5,781	15
`<unknownlink>`	1,308	1,458	1,271	7
`<outsidelink>`	807	851	778	5
`<imagelink>`	197	212	197	15
`<languagelink>`	79	1,147	1,147	1
`<wikipedialink>`	59	60	58	1
`<weblink>`	27	28	26	2
Total	8,263	12,624	9,232	-

Table 1: Statistics of the types of Links in the 90 un-orphaned LTW articles on the file level

For example, if we regard all the links as one distribution, then the `<languagelink>` has 79 different types (appearing once), but the same types are used 1,147 times, of which the single link `<languagelink lang="de">` is used as often as 66 times, which means the same language links are reused in the articles. When we look at each file separately, then a language link appears only once in a file.

In the INEX Wikipedia collection, there three type of links which are used for link detection at the LTW task: `<collectionlink>`, `<wikipedialink>`, and `<unknownlink>`. The `<collectionlink>` comprises of the bulk of the links in the orphaned articles (70.0%). When looking on a global level at all orphaned articles, then there are 5,786 unique types of collection links, out of the total of 8,868. The number of collection links that only occurs once is 4,275, which is 73.9% of the different types of collection links, and 48.2% out of all collection links.

Freq.	**Anchor term**	**Target file**
51	"2004"	35524.xml
48	"United States"	31882.xml
40	"2005"	35984.xml
21	"France"	10581.xml
21	"United Kingdom"	31717.xml
18	"2003"	36163.xml
17	"2001"	34551.xml
16	"Japan"	15573.xml

Table 2: Frequency of top 8 anchor terms on collection level.

Freq.	**Anchor term**	**In topic**	**Target file**
15	"Florida"	150340.xml	10829.xml
12	"Miami Beach"	150340.xml	109449.xml
12	"2004"	1092923.xml	35524.xml
10	"California"	150340.xml	5407.xml
9	"2005"	1092923.xml	35984.xml
9	"USA"	150340.xml	31882.xml
9	"2004"	2542756.xml	35524.xml
8	"Long Beach"	150340.xml	94240.xml

Table 3: Frequency of top 8 anchor terms on file level.

3.1.2 Link repetition

However, not every type of collection link appears once. The collection link to article 35524.xml is occurring most often on the collection level: 51 times, but it surprisingly does not to exist in the 2007 INEX collection that we used. We observe that many links that re-occur are named entities of years and geographical names like that of countries. Table 3 shows that over 3,000 of in total 8,868 links are links that are repeated. This is a substantial amount. On average, there are 98.5 *outgoing* collection links per topic, of which 64.3 per topic are unique, thus occurring once.

Many of the same types of links on a global level are reused in the files, such as links referring to years and dates which are almost always linked. Supporting evidence is given in Table 2. Moreover, 5,781 of the 8,868 collection links appear only once (65.2%) when looking on the file level, see Table 1, an outlier is the collection link 10829.xml (*"Florida"*), which is occurring 15 times in the topic 150340.xml (*"Miss Universe"*). The reason is that the topic *"Miss Universe"* has a very high link density, and the anchor term *"Florida"* occurs in total 15 times in the file, and thus is linked in all instances. A distribution plot is depicted for all link (re-)occurences in Figure 1. Most of the links occur only once, however, a substantial subset re-occurs.

3.1.3 Links in relation to article length

We observed that the link density in Wikipedia articles is mostly consistent and dependent on the length of an article. The length of an article is calculated by discarding all the XML structure, so we only obtain the cumulative length of all the text nodes in a file.

We found that there is a significant strong positive rela-

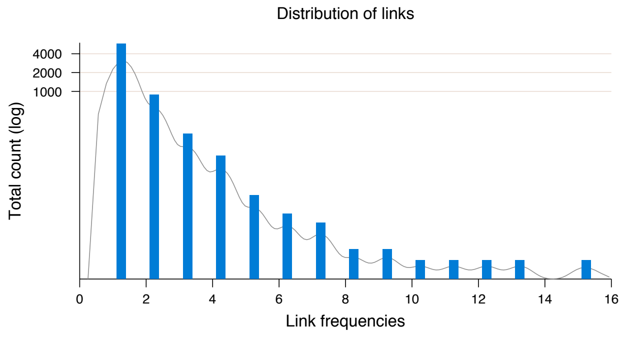


Figure 1: Distribution of all link frequencies

tionship between the length of a Wikipedia article (excluding structure) and the number of links appearing in that article (Pearson correlation coefficient $r = 0.78$ at $p < 0.01$, or Spearman's rho $= 0.85$ at $p < 0.01$), i.e. longer articles have more links than shorter articles, see Figure 2. Moreover, the average length of an anchor text is 12.3 characters, only 62 (0.7%) collection links are 3 characters or shorter.

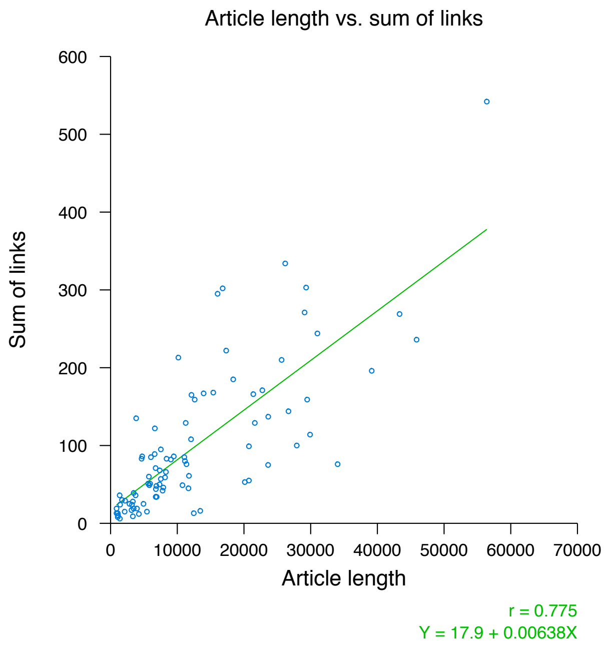


Figure 2: Strong positive correlation between article length and number of links.

The link density can be measured simply using a ratio,

$$\text{Link density ratio} = \frac{\text{total links}}{\text{article length}} \tag{1}$$

A distribution of the occurrences of the different types of links is presented in Figure 1 and an overview of the length distribution of the articles is given in Figure 3. We see in Figure 4 that the majority of the topics have a similar link density.

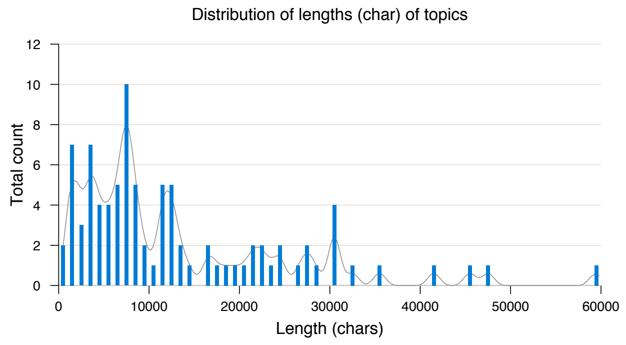


Figure 3: Distribution of article length of 90 topics

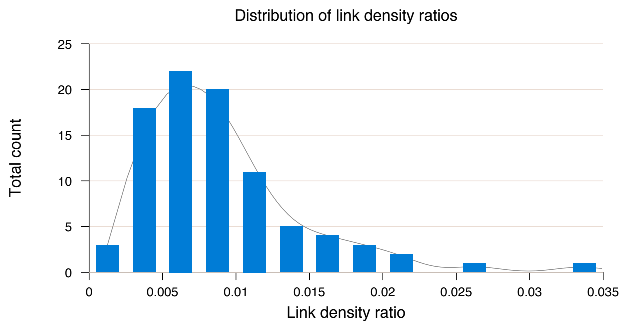


Figure 4: Distribution of link density ratios of 90 topics

3.1.4 Variables

In our experiments we check what the effect is of using 2 *dependent variables*, namely (1) "anchor distance" and (2) the number of "repeated candidate links", on the link detection performance. This performance is measured by comparing them against the ground truth of real links, which makes the number of real links our *independent variable*. Our definitions of the 2 dependent variables:

Anchor distance (AD) The distance between two anchor texts that refer to the same destination node, or A and A', can be calculated. We define it as the function, which we call the "Anchor Distance" AD, which is calculated as

$$AD(A, A') = rindex(A') - rindex(A), \tag{2}$$

where *rindex* is a function that determines the index of the last occurrence of a letter or the substring in the whole file. A substring of a string $T = t_1 \ldots t_n$ is a string $\hat{T}$ that is a subset of T, or $\hat{T} = t_{i+1} \ldots t_{m+i}$ where $0 \leq i$ and $m + i \leq n$.

Repeated candidate links (RCL) A link that has been detected with our method is a link candidate, which does not necessarily have to be a real link. A repeated link candidate is a candidate link that occurs more than once in a topic.

The AD is directly related to the concept of link density, e.g. a greater AD means that the link density is less, while a smaller AD show that the link density is greater. If a link occurs only once in an article, then it means that the distance is 0. Table 1 shows that the majority of a collection link occurs only once in an article, and that in one article the same collection link appears up to 15 times. Table 4 gives descriptive statistics over the 2 dependent variables over all detected candidate links that are repeated in the orphaned topics, and additional information about the article length is given. We did not choose to make the article length a third dependent variable, because it does not relate to individual candidate links, but only topics as a whole. Moreover, article length is implicitly connected with anchor distance and repeated candidate links. One candidate link is linked in the beginning and end of a file and has an extreme anchor distance.

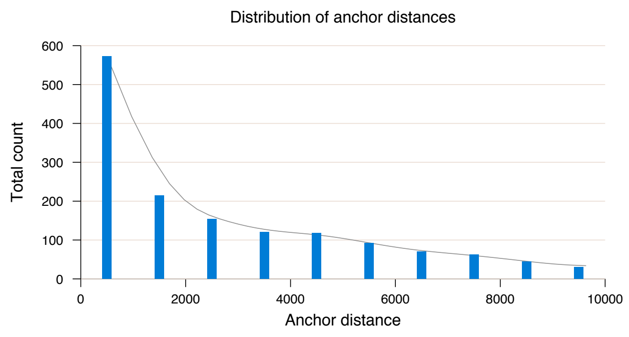


Figure 5: Distribution of anchor distances for an anchor a

	Anchor distance	*Article length*	*Link candidates*
Mean	9,382.97	13,454.58	10.76
Std.	11,418.94	12,191.25	29.71
Min	59	951	2
Med.	5,195	8,634	3
Max.	58,241	58,984	544

Table 4: Statistics of anchor distances (char), article lengths (char) and detected link candidates over the 90 orphaned topics.

3.2 Link Detection Method

3.2.1 Identification of related documents

We use the same method as outlined in [3], where we used the Vector Space Model (VSM) to retrieve related documents (articles) by using the whole article as a query to the index, where the index terms were stemmed using the Porter Stemmer, but no stopwords were removed. Our vector space model is the default similarity measure in Apache Lucene [12], i.e., for a collection D, document d, query q and query term t:

$$sim(q,d) = \sum_{t \in q} \frac{tf_{t,q} \cdot idf_t}{norm_q} \cdot \frac{tf_{t,d} \cdot idf_t}{norm_d} \cdot coord_{q,d} \cdot weight_t, \tag{3}$$

where

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

We also assume that articles that link to each other are somehow related textually. In [2] it is stated that the stronger the similarity, the better the quality of the relation is between two nodes. We adopt a *breadth m–depth n* technique for automatic text structuring for identifying candidate anchors and text node, i.e. a fixed n number of documents accepted in response to a query and a fixed m number of iterative searches. So the similarity on the document level and text segment level is used as evidence.

It is reported in [3] that when using the Vector Space Model, the best performance is achieved by retrieving the top 300 results. We use the same threshold in these experiments. This link detection experiment focuses only on outgoing links of the type collection `<collectionlink>`, which consists of the bulk of the links in the INEX Wikipedia collection. We do not allow a Wikipedia article to link to itself.

3.2.2 Identification of anchor texts

For our experiment, we only detect outgoing links by using the structure of the documents. An outgoing link is a link from an anchor text in the topic file to the *Best Entry Point* of existing related articles, which in our case was always the text-node of the */article[1]/name[1]* element. There is an outgoing link for topic t, when $S_{1 \cdots n} = T_{q \cdots r}$, where S is the title of a foster article, and T is a line in a orphan article. We assume that the first occurence of an anchor text is also a link. When there are multiple candidate anchors in a file, we learn and apply our link density parameters.

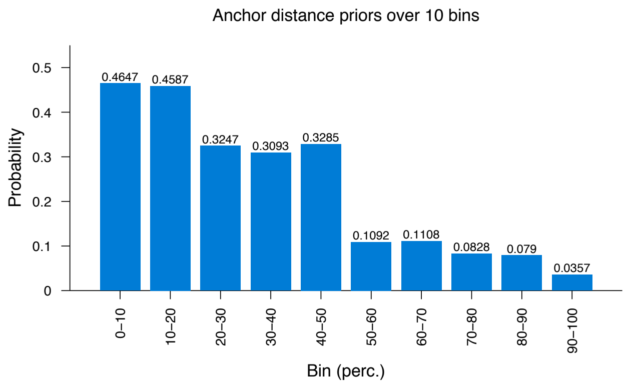
We extract for each topic the title enclosed with the `<name>` tag with a regular expression and store that in a hash-table for substring matching. We apply case-folding, and remove any existing disambiguation information put between brackets behind the title, e.g. *"What's Love Got to Do with It (film)"* becomes the substring *"What's Love Got to Do with It."* We only do exact string matching, and do no take into account linguistic features such as morphological variations between anchor terms or an other kind of normalization.

3.2.3 Priors

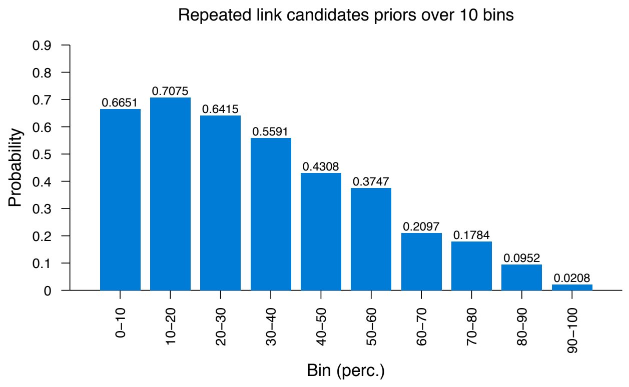
We compute and generate 2 prior plots, which are the anchor distance prior in Figure 6(a) and the repeated link candidates in Figure 6(b).[3] Each of these plots has 10 bins, where bin 1 consists of the bottom 10% of the anchor distances, bin 2 consists of the bottom 10-20 percent of the anchor distances, etc. The same is true for repeated link candidates, e.g. bin 10 in Figure 6(b) contains the top 10% of most frequent repeated link candidates.

We see that the probability that a link is repeated is higher when either the anchor distance is shorter, or the number of repeated candidate links is smaller. This is remarkable, and it may be due to an artifact of the topics, as we imagined that when the anchor distance becomes bigger, then it would become more probable that a link is repeated. We assumed

[3]We use here "prior probability" loosely. Since we are only interested in the shape of the distribution, we do not transform it into a probability distribution (which is in itself a simple mathematical exercise).



(a) Anchor distance priors over 10 bins



(b) Repeated link candidates priors over 10 bins

Figure 6: Priors partitioned in 10 bins.

the same for repeated candidate links. This requires more thorough analysis.

We improve our link detection approach as described in [3] by using the priors to make a Boolean choice: we either link a repeated candidate link, or not. We apply them on 'orphaned' topics, where all XML structure is removed, including any markup of the wikilinks. We do substring matching with the titles of the destination (target) articles to identify the anchor texts. We store these titles in an in-memory hash-table.

During the link detection procedure, we use the priors to make boolean choices on whether to link an anchor term that re-occurs in a topic given one of the 2 *dependent variables* as previously discussed, or P(repeated link | *dependent variable*).

3.3 Baselines and runs

We outlined in the introduction that there are 2 opposite approaches for dealing with link density: we can only link each anchor once, or we can always link a detected anchor. As we have pointed out before; both are not optimal. To compare our runs we use 2 baselines that are based on both polarities.

Baseline 1: "Link once" This baseline simulates the minimal link density. Each detected anchor is only made once a link.

Baseline 2: "Always linking" This baseline simulates the maximal link density. Each detected anchor is made a link.

Moreover, we have the following 4 runs which are in-between both baselines. These runs are based on the prior plots. Since we have 2 dependent variables, we have 2 variants for each run.

Run 1 We match the 2 bins with the highest priors.

Run 2 We match the 6 bins with the highest priors.

4. EVALUATION AND RESULTS

4.1 Evaluation

Our method is evaluated on 'cleaned' topics, where the collection link `<collectionlink>` markup has prior been removed. The original topics with markup are used as *qrels*. The official INEX Link-the-Wiki metrics only measure unique links between Wikipedia articles and do not take into account link density and the detection of repeated links. Using these official metrics, we reported at INEX in [3] a Mean Average Precision (MAP) value of 0.1825 and R-Prec value of 0.2233. It means that when the LTW task would also take into account link repetition and the issue of link density, we would have achieved higher scores.

Our evaluation is restricted to the number of links that are actually present in the un-orphaned pages, or A. Furthermore, our research is focused on investigating the link density in XML files, and not the accuracy of the actual detected links. Therefore to check this effect, we also only evaluate when we detected a link. Table 1 makes clear that most of the links in the topics appear once, so a minority of the links in the articles are actual repeated links.

We use the standard IR metrics for evaluating our methods. We use Precision P and Recall R. Precision is the number of detected true positive links tp divided by the sum of true positives and false positives fp, or all detected links D.

$$P = \frac{tp}{tp + fp} \tag{4}$$

where

$$fp(D, A) = \begin{cases} D - A & \text{if } D > A \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Recall is the number of true positive links divided by the sum of true positives and false negatives fn.

$$R = \frac{tp}{tp + fn} \tag{6}$$

where

$$fn(D, A) = \begin{cases} A - D & \text{if } D < A \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

This evaluation means that when we underlink candidate anchor terms, we hurt the recall, but when we overlink a candidate term, then the precision drops. Finally, we use the weighted harmonic mean of precision and recall, the balanced F-score F

$$F = \frac{2 \cdot (P \cdot R)}{P + R} \tag{8}$$

4.2 Results

The results are shown in Table 5. When we focus on the performance of the baseline runs, we see that the baseline runs perform relatively well. The main reason is that most of the links in the topics occur once, but obviously this goes at the expensive of the recall. Baseline 2 outperforms baseline 1 because a very high recall is achieved. This causes a slight drop in the precision as compared with the 1st baseline.

When we look at our runs, we see that some of them achieve higher precisions. A higher precision is more valued in our evaluation, because it means that the links are properly placed in terms of frequency and density. Run 1 (RCL) performs best overall, which indicates that the actual number of repeated candidate links is related to detect whether a link should re-occur. We also improve the link detection performance by taking into account the anchor distance in Run 2 (AD).

Run	Precision	Recall	F-Score
Baseline 1	0.8459	0.8043	0.8206
Baseline 2	0.7526	0.9967	0.8053
Run 1 (AD)	0.7635	0.8750	0.7790
Run 2 (AD)	0.8517	0.8126	0.8279
Run 1 (RCL)	0.8445	0.8286	0.8295
Run 2 (RCL)	0.8517	0.8126	0.8279

Table 5: Overal results for link detection.

In Table 6 we only present the top 8 detected links sorted by anchor distance with the 2nd baseline run. This table illustrates an interesting finding, which is the clear relation between link detection and the topicality of documents. All the detected candidate links are related to the topic "communism". We also see that we obviously overlink overwhelmingly. Two out of the 8 detected link candidates are actually false links when using the un-orphaned topics as 'ground truth'. However, when looking at the anchor terms, both of them seem very plausible links. It makes clear that to really determine whether a link is needed, user assessments are required.

AD	#Real	RCL	Topic	Anchor
58,241	2	15	15641.xml	russia
58,057	0	5	15641.xml	communist party
57,912	3	6	15641.xml	joseph stalin
57,143	2	3	15641.xml	leon trotsky
56,862	2	3	15641.xml	stalinism
56,212	2	9	15641.xml	moscow
55,840	0	5	15641.xml	cult of personality
51,018	2	34	15641.xml	soviet union

Table 6: Zooming on 8 results with longest anchor distance (AD) from baseline 2, where article length is 58,984.

4.3 Discussion

A limitation of our experimental setup is that we do not take into account the variable of "Intuitiveness." According to Wikipedia's guidelines, piped links should be kept as intuitive as possible. Piped links should not made "easter egg" links, which require the reader to follow them before understanding what's going on.

Our link detection method does not deal with violations of these guidelines as we do exact substring matching, and such violations are count as true positives in the automatic evaluation, and subsequently these hurt the performance of our method. Example 1 is used by Wikipedia as an instance of such a link.

(1) ... and by mid-century the puns and sexual humor were (with only a few [[**Thomas Bowdler** | **exceptions**]]) back in to stay.

The reference to *"Thomas Bowdler"* is not seen, unless the reader clicks on it or hovers over the link. If there are cases of such references, then the article should be explicitly linked by using a "see also" as in Example 2, or rephrased as in Example 3. In any case, the exact anchor terms should be made clear explicitly. Using Wikipedia's guidelines, noise in the data, such as variants of the same terms should not occur often, but dealing with such noise will certainly improve the accuracy of link detection.

(2) ... and by mid-century the puns and sexual humor were (with only a few exceptions; see [[**Thomas Bowdler**]]) back in to stay.

(3) ... and by mid-century the puns and sexual humor were back in to stay, [[**Thomas Bowdler**]] being an exception.

Another limitation of our study is that we did not properly deal with overlapping anchors, which should be avoided or parsed correctly. In the topic *"Educational progressivism"* (10005.xml) we identify 2 links in the same substring *"education reform"*, namely (1) *"education"* and (2) *"education reform"*. Example 4 shows these link candidate instances in simplified XML form. A solution may be to always select the longest substring.

(4) `<link>` `<link>` education `</link>` reform `</link>`

Finally, related to Example 4 is the issue of proper segmentation of the anchor terms. If we only match substrings that are separated with non-word boundaries, then we will not find anchor terms like *"Yahoo!"*. That is why we do plain substring matching, where the trade-off is generating too many candidate anchor terms (and thus links).

5. CONCLUSIONS AND FUTURE WORK

In this paper we described our work on predicting link density in XML documents for automatic link-detection. We raised 3 questions, and we address them here.

- Does link repetition occur, and how often?

In our analysis we showed that the same links do re-occur in the same documents. A substantial subset of the number of links are actual repeated links.

- How can we predict when to link in a XML document?

The main challenge in link detection is the detection of anchor terms. To find relevant anchor terms, we first cluster related documents using the Vector Space Model. Using the structure of XML documents, we extracted relevant substrings in the `<name>` nodes of Wikipedia articles. We assumed that a link should be created at the first instance.

To predict when a repeated link candidate should be actually made a link, we conducted a study with 2 variables. There variables are the distance between 2 of the same anchor terms, and the total number of possible link candidates in a file. We showed that both variables matter in dealing with repeated links.

- Will link detection in XML documents improve by taking into account repetitions of links?

We gave an outline of our approaches for automatic link-detection by taking into account some structure in the XML documents and context with link density analysis. We compared our runs with 2 baselines. The results are evaluated with precision, recall and their weighted harmonic means. Links are repeated in XML documents like Wikipedia articles, and there is also a user need for repeated links. Our preliminary experiments showed that when we take into account repeated links in the 'ground truth', we can achieve better link detection performance compared to the baselines of 'linking once' and 'link always'.

We presented our preliminary work on this subject. For future work, we would like to conduct more studies with different samples of documents, and test it more thoroughly. We also would like to apply and test our approaches with users on real-world problems with other XML datasets, such as linking archival finding aids, where we presented our conceptual framework in [15]. Moreover, we will further improve our method by making more use of the context of the anchors of the hyperlinks. Detecting variants of the same candidate anchor terms will also be investigated.

6. ACKNOWLEDGMENTS

Junte Zhang was supported by the Netherlands Organization for Scientific Research NWO under grant # 639.072.-601. Jaap Kamps was supported by the NWO grants # 612.-066.513, 639.072.601, and 640.001.501, and by the E.U.'s 6th FP for RTD project MultiMATCH contract IST-033104.

7. REFERENCES

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

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

[3] K. N. Fachry, J. Kamps, M. Koolen, and J. Zhang. Using and detecting links in wikipedia. In N. Fuhr, M. Lalmas, A. Trotman, and J. Kamps, editors, *Focused access to XML documents: 6th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2007)*, Lecture Notes in Computer Science. Springer, 2008.

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

[5] S. Geva. GPX@INEX2007: Ad-hoc Queries and Automated Link Discovery in the Wikipedia. In N. Fuhr, M. Lalmas, and A. Trotman, editors, *Pre-Proceedings of INEX 2007*, pages 403–409, 2007.

[6] M. Henzinger. Hyperlink analysis on the world wide web. In *HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*, pages 1–3, New York, NY, USA, 2005. ACM.

[7] K. Instone. Too much hypertext or too little? *SIGWEB Newsl.*, 3(3):22, 1994.

[8] K. Y. Itakura and C. L. A. Clarke. University of Waterloo at INEX2007: Ad Hoc and Link-the-Wiki Tracks. In N. Fuhr, M. Lalmas, and A. Trotman, editors, *Pre-Proceedings of INEX 2007*, pages 380–387, 2007.

[9] N. Jenkins. Can we link it, 2008. `http://en.wikipedia.org/wiki/User:Nickj/Can_We_Link_It`.

[10] D. Jenkinson and A. Trotman. Wikipedia Ad hoc Passage Retrieval and Wikipedia Document Linking. In N. Fuhr, M. Lalmas, and A. Trotman, editors, *Pre-Proceedings of INEX 2007*, pages 365–379, 2007.

[11] J. Kamps and M. Koolen. The importance of link evidence in Wikipedia. In C. Macdonald, I. Ounis, V. Plachouras, I. Rutven, and R. W. White, editors, *Advances in Information Retrieval: 30th European Conference on IR Research (ECIR 2008)*, volume 4956 of *Lecture Notes in Computer Science*, pages 270–282. Springer Verlag, Heidelberg, 2008.

[12] Lucene. The Lucene search engine, 2007. `http://lucene.apache.org/`.

[13] A. Trotman and B. Sigurbjörnsson. Narrowed Extended XPath I (NEXI). In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *INEX*, volume 3493 of *Lecture Notes in Computer Science*, pages 16–40. Springer, 2004.

[14] Wikipedia. The free encyclopedia, 2006. `http://en.wikipedia.org/`.

[15] J. Zhang, K. N. Fachry, and J. Kamps. Automatic link-detection in encoded archival descriptions. In L. L. Opas-Hänninen, M. Jokelainen, I. Juuso, and T. Seppänen, editors, *Digital Humanities 2008, Conference Abstracts*, pages 226–228, Oulu, Finland, 2008. The University of Oulu.

Author Index