ADCS 2011

Proceedings of the Sixteenth Australasian Document Computing Symposium

2 December 2011

Edited by
Sally Jo Cunningham, Falk Scholer, Paul Thomas

Proceedings of the Sixteenth Australasian Document Computing Symposium

Australian National University, Canberra, ACT
2 December 2011

Published by
School of Computer Science and IT, RMIT University
Melbourne VIC 3001, Australia

Editors

Sally Jo Cunningham
Falk Scholer
Paul Thomas

ISBN: 978-1-921426-92-6
http://www.cs.rmit.edu.au/adcs2011

Proceedings of the Sixteenth Australasian Document Computing Symposium

Australian National University, Canberra, ACT
2 December 2011

Chairs' preface

These proceedings contain the papers of the Sixteenth Australasian Document Computing Symposium hosted by the Australian National University.

A number of high quality submissions were received this year. Of 24 submitted papers, 10 were accepted for full presentation at the symposium (42%) and 5 were accepted as short papers (21%). The full written version of each submission received at least two anonymous reviews by independent, qualified experts in the area; several received three reviews. Dual submissions were explicitly prohibited.

The symposium provides an opportunity for document computing practitioners and researchers to get together and informally share ideas and enthusiasm. In 2011 we have again co-located ADCS with The Australasian Language Technology Workshop, sharing a joint paper session and social events.

We would like to thank the members of the program committee and the extra reviewers for their efforts.

We would also like to thank Google and NICTA for their generous support of the event.

Conference chair

Paul Thomas	CSIRO / ANU

Programme chairs

Sally Jo Cunningham	University of Waikato
Falk Scholer	RMIT University

Programme committee

Peter Bruza	Queensland University of Technology
J. Shane Culpepper	RMIT University
David Eyers	University of Otago
Shlomo Geva	Queensland University of Technology
Yun Sing Koh	Auckland University of Technology
Irena Koprinska	University of Sydney
Alexander Krumpholz	CSIRO/ANU
Alistair Moffat	University of Melbourne
Laurence Park	University of Western Sydney
Luiz Augusto Pizzato	University of Sydney
Gitesh Raikundalia	Victoria University
Tom Rowlands	CSIRO/ANU
Scott Sanner	NICTA
James A. Thom	RMIT University
Andrew Trotman	University of Otago
Andrew Turpin	University of Melbourne
William Webber	University of Melbourne
Mingfang Wu	RMIT University
Kalina Yacef	University of Sydney
Justin Zobel	University of Melbourne

Additional Reviewers

Ying-Hsang Liu	Charles Sturt University
Simon Puglisi	RMIT University
Mark Sanderson	RMIT University
Stephen Wan	CSIRO
Ross Wilkinson	Australian National Data Service

ADCS steering committee

Peter Bruza	Queensland University of Technology
Shlomo Geva	Queensland University of Technology
Alistair Moffat	University of Melbourne
Falk Scholer	RMIT University
James Thom	RMIT University
Paul Thomas	CSIRO
Andrew Trotman	University of Otago
Andrew Turpin	University of Melbourne

Information Retrieval in Large Organisations
(ADCS 2011 Invited Keynote)

Simon Kravis
Fujitsu Australia Limited
simon.kravis@au.fujitsu.com

Abstract:

Information retrieval within large organisations is usually from repositories with little or no hyperlinking of documents. Tools which are successful for public-facing web sites are frequently unsuccessful inside organisations through poor ranking of search results, high levels of duplication and near-duplication and the frequent occurrence of documents which are not completely described by the text which they contain. Metadata-based retrieval is limited by poor coverage from manually assigned metadata and the limited availability of automated metadata assignment.

The usefulness of web spam

Timothy Jones
Australian National University
Canberra, Australia
tim.jones@anu.edu.au

Paul Thomas
CSIRO
Canberra, Australia
paul.thomas@csiro.au

David Hawking
Funnelback Pty Ltd
Canberra, Australia
david.hawking@acm.org

Ramesh Sankaranarayana
Australian National University
Canberra, Australia
ramesh@cs.anu.edu.au

Abstract *Spam comprises at least 60% of the public web, and search engine companies invest considerable effort in rejecting these apparently useless pages. But how bad are spam pages in search results? Can spam be dealt with as a side-effect of dealing with page utility, or is the relationship more complex?*

Thirty-four volunteer judges rated selected individual documents first on usefulness to a specified task and then on degree of "spamminess". Our results show that the relationship between spamminess and utility is far from clear cut; judges found that an important proportion of spam documents were useful. We conclude that evaluation should consider both utility and spamminess, as separate factors; and that search engines should not summarily discard spam pages but should take their utility into account as well.

Keywords User Studies Involving Documents; Web Documents

1 Introduction

Between 60 and 80% of pages on the web may be spam, as of 2009 [4]. Detecting spam documents on the web has therefore received much recent attention [2, 3]. However, relatively little attention has been paid to the problem of spam nullification—understanding how to correctly deal with spam documents once they have been detected [2, 6]. A common assumption is that spam pages should either be removed from the index or filtered out at query time, and never shown to users. However, this may be counter-productive if and when a spam page turns out to be useful, and both search engines and search evaluations should take this into account.

In order to understand how to correctly deal with spam documents, it is important to investigate the relationship between spam and utility. We conducted a labelling experiment to investigate whether there is a relationship between a page's usefulness and spam score. Judgements were made on a carefully selected pool of documents, which included known relevant documents, known irrelevant documents, and known spam documents.

Note that in this study we consider judge's ratings of individual documents in isolation. A companion study of the relative effect of spam and irrelevant documents on user satisfaction with search engine result pages is reported in [5].

Proceedings of the 16th Australasian Document Computing Symposium, Canberra, Australia, 2 December 2011.

2 Queries and documents

To build the pools, popular queries were extracted from the FAST search engine query log [8]. The list of queries was pruned to remove adult content and likely single answer navigational queries (such as "hotmail.com"), and then unique queries were ranked by frequency. Five queries that lent themselves to information gathering tasks were hand selected from the top 20 of the pruned list, and task statements were created for each of these queries. The queries were "free posters", "moon landing", "online tv", "recipes", and "dictionaries". Task statements are included in Appendix A.

For each query, we built a pool of nine documents.

- Two documents were chosen from the first page of results of each of two major search engines, for a total of four relevant documents.
- Three documents were chosen from later pages of search results. These were chosen to be irrelevant, but still to be plausible results—e.g., they still contained related terms.
- Two spam documents were retrieved from an index of pages from the UK-2006 collection which had been labelled as spam [3].

These ratios were chosen based on observations of commercial and our own search engines in previous experiment.

3 Judgements

Judgements were obtained in a two phase process. In the first phase, judges were presented with each query and the corresponding task description. Each document from the current pool was presented as if it were a search result, with title, query-biased summary, and a link to the document itself. Document ordering within each pool was randomised. Judges were then asked to rate each result on the scale {*very useful, useful, ok, not useful, totally useless*} according to how useful they believed it would be for completing the task. This five point scale is inspired by the scale used by Wu and Davidson [9], but we use the word *useful* instead of *relevant*. Judges were not given any instruction in what "useful" meant.

In the second phase of the experiment, we educated judges to assess pages for spam content, using the instructions presented to judges for the UK-2006 and UK-2007 web spam collections [3]. Judges were presented with each document from each result pool again, and asked to rate documents on the scale {*completely spam, borderline spam-leaning, borderline normal-leaning, normal*}. These labels were inspired by the labels from the UK collections, but we include two levels of borderline labels because of the anticipated disagreement on pages labeled *borderline* [3]. We did not include the *can-not-classify* label, because we ensured all results in the pools were available at the time of the experiment. Within-pool document ordering was again randomised.

34 volunteer judges, largely postgraduate students, were solicited from the university community. Judges were compensated with a movie ticket upon completion of their judgements. Some judges did not complete the judgement process, causing some of their judgements to only have a usefulness score. Removing these incomplete judgement pairs, the 34 judges created a total of 1123 ⟨*judge, document, usefulness score, spam score*⟩ judgements across the 45 documents.

This is a relatively high dropout rate: we have lost 27% of possible judgements. Judges worked remotely and unsupervised, so it is likely that some simply got bored. On the other hand, this suggests that those judgements we did collect are from motivated judges. Since documents were presented in random order, we do not expect any significant bias due to attrition.

Since our volunteers judged each page for utility before judging them for spamminess, there is a chance of some carry-over effect. Having decided a page is useful, and with a notion of spam as useless by definition, judges may be prone to labelling *very useful* or *useful* pages as non-spam where they would have labelled them spammy without the prior prompt. This may mean that there are even more useful spam pages than are counted here. In this respect, our conclusions are possibly conservative.

4 Results

Examining the spam scores for all the judgements at each level of utility produces Figure 1(a). The distribution of spam scores varies significantly across levels of utility (χ^2 test, $p \ll 0.0005$), indicating that spam scores vary somehow with utility; it is clear that in fact the more useful pages are less likely to be spam. A similar relationship is visible in Figure 1(b), where spam labels explain some of the difference in utility scores and spammier pages are less likely to be useful.

Note however an interesting minority in either case: when a document was judged *very useful*, in more than 20% of cases it was also judged *borderline-spam-leaning* or *completely spam*. In the case of judgements of *completely spam*, over 15% were also judged *useful* or *very useful*. Clearly, not all spam is useless and some is in fact useful.

Our judges did not tend to agree with each other in their labelling. We recorded Krippendorff's α of 0.249 for spam judgements, and 0.483 for relevance judgements.[1] This is low agreement, especially for "spaminess", which suggests that spam is hard to spot even when judges are given careful instruction. Castillo et al. note [3]

> "a common problem raised by the judges was that the evaluation of borderline cases is very subjective. Indeed, many Web sites that use spam techniques also provide some contents, so that it is very difficult to classify them as spammers."

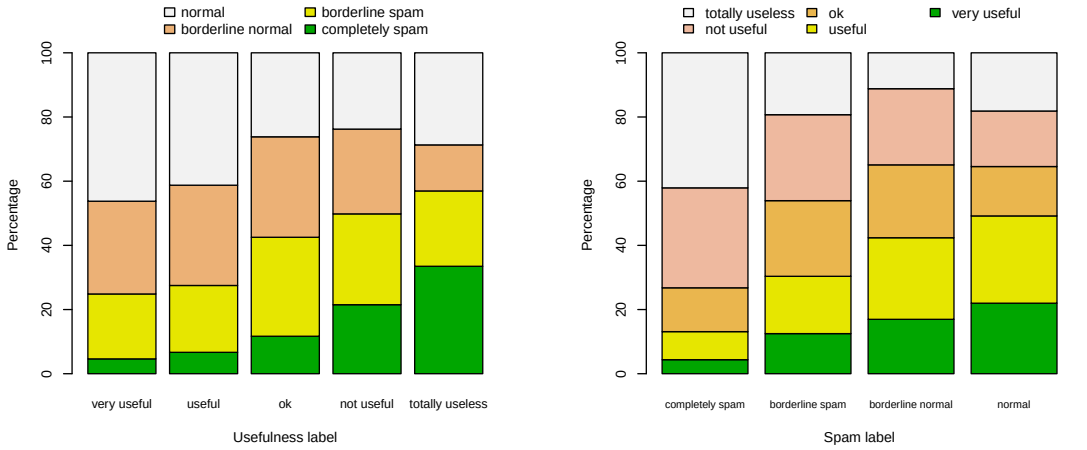
We agree. Inter-judge agreement is likely to be low for most experiments of this nature: Castillo et al. report Fliess's K of 0.56, "moderate" agreement, on a simpler problem with a three- (not four-) point scale and with only two judges per document (not up to 34).

5 Discussion

The percentage of *totally useless* documents that were labelled spam is nearly 60%, which is close to the background probability that a given page from 2009 was spam [4]. There appears to be no relationship between documents marked as *totally useless* and any spam label other than *completely spam*—as one would expect, just because a document is not spam does not mean it is useful.

Observing Figures 1(a) and 1(b), only a small percentage of *completely spam* documents are considered *useful* or better. Additionally, in Figure 1(b), we see considerably more documents marked as *totally useless* in the *completely spam* category than in any other spam category. These two observations imply it may be safe

[1] α, which measures inter-judge agreement, ranges from 0 (data are random) to 1 (judges are perfectly in agreement) and can be adapted to multiple judges, missing observations, and ordinal rather than categorical assessments [1, 7].



(a) Judgements of spamminess, conditioned on level of utility.

(b) Judgements of utility, conditioned on level of spamminess.

Figure 1: All 1123 ⟨ utility, spamminess ⟩ judgements.

to remove the spammiest documents from the index, as they are most likely not useful.

However, although there is a slight trend that useful documents have lower spam scores (Spearman's $\rho = -0.26$), it is difficult to make reliable assumptions about the potential spam score of a document which is a little bit useful; or about the usefulness of a document which is a little spammy. Importantly, more than 20% of *very useful* documents were labelled *borderline-spam-leaning* or *completely spam*.

Cormack et al. [4] found that their spam filter (used to produce the ClueWeb'09 spam labels) was also effective at detecting documents that had been labelled irrelevant by TREC assessors—although less effective than it was at detecting spam documents. This suggested that spam documents may be generally irrelevant, which is a stronger effect than we see here. Due to the larger sample used by Cormack et al., and our hand-crafted document selection method, their results may be biased towards including "spammier" spam examples. It is possible that a large fraction of real-world cases are both obviously spam and obviously useless (e.g. pages from a link farm, as seen in Figure 2(a)). Even if these pages do dominate, it is important not to ignore the potential usefulness of completely or borderline spam documents.

The low inter-judge agreement seen here and in the UK-2006 collection reinforces this: if we believe a page is spam, and remove it, not everyone will agree with the choices we make.

In related work [5], we have seen that users are sensitive to spam in result sets, so in general spam should be supressed. However, discarding all spam pages will mean that some useful pages are thrown away. Neither utility nor spamminess can be the only factor in a good ranking—search engines, and evaluation measures, must account for both.

What are the useful spam pages? A manual inspection of the pages that were labelled as *borderline-spam-leaning* or *completely spam* and with a usefulness score of *ok* or better indicated that these pages tended to contain excessive advertising but also provide some useful service such as a store or edited content. Many of these pages were much more sophisticated than a simple copy of Wikipedia content. Figure 2(b) shows one such page, where users located in the US can obtain streaming television (although our judges were located in Australia, they still scored the page "useful" on average. Use of a proxy located in the US confirms that streaming television is available).

6 Conclusions

Our judges found that an important proportion of spam documents (around 13% to 40% depending upon the spam rating) were either *useful* or *very useful* to the specified task. This suggests that pages classified as spam should not be summarily excluded from search engine indexes. Since there is no clear-cut relationship between spamminess and utility, ranking algorithms and quality evaluation campaigns for Web search should take into account both utility and spamminess; neither alone will suffice.

(a) Example *completely spam* and *totally useless* page (averaging ratings across all users). This page offers no content itself, not even copied from elsewhere, so it is not useful; it serves only for advertising.

(b) Example *borderline-spam-leaning* but *useful* page (averaging ratings across all users). Although the page is heavily laden with advertising, and appears to be spam content, streaming television is available in one click.

Figure 2: Sample spam pages.

References

[1] Ron Artstein and Massimo Poesio. Inter-coder agreement for computational linguistics. *Computational Linguistics*, Volume 34, Number 4, pages 555–596, December 2008.

[2] Carlos Castillo and Brian D. Davison. Adversarial web search. *Foundations and Trends in Information Retrieval*, Volume 4, pages 377–486.

[3] Carlos Castillo, Debora Donato, Luca Becchetti, Paolo Boldi, Stefano Leonardi, Massimo Santini and Sebastiano Vigna. A reference collection for web spam. *SIGIR Forum*, Volume 40, Number 2, pages 11–24, December 2006.

[4] Gordon V. Cormack, Mark D. Smucker and Charles L. A. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. `http://arxiv.org/abs/1004.5168`, 2010.

[5] Tim Jones, David Hawking, Paul Thomas and Ramesh Sankaranarayana. Relative effect of spam and irrelevant documents on user interaction with search engines. In *Proc. CIKM*, 2011.

[6] Timothy Jones, Ramesh Sankaranarayana, David Hawking and Nick Craswell. Nullification test collections for web spam and SEO. In *Proc. 5th Int'l Workshop on Adversarial Information Retrieval on the Web*, 2009.

[7] Klaus Krippendorff. *Content Analysis, An Introduction to Its Methodology*. Sage Publications, California, USA, second edition, 2004.

[8] Amanda Spink, Seda Ozmutlu, Huseyin C. Ozmutlu and Bernard J Jansen. US versus European web searching trends. *SIGIR Forum*, Volume 36, pages 32–38, September 2002.

[9] Baoning Wu and Brian D. Davison. Undue influence: eliminating the impact of link plagiarism on web search rankings. In *Proc. SAC*, 2006.

A Task statements

The five tasks allocated were:

1. ("Free posters") You'd like to download and print some posters to decorate the walls of a teenager's bedroom. Please rate each result considering how useful it is for finding **free posters**.

2. ("Moon landing") July the 21st was the 40th Anniversary of the Apollo 11 Moon landings. With the recent coverage, you're interested in a summary of the events of the Moon landing. Please rate each result considering how useful it is as a summary of the Apollo 11 **moon landing**.

3. ("Online TV") You've heard that you can watch TV online, instead of using a television. Please rate each result considering how useful it is for watching **TV online**.

4. ("Recipes") You are looking for a site with a large collection of recipes to add to your home cookbook. Please rate each result considering how useful it is for finding **recipes**.

5. ("Dictionaries") You are looking for English to non-English dictionaries, so that you can file them away for later use. The more non-English languages, the better. Please rate each result considering how useful it is for finding bilingual **dictionaries**.

Indexing without Spam

Guido Zuccon
The Australian e-Health Research Centre
CSIRO
QLD, Australia
Guido.Zuccon@csiro.au

Anthony Nguyen
The Australian e-Health Research Centre
CSIRO
QLD, Australia
Anthony.Nguyen@csiro.au

Teerapong Leelanupab
Faculty of Information Technology
KMITL
Thailand
t.leelanupab@it.kmitl.ac.th

Leif Azzopardi
School of Computing Science
University of Glasgow
Scotland, UK
Leif.Azzopardi@glasgow.ac.uk

Abstract

The presence of spam in a document ranking is a major issue for Web search engines. Common approaches that cope with spam remove from the document rankings those pages that are likely to contain spam. These approaches are implemented as post-retrieval processes, that filter out spam pages only *after* documents have been retrieved with respect to a user's query. In this paper we propose removing spam pages at indexing time, therefore obtaining a pruned index that is virtually "spam-free". We investigate the benefits of this approach from three points of view: indexing time, index size, and retrieval performance. Not surprisingly, we found that the strategy decreases both the time required by the indexing process and the space required for storing the index. Surprisingly instead, we found that by considering a spam-pruned version of a collection's index, no difference in retrieval performance is found when compared to that obtained by traditional post-retrieval spam filtering approaches.

Keywords Information Retrieval; Index Pruning; Spam; Web search; Efficiency.

1 Introduction

The presence of spam in a document ranking is a major issue for Web search engines. For example, in TREC 2009 Web Track, which is based on a large crawl of the Web (i.e. the ClueWeb collection, about 1 billion web pages), it has been noted that spam harmed performance of retrieval systems [10, 11]. To cope with this problem, participants to the TREC Web Track have implemented strategies of post-retrieval processing that filter out pages deemed as spam [9, 12]. Similarly, Cormack et al. have produced four spam lists for the ClueWeb collection. These consist of a label associated to each document indicating the likelihood of the document being spam [8]. They also showed that by removing spam employing a post-retrieval process substantially improves retrieval performance of TREC 2009 systems.

In this paper we tackle the problem of spam in Web collections (specifically the ClueWeb collection) from a different perspective. While previous approaches filtered out spam from the retrieval results using a post-retrieval process, we modify traditional indexing procedures, by introducing a spam filtering step at indexing time. By doing so, web pages that are deemed to be spam with respect to some threshold level are not indexed, and therefore, neither are they retrieved. By not considering spam documents at indexing time, modified index statistics are obtained, if compared to indexes obtained using traditional indexing procedures. Our strategy acts as an index pruning technique, driven by the spam scores of documents. We expect our technique to benefit by the common characteristics of indexing pruning strategies: reduced index size and reduced indexing time. However, index pruning techniques have been shown to degrade retrieval performance of systems. But, does pruning with respect to spam documents harm retrieval performance as well?

To investigate benefits and drawbacks of *"indexing without spam"*, we conducted a number of retrieval experiments across several traditional retrieval models on the TREC ClueWeb 2009-2010 Web Track dataset. We found that the use of indexes used through our indexing without spam procedure does not harm retrieval performance. Instead, often this procedure offers better effectiveness than traditional indexing with or without post-retrieval removal of spam documents, while reducing indexing and retrieval time.

The paper continues as follows. In section 2 we outline the method of indexing without spam. Subsequently we present related works in the area of in-

Proceedings of the 16th Australasian Document Computing Symposium, Canberra, Australia, 2 December 2011.

dex pruning and spam identification and removal in the context of the ClueWeb collection and the TREC Web Track (section 3). Our research questions are stated in section 4, while section 5 presents and discusses our experimental settings and results. Finally, the paper concludes in section 6.

2 Indexing without Spam

Assume that a spam list for a corpus is given, where a score is associated to each document indicating its likelihood of being spam. For ease of exposition, let assume that a low spam score indicates that a document is very likely to be spam, with 0 being associated with the documents that are most probable spam. Similarly, a high score would suggest that a document is unlikely to be spam, with 99 indicating documents that are least probable to be spam.

Given these settings, we propose to modify the traditional indexing processes, such that both corpus and spam list are provided as input to the indexer. Thereafter, the indexer is instructed to index only those documents in the corpus for which their spam list scores are higher than a threshold th. Algorithm 1 outlines the pseudocode of such indexing procedure, where $\mathcal{SL}(d)$ returns the spam score associated with document d.

This approach resembles the notion of index pruning, where the inverted index is compressed by avoiding storing some of the statistical information associated to terms or documents.

Algorithm 1 Indexing without spam

Input: a corpus of documents $\mathcal{D}$; a spam list $\mathcal{SL}$ containing pairs of documents and spam scores; a threshold value th
Output: an index $\mathcal{I}$

```
for all d ∈ D do
  if SL(d) > th then
    index d in I
  else
    ignore d
  end if
end for
return I
```

3 Related work

3.1 Index Pruning

As the approach presented in section 2 is similar to pruning an index (lossy compression), we briefly revise and compare some index pruning strategies proposed in the literature. In particular, we focus on static index pruning strategies, as opposed to dynamic strategies which are applied at query time (for example [13]).

The strategy of posting pruning by sparsification of the index table, as proposed by Carmel et al. [2], consists of calculating the importance of individual postings in the index, and then removing from the index those postings that are less informative. This results in the removal of the associated terms from documents, thus varying the statistics associated to document lengths and term frequencies, and possibly lead to the complete removal of a term from the index. Therefore, this pruning strategy acts at the term level.

While the sparsification of the index table (at term level) does not necessarily imply the complete removal of a term from the index, the term pruning strategy of Blanco and Barreiro [1] prescribes that uninformative terms should be completely removed from the index. Therefore, such terms are treated similarly to stopwords. This approach radically modifies the statistics of the pruned index, affecting the document length statistics. Ultimately, the strategy might affect the presence of some documents, if these contained only terms that have been pruned.

While the previous approaches act at a term level, the strategy proposed by Zheng and Cox prunes the index at a document level [20]. A score is computed for each document in the collection, based on the entropy of the terms in the documents. Documents are thereafter selected or rejected for indexing depending upon their score being higher or lower than a set threshold. This solution affects the presence of documents, the average document length, and the inverse document frequency.

Common to index pruning strategies is that a trade-off is found between index size and retrieval performance: higher levels of index pruning (i.e. smaller indexes) translate in worse retrieval performance, with no-pruning obtaining usually the best retrieval performance.

The document pruning strategy of Zheng and Cox is the closest pruning strategy to that proposed in this paper. In contrast to Zheng and Cox, here we do not consider the entropy of terms within documents for deciding which documents should be excluded from the indexing process. Instead, we reject documents for indexing according to their scores with respect to spam features.

3.2 Spam Identification and Removal

The presence of spam among top ranked search results is a problem that harms the retrieval effectiveness of Web search engines. Participants to the TREC 2009 Web retrieval Track have noted the impact of spam on system effectiveness [10, 11], while others attempted to prune spam documents from the ranking through a post-retrieval process [9, 12]. In particular, Hauff and Hiemstra used a spam detection algorithm that relies on page content, page title features and URL form [9]. Once spam documents were identified, they were removed by the document ranking through a post-retrieval processing stage. Similarly, Lin et al. used Yahoo!'s proprietary adult, spam, and document quality classifiers to identify and remove spam once documents have been retrieved by their system [12].

While several methods have been proposed in the literature to identify spam documents (e.g. [3, 14]), we focus on the work of Cormack et al. [8]. This is because they have already successfully applied their spam detection and filtering methods to web retrieval (and in particular to the same dataset we employ in this study) and studied the effect of spam on retrieval performance. In particular, Cormack et al. adapted content-based email spam filters based on three different training sets to the task of identifying spam web pages in ClueWeb. The training sets were:

- UK2006: a (small) set of spam and non-spam labels for web pages, containing 767 spam pages and 7,474 non spam pages.
- Britney: a set of training examples heuristically built from popular search queries as reported by popular web search engines' statistics. Following this method, the top 10 ranked pages retrieved by the Indri retrieval system in answer to the popular queries were deemed as being spam. While, 10,000 documents belonging to the collection for which their URIs were found to be also present in the Open Directory Project[1] were classified as non-spam.
- Group X: this training set consists of 756 documents for which human assessments regarding the presence of spam were collected by Cormack et al.

Furthermore, the single scores obtained by the three filters can be combined together through a naïve Bayes combination: this approach (called *"Fusion"*) has been shown to be the most effective in the context of spam identification and document retrieval when using the ClueWeb dataset. In the experiments reported in section 5, we employ the spam list generated with the Fusion method, which is publicly available[2]. For each document in ClueWeb, the spam list contains a percentile score, which indicates the percentage of the documents in the corpus that are "spammier". For example, if a document received a percentile score of 95 it means that it belongs to the 5% of the documents that are least probable to be spam. While, if a low percentile score is associated to a document, then this is likely to be a spam page: a document with score 0 belongs to the "spammiest" 1% of the documents.

The research we have reviewed in this section provides us the motivations for studying the effect of spam on indexing, both in terms of indexing and retrieval performance obtained by employing a spam-pruned index.

4 Research Questions

Previous approaches that account for spam in document retrieval have tackled the problem at the *retrieval phase*: documents are retrieved in response to a query, those that are identified as possible spam are filtered out, and the remaining documents are returned to the user that initially issued the query. In this work we investigate an alternative solution, where documents that are identified as being spam are removed during the *indexing phase*. Specifically, when indexing a collection, we suggest that documents that may be considered to contain spam are filtered out and therefore not indexed at all. This approach leads to the construction of an index that does not contain spam documents (as identified by the spam identification algorithm that is employed). Furthermore, the resulting index statistics, such as term counts, idf-s, average document lengths, etc, are effectively affected by the absence of evidences that would normally have been drawn from documents considered as spam.

In this paper, we investigate the differences between the two approaches to document retrieval with spam removal. Specifically, we explore the following research questions:

RQ1 How does spam removal at indexing time affect the *indexing process*?

RQ2 How does spam removal at indexing time affect the *indexes* that are created? And, how different are these indexes from those created using the standard indexing procedure?

RQ3 How does spam removal at indexing time affect *document retrieval*? Do index statistics obtained by an index that has been pruned from spam at indexing stage lead to worse retrieval performance, if compared to that obtained using retrieval-time spam removal? Does spam removal at indexing time affect retrieval time as well?

5 Experiments

5.1 Experimental Methodology

To empirically investigate our research questions, we employed the ClueWeb 09 collection (we used Category B only in this initial investigation), consisting of more than 50 million web pages. We indexed documents using Indri 5.0 [17][3], after stop-words removal and stemming conflation with the Krovetz stemmer. In the retrieval experiments, we used the TREC 2009 and 2010 Web Track topics[4], and four standard retrieval models as implemented by Indri: Okapi BM25 [16], Unigram Language Model with Dirichlet smoothing (LMDIR), Unigram Language Model with Jelineker-Mercer smoothing (LMJM) [18], and Unigram Language Model with two stage smoothing (LM2S) [19]. For the retrieval methods based on language model, we used the parameter

[1] `http//rdf.dmoz.org`

[2] `http://durum0.uwaterloo.ca/clueweb09spam/`

[3] `http://lemurproject.org/indri.php`

[4] Specifically, we used all topics released in TREC 2009 Web Track and 48 topics released in TREC 2010 Web Track: topics 95 and 100 from TREC 2010 were excluded because no relevance assessments were provided for them.

values that obtained the best retrieval effectiveness in the experiments of Zhai and Lafferty on the TREC 8 Web collection with short queries [18]:

- LMDIR: $\mu = 3,000$
- LMJM: $\lambda = 0.01$
- LM2S: $\lambda = 0$, $\mu = 3,000$

For BM25 we used the standard settings suggested in [15], i.e.:

- BM25: $b = 0.5$, $k_1 = 2$, $k_2 = 0$, $k_3 = 8$

To identify spam, we employed the "Fusion" spam list created by Cormack et al. [8], where a percentile score is associated with each ClueWeb document indicating its level of spam, as described in section 3.2. Note that the percentile scores refer to the whole ClueWeb dataset, while here we consider category B only. As a result, in our experiments, the number of pages that have a percentile score of x or lower is not equivalent to the $x\%$ of pages in the category B dataset[5].

These common settings have been used to implement the following 3 systems:

System S_1: a traditional information retrieval system, where all the documents contained in the collection are indexed and when a query is issued retrieval is performed according to one of the standard models listed above.

System S_2: a modification of System S_1, where a post-retrieval process is activated to remove those documents that are identified in the spam list as containing spam with a percentile score smaller or equal to a threshold th. This system implements the *post-retrieval spam removal* approach.

System S_3: a modification of System S_1, where documents that are identified in the spam list as containing spam with a percentile score smaller or equal to a threshold th are not considered during the indexing process. This system implements our *indexing without spam* approach.

Systems S_1 and S_2 are based on the standard Indri 5.0 toolkit, where for System S_2 we implemented a post-retrieval filtering process that removes from the result rankings those documents that were flagged as spam. To implement System S_3, we modified the indexing process of Indri 5.0 toolkit, including a filtering stage to exclude spam documents before the system performs indexing. The spam list $\mathcal{SL}$ is read in memory using the C++ operator >>, and document identifiers that refer to those documents which do not have to be indexed are stored into a standard C++ `std::map<string,bool>` container. The storing operation (operator `[]`) is logarithmic in the size of `map`. At indexing time, once a document identifier is read by the indexing process, a `count` operation is performed on the `map` container: if the result of this operation returns a value greater than zero (i.e. the `map` contains the searched document) the document is not indexed, and the next document is processed. The `count` method requires logarithmic time in the size of the `map`.

In our experiments we empirically investigate four settings of the threshold parameter[6] th, i.e. $th = \{20, 45, 70, 95\}$. Moreover, we studied the performance of S_1, which corresponds to S_2 and S_3 when $th = 0$.

To assess the difference between the common indexing procedure and the approach based on indexing without spam, we record the time required for indexing by the standard Indri 5.0 toolkit and by our own modification of that software for all levels of the spam threshold th. Indexing and retrieval were performed on a high performance server, fitted with 16 Intel Xeon X7350 (2.93GHz) CPUs and 128 GB of RAM. For each setting of th, indexing and retrieval were performed separately, and no other user-process was in execution on the server when indexing the collection.

In our empirical study, indexing was performed only once for each level of th: thus the time we recorded is that of a single execution of the indexing process. A more accurate approach would be to consider a number of repetitions of the indexing process for each level of th, and report the mean time required to build each index. However, we do not follow this approach at this stage. To allow the reproducibility of the experiments, we report that Indri's indexing parameters `memory` was set to `50G` and `storeDocs` to `false`, across all indexing processes. The `memory` parameter provides a soft upper bound on the memory consumption of the indexer process (the total usage would be up to three times the parameter value). The `storeDocs` parameter set to `false` indicates that the original documents were not archived within the folder containing the Indri index.

On the contrary, we repeated the retrieval process 10 times for each setting. When reporting retrieval time statistics, we consider the mean value obtained from 10 iterations of the retrieval process.

The formats of the relevance assessments (i.e. qrels) for TREC 2009 and TREC 2010 ad-hoc tasks are different, as TREC 2009 qrels are suited for computing statMAP, while TREC 2010 qrels are tailored to standard MAP [6, 7]. This does not allow us to use a common set of ad-hoc measures across both topic sets. To overcome this issue and assess the retrieval effectiveness of the systems, we consider the TREC Web diversity task, although our systems do not perform any diversification of the document rankings. The diversity

[5]The actual percentages of pages that are considered as spam for $th = 20, 45, 70, 95$ are approximately $8\%, 22\%, 42\%, 80\%$, respectively.

[6]Note that $th = 70$ has been suggest to be the most effective threshold value [8].

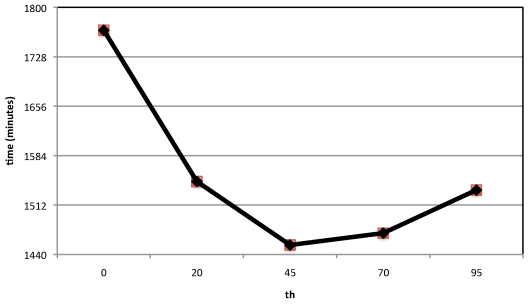


Figure 1: Indexing time for ClueWeb obtained using Systems S_1 and S_2 (which both correspond to value $th = 0$) and System S_3 for different values of the threshold th.

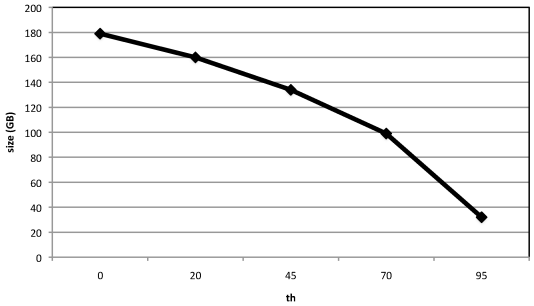


Figure 2: Size of the index of ClueWeb for Systems S_1 and S_2 (which both correspond to the value $th = 0$) and for System S_3 for different values of the threshold th.

qrels consist of the indication of document's relevance to a set of query-intents of a query, rather than to the query itself. The use of this retrieval task enables us to consider a common set of measures on the TREC 2009 and 2010 topic sets; in particular, we report the retrieval effectiveness of systems combining the effectiveness measured on both topic sets. The measures employed to assess the retrieval performance of systems are ERR-IA@10 [4] and α-nDCG@10 (with $\alpha = 0.5$) [5]. To provide an indication of how systems would perform in the task of standard ad-hoc retrieval, we produced standard qrels suited for computing (standard) MAP by considering a document relevant to a query if it is relevant to one or more subtopics of that query. MAP is then reported along with the selected diversity measures.

5.2 Results and Discussion

5.2.1 RQ1: Indexing Process

In figure 1 we report the time required for indexing the ClueWeb collection by Systems S_1, S_2 and S_3. For the latter system, we report the indexing time with respect to different settings of the threshold parameter th.

In answer to RQ1, removing spam during the indexing stage provides gains in terms of index efficiency, i.e. by lowering the indexing time, as demonstrated by the time required for indexing when using System S_3. In fact, System S_3 had been up to 20% times faster than System S_1 and System S_2 when indexing the ClueWeb collection. This result is not surprising, because System S_3's indexing process skips the documents that have been flagged as spam, and therefore System S_3 ends up processing less documents than Systems S_1 and S_2. However, note when using System S_3, the indexing time does not linearly decrease by increasing the value of the threshold th. Instead, high values of th for System S_3 may require more time for indexing the collection than that required by the same system for lower th. This is the case when $th = 95$: under this setting, S_3 requires 62 minutes more for indexing the collection than the same system with $th = 45$. This may be caused by the procedure we used for loading the file containing the list of documents which do not have to be considered for indexing. In fact, when $th = 95$ the list of documents to be excluded from indexing is large (998 MB against 271 MB for the list associated with $th = 45$) and the loading procedure may require more time than that saved for not indexing those documents.

5.2.2 RQ2: Indexes

In figure 2 we report on the size of the ClueWeb indexes as produced by Systems S_1, S_2 and S_3. For the latter system, we report the index sizes with respect to different settings of the threshold parameter th. Similarly, in table 1 we report the following index statistics: total number of indexed documents, total number of indexed terms, number of unique terms indexed.

In answer to RQ2, we found that the removal of spam during the indexing phase reduces the size of the index. Indexes created for the ClueWeb collection by System S_3 are up to 82.1% smaller than those created by Systems S_1 and S_2, when $th = 95$. As for question RQ1, this result is not surprising because in System S_3 up to 95% of the ClueWeb documents do not get indexed when $th = 95$: their statistics do not get recorded within the created index, and therefore the dimension of the index is smaller when compared to the standard index obtain by Systems S_1 and S_2. Furthermore, note that the decrease in index size in not linear with the increase of the value of threshold th: higher values of th provide higher index resizing ratios than lower ones. This is consistent with the statistics observed in table 1.

5.2.3 RQ3: Retrieval

The observations in sections 5.2.1 and 5.2.2 lead to the claims that filtering spam at indexing time provides advantages in terms of lowered indexing time (for some settings of th) and decreasing indexing size (consistently over all settings of th). But:

- does this translate into a reduced retrieval time as well?
- and, do these benefits harm retrieval effectiveness? or conversely, also is retrieval effectiveness

Systems	Statistics		
	Tot. # Indexed Documents	Tot. # Indexed Terms	Tot. # Indexed Unique Terms
S_1, S_2	50,220,423	40,417,956,329	87,331,162
S_3, $th = 20$	46,432,700	36,655,900,328	61,903,774
S_3, $th = 45$	39,303,448	30,833,198,931	48,382,773
S_3, $th = 70$	29,038,220	22,814,917,151	33,315,189
S_3, $th = 95$	10,008,217	7,278,840,138	11,532,356

Table 1: Index statistics (total number of indexed documents, total number of indexed terms, number of unique terms indexed) for the indexes of ClueWeb created by Systems S_1, S_2 and S_3.

		S_2				S_3			
	S_1	$th = 20$	$th = 45$	$th = 70$	$th = 95$	$th = 20$	$th = 45$	$th = 70$	$th = 95$
ERR-IA@10	.0878	.1570	.1771	.1922	.1591	.1569	.1772	.1926	.1564
α-nDCG@10	.1495	.2427	.2571	.2666	.2195	.2425	.2571	.2669	2168
MAP	.1693	.1891	.1792	.1376	.0491	.1901	.1821	.1411	.0505

Table 2: Retrieval effectiveness of System S_1, S_2 and S_3 for different values of th obtained when using LMDIR.

enhanced by considering a system that removes spam at indexing time?

These issues were explored when answering RQ3.

In figures 3 and 4 we plot the retrieval time of the four considered retrieval methods when using S_1 (specific case of S_3 with $th = 0$), S_2 and S_3, for varying values of the threshold th. For LMDIR, LM2S and BM25, there are marginal differences in retrieval time between systems S_1 and S_3, with respect to all values of th. On the contrary, differences are found for LMJM, where indexes created with higher th support faster retrieval, with $th = 95$ achieving a 93% speed-up with respect to system S_1 (8 seconds against 46 seconds). Retrieval times required by system S_2 are far greater than those required by S_1 and S_3. This is because in S_2 the removal of documents containing spam is performed at retrieval time. This process requires to perform a first pass of retrieval according to S_1. Then, the list of documents to be removed is loaded in memory, and documents are compared against the list. Documents considered as spam are then removed to only retrieve documents that are characterised by spam-scores greater than th. As the figures show, at parity of th, this process is considerably slower than using S_3: for example, for $th = 0$, retrieving using S_2 and LMJM is almost 364 times slower than the correspondent S_3 setting.

In table 2 we report the retrieval performance in terms of ERR-IA@10, α-nDCG@10 and MAP obtained by the three systems with different values of th and when using LMDIR. Similarly, tables 3, 4 and 5 report the retrieval performance obtained when using LMJM, LM2S and BM25, respectively.

The reported results show that retrieval effectiveness as measured by ERR-IA@10 and α-nDCG@10 is maximum when considering Systems S_2 and S_3 with

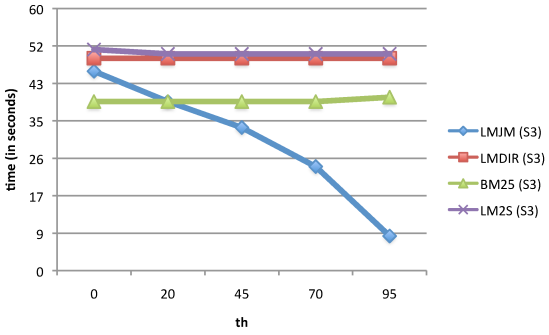


Figure 3: Retrieval time for 100 TREC 2009 and 2010 Web Track topics obtained using Systems S_1 (corresponding to the value $th = 0$) and S_3 for different values of the threshold th and for different standard IR retrieval models.

$th = 70$, regardless of the retrieval model implemented (i.e. LMDIR, LMJM, LM2S, BM25), while usually System S_1 obtains the lowest performance. In particular, we found that removing too many documents because of stringent spam threshold (i.e. $th = 95$), either during indexing or at post-retrieval phase, provides worse retrieval effectiveness than removing documents with $th < 70$. However, when a value of the threshold greater than 70 is used, Systems S_2 and S_3 yet generally obtained better retrieval performance than System S_1. When MAP is considered, best performance are obtained by low values of th (e.g. $th = 20$). However, for LMJM and BM25, no improvements in MAP are found for any setting of th.

When compared across the same levels of spam threshold th, Systems S_2 and S_3 exhibit similar retrieval performance. Specifically, S_3 is slightly more effective than S_2 when $th = 70$ is used, while S_2 is more effective than S_3 when $th = 95$. However,

		S_2				S_3			
	S_1	$th = 20$	$th = 45$	$th = 70$	$th = 95$	$th = 20$	$th = 45$	$th = 70$	$th = 95$
ERR-IA@10	.0804	.0846	.0913	.1105	.0815	.0846	.0913	.1105	.0807
α-nDCG@10	.1262	.1368	.1411	.1543	.1178	.1368	.1411	.1543	.1166
MAP	.0905	.0866	.0775	.0589	.0237	.0870	.0791	.0614	.0255

Table 3: Retrieval effectiveness of System S_1, S_2 and S_3 for different values of th obtained when using LMJM.

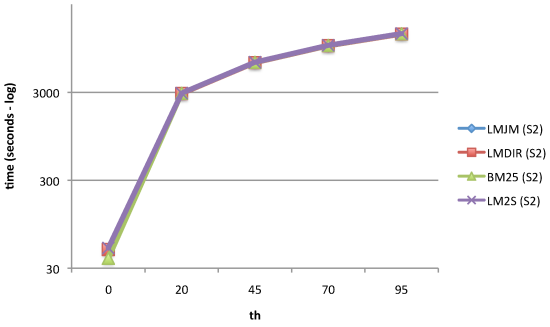


Figure 4: Retrieval time for 100 TREC 2009 and 2010 Web Track topics obtained using Systems S_1 (corresponding to the value $th = 0$) and S_2 for different values of the threshold th and for different standard IR retrieval models. Time in seconds is reported in logarithmic scale. Note that the retrieval times for S_2 are dominated by the process of loading in memory the file containing the list of spam documents.

differences are small and not statistically significant, when analysed using a two-tailed paired t-test.

In answer to RQ3, we found that indexing without spam does not harm retrieval effectiveness. Instead, the retrieval effectiveness of System S_3, which implements the indexing without spam procedure, is higher than that of System S_1 for specific values of th. We also found that System S_3 delivers similar retrieval performance than System S_2 when considering the same spam threshold th. However, when the retrieval times are considered, we found that S_1 is significantly faster than S_2. System S_3 generates indexes that are smaller than those of S_1: in general, this does not influence retrieval time for LMDIR, LM2S, and BM25. However, the retrieval time required when using LMJM decreasing with the increase of th. This difference in retrieval times may may be due to the specific implementation of the considered retrieval models in Indri. We also found that the time required for creating indexes when using System S_3 was generally lower than that required for Systems S_1 and S_2.

6 Conclusions

In this paper we have proposed a modification of traditional indexing procedures, called indexing without spam, where only documents considered as not containing spam are indexed. We have shown that this approach modifies the index statistics when compared to traditional indexing procedures, and we have linked our approach with methods of index pruning, and in particular with [20].

To assess the effectiveness of our proposal, we set up a thorough empirical comparison using the TREC ClueWeb collection and the TREC 2009 and 2010 Web topics. Results suggest that indexing without spam is more effective than standard indexing procedures combined with post-retrieval spam removal, when considering retrieval measures such as ERR-IA, α-nDCG and MAP.

An important observation is that indexing pruning strategies are believed to harm retrieval performance, e.g. [20], while obtaining gains in throughput. However, empirical results demonstrate that indexing without spam prunes indexes without harming retrieval effectiveness: often our approach delivers better performance than the other considered strategies.

Future work aims to extend this study by considering:

- more executions of the indexing procedures, so as to obtain more reliable indications of the time required for indexing in each experimental setting;
- ClueWeb part A, which consists of more than one billion web pages. In particular, it is interesting to explore whether the index statistics obtained through the indexing without spam procedure are sensibly more effective then those collected by standard indexing procedures when used for retrieving documents.
- a more systematic exploration of how document removal affects retrieval effectiveness.

References

[1] Roi Blanco and Álvaro Barreiro. Static pruning of terms in inverted files. In *ECIR '07*, pages 64–75. 2007.

[2] David Carmel, Doron Cohen, Ronald Fagin, Eitan Farchi, Michael Herscovici, Yoelle S. Maarek and Aya Soffer. Static index pruning for information retrieval systems. In *SIGIR '01*, pages 43–50, 2001.

[3] Carlos Castillo, Debora Donato, Aristides Gionis, Vanessa Murdock and Fabrizio Silvestri. Know your neighbors: web spam detection using the

		S_2				S_3			
	S_1	$th = 20$	$th = 45$	$th = 70$	$th = 95$	$th = 20$	$th = 45$	$th = 70$	$th = 95$
ERR-IA@10	.0878	.1570	.1771	.1922	.1591	.1569	.1772	.1926	.1564
α-nDCG@10	.1495	.2427	.2571	.2666	.2195	.2425	.2571	.2669	.2168
MAP	.1693	.1891	.1792	.1376	.0491	.1901	.1821	.1411	.0505

Table 4: Retrieval effectiveness of System S_1, S_2 and S_3 for different values of th obtained when using LM2S.

		S_2				S_3			
	S_1	$th = 20$	$th = 45$	$th = 70$	$th = 95$	$th = 20$	$th = 45$	$th = 70$	$th = 95$
ERR-IA@10	.1452	.1824	.2016	2026	1531	1831	.2015	2035	.1496
α-nDCG@10	.2207	.2636	.2720	.2740	.2098	.2653	.2715	.2756	.2056
MAP	.1528	.1492	.1338	.0984	.0390	.1498	.1360	.1012	.0392

Table 5: Retrieval effectiveness of System S_1, S_2 and S_3 for different values of th obtained when using BM25.

web topology. In *SIGIR '07*, pages 423–430, 2007.

[4] Olivier Chapelle, Donald Metlzer, Ya Zhang and Pierre Grinspan. Expected reciprocal rank for graded relevance. In *CIKM '09*, pages 621–630, 2009.

[5] Charles L. A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Buttcher and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR '08*, pages 659–666, Singapore, 2008.

[6] C.L. Clarke, N. Craswell and I. Soboroff. Overview of the trec 2009 web track. In *Proc. of TREC 2009*, 2009.

[7] C.L.A. Clarke, N. Craswell, I. Soboroff and G.V. Cormack. Overview of the trec 2010 web track. In *Proc. of TREC 2010*, 2010.

[8] Gordon Cormack, Mark Smucker and Charles Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *JIR*, pages 1–25, 2011.

[9] Claudia Hauff and Djoerd Hiemstra. University of twente @ trec 2009: Indexing half a billion web pages. In *Proc. of TREC 2009*, 2009.

[10] J. He, K. Balog, K. Hofmann, E. Meij, M. de Rijke, M. Tsagkias and W. Weerkamp. Heuristic ranking and diversification of web documents. In *Proc. of TREC 2009*, 2009.

[11] Rianne Kaptein, Marijn Koolen and Jaap Kamps. Result diversity and entity ranking experiments: Anchors, links, text and wikipedia. In *Proc. of TREC 2009*, 2009.

[12] J. Lin, D. Metzler, T. Elsayed and L. Wang. Of Ivory and Smurfs: Loxodontan MapReduce Experiments for Web Search. In *Proc. of TREC 2009*, 2009.

[13] Alistair Moffat and Justin Zobel. Self-indexing inverted files for fast text retrieval. *ACM Trans. Inf. Syst.*, Volume 14, pages 349–379, October 1996.

[14] Alexandros Ntoulas, Marc Najork, Mark Manasse and Dennis Fetterly. Detecting spam web pages through content analysis. In *WWW '06*, pages 83–92, 2006.

[15] S. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. and Tr. in IR*, Volume 3, Number 4, pages 333–389, 2009.

[16] K. Sparck-Jones, S. Walker and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments: Part 1. *IP&M*, Volume 36, Number 6, pages 779 – 808, 2000.

[17] T. Strohman, D. Metzler, H. Turtle and W. B. Croft. Indri: A language model-based search engine for complex queries. *ICIA '04*, 2004.

[18] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR '01*, pages 334–342, 2001.

[19] ChengXiang Zhai and John Lafferty. Two-stage language models for information retrieval. In *SIGIR '02*, pages 49–56, 2002.

[20] Lei Zheng and Ingemar Cox. Entropy-based static index pruning. In *ECIR '09*, pages 713–718. 2009.

Automatic identification of the most important elements in an XML collection

Alexander Krumpholz
CSIRO ICT Centre and
Australian National University
Canberra, Australia
krumpholz@acm.org

Amir Hadad
RSCS
Australian National University
Canberra, Australia
amir.hadad@anu.edu.au

Nina Studeny
University of Applied Science
Technikum Wien
Vienna, Austria
nina.studeny@gmail.com

Tom Gedeon
RSCS
Australian National University
Canberra, Australia
tom.gedeon@anu.edu.au

David Hawking
Funnelback and Australian National University
Canberra, Australia
david.hawking@acm.org

Abstract *An important problem in XML retrieval is determining the most useful element types to retrieve – e.g. book, chapter, section, paragraph or caption. An automated system for doing this could be based on features of element types related to size, depth, frequency of occurrence, etc. We consider a large number of such features and assess their usefulness in predicting the types of elements judged relevant in INEX evaluations for the IEEE and Wikipedia 2006 corpora. For each feature we automatically assign Useful / Not-Useful labels to element types using Fuzzy c-Means Clustering. We then rank the features by the accuracy with which they predict the manual judgments. We find strong overlap between the top-ten most predictive features for the two collections and that seven features achieve high average accuracy (F-measure $> 65\%$) acrosss them. We hypothesize that an XML retrieval system working on an unlabelled corpus could use these features to decide which retrieval units are most appropriate to return to the user.*

Keywords XML Retrieval, Fuzzy C-Means Clustering, F-Measure.

1 Introduction

Information retrieval (IR) systems attempt to find the documents in a corpus which best match a given query. In traditional IR systems the document is the obvious unit of retrieval. However, when documents are explicitly structured, e.g. in the Extensible Mark-up Language (XML), it may be more natural to retrieve sub-elements, such as sections of a paper, or chapters of a book. This raises a number of additional questions: What is the optimal granularity of result elements? Should retrieval results be presented which overlap or subsume each other? What element types make or do not make good retrieval units?

In the present work we address the latter question using resources developed by INEX (Initiative for the Evaluation of XML Retrieval)[4]. The INEX organisers have provided XML corpora and participants have contributed queries and assessments in an annual cycle since 2002.

Given the large number of distinct element types typically found in an XML corpus (e.g. 1257 in the 2006 INEX Wikipedia corpus), an automatic method for determining which element types (identified by their tag) make useful units of retrieval would be of considerable value.

In a first step toward achieving this we calculate feature scores for each tag in a corpus. For each feature we use a the Fuzzy c-Means (FCM) clustering method to label each tag as Useful or Not-Useful. We then compute the accuracy (F-measure) with which the automatically assigned labels align with the sets of tags appearing in the official INEX judgments. We do this for both the IEEE and Wikipedia 2006 corpora from INEX[1] and identify the features which best predict the usefulness of an element type as a unit of retrieval.

Proceedings of the 16th Australasian Document Computing Symposium, Canberra, Australia, 2 December 2011.

[1] The only XML corpora for which we had topics and assessments at the time. The XML version of Wikipedia pages was compiled by Ludovic Denoyer [3].

Naturally, a method for estimating the usefulness of element types as retrieval units must be combined with a normal retrieval system which estimates the relevance of the content of an element to the query.

In previous work in this area:

- Mihajlović et al. [5] examined structural knowledge for Information Retrieval in XML Databases, but explicitly exclude background statistics like element frequency.
- Ali et al.[1] performed statistical analyses of XML structure. They used structural summaries of the XML documents in the corpus to answer queries with structural constraints.

2 Data used

Collection	No. doc.s	No. unique tags
IEEE	16,819	178
Wikipedia	659,388	1.257

Table 1: Basic characteristics of the INEX collections examined.

The IEEE collection comprises journal articles which are well marked-up in XML, including citations. The Wikipedia articles are far less homogeneously structured.

3 Candidate features

An XML file is made up of different elements. Each element exists in a context defined by its parent nodes (elements) and its child nodes. Any new collection to be indexed will have new element types with unknown a-priori probability for their likelihood to be relevant.

To gain more information about each element type as well as to be able to classify certain nodes, we analysed XML elements of different XML corpora. For each element we collected various characteristics for comparison later on. The main characteristics relevant to this study were:

Name The name of the element is saved to identify it.

Frequency The number of occurrences of this element within the whole corpus.

Size (CharExclKids, CharInclKids, AVGCharExclKids, AVGCharInclKids) The size is counted in characters. Two values are collected in character size: Character excl. Character of child elements – meaning that the text had to be in the very element itself – and Character incl. Character of child elements – meaning the sum of characters occurring in the element itself or in any of its child elements. Listing 1 shows an example for an XML element without text nodes.

Listing 1: XML element with only child nodes.

```
<doc>
    <from>Peter</from>
    <body>Just a body</body>
</doc>
```

In this case the characters exclusive of child element characters for 'doc' would be 0, whereas the characters including the characters of the child elements equals 15. The size was believed to be the most important charateristic of an element. Too small elements might include good key words, but are too small to return to the user since they might not include all the needed information or might be useless out of context. As well, elements, which are too large include most likely good information, but it might be hard for the user to locate it.

Text nodes (TextNodeOcc) The number of text fields within an element. Two text fields can just be separated by other elements (children). Therefore this value can never be higher than the number of children nodes of an element + 1. Listing 2 shows an example.

Listing 2: XML element with text and child nodes.

```
<doc>
    first text node
      <from>Peter</from>
      <body>Just a body</body>
    second text node
</doc>
```

Child nodes (CountKids, AVGCountKids) The number of child nodes was saved, as well as the number of different types of children occurring within an element. For each child element different attributes were saved as well, like the occurrence, if this child element occurred every time in this certain parent element, or if its content was always numeric. Minimum, maximum, average and median values were generated.

Attributes (AttCount) The attributes were saved just the way the children nodes are, so their number as well as their label were marked down for later evaluation. The number of attributes that had been found was stored as AttCount.

Depth A list keeps track of the depth in which the element occurred and delivered the min, max, average and median values.

File Occurrence (FileOcc) A Boolean variable supported the process of finding 'Must elements', meaning elements, which occured in every file. The number of files where the element was found was saved as well.

Number of different child nodes of this element (NumbKidTypes) The number of child element types the XML element had within the collection.

In order to be able to sub-sequentially evaluate which elements are more important than others, the

INEX assessments were used. The elements were enriched by information about their assessment as being relevant.

Relevant If an element was ever marked in an assessment, this value was set to 1, else 0

Occurrence in Assessment(OccAss) The number of times an element is marked relevant.

Occurrence as Cover node We defined a cover node to be the node that covers all data marked relevant in a single assessment, i.e. the root node of the minimal sub tree containing relevant sections. When a passage was marked in an assessment it included most of the times a number of nodes, as shown in figure 1. The cover node would have been in this case the best node to return to the user, since it just enfolded all relevant information. In this case the C element is the cover node.

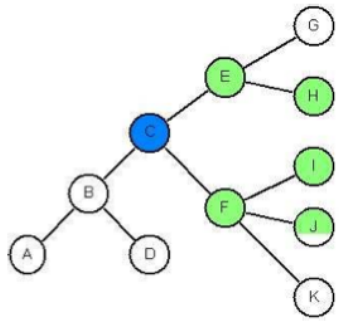


Figure 1: The cover node C, covering the whole passage marked as relevant by an INEX assessor.

4 Method

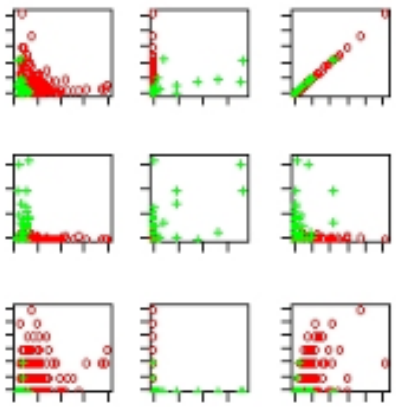


Figure 2: Small extract of the scatter plot matrix. Each plot compares the distribution of feature values for elements judged relevant and those judged irrelevant. [Unfortunately, this figure must be viewed in colour.]

Initially we compared the collected characteristics in a huge scatter plot matrix to get a better understanding of the collections. A small extract can be seen in figure 2. It shows elements marked as relevant in another colour to allow the visual identification of relevant element characteristics.

This approach indicated interesting characteristics, but we needed an objective and deterministic way of identifying the most useful features (properties). We employed Fuzzy c-Means clustering (FCM), first introduced by Bezdek in 1981 [2].

As the first step, we created two clusters for each individual feature based on the value calculated for each tag. As the second step, we divided the tags into relevant and non relevant set of tags based on these two clusters and their cut-point for each feature. As the final step, we measured the alignment of the automatically labeled tags with the published INEX relevance labels. We measured Precision, Recall and F-Measure and sorted the features on decreasing F-measure.

We applied this method for both IEEE and Wikipedia 2006 INEX collections. Among the results judged relevant for Wikipedia at INEX, 72 out of 1257 element types appeared. The corresponding ratio for IEEE was very different: 122 out of 177.

5 Results and Discussion

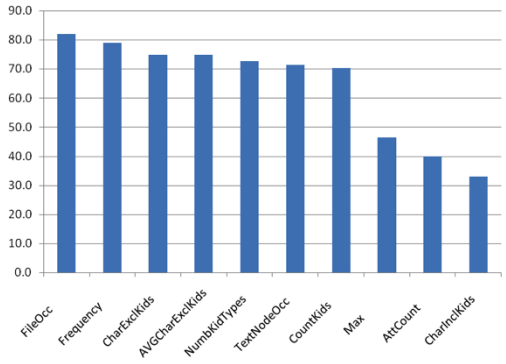


Figure 3: Top 10 F-Measure percentages for Wiki corpus features.

Figure 3 and figure 4 show the top 10 F-measure percentages for Wiki and IEEE corpora. These F-measure values were based on calculated precision and recall for the features. Note that despite the marked differences between the two corpora, eight features appear in both top-ten lists.

Finally we selected the 8 common features between the two corpora as the best list of features which can represent a corpus. We ranked these features based on the average of their two F-measure values. It can be seen that seven of the features achieve an F-measure score in excess of 65%. We propose that these features con be used to identify and select important tags for other corpora automatically.

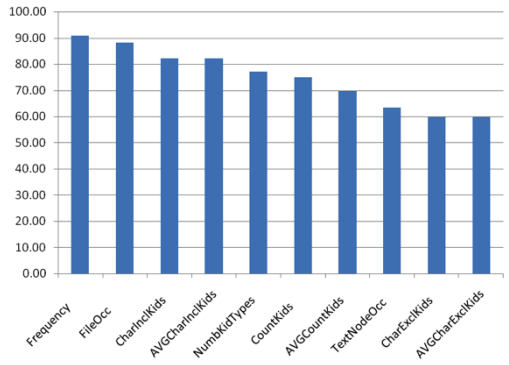


Figure 4: Top 10 F-Measure percentages for IEEE corpus features.

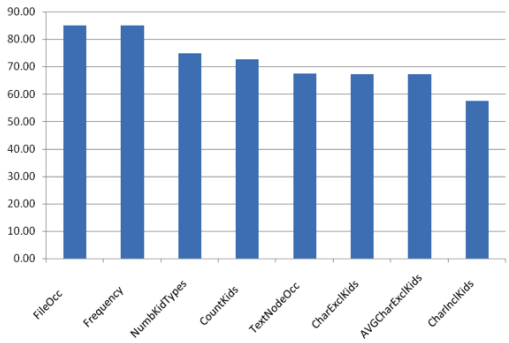


Figure 5: F-Measure percentages averaged across Wikipedia 2006 and IEEE for the eight features in common between the two Top Tens.

In a hypothetical retrieval system which calculated probabilities of relevance to a query for all elements, estimates of the usefulness of each element type as a retrieval unit could be fed into the ranking function as prior probabilities.

One limitation of our work is that we used the name of a tag to uniquely identify an element type. In general, XML allows the same tag name to appear as an element at different levels in the hierarchy. For example, the element <name> as child of an element <person> might have different child nodes and attributes, than the element <name> within <project>. This limitation would need to be removed in transferring our method into practice.

Obviously, Fuzzy c-Means clustering is far from the only possible method which could be used to select features. Future work may discover alternative methods which outperform even the relatively promising results reported here.

6 Conclusion

We have applied Fuzzy c-Means clustering to a number of statistical features of element types within an XML corpus in an attempt to label the element types as "useful unit of retrieval" or otherwise. We computed the accuracy with which these automatically assigned labels align with the manual judgments in two very different INEX test collections. We found substantial overlap between the best features across the two collections. We identified seven features whose average prediction accuracy (F-measure) across the collections exceeded 65%.

We hypothesise that these features could be used to improve performance of an XML retrieval system operating over a corpus for which no judgments are available.

References

[1] Mir Sadek Ali, Mariano P. Consens, Xin Gu, Yaron Kanza, Flavio Rizzolo and Raquel Kolitski Stasiu. Efficient, effective and flexible XML retrieval using summaries. In *INEX 2006 Revised and Selected Papers*, 2006.

[2] J.C. Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Kluwer Academic Publishers Norwell, MA, USA, 1981.

[3] L Denoyer and P Gallinari. The wikipedia XML corpus. *ACM SIGIR Forum*, Jan 2006.

[4] N. Fuhr, N. Gövert, G. Kazai and M. Lalmas. INEX: INitiative for the Evaluation of XML retrieval. In *Proceedings of the SIGIR 2002 Workshop on XML and Information Retrieval*, 2002.

[5] V Mihajlovic, D Hiemstra, H Blok and P Apers. Utilizing structural knowledge for information retrieval in XML databases. *wwwhome.cs.utwente.nl*, Jan 2005.

Language Independent Ranked Retrieval with NeWT

J. Shane Culpepper
RMIT University
Victoria, Australia 3000
shane.culpepper@rmit.edu.au

Michiko Yasukawa
Gunma University
Kiryu, Japan 376-8515
michi@cs.gunma-u.ac.jp

Falk Scholer
RMIT University
Victoria, Australia 3000
falk.scholer@rmit.edu.au

Abstract *In this paper, we present a novel approach to language independent, ranked document retrieval using our new self-index search engine,* NeWT*. To our knowledge, this is the first experimental study of ranked self-indexing for multilingual Information Retrieval tasks. We evaluate the query effectiveness of our indexes using Japanese and English. We explore the impact that linguistic processing, stemming and stopping have on our character-aligned indexes, and present advantages and challenges discovered during our initial evaluation.*

Keywords Text Indexing, Language Independent Text Indexing, Data Storage Representations, Experimentation, Measurement, Performance, Data Compression

1 Introduction

In this paper, we present a new self-index search engine, NeWT, which implements a novel approach to language independent, ranked document retrieval. NeWT is a hybrid search engine derived from an FM-*index* [Ferragina and Manzini, 2000] for fast substring matching, and a *wavelet tree* [Grossi et al., 2003] to support document level ranking. To our knowledge, this is the first comprehensive evaluation of ranked self-indexes using multilingual document collections.

We focus on an unexplored key aspect of supporting traditional ranked document retrieval using self-indexes – bag-of-words queries. Past work on ranked document retrieval in self-indexes has focused primarily on top-k phrase querying using simple TF×IDF similarity metrics [Hon et al., 2009, Culpepper et al., 2010]. In previous self-indexing approaches, the top-k documents are retrieved using raw frequency counts. While top-k retrieval using raw counts is a valuable first step to developing new approaches for efficient and effective ranked document retrieval, these methods have never been properly evaluated for traditional retrieval effectiveness on any reasonably sized document collection.

We extend the work of Culpepper et al. [2010] to support the Okapi BM25 [Robertson et al., 1994] similarity metric along with generic bag-of-words querying capabilities. We also perform the first comprehensive ranking evaluation using three different test collections: the TREC 7 & 8 *ad hoc* collection, the NTCIR-9 GeoTime collection of English and Japanese documents, and the JA-Category subset of the ClueWeb09 collection, as described in the NTCIR-9 INTENT track [Song et al., 2011]. As part of our evaluation, we expose and explore an interesting substring problem unique to character-aligned bag-of-words querying approaches.

Our Contributions. We present the first self-index capable of supporting language independent bag-of-words queries in a top-k ranked document framework. We provide the first comparative evaluation of self-indexes for top-k ranked retrieval against state-of-the-art inverted indexes on the TREC and NTCIR data collections.

We also explore the viability of constructing a monolithic multilingual index capable of delaying linguistic and morphological processing until *query time*. This is an important first step in building a new class of indexes for ranked text retrieval that make no a priori assumptions about the use of the text at indexing time, enabling decisions about complex linguistic processing to be deferred until query time.

2 Background

2.1 Inverted Indexes

For bag-of-words information retrieval tasks, indexing and querying is usually accomplished us-

Proceedings of the 16th Australasian Document Computing Symposium, Canberra, Australia, December 2, 2011. Copyright for this article remains with the authors.

ing an inverted index [Zobel and Moffat, 2006]. An inverted index contains a vocabulary of *terms* mapped to a list of tuples containing document and frequency data for the entire text collection. A term can be anything, but for simplicity we consider each term to be a single word in the linguistic sense.

For English text, stemming, stopping, case folding, and removing diacritics are standard steps in text preprocessing for inverted files [Witten et al., 1999, Büttcher et al., 2010, Croft et al., 2010], and is always done at indexing time. However, this preprocessing requires specific domain knowledge to be applied *at indexing time*. A considerable body of literature now exists on the advantages and disadvantages of different preprocessing techniques. For instance, stemming [Lovins, 1968, Porter, 1980, Frakes, 1984, Paice, 1990, Krovetz, 1993] of English text tends to help for large collections [Hull, 1996], but not necessarily in smaller collections [Harman, 1991]. The conventional wisdom on using stop words for English text is also context dependent. For some queries, using stop words can dramatically increase effectiveness, but for others stop words have a negative impact. Retaining stop words is also important for phrase queries [Witten et al., 1999, Manning et al., 2008].

In CJKV languages, segmenting text into terms is not always possible since words are not space delimited. Instead of mapping a single word in a language to a term, an alternative approach is to construct inverted indexes using character n-grams. This approach has a long history in Information Retrieval, particularly for language independent indexing [Burnett et al., 1979, Willett, 1979, de Heer, 1982, Hollink et al., 2004]. The general idea of n-grams is to index overlapping substrings of fixed length, and support bag-of-words querying using the same merging or intersection techniques as inverted files. One advantage of this approach is that language-specific parsing can be avoided at index time, and multiple languages can be supported simultaneously. However, the effectiveness of this approach is strongly dependent on the underlying language of the document [McNamee and Mayfield, 2004, McNamee et al., 2008]. In particular, the retrieval effectiveness in English text using n-grams is mixed, and often worse than term-based approaches, limiting widespread adoption [Büttcher et al., 2010].

English and many European languages can be described as *space-delimited languages*, wherein words are separated by white space. In English, a term in an inverted index is simply a word, which is a sequence of characters preceded and followed by whitespace [Dale et al., 2000]. However, east Asian languages such as Chinese, Japanese, Korean, and Vietnamese (CJKV languages), and agglutinated European languages such as Finnish or Turkish can be classified as *unsegmented languages* wherein words are not neatly delimited by spaces.

In Japanese, text is mainly composed of Kanji (Chinese characters) with Hiragana syllables for inflectional endings and function words [Manning et al., 2008]. Because words in Japanese are written in succession, with no indication of word boundaries, the word segmentation process is generally performed on Japanese documents by using morphological analyzers such as ChaSen[1] and MeCab.[2] However, the output of these morphological analyzers is not always correct. Some morphemes suggested by morphological analyzers are overly segmented, especially when documents contain unknown words [Yasukawa and Yokoo, 2010]. Furthermore, output texts are under-segmented or not segmented at all when web pages omit punctuation marks or white space in the text for the sake of simplicity or layout. In newspaper articles, unique proper nouns are commonplace. In patent documents, uncommon compound words are widely used to describe a developed technology. When wrongly segmented morphemes are stored as index terms, search results are often poor.

A considerable body of research comparing the trade-offs when applying n-gram or term-based methods preprocessing to Chinese exists (see, for instance, Kwok [1997], Nie et al. [2000], Luk and Kwok [2002]). Generally, n-gram based methods increase recall, while word-based methods improve precision [Luk and Kwok, 2002]. However, recent work has focused primarily on improving word segmentation using morphological analysis, since character and n-gram indexes can impose a significant space overhead.

2.2 Self-Indexes

Inverted indexes are a robust solution for search problems dealing with readily-parsable natural language text [Zobel and Moffat, 2006]. However, forcing unsegmented languages to rely on morphological analysis is not always the best option. Recently, *self-indexing* algorithms have emerged as a viable alternative to inverted indexes [Navarro and Mäkinen, 2007]. Self-indexing algorithms are capable of supporting character level matching similar to an n-gram index, but with the same bounded efficiency as a suffix array. Significant performance gains using these new data structures for basic pattern matching are possible, but efficiently supporting ranked queries on massive data sets using these novel

[1] http://chasen-legacy.sourceforge.jp/
[2] http://mecab.sourceforge.net/

data structures is still relatively unexplored [Hon et al., 2010]. Self-indexes use space close to what can be obtained by the best possible data compression algorithms. Previous work on self-indexes has focused predominantly on classical pattern matching problems. Self-indexes are extraordinarily efficient for simple pattern matching operations, but are capable of much more.

In this paper, we investigate the problem of using self-indexing algorithms to solve the **ranked document search** problem. The document collection $\mathcal{T}$ is a contiguous string drawn from an alphabet Σ^*, where $\sigma = |\Sigma^*|$. Each document in $\mathcal{T}$ is separated by a unique end of document symbol. In practice, Σ^* can be characters (UTF8 or ASCII), bytes, integers, or even terms.

Definition 1 *A* **ranked document search** *takes a query $q \in \Sigma^*$, an integer $0 < k \leq d$, and a text $\mathcal{T} \in \Sigma^*$ that is partitioned into d documents $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_d\}$, and returns the top-$k$ documents ordered by a similarity measure $\hat{S}(q, d_i)$.*

In order to solve the **document search problem**, unadorned self-indexing algorithms are not sufficient. The index also requires a *document array* – a mapping between every suffix in $\mathcal{T}$ to a corresponding document identifier [Mithukrishnan, 2002, Välimäki and Mäkinen, 2007, Culpepper et al., 2010]. By representing the document array with a wavelet tree, all occurrences of a substring $\mathcal{P}$ in each distinct document, and across the whole collection, can be counted in $\mathcal{O}(\log u)$ time, where $u \leq min(\sigma, d)$ using $\mathcal{O}(nH_k(\mathcal{T}) + n \log d) + o(n \log \sigma) + o(n \log d)$ bits of space. Here, $H_k(\mathcal{T})$ represents the kth order empirical entropy of the compressed text. The document and collection occurrence counts can then used to calculate TF×IDF based $\hat{S}(q, d_i)$ metrics at query time. A comprehensive discussion of prior work on applications of self-indexes to the **ranked document search** problem is beyond the scope of this paper. In this work, we use an enhanced version of the *greedy* top-k approach described in Culpepper et al. [2010].

2.3 Ranking in Self-Indexes

Effective and efficient document ranking in a self-index is an interesting open problem. For succinctness, we use "term" to represent any word, substring, or phrase as self-indexing methods can support any character sequence. All prior published work on ranked self-indexes uses a trivial TF×IDF ranking metric whereby the absolute term, substring, or phrase frequency is calculated, along with the number of distinct documents in which the term, substring, or phrase appears. Ranked self-indexes are capable of searching for a phrase, a bag of words, or a bag of substrings with equal ease.

For our experiments, our $\hat{S}(q, d_i)$ ranking function is a variant of BM25. To our knowledge this is the first ranked document self-index using BM25. We have began experimenting with other ranking models such as Dirichlet language model ranking, but BM25 is simple and effective for bag-of-words queries. Our $\hat{S}(q, d_i)$ ranking function has the following formulation:

$$\text{BM25} = \sum_{t \in q} \log \left(\frac{N - f_t + 0.5}{f_t + 0.5} \right) \cdot \text{TF}_{\text{BM25}}$$

$$\text{TF}_{\text{BM25}} = \frac{f_{t,d} \cdot (k_1 + 1)}{f_{t,d} + k_1 \cdot ((1 - b) + (b \cdot \ell_d / \ell_{avg}))}$$

Here, N is the number of documents in the collection, f_t is the number distinct documents appearances of t, $k_1 = 1.2$, $b = 0.75$, ℓ_d is the number of UTF8 symbols in the documents, and ℓ_{avg} is the average of ℓ_d over the whole collection. For self-indexes, there is an efficiency trade-off between locating the top-k $f_{t,d}$ values and accurately determining f_t since the index can extract exactly k $f_{t,d}$ values without processing every document. We leave the task of finding the most efficient trade-off between these competing requirements, and devising ranking metrics that do not make term-based *independent and identically distributed* assumptions, as topics for future work.

3 Experimental Framework

3.1 Text Collections

TREC 7 and 8 *Ad Hoc*. We conduct our English language experiments using the TREC 7 and 8 *ad hoc* collections. These use around 2 GB of newswire data from the *Financial Times*, *Federal Register*, *LA Times*, and *Foreign Broadcast Information Service*, consisting of around 528,000 documents in total [Voorhees and Harman, 1999]. A total of 100 test topics and associated relevance judgments are available for these collections (topic numbers 351–400 for TREC 7, and 401–450 for TREC 8). In our experiments, we use the *title* field of each topic; this gives a short query consisting of 2 to 3 keywords, on average, and is representative of the kinds of queries that users enter into popular web search engines.

NTCIR 2011 GeoTime. The NTCIR GeoTime task supports the investigation of search based on geographic and temporal constraints. Each information need therefore includes both time and space aspects. The GeoTime 2011 task used two document collections: English and Japanese. 25 information needs were made available in both

languages, enabling the investigation of retrieval for the same queries in different languages, as well as the potential for cross-lingual retrieval [Gey et al., 2011]. The English collection consists of newswire data from the *New York Times*, *Korea Times*, *Mainichi*, and *Xinhua*, together covering a time period from 1998 to 2005, and amounting to around 800,000 documents. The Japanese collection consists of newswire data from *Mainichi*, also covering a time period from 1998 to 2005, and including around 800,000 documents.

NTCIR 2011 INTENT in Japanese. One of the subtasks of the NTCIR 2011 INTENT task was Document Ranking in Japanese [Song et al., 2011]. The Japanese collection in the task consists of the Japanese segment of the ClueWeb09 collection[3], which entails 67.3 million documents. A total of 100 test topics and associated relevance judgments are available for the collection (topic numbers 0101–0200 for NTCIR 2011 INTENT).

3.2 Query Processing

In order to test the effectiveness of bag-of-words queries, the same query processing method was used for each collection. We use the Indri Search Engine[4] as the state-of-the-art baseline for our experiments. When stemming is required, we use Krovetz stemming in order to remain consistent with the Indri baseline [Krovetz, 1993]. English queries for NEWT and Indri were initially treated as a bag-of-words with no stemming or stopword removal. Each word was treated as a single term. The Japanese queries were initially partitioned into terms using ChaSen/MeCab with the standard IPA dictionary[5]. Our preliminary testing led to the discovery of an unexpected issue, which we will refer to as the *substring problem*.

The substring problem. The preliminary evaluation of simple Japanese and English queries in the test collections resulted in unusually poor effectiveness. Further investigation revealed a problem with both query sets, which is rather obvious in hindsight. Short query terms in English or Japanese are also substrings of other terms. These false matches degrade the discriminatory power of certain terms, and the problem becomes more pronounced with longer queries. The longer the query, the more likely at least one of the terms is a substring of another common term in the collection. For example, the acronym "ana" (All Nippon Airways) found in the GeoTime-034 query is also a substring of the word "analytic", and, worse, it is an overlapping substring in words like "banana / bananna". This problem can be somewhat mitigated using whitespace padding, as we will see shortly.

[3]http://lemurproject.org/clueweb09/
[4]http://sourceforge.net/projects/lemur/
[5]http://sourceforge.jp/projects/ipadic/

However, there is no white space in the Japanese collection used in NEWT, so word boundaries are particularly problematic in the Japanese collection. Consider the following example scenario. The Japanese word "ナス" meaning "eggplant" is the same as the middle of the word "バナナスムージー" meaning "banana-smoothie". Consequently, search results with NEWT are intermingled with documents that are not relevant, but were coincident with the short words in queries. This problem is further exacerbated by the fact that short substrings in Japanese queries can traverse word boundaries since there are no spaces in the index.

Conversely, substring matches in Japanese texts can also have advantages over morphological parsing. For example, suppose that document X contains the Japanese word "バナナスムージー" meaning "banana-smoothie," and document Y contains the Japanese word "ストロベリースムージー" meaning "strawberry-smoothie". Because the Japanese Katakana loan word "スムージー" meaning "smoothie" is not included in the standard dictionary for morphological analyzers, both compound words would not be segmented and become terms in an inverted index. Under this circumstance, the query term "スムージー" meaning "smoothie" finds neither document X nor document Y, because the query term is not an exact match to the terms in inverted indexes. On the other hand, in NEWT, the short query substring will correctly match any documents that contain "smoothie," even as part of a compound word. Moreover, a document for "スムージー の 材料" meaning "ingredients for smoothie" is also found in self-indexes without any analysis or parsing of the sentence being performed. However, it is not clear if these substring matches overcome the rank pollution imposed by cross-term or intra-term matching of very short substrings.

English query processing. In order to circumvent the substring problem in the English query set, we experiment with term padding, whereby each term in the query string uses additional whitespace:

1. *prefix* - add a single whitespace at the beginning of each query term.
2. *suffix* - add a single whitespace at the end of each query term.
3. *space* - add a single whitespace at the beginning and end of each query term.

We investigate each of these query expansions in conjunction with a mixture of the four different indexing strategies: stemming, nostemming, stopping, and nostopping. Each of these alternatives is discussed further in Section 4. These modifications significantly improve the effectiveness of

NEWT on the TREC collection, but the results were still lackluster for the GeoTime collection. The problem with the GeoTime query set seems to be that each topic is phrased as a question, instead of a set of keywords. For example, the GeoTime-026 topic is "Where and when did the space shuttle Columbia disaster take place?". Since many of the words in the topic description are not relevant to the query, we generated a manual set of keyword queries using only a subset of important words from each topic. So, GeoTime-026 was reduced to "space,shuttle,columbia,disaster". These keyword queries are then run with Indri and NEWT and compared explicitly.

Japanese query processing. To examine the substring problem in the Japanese query set, we evaluate the impact of morphological analysis on effectiveness, and then test four possibilities:

1. ChaSen/MeCab word segmentation on both documents and queries.
2. ChaSen/MeCab word segmentation on documents and no word segmentation on queries.
3. ChaSen/MeCab word segmentation on queries and no word segmentation in documents.
4. No word segmentation on documents or queries.

For the GeoTime collection, bag-of-words queries from the topic description and a manual set of keyword queries were prepared as described above for the English queries. For the JA-Category subset of the ClueWeb09 collection, we used the official Japanese topics in the NTCIR-9 INTENT task. Since those topics were selected from the user query log of a commercial search engine, no manual keyword selection was necessary.

4 Results

4.1 English Newswire

Text preprocessing at indexing time. A variety of text preprocessing approaches are commonly applied to raw natural language text with the aim of improving subsequent retrieval from a collection. We investigate the impact of applying stopping and stemming, at indexing time, for both the the inverted index (Indri) and self-index (NEWT) approaches. The results are shown in Table 1, with performance measured using precision at 10 documents retrieved, mean average precision, and *normalized discounted cummulative gain* (nDCG) for the TREC 7 and 8 newswire collections.

In our experiments, the best text preprocessing options for Indri are to apply stemming but not stopping; for NEWT, the best performance is achieved by applying both stemming and stopping. These correspond to rows 3 and 8 in Table 1. The difference between the results is not statistically significant (paired t-test, $p > 0.1$).

Query processing. A key potential advantage presented by self-indexing approaches is the opportunity for flexible query processing at retrieval time, without the requirement of collection preprocessing at indexing time. As explained in Section 3.2, by default NEWT will match *any* substring that corresponds to a query term. While this can be useful, it may also lead to unexpected outcomes for short query terms. We therefore compare the default NEWT method (*plain*) with three other query processing techniques: *prefix*, *suffix*, and *space* (see Section 3.2 for a full description). The results for the TREC 7 and 8 collections are shown in Table 2. Note that, as the aim is to avoid the need to commit to any particular type of preprocessing at indexing time, all results are based on the original collection (no stemming or stopping).

The Indri and NEWT *plain* runs are of roughly similar effectiveness. Adding *prefix* or *suffix* query processing increases performance slightly compared to *plain*. However, *space* query processing leads to a substantial boost, with statistically significant improvements for all three effectiveness measure for the TREC 7 data (paired t-test, $p < 0.05$), and a significant improvement for nDCG on the TREC 8 data.

4.2 GeoTime

English GeoTime. We compare the inverted index and self-index approaches on a more complex family of queries – with geographic and temporal constraints – through experiments on the NTCIR 2011 GeoTime English task. The results are shown in Table 3, as measured by mean average precision, mean reciprocal rank, and normalised discounted cumulative gain at cutoff level 10 as well as over the full ranked list.

The performance of Indri and NEWT is comparable over the *full* GeoTime 2011 queries when queries are processed using the *space* approach. Specifically, there are no statistically significant differences when using the *suffix* or *space* query processing approaches, while using the *plain* or *prefix* query processing approached leads to significantly worse performance on the nDCG measures (paired t-test, $p < 0.05$).

As the queries for the GeoTime task are very verbose and include many instruction words, we also carried out experiments with these queries when they are reduced to keywords only (see Section 3.2). Removing the instruction words leads to substantially higher performance for both the inverted index and self-index approaches

System	Preprocessing	TREC 7 P@10	TREC 7 MAP	TREC 7 nDCG	TREC 8 P@10	TREC 8 MAP	TREC 8 nDCG
Indri	nostem, nostop	0.3760	0.1570	0.3885	0.4280	0.1938	0.4398
Indri	nostem, stop	0.3760	0.1570	0.3885	0.4220	0.1917	0.4367
Indri	stem, nostop	0.4000	0.1717	0.4185	0.4520	0.2256	0.4948
Indri	stem, stop	0.3960	0.1685	0.4153	0.4420	0.2230	0.4917
NEWT	nostem, nostop	0.3740	0.1711	0.4178	0.4080	0.1881	0.4463
NEWT	nostem, stop	0.2837	0.1252	0.3445	0.2729	0.1226	0.3405
NEWT	stem, nostop	0.3700	0.1466	0.3897	0.3980	0.2000	0.4569
NEWT	stem, stop	0.4040	0.1736	0.4275	0.4340	0.2122	0.4715

Table 1: Effectiveness results for Indri and NEWT, *with preprocessing of text* at indexing time, measured by precision at 10 documents retrieved, mean average precision, and normalised discounted cumulative gain. Preprocessing variants are combinations of stemming and stopping.

System	TREC 7 P@10	TREC 7 MAP	TREC 7 nDCG	TREC 8 P@10	TREC 8 MAP	TREC 8 nDCG
Indri	0.3760	0.1570	0.3885	0.4280	0.1938	0.4398
NEWT *plain*	0.3740	0.1711	0.4178	0.4080	0.1881	0.4463
NEWT *prefix*	0.4040	0.1773	0.4222	0.4400	0.1974	0.4614∗
NEWT *suffix*	0.3920	0.1653	0.4083	0.4120	0.1954	0.4462
NEWT *space*	0.4300∗	0.1763∗∗	0.4227∗∗	0.4400	0.2093	0.4702∗

Table 2: Effectiveness results for *query processing* approaches with NEWT for the TREC 7 and TREC 8 newswire documents. No preprocessing of the collection was carried out at indexing time. NEWT query processing variants are *plain*, *space*, *prefix*, and *suffix*, as described in Section 3.2. ∗ and ∗∗ indicate statistical significance relative to the Indri run at the 0.05 and 0.01 levels respectively, based on a paired t-test.

compared to using the full queries. The relative performance between Indri and NEWT on these *keyword* queries is very similar, with no statistically significant differences between them.

Japanese GeoTime. We also investigate the impact of word segmentation on documents and queries in Japanese. For documents in the the inverted index (Indri), word segmentation was applied using the Japanese morphological analyzer ChaSen. For documents in the self-index (NEWT), no word segmentation was applied. The results are shown in Table 4, with performance measured by precision at 10 documents retrieved, mean average precision, and normalised discounted cumulative gain, for the the NTCIR 2011 GeoTime Japanese collections.

As expected, the best method for the inverted index approach is to segment both documents and queries, while for the self-indexing approach good performance is achieved with no segmentation being applied. The difference in performance between these two approaches is not statistically significant (paired t-test, $p > 0.1$).

Unsurprisingly, mixing segmentation for only one of either documents or queries leads to a drop in performance for both indexing methods, as the probability of mismatches is sharply increased.

4.3 Japanese Web Pages

We conducted experiments using the NTCIR 2011 INTENT collection. For documents in the the inverted index (Indri), word segmentation is applied using MeCab. For documents in the self-index (NEWT), no word segmentation is applied. Separate query sets were then tested with and without segmentation using MeCab. The results are shown in Table 5, with performance measured by P@10, MAP, and nDCG for the the NTCIR 2011 INTENT Japanese collections.

The best performance for the inverted index approach (Indri) is when both queries and documents are segmented; for the self-index approach (NEWT) performance is best when no segmentation is carried out at either stage. The difference between these two runs is not statistically significant for any of the measures, except for nDCG@1000 (paired t-test, $p < 0.01$).

As for the GeoTime Japanese results, mixing segmentation approaches between documents and queries harms both indexing approaches.

5 Conclusions

In this paper, we present results for our new experimental ranked self-index NEWT. We show that NEWT is capable of supporting bag-of-words queries

System	Query type	MAP	RR	nDCG@10	nDCG
Indri	full	0.1931	0.4922	0.2782	0.3635
NEWT *plain*	full	0.1519	0.2481	0.1687∗	0.2430∗∗
NEWT *prefix*	full	0.1591	0.3411	0.1937∗	0.2622∗
NEWT *suffix*	full	0.1666	0.4110	0.2043	0.2919
NEWT *space*	full	0.1847	0.4742	0.2625	0.3348
Indri	keyword	0.2846	0.6950	0.4159	0.5344
NEWT *plain*	keyword	0.2541	0.6448	0.3572	0.4817
NEWT *prefix*	keyword	0.2689	0.6568	0.3892	0.5029
NEWT *suffix*	keyword	0.2737	0.6953	0.4105	0.5175
NEWT *space*	keyword	0.2822	0.7131	0.4133	0.5235

Table 3: Effectiveness results for Indri and NEWT, with *no text preprocessing*, for the GeoTime'11 collection for English topics. NEWT query processing variants are *plain*, *space*, *prefix*, and *suffix*, as described in Section 3.2. ∗ and ∗∗ indicate statistical significance relative to the Indri run at the 0.05 and 0.01 levels respectively, based on a paired t-test.

System	Query type	Document	Query	MAP	RR	nDCG@10	nDCG
Indri	full	segmented	segmented	0.3229	0.6056	0.3731	0.5144
NEWT	full	no segmentation	segmented	0.0447	0.1256	0.0656	0.1093
Indri	keyword	segmented	no segmented	0.2568	0.5163	0.3181	0.4261
NEWT	keyword	no segmentation	no segmentation	0.3535	0.6869	0.3851	0.5633

Table 4: Effectiveness results for Indri and NEWT, for the Japanese GeoTime'11 collection.

with effectiveness similar to traditional inverted indexing methods, and present the first evaluation of top-k ranked retrieval in large document collections using self-indexes. We then explore potential solutions to an unexpected substring problem inherent to all character-based indexing algorithms.

This work is an important first step in constructing an entirely new class of indexing algorithms and search engines that are capable of delaying linguistic processing until query time. By minimizing processing at index time, multiple text formats and languages can be supported simultaneously. Our indexes implicitly retain the original substring ordering, so new ranking methods that exploit term proximity or linguistic syntax can also be devised in future work.

6 Acknowledgments

The first author was supported by the Australian Research Council.

References

J. E. Burnett, D. Cooper, M. F. Lynch, P. Willett, and M. Wycherley. Document retrieval experiments using indexing vocabularies of varying size. I. Variety generation symbols assigned to the fronts of index terms. *Journal of Documentation*, 35(3):197–206, September 1979.

S. Büttcher, C. L. A. Clarke, and G. V. Cormack. *Information Retrieval: Implementing and evaluating search engines.* MIT Press, Cambridge, Massachusetts, 2010.

W. B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in practice.* Addison Wesley, Boston, 2010.

J. S. Culpepper, G. Navarro, S. J. Puglisi, and A. Turpin. Top-k ranked document search in general text databases. In *ESA, Part II*, volume 6347 of *LNCS*, pages 194–205. Springer, 2010.

R. Dale, H. Moisl, and H. Somers. *Handbook of Natural Language Processing.* CRC Press, New York, NY, 2000.

T. de Heer. The application of the concept of homeosemy to natural language information retrieval. *Information Processing & Management*, 18(5):229–236, 1982.

P. Ferragina and G. Manzini. Opportunistic data structures with applications. In *FOCS*, pages 390–398. IEEE, November 2000.

W. B. Frakes. Term conflation for information retrieval. In *SIGIR*, pages 383–389, August 1984.

F. Gey, R. Larson, J. Machado, and M. Yoshioka. NTCIR9-GeoTime overview - evaluating geographic and temporal search: Round 2. In *Proceedings of NTCIR-9 Workshop Meeting (in printing)*, December 2011.

R. Grossi, A. Gupta, and J. S. Vitter. Higher-order entropy-compressed text indexes. In *SODA*, pages 841–850, January 2003.

System	Document	Query	MAP	RR	nDCG@10	nDCG
Indri	segmented	segmented	0.3413	0.9255	0.3553	0.5298
Indri	segmented	no segmentation	0.1550	0.3924	0.1509	0.2169
NEWT	no segmentation	segmented	0.3044	0.8251	0.2629	0.4433
NEWT	no segmentation	no segmentation	0.3651	0.9675	0.3182	0.4638

Table 5: Effectiveness results for Indri and NEWT, for the Japanese NTCIR 2011 INTENT collection.

D. Harman. How effective is suffixing? *Journal of the American Society for Information Science*, 42(1): 7–15, 1991.

V. Hollink, J. Kamps, C. Monz, and M. de Rijke. Monolingual document retieval for European languages. *Information Retrieval*, 7(1-2):33–52, 2004.

W.-K. Hon, R. Shah, and J. S. Vitter. Space-efficient framework for top-k string retrieval problems. In *FOCS*, pages 713–722. IEEE, October 2009.

W.-K. Hon, R. Shah, and J. S. Vitter. Compression, indexing, and retrieval for massive string data. In *CPM*, volume 6129 of *LNCS*, pages 260–274. Springer, June 2010.

D. A. Hull. Stemming algorithms: A case study for detailed evaluation. *Journal of the American Society for Information Science*, 47(1):70–84, 1996.

R. Krovetz. Vierwing morphology as an inference process. In *SIGIR*, pages 191–202, June 1993.

K. L. Kwok. Comparing representations in Chinese information retrieval. In *SIGIR 1997*, pages 34–41. ACM Press, August 1997.

J. B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11(1–2):22–31, 1968.

R. W. P. Luk and K. L. Kwok. A comparison of Chinese document indexing strategies and retrieval models. *ACM Transactions on Asian Language Information Processing*, 1(3):225–268, 2002.

C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.

P. McNamee and J. Mayfield. Character n-gram tokenization for European language text retrieval. *Information Retrieval*, 7(1–2):73–97, 2004.

P. McNamee, C. Nicholas, and J. Mayfield. Don't have a stemmer? Be un+concern+ed. In *SIGIR*, pages 813–814. ACM Press, July 2008.

S. Mithukrishnan. Efficient algorithms for document retrieval problems. In *SODA*, pages 657–666, January 2002.

G. Navarro and V. Mäkinen. Compressed full-text indexes. *ACM Computing Surveys*, 39(1):2–1 – 2–61, 2007.

J.-Y. Nie, K. Gao, J. Zhang, and M. Zhou. On the use of words and n-grams for Chinese information retrieval. In *IRAL 2000*, pages 141–148. ACM Press, 2000.

C. D. Paice. Another stemmer. *ACM SIGIR Forum*, 24(3):56–61, 1990.

M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.

S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *TREC-3*, 1994.

R. Song, M. Zhang, T. Sakai, M. Kato, Y. Liu, M. Sugimoto, Q. Wang, and N. Orii. Overview of the NTCIR-9 INTENT tasks. In *Proceedings of NTCIR-9 Workshop Meeting (in printing)*, December 2011.

N. Välimäki and V. Mäkinen. Space-efficient algorithms for document retrieval. In *CPM*, volume 4580 of *LNCS*, pages 205–215. Springer, July 2007.

Ellen M. Voorhees and Donna K. Harman. Overview of the Eighth Text REtrieval Conference (TREC-8). In *TREC-8*, pages 1–24, 1999.

P. Willett. Document retrieval experiments using indexing vocabularies of varying size. II. Hashing, truncation, digram and trigram encoding of index terms. *Journal of Documentation*, 35(4):296–305, December 1979.

I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, San Francisco, second edition, 1999.

M. Yasukawa and H. Yokoo. Composition and decomposition of Japanese katakana and kanji morphemes for decision rule induction from patent documents. In *ADCS*, pages 28–35, December 2010.

J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Computing Surveys*, 38(2):6–1 – 6–56, 2006.

Mobile Applications of Focused Link Discovery

Darren Wei-Che Huang
Shlomo Geva, Kelly Y. Itakura
Queensland University of Technology
Brisbane, Australia
{s.geva,kelly.itakura,w2.huang}@qut.edu.au

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

Abstract *Interaction with a mobile device remains difficult due to inherent physical limitations. This difficulty is particularly evident for search, which requires typing. We extend the One-Search-Only search paradigm by adding a novel link-browsing scheme built on top of automatic link discovery. A prototype was built for iPhone and tested with 12 subjects. A post-use interview survey suggests that the extended paradigm improves the mobile information seeking experience.*

Keywords: Focused Link Discovery, Wikipedia, Mobile Information Seeking, User Studies Involving Documents.

1. Introduction

1.1. Background and Motivation

The Internet can now be accessed through mobile devices. Services such as email, web search, blogs, and instant messaging are all available on smart phones. Marketing campaigns by communications companies have persuaded us that we should access information from wherever we are, and we now can, and do. Because of this the nature of mobile information seeking has changed. However, search and navigation on a small mobile device remains awkward [11], especially as many service providers have not yet adapted to the small screen of a mobile device.

Recently there has been a substantial improvement in mobile provision by popular providers such as *Facebook*, *Skype*, *Twitter*, *Wikipedia*, *Windows Live Messenger*, and the search engines. They each have mobile versions with enhanced interaction. It is not, for example, uncommon to see users using mobile maps to look for shops, petrol stations, and restaurants, or social software to check for contacts. But, problematically, the screen on a smart phone is small.

Focused link discovery research has examined the automatic identification of new anchor texts in an existing document for linking to a *best entry point* (BEP) from where a user should start reading in a target document. This is an example of how existing technology can be adapted to facilitate better mobile search on a small screen[1].

This paper discusses a novel mobile knowledge seeking application that uses link discovery to alleviate the small screen search problem. To navigate via links, a four-page link browsing scheme is proposed. The first of these four pages is the document view displayed exactly as usual, but with additional hypertext links optionally added. The second displays the article's anchor texts. The third displays details of target documents. The final page is a history view. New links are identified using previously published link discovery algorithms already shown to be good on the Wikipedia (our test collection) [8].

We tested our prototype application with 12 participants and present our findings: that the four page scheme with link discovery enhances the information seeking experience.

1.2. Related Work

Arvola et al. [2] proposed a framework for evaluating passage and XML retrieval search engines. They used a mobile browsing scenario to simulate how the relevant content should be extracted and localized (to reduce the effort of reading on a small screen).

They suggested interaction followed two phases: the fetch phase, in which the user enters a query into a relevance ranking search engine; and the browse phase which presents a set of non-overlapping passages in a retrieved document.

In their interface, the effort of finding relevant content within a document is decreased because they insert links to the next relevant passage at each relevant passage. In this way the mobile user is directed to the relevant parts of a document rather than the beginning of the document. This framework removed the first vertical scroll to the first relevant passage, but also preserves the original document order and structure. The user can also determine if there is any further relevant information in the document without scrolling to the bottom.

Their motivation was the design of an evaluation framework (with metrics) for focused retrieval. They do not investigate how focused retrieval helps to alleviate mobile search difficulties.

The concept of fetching relevant documents and browsing relevant passages proposed herein is similar to that of Arvola et al. but we improve mobile information seeking using focused link discovery. This,

Proceedings of the 16th Australasian Document Computing Symposium, Canberra, Australia, 2 December 2011.

coupled with our four-page browsing scheme, makes it possible for the user to browse related content without performing additional searches – without typing new queries on a small mobile keyboard or screen.

Mobile search is still an unsolved problem. In 2010 Steve Jobs stated that "On mobile, search hasn't happened. People aren't searching on their phones." [3].

While Jobs raises questions about the utility of mobile search, Microsoft, Google, and Yahoo! all see the value of mobile search services across multiple devices. They optimized their search interfaces to include localised search, addresses, and maps. Users now expect information to be a single click away from an interactive mobile interface. Making it so reduces the effort involved in search and increases access to information. The shift to mobile has altered the way users think about, and use, search.

We use the Wikipedia as a sample document collection in our experiments. Several existing mobile Wikipedia applications already exist. For example, there is a mobile version of the English Wikipedia supporting different mobile devices (iPhone, Android, Palm WebOS, Opera Mini and NetFront). There are also third-party applications including: *Wapedia*, *Sevenal*, *iWiki*, *Boopsie*, *WikiPock* and *HAWHAW*.

Our application is novel because it emphasizes two aspects. The first is the automatic identification of new links that are inserted from displayed documents to Wikipedia documents. The second is a new way to interact with those links.

Mobile applications that augment the information on the device have already been proposed. Phone-derived GPS coordinates are an essential part of many augmented reality applications [6,7]. Our application augments text documents by inserting new links.

2. Focused Link Discovery

Instead of requiring the user to enter queries on a tiny keyboard, we re-cast search as browse. To do so it is necessary to augment documents by adding new links that point to related content.

The branch of IR known as Automatic Link Discovery studies the automatic identification and maintenance of hypertext links in a collection of documents. Successful algorithms eliminate the human effort required to build a highly accurate link-graph, and to keep the graph up-to-date[1].

Going beyond ordinary document level link discovery, Focused Automatic Link Discovery identifies where, within the target document, the user should start reading. This location is known as the best entry point, or BEP. BEP links are more versatile than whole-document links (especially for a mobile device) because the user does not need to scroll (or skim read) through irrelevant content in order to find relevant content. It is reasonable to expect focused automatic link discovery to be beneficial in reducing user effort when finding information while on the move.

Central to this research is the evaluation of these automatically identified links. The INEX link-the-wiki track offers a platform for researchers to collaboratively assess link discovery systems [9]. In addition to having examined document-to-document links (which began in 2007), the evaluation examined anchor-to-BEP links in 2008 and 2009.

3. Conceptual Framework

In order to demonstrate the applicability of focused automatic link discovery to a mobile environment we designed and built a simulation of an iPhone application. The conceptual flow of the application is illustrated in Figure 1. When an article is loaded into the system, the text is parsed by the text scanner. The anchor indicator then identifies potential anchor terms and phrases, which are then filtered by the anchor filter. The filtering process, for example, resolves potentially overlapping anchors such as "apple", "computer", and "apple computer". This is a form of contextual disambiguation.

The link detector identifies candidate target documents for the anchors and produces link candidates. The candidates are then selected and filtered by the filtering component which ranks candidate links, categorizes relevant XML elements and documents, and prepares a summary of each target document.

4. Mobile Information Seeking

4.1. Mobile User Interface

This section describes our user interface for small mobile screens. Our application starts with the *One Search Only* screen [10]. This screen, also known as the *Home* screen is shown in Figure 2 (left).

There are three ways to start the information seeking process. The first is to enter a query into the search box (depicted, left). The second is to enter a URL. The third is to select a bookmark. The *Transfer* button (top right) is used to apply focused automatic link discovery to the currently displayed document.

Figure 2 (middle) shows a Wikipedia infobox while the table of contents with links is displayed in Figure 2 (right) for quick access to document subsections. New links in the infobox have been identified by our focused automatic link discovery system. Since the anchors on a small screen are themselves small, they are difficult to click. One of our aims is to reduce this difficulty; we discuss our solution in the next section.

4.2. Mobile Link Browsing Scheme

Our goal is to simplify the operation of information seeking on mobile devices. Link discovery identifies related content while a simple navigation process allows the user to browse through that content. Our interface is simple and intuitive and alleviates the diffi-

[1] Link discovery algorithms are beyond the score of this paper, the interested reader is referred to the INEX track overviews [8].

culties associated with the more common query-typing, result list reading, and document skim-reading iterative process. An initial search is necessary, but it is expected to be the only search consequently it is a *One Search Only* system [10].

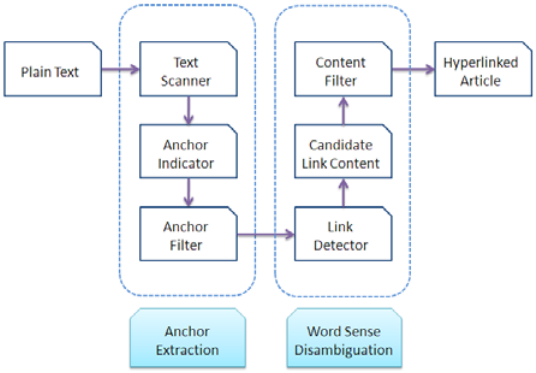


Figure 1. Conceptual flow of mobile link discovery

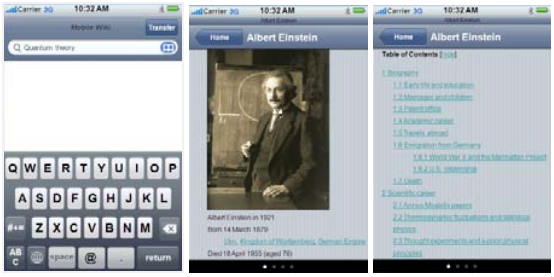


Figure 2. Home and Article content screens

There are two possible responses to the one search as shown in Figure 3: either a relevance ranked list of documents (case 1); or if there is only one result, then that document is displayed (case 2). In case 1, the user can scroll up and down the results list and choose a document to view. That document is displayed on screen with its original links shown. By clicking the *transfer* button new links are added. For case 2, the article and links are immediately displayed on screen.

Figure 4 shows the conceptual flow of the link-browsing scheme. There are 4 screens. Articles can be displayed in 3 ways: (1) the article content; (2) the article's anchor terms; (3) and the article's links. And there is also a history screen (4). The user switches between them using the iPhone *swipe* gesture.

In the first screen, the article content screen, the article is not shown from the beginning, but rather the screen it automatically scrolled to the most relevant content (the BEP[2]). This happens regardless of whether the article is displayed as a consequence of a search or a link-click. Because of this the user can immediately read the relevant portion of the articles without ever scrolling.

The second screen displays the set of anchor-texts. These can be listed in either document order or alphabetical order.

[2] Algorithms for computing the BEP are beyond the scope of this contribution. The interested reader is referred to the proceedings of INEX between 2007 and 2010

The third screen displays the titles and a summary of the target documents that are linked-to by the current article. Each summary, prepared by the Content Filter, can be hidden or displayed by clicking an icon on the right hand side of each link.

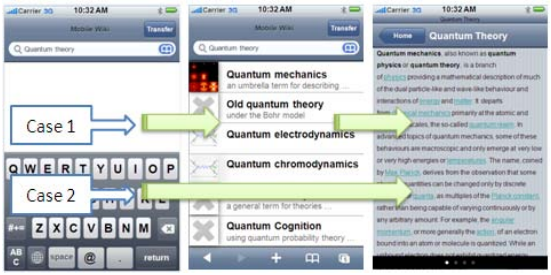


Figure 3. Search screen for mobile information seeking

The fourth screen shows the browsing history, and is a list of all the articles the user has viewed. This history list can be used to return to any document the user has viewed in the current session.

On the article screen and the anchor terms screen, the anchor to target relationship is many-to-many. That is, many anchors may target the same document, but also each anchor may target multiple documents. To help navigate this we introduce the *anchor-targets* page.

The bottom left of Figure 4 shows an example *anchor-targets* page. A list of document titles and summaries is given allowing the user to choose which target they prefer.

Several navigation features are common to many of the screens. There is *Back* button to return to the previous screen. The round icon on the *anchor terms*, *anchor links* and *anchor-targets* page indicates whether the document has previously been viewed or not. At the top of each navigation-page there is a *Home* button that returns to the search page.

5. Evaluation

We implemented the simulation of the iPhone application using Xcode and evaluated it using 12 volunteers (Nielsen & Landauer [12] suggest 5 users sufficient to find most usability problems). Our goal was to test the user experience using the 4-page interface, The links themselves were pre-discovered offline using the algorithm of Huang et al. [8]. The user testing experimental software did not actually discover links in real-time, but rather restricted users to a set of pre-linked pages and one level deep link following. The interface however was not restricted in design and recursive browsing can be implemented in a straight forward manner. None of the volunteers had used this mobile link discovery application previously. Half had an IT degree while the others had a non-IT background. Each was asked to experiment and to become familiar with the tool.

We were specifically seeking qualitative information on how the small screen affects the operation of

the mobile link browsing system (and therefore how to improve it) before undertaking a full implementation.

A questionnaire was prepared and an interview survey was conducted. The questions included prior experience of existing mobile search applications, the experience of using mobile version of Wikipedia, the use of the link browsing scheme, the experience of using the four-layer navigation process, how the link browsing scheme improved mobile information seeking, which layers were used most and most useful, and the pros and cons of these four layers.

Figure 4. Conceptual flow of the link browsing scheme

Table 1. Usability survey results

Questions	Strong	Average	Weak
Current Mobile Search			
How often do you use it?	5	2	5
How are they?		7	5
Mobile Wikipedia			
How often do you use it?	1	4	7
Is it useful for browsing information?	2	8	2
4-Layers Link Browsing Scheme			
Easy to understand and use?	9	2	1
Is it useful for mobile information seeking?	9	2	1
Article Content Layer – positive	10	2	
Anchor Terms Layer – positive	10	2	
Anchor Links Layer – positive	11		1
History Layer – positive	11		1

Table 1 summarises the results. Most of our subjects were aware of specialised mobile search applications but many only used these for getting general information (e.g. bus/movie timetable, weather, maps and news). Most did not directly use mobile Wikipedia except when linked-to by sources such as Google.

There was strong support for the four-layer navigation and so it is reasonable to conclude that it is an effective method of interaction. The *article content* screen and the *anchor terms* screen were viewed positively. Through informal discussion participants indicated that it was easier and quicker to find relevant anchors in the list of anchor terms than in the *article content* screen.

Participants indicated that multiple links per anchor was better than one-to-one linking. This is of particular note because it is not the normal state of affairs for the web. Multiple targets per anchor could be implemented in JavaScript and might be of particular value to information sources such as the Wikipedia – however it is not seen there.

The *anchor links* and *history* screens were the layers most frequently used.

These results suggest that the link browsing scheme is practicable for mobile users, and an improvement on conventional mobile search.

6. Conclusions and Future Work

Using a mobile phone to search the Internet is awkward because the screen is small and typing is difficult. In order to make it easier to find information using these devices we suggested turning search into browse.

If related documents are extensively interlinked then the quantity of queries needed in order to satisfy an information need could be reduced. The user would be able to click rather than forced to search. But documents are not extensively linked and it is highly unlikely that authors will change them to be so.

To address this we suggested the use of automatic link discovery technology. This technology adds links between documents by identifying anchor text in one and a related best entry point in another.

But clicking links on a small screen is also difficult. To address this we introduced a novel multi-screen link browsing scheme.

To test our theories we designed and implemented a simulation for an iPhone application with the two technologies: focused automatic link discovery; and a four-screen link browsing scheme. The four-screens were the article view, the anchor terms; the article's links, and the history screen. Navigation between screens was with the iPhone *swipe* gesture. We note that link discovery algorithms are beyond the scope of this paper and so we use a previously published algorithm previously shown to be effective.

The application was tested with 12 subjects of varying backgrounds. A post use survey was conducted.

The survey results show that the link browsing scheme is easy to understand and easy to use. Most of the subjects believed the link browsing scheme was useful for information seeking in a mobile environment. It is conceptually simple and practical, suggesting that further research into the *One Search Only* paradigm is warranted.

But, the experiment we conducted was with only a small user base (of 12) and consequently can only be considered as a proof of concept. A substantially larger number of participants are necessary before any firm conclusions can be drawn.

Our application functions in one of two possible modes. The first is as a mobile (link-enhanced) version of the Wikipedia. The second is as an application that can add links from any page to the Wikipedia (similar to the Wikify at the University of Waikato).

We believe this kind of service will be invaluable in the future as other Wikipedia like resources become available (such as dictionaries).

In future work we will examine whether or not our interface is generally useful or just for the Wikipedia. We will do this by turning our prototype iPhone application into a fully functional application and rolling-out to a larger user base.

References

[1] Huang, W. C., Xu, Y., Trotman, A, and Geva, S. Mobile Link Browsing: Efficient Interaction with Mobile Contents, In Joint HCSNet-EII Workshop 2008, Australia.

[2] Arvola, P., Kekalainen, J. and Junkkari, M. Expected reading effort in focused retrieval evaluation, Information Retrieval: Focused Retrieval and Result Aggregation, 13(5), 2010, 460-484.

[3] Subramanian, K. Clash of the Titans: The Battle to Become the Mobile Search Leader, June 2010, http://techcrunch .com/2010/06/27/mobile-search-clash/.

[4] eMarketer The Search Wars Are Going Mobile, 2007, http://www.emarketer.com/Article.aspx?R=1005083.

[5] Wikipedia Help: Mobile Access, 2010, http://en.wikipedia. org/wiki/Help:Mobile_access.

[6] Jones, M., Buchanan, G. and Thimbleby, H. Sorting Out Searching on Small Screen Devices, *In Proceedings of the 4th International Symposium on Mobile HCI*, Springer, 2002, 81-94.

[7] Sweeney, S., Crestani, F. Supporting Searching on Small Screen Devices Using Summarization, *Lecture Notes in Computer Science*, Udine, Italy, 2954. Springer, Berlin, 187-201.

[8] Huang, W.C., Trotman, A, and Geva, S. Overview of INEX 2009 Link the Wiki Track, Lecture Notes In Computer Science.

[9] Huang, W. C., Trotman, A, and Geva, S. The Methodology of Manual Assessment in the Evaluation of Link Discovery, In Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

[10] Huang, W. C., Trotman, A., and Geva, S. (2008) Efficient Information Seeking on a Mobile Device, Proceedings of SIGIR 2008 Workshop on Mobile Information Retrieval (MobIR'08)

[11] Dunlop, M., and Brewster, S., (2002). The Challenge of Mobile Devices for Human Computer Interaction. Personal Ubiquitous Comput. 6:4(235-236)

[12] Nielsen, J., and Landauer, T. K. (1993) A mathematical model of the finding of usability problems. In Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems (CHI '93), pp 206-213

A Workflow for Document Level Interoperability

David Bainbridge
University of Waikato
Hillcrest Road
Hamilton, NZ
davidb@cs.waikato.ac.nz

Sally Jo Cunninghom
University of Waikato
Hillcrest Road
Hamilton, NZ
sallyjo@cs.waikato.ac.nz

Abstract *This article describes a software environment called the Exchange Center that helps digital librarians manage the workflow of sourcing documents and metadata from various repositories. The software is built on Greenstone but does not require its use as the final digital library server. After describing the software architecture we provide two scenarios of its use: a private library of recipes, which ultimately involves collaboration with other cooks; and a digital library that aggregates the collections of various host institutions that use different repository software.*

Keywords Digital Library Interoperability, Software Architecture, Workflow

1 Introduction

The term Digital Library (DL) can mean different things to different people. Even within the profession there are a variety of different definitions [15]. This has lead to a sizable—some might say dizzying—array of software solutions for libraries (archives, and others) to choose from, as a gamut of comparison articles attest to, e.g., [7, 6, 3]. Recurring in these articles are the themes of importing, exporting, and interoperability; yet, surprisingly these are areas that are (on the whole) under-represented in the software solutions available.

Building on our experiences with Greenstone [14], this article describes a software environment that we call the Exchange Center, designed to help digital librarians manage the workflow of documents and metadata in forming various types of digital repositories. Tasks include sourcing from external digital libraries, adding local content where necessary, and manipulating the resulting collection into formats that are suitable for importing into other digital library repositories. Such work involves interoperability at a practical level, reflecting the real need of practicing digital librarians to span different standards and also work within different digital library software applications.

This paper develops the work previously reported as an extended abstract [1], and is structured as follows. First we provide an overview of pertinent features of the existing Greenstone [15] infrastructure that supports the Exchange Center. Then we describe the graphical interoperability environment. Key features are the import and export stages, which we describe in detail from the user's point of view. Several protocols are supported for importing from external repositories: HTTP, OAI-PMH [9], Z39.50 [10], and SRU [12]. Export options include DSpace [11], and the Fedora [8] variant of METS [4]; XSL Transforms [13] are used wherever possible to aid extensibility. The paper concludes by describing two scenarios that illustrate how the Exchange Center can be put to use, along with a discussion of issues that arise.

Proceedings of the 16th Australasian Document Computing Symposium, Canberra, Australia, 2 December 2011.

2 Software infrastructure

Greenstone is intended to be a flexible software architecture for digital library construction and delivery. Pertinent to the present project is its importing component. This can stand alone—it does not have to result in a Greenstone collection—and includes a system of plugins for processing documents and metadata that is both extensive and extensible. There are plugins for literally dozens of widely-used formats, some of which we draw upon in the scenarios below. The import process combines documents and metadata into the METS format (Greenstone profile).[1] Collections can be exported in other formats, for example DSpace (ready for the DSpace batch import program) [17]. The software includes an OAI server and a Z39.50 server, which are alternative ways of getting collection information out of Greenstone, as well as Greenstone's native Reader's Interface. See *www.greenstone.org* for further details of this software.

On top of this infrastructure is the Greenstone Librarian Interface (GLI) [16]. This is an interactive application that provides a graphical environment for digital librarians and others to gather, enrich, design, and create collections. Written in Java, in its standard deployment it performs its work using the local file network: this is where it finds the documents and where it builds the collection. GLI was originally designed

[1] Historically, because Greenstone predates METS, a canonical XML file format known as the Greenstone Archive format is used by default, but METS can be specified instead using a switch on the import program.

specifically to create Greenstone collections from existing documents and metadata. However, this paper focuses on how we have adapted it to be used to assemble information for other digital library systems too. Since it was designed with flexibility in mind, it is not difficult to re-target it to fulfill this new role of a document and metadata exchange center.

Figure 1 shows the re-purposed graphical environment, which we call the Exchange Center. Along the top are *Download*, *Gather* and *Enrich* tabs. *Gather* refers to the act of gathering together documents and metadata into a digital library collection. It involves dragging individual files, folders, or entire file hierarchies from a file browser and dropping them into the nascent collection. *Download* invokes utilities that bring in documents or metadata over various standard protocols; this is a precursor to the gathering stage for collections in which the user must reach out to other digital libraries—or to the web at large—to access their content. The downloaded files are placed in a cache where they can be examined and dragged into the collection. Finally, *Enrich* is used to add metadata manually to documents in the collection. If there are existing metadata files, they can be dragged into the collection just as documents are. However, users often need to add metadata to documents interactively. GLI is agnostic towards the actual metadata set (or sets) deployed, and facilities exist to define new ones if desired. Exporting the collection in different forms is achieved through an *Export* item on the *File* menu.

The full Greenstone Librarian Interface, in addition to these three tabs, contains two others that take the user through the design and building process for a Greenstone collection. These are not used in the Exchange Center, and consequently are suppressed.

3 Importing from external repositories

This section details how we augmented the existing GLI capabilities to reach out to external digital libraries and repositories. The implementation of this was split into two stages: core infrastructure and graphical environment. The first stage addresses the actual remote information gathering process, while the second stage looks at adding this functionality to the user interface in an extensible way. We start with the latter to convey the end result of how things pan out from a user's point of view, before detailing how the infrastructure supports this.

3.1 Graphical environment

Figure 1 shows the download panel in use. At the upper left is the operation area. This presents a list of protocols that can be used: currently Web, OAI, Z39.50, and SRU. In this case OAI is selected. At the upper right is the configuration area, in which various settings relevant to the selected protocol are available. Here the user has entered the name of an OAI server, *http://memory.loc.gov/cgi-bin/oai2_0*, decided to download a maximum of 200 metadata records from the set *cwp* (Civil War Photographs), and pressed the *Download* button. In the lower portion of the panel is the download area. Here a progress bar has appeared (the lower one), providing feedback about how the download operation is proceeding. The upper progress bar shows the result of importing metadata over the SRU protocol, which was invoked earlier and is now complete.

The **Configuration area** contains four download methods for users to select and a set of configuration options that change depending on which method is selected.

For downloading over the web we use a standard web-mirroring program, which is controlled through a form-based panel. There are two arguments, *url* and *depth*. The first must be specified; it gives the web location from which pages will be downloaded. The second is optional, and specifies how many levels of information are to be obtained. For example, if *url* points to a particular home page and *depth* is set to 0, the system will only download that page. However, if *depth* is set to 1, then the system will download the home page as well as all the pages within that site that the home page links to.

We have already examined the OAI mode, which is selected in Figure 1. As well as the (compulsory) *url* argument and the (optional) *set* and *max_records* arguments, which have already been seen in use, there is a further argument, *get_doc*. This determines whether the user wishes to download the source documents along with the OAI records. Although OAI-PMH is a metadata-only standard, many users follow the informal convention that the *Identifier* field, if present, gives the URL of the document to which that metadata refers. From the point of view of the destination digital library, a useful source for full-text indexing. Setting the *get_doc* flag causes them to be downloaded also and bound to the metadata.

Settings for the Z39.50 mode are slightly more complex. There are four compulsory arguments and one optional one. The *host* and *port* arguments are used to specify the location of a Z39.50 server. The server might contain more than one database, and the *database* argument specifies which one to access. The *find* argument allows users to input the keywords they wish to search for and also supports advanced query syntax. Finally, users can enable the optional argument *max_records* to specify the maximum number of records to download. SRU follows a similar route. It has exactly the same options, with the server interchange realized through the same underlying open-source utility, YAZ (see *www.indexdata.dk* for more information).

The **Operation area** lies directly beneath the Configuration area in Figure 1 and contains three buttons. *Download* starts the actual download operation according to the setup in the configuration area, and adds a

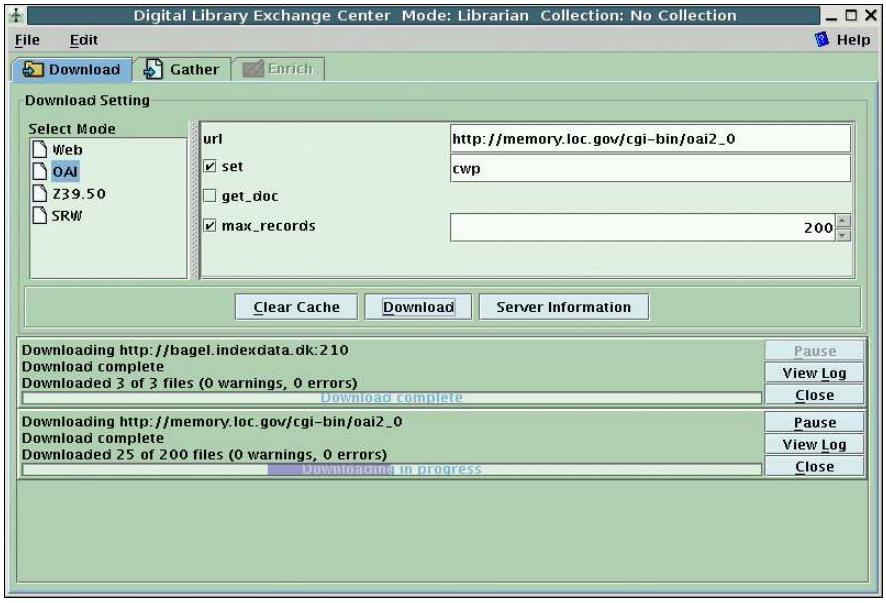


Figure 1: Exchange Center interface – Download Panel.

new job to the progress area. The files are placed in a folder called *Downloaded Files* which is present in the Gather panel, allowing users access to them. This panel is shown in Figure 2. As mentioned previously, the user drags files into the collection. The file system is presented on the left-hand side. At the top level is the user's local file space and home folder (second and third entries), documents in existing Greenstone collections (first entry), and downloaded files (fourth entry), which is shown expanded in Figure 2.

Returning to Figure 1, the *Clear Cache* button deletes all the information in the *Downloaded Files* folder. Finally, the *Server Information* button provides some basic information about the specified information source, which is helpful in targeting a particular set on an OAI server, for example, or determining the databases that exist on a Z39.50 server.

The **Download progress area** shows all the download jobs that have been issued so far. They are stored in a process queue, and each one is processed in turn. For each job there is an associated information bar, beside which are three buttons. The information bar displays the status and progress of the current download job.

Downloading proceeds asynchronously with the rest of the interface. While the application is busy working through the download queue importing data files, the user is free to interact with the other tabs in the interface. For example, he or she can begin to gather files into the collection for processing by the Exchange Center, or enrich the metadata of files that have already been included.

Three buttons are visible in Figure 1: *Pause*, *View Log* and *Close*. The first temporarily stops the download process, and yields priority for downloading to the next download job. *View Log* displays the log information for the current download operation, which gives a detailed account of progress. *Close* deletes the current download job and removes it permanently from the panel. Any files that have already been downloaded will remain in the *Downloaded Files* folder.

3.2 System Description

This interface relies on Greenstone's core infrastructure to perform the actual download operations. Figure 3 shows a flow diagram for the *Download* panel along with the corresponding core infrastructure scripts for supporting the download related processes. Once the Download panel is initiated by users, the system defaults to Web download mode. The arguments are obtained by using the command-line script *downloadinfo.pl* described below, and the system will use the obtained information to generate the GUI components in the configuration options area of the Download panel. This technique is used to aid extensibility.

Once a mode is selected, users need to fill out the compulsory options in order to retrieve the server information or download the actual documents. By pressing the "Server Information" button, the system retrieves some basic information from the information source. This process is supported by the core infrastructure *downloadfrom.pl* script with the "-info" flag. By pressing the "Download" button, the system starts the download proper (by running *downloadfrom.pl* without the "-info" option).

3.3 Core infrastructure

The Greenstone building infrastructure consists of a set of utilities, written in Perl, for building, unbuilding,

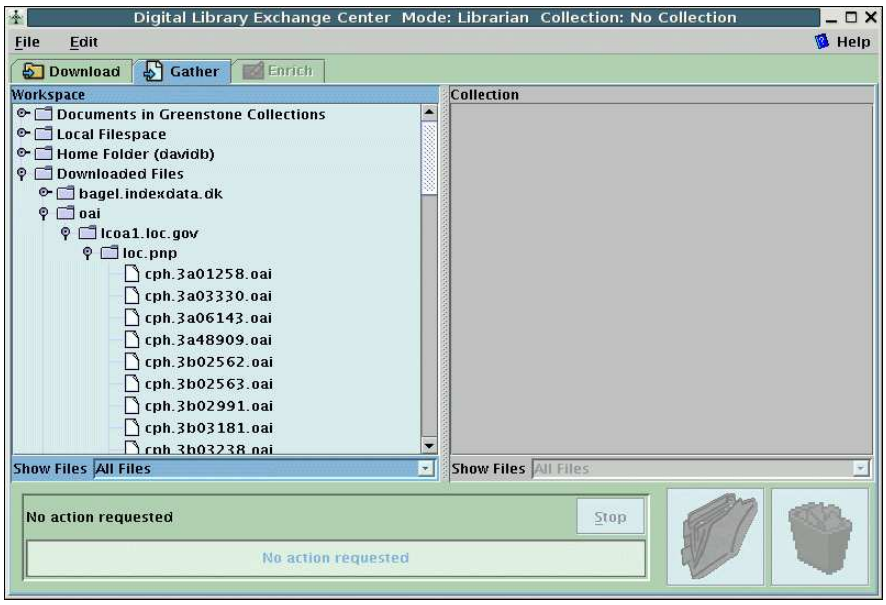


Figure 2: Gather Panel – downloaded files.

creating CD-ROM images, and the like. An existing command for downloading XML records from an OAI server already existed and this was generalized to become *downloadfrom.pl*.

Much of the work for *downloadfrom.pl* resides in a set of Perl modules. These modules use an inheritance hierarchy to abstract to the core what it means to contact and download documents and metadata from other digital libraries, thereby paving the way for further classes of protocol, such as that used by Fedora to be added. We return to this point in the conclusion.

4 Exporting for external digital libraries

For the export ability of the Exchange Center development, modest upgrading of the existing software was undertaken. Figure 4 shows the process of exporting a collection, through the graphical environment, to the Greenstone profile of the METS. The dialog is initiated by the *Export* option of the *File* menu. In the resulting popup window, users can select the format—currently a choice between DSpace and METS—and the collections they wish to export. Optionally, the user can stipulate an XSLT file that is applied to the generated file. This opens up the possibility of generating file formats other than the two explicitly mentioned. For instance, an XSLT of modest complexity can transform the Greenstone METS format into Fedora METS.

Compared with the plugin architecture that is the powerhouse of the importing process, the exporting infrastructure is less developed. Figure 5 shows the process in schematic form. At its heart, the internal document model—generated as a consequence of importing—is traversed by a "back-end" that generates the necessary files. The *docsave.pm* module is the initiation point for this. There are different back-ends for the different formats supported. This means in principle a new traversal routine must be added for each new export format supported. The ability to apply an XSLT softens this requirement slightly, however, and provides some degree of extensibility. While a back-end for DSpace had already been written, as a proof of concept an XSLT was written to demonstrate how it could be done without modifying Greenstone's core Perl infrastructure, mapping the Greenstone METS profile into that used by DSpace. These back-end modules are implemented as a system of "plugouts"—a complementary process to plugins, that provides a rich and versatile way of exporting collection content.

5 Scenarios

To illustrate different uses of the Exchange Center we describe two scenarios.

5.1 Web to Fedora

George is a enrolled in a Masters of Library Information Science programme, currently taking a course in information management. For one assignment he must organize a personal collection using one of the software digital library packages they have studied in the course. Since George loves to cook, he chooses this pastime as the basis for his assignment.

When George cooks, he uses a variety of resources: cookbooks, recipes he has jotted down on scraps of paper and compiled into a scrapbook, and recipes that he finds and normally bookmarks from the web. For this assignment he decides to organize all this content using Fedora as the document repository.

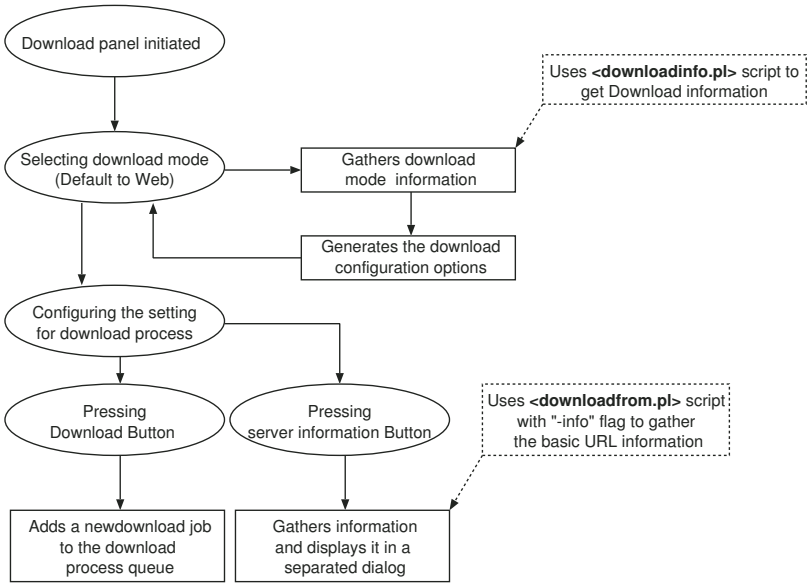


Figure 3: System flow diagram for Download Panel.

He begins by digitizing the pages of cookbooks that contain recipes he has used at least once. Sometimes he expands his policy and digitizes a popular section entirely. He types in the recipes from his scrapbook as plain text files, performing a modest amount of formatting with tabs and line breaks. For example, the first line of the file is the recipe's title. He uses the Exchange Center's *Gather* panel to get these page images and files into the collection, and then the *Enrich* panel to add a small amount of metadata. He does not need to enter title information for the text files, because these are automatically determined from the first line of text.

For the web recipes he retrieves the items using his bookmarks and, where needed, Internet searching. Using the *Download* panel he selectively downloads individual recipes, recipe index pages plus the recipes they point to, and, when the case merits it, entire websites. All these he drags from the *Download Folder* into the collection, except that he takes the opportunity to tidy up as he goes, deleting chaff such as preface material and other non-recipe pages. Again he assigns metadata manually using the *Enrich* panel, providing a unifying structure for all the digital objects in his collection. He exports the collection to Fedora and starts to use it.

Unbeknownst to George, his friend Mildred who is also taking the course, has begun a similar project with her recipes; however, she has chosen to use DSpace. When they realized what each other had done (how they laughed!) they swapped website addresses of their repositories with a view to incorporating a selection of tasty morsels from each other's collection.

Mildred searches and browses George's collection and contacts him with a list of recipes she would like, identified by their unique IDs. George uses this information to start a new Exchange Center collection into which he drags and drops the recipe files from his main collection. When copying a document from an existing collection, the Exchange Center automatically transfers the metadata too. George then chooses the *Export* option from the *File* menu, selecting DSpace as the export format. If there was a mismatch between the metadata fields he had used and those Mildred had chosen, he could have reconciled the differences through an XSLT. The exported DSpace files are then zipped up and e-mailed to Mildred.

To transmit some of Mildred's records to George, they use OAI-PMH to facilitate the transfer. This involves a similar workflow to the next scenario, where the Exchange Center is used to import from a federation of OAI repositories, consequently we defer describing this particular transfer of information until after the second scenario.

Discussion

George and Mildred achieved their goal with a small amount of effort, despite the fact that they had not come up with a joint plan in advance. The procedure was a little less than elegant in that when Mildred chose some of George's recipes, she found them using Fedora's Reader's interface and then George had to copy them from the collection using the Exchange Center interface. This would have been easier if the environment had included a "berry basket" facility (akin to the popular eCommerce metaphor of placing items in the shopping cart) that allowed Mildred to select juicy documents and export them, for example by email. But nevertheless the procedure they devised was practical, and worked reasonably well.

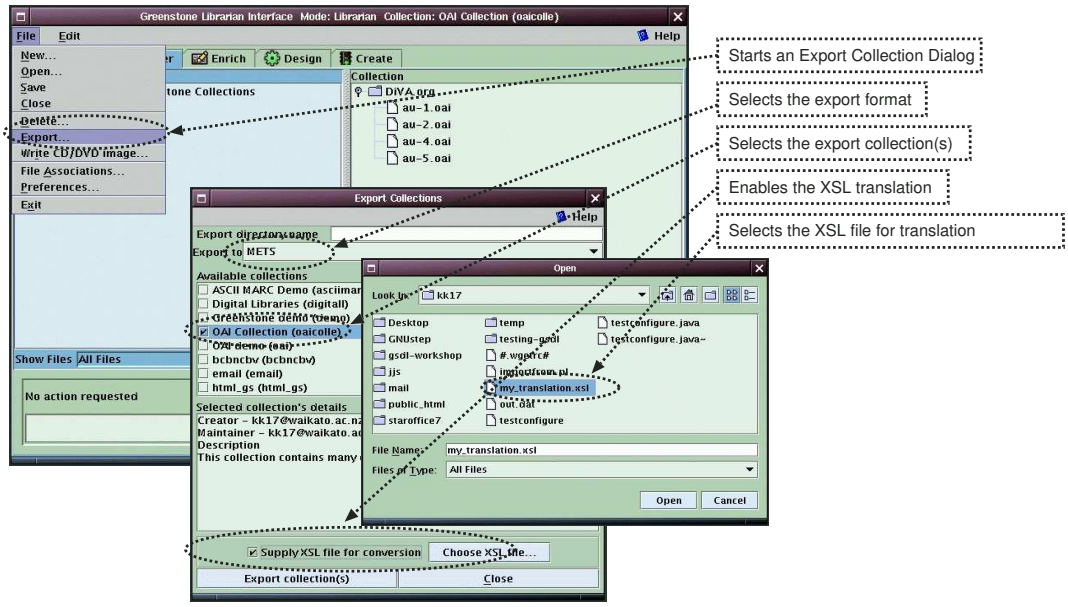


Figure 4: Graphical exporting process.

5.2 An OAI-based institutional repository network

A national consortium of university libraries has decided to establish a network of institutional repositories. Each university sets up and manages its own repository. It is free to choose its own software solution, with the proviso that it must support OAI-PMH, with MODS metadata [5], and a persistent URL to the source document in the *Identifier* field. DSpace and Greenstone are popular choices for the individual libraries. DSpace has a web interface through which people can enter new documents; Greenstone supplies similar functionality using a submission system called the Depositor [2] in combination with an applet version of GLI for building and maintaining collections on a remote server. The plan is to create a single website for aggregated searching and browsing once the individual institutions' sites are up and running.

The Exchange Center described in this paper is chosen as the interoperability environment in which to develop the aggregated site. The *Download* panel is used to import OAI records from the various university servers to a central location on the file system (the download folder). These are then copied across verbatim into the collection. One of the Exchange Center's many plugins is an OAI plugin which reads OAI records and associated documents, if any, and it is used to form the collection. However, the OAI plugin supports Dublin Core metadata, not MODS. Fortunately all XML-enabled plugins have a switch that allows the user to supply an XSLT transform, and this is used to provide a crosswalk from MODS to Dublin Core. Guidelines from the Library of Congress are used in establishing the mapping. Some information is lost, but that is not important in this context because the individual institutions are responsible for archival copies of their documents and metadata; the aggregated site is just to provide unified access to all the information.

Irrespective of which university it originated from, the Dublin Core version of the metadata is stored locally at the central site. This makes it easy to manipulate and display. However, the source documents themselves reside on the relevant host institution's server. It is those servers, not the central one, that will serve the actual documents from the aggregated collection to users. Some form of bridging link is required. Using standard features of the Greenstone system, an *Source_URN* metadata field can be established that contains each document's location. It is easy to incorporate this into the web pages served up by the system—for example, by hyperlinking the title—in such a way that the document is downloaded from the relevant host institution should the user click on the link.

Discussion

If full-text indexing search were desired for the aggregated document collection, the *get_doc* feature would need to be activated. Recall that this makes the download code check each record's *Identifier* field and, if it is a valid URL, download the file and tie it to the record as its source document. This is all that is needed for the collection that is built to provide full-text searching. Furthermore, if desired a cached version of the document could be offered to readers as an alternative to the host institution's copy in case its server went down.

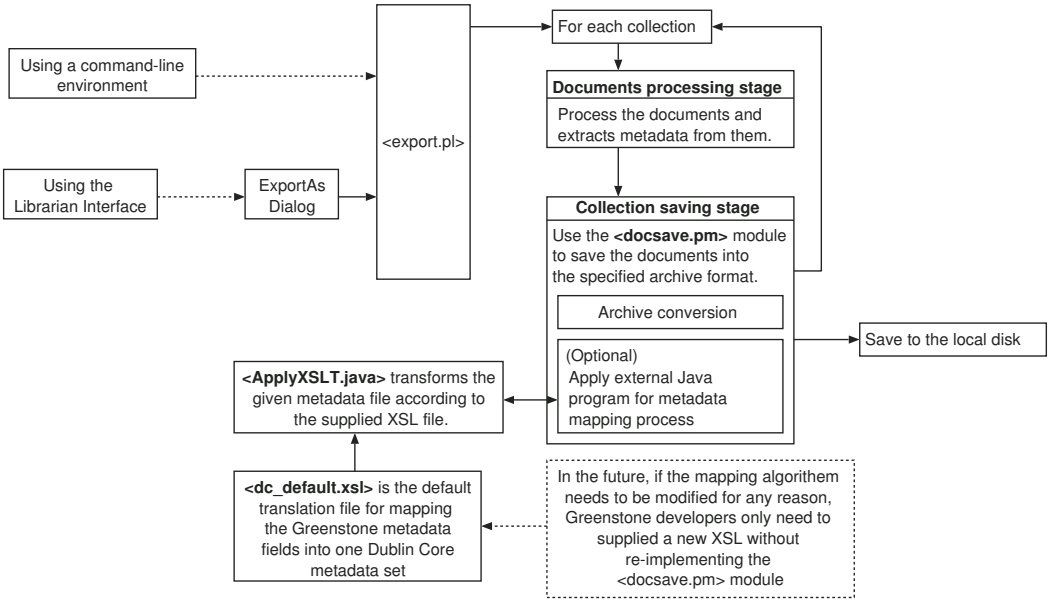


Figure 5: The exporting process.

In moving to the production version of the aggregated site, manual interaction using the *Download* panel would be discontinued in favor of setting up a script that runs the relevant Perl script (*downloadfrom.pl*) automatically every night. A finishing touch for the project is to brand the various host sites, and the central server, with the appropriate institutional logos, backgrounds, and color schemes. This is straightforward to accomplish in both DSpace and Greenstone.

As an alternative, it might be desired to present the aggregated collection using DSpace rather than Greenstone. This would involve using the Exchange Center to accumulate the documents and translate the metadata into Dublin Core. But instead of building the collection with Greenstone, the *Export* option would be used to export it in a form that is suitable for DSpace's batch import program. The above remarks about downloading the full text for indexing apply equally in this scenario, and of course the production version of the system would use automated scripts to perform the exporting and importing into DSpace.

Now we have described the basic approach the Exchange Center uses for OAI repositories, we can now return to the role of this protocol in the recipe scenario. In many ways the workflow for importing recipes from Mildred's collection into George's one is simpler: there is only one site to connect to and one download session is enough to transfer the metadata and source documents (using the *get_doc* option). But here's the rub. To achieve the selective inclusion of recipes accompanied with the assigned metadata, the OAI download step needs to download *all* the metadata records and *all* the source documents, even though only a small portion will ultimately be selected. It would be better if only the metadata records were requested (i.e., no *get_doc*). This metadata can then be used to choose what goes into George's collection and it is only when the collection is built that the source documents are retrieved. This would necessitate that the OAI plugin supports the *get_doc* option as well, and while not difficult to arrange it has not been done to date.

6 Summary

In summary, we have described an interactive system, the Exchange Center, that is intended to help practicing digital librarians solve practical interoperability problems. It is based on the Greenstone infrastructure, but is by no means restricted to projects that build Greenstone collections. The software already has an extensive and mature infrastructure for information gathering (including from external sites), enriching downloaded documents with manually-entered metadata, and processing documents in a wide variety of formats, including image, audio, and video formats, into a standard form. This project makes this infrastructure available to users of other digital library systems by permitting the collections to be exported in various forms.

Information can be downloaded over HTTP using standard web mirroring facilities, from OAI metadata repositories (including the documents themselves where applicable), and from Z39.50/SRU library servers. Of course, documents and metadata in local files can also be included in collections. While Greenstone's common internal format predates the design of METS, an option allows users to specify that METS should be used for storing the collection instead. This facilitates greater use of existing

standards. Collections can be exported using an OAI server, a Z39.50 server, in various METS profiles (using XSLT translation modules), and in a form suitable for DSpace's batch input process.

The system is by no means restricted to the current set of input and output protocols and formats. Extensibility is absolutely fundamental to Greenstone's design, and the Exchange Center inherits this philosophy. By implementing and demonstrating four very different protocols, HTTP, OAI-PMH, Z39.50, and SRU we have illustrated the flexibility and extensibility of the framework. These protocols span a wide spectrum of complexity, from pure browsing with HTTP, through the hybrid OAI-PMH which includes selective filtering, all the way to full and comprehensive search with Z39.50. In the future, other information gathering methods could easily be added to this suite without excessive effort. An example is the Web-services based protocol used by the Fedora project; it could be described as intermediate between OAI-PMH and Z39.50 in complexity, for it provides querying ability, like Z39.50, but is borne over HTTP, like OAI-PMH.

The Exchange Center's *Download* panel provides easy-to-use interactive access to the features of these protocols. This radically lowers the complexity of information gathering from non-local information sources, which previously had to be done by invoking arcane scripts. We hope it will empower practicing digital librarians to undertake practical interoperability projects.

References

[1] D. Bainbridge, K.Y. Ke and I.H. Witten. Document level interoperability for collection creators. In *Joint ACM/IEEE Conference on Digital Libraries*, pages 105–106, Chapel Hill, 2006.

[2] David Bainbridge, Wendy Osborn, Ian Witten and David Nichols. Extending Greenstone for institutional repositories. In *Digital Libraries: Achievements, Challenges and Opportunities*, Volume 4312 of *Lecture Notes in Computer Science*, pages 303–312. Springer, 2006.

[3] Goutam Biswas and Dibyendu Paul. An evaluative study on the open source digital library softwares for institutional repository: Special reference to Dspace and Greenstone digital library. *International Journal of Library and Information Science*, Volume 2, Number 1, pages 1–010, 2010.

[4] M.V. Cundiff. An introduction to the Metadata Encoding and Transmission Standard (METS). *Library Hi Tech*, Volume 22, Number 1, pages 52–64, 2004.

[5] Richard Gartner. MODS: Metadata Object Description Schema. *JISC*, 2003. http://www.jisc.ac.uk/uploaded_documents/tsw_03-06.pdf.

[6] Dion Hoe-Lian Goh, Alton Chua, Davina Anqi Khoo, Emily Boon-Hui Khoo, Eric Bok-Tong Mak and Maple Wen-Min Ng. A checklist for evaluating open source digital library software. *Online Information Review*, Volume 30, Number 4, pages 360–379, 2006.

[7] Yan Han. Digital content management: the search for a content management system. *Library Hi Tech*, Volume 22, Number 4, pages 355–365, 2004.

[8] C. Lagoze, S. Payette, E. Shin and C. Wilper. Fedora: An architecture for complex objects and their relationships. *Journal of Digital Libraries*, Volume 6, Number 2, pages 124–138, 2006.

[9] C. Lagoze and H. Van de Sompel. The open archives initiative: building a low-barrier interoperability framework. In *Joint ACM/IEEE Conference on Digital Libraries*, pages 54–62, Roanoke, Virginia, June 2001.

[10] NISO. *Information Retrieval (Z39.50): Application Service Definition and Protocol Specification*. National Information Standards Organization Press, 1996.

[11] M. Smith, M. Bass, G. McClella, R. Tansley, M. Barton, M. Branschofsky, D. Stuve and J.H. Walker. DSpace: An open source dynamic digital repository. *D-Lib Magazine*, Volume 9, Number 1, 2003. (doi:10.1045/january2003-smith).

[12] T. Storey. Moving Z39.50 to the web. *OCLC Newsletter*, Volume 263, 2004.

[13] D. Tidwell. *XSLT*. O'Reilly, 2001.

[14] Ian H. Witten and David Bainbridge. A retrospective look at Greenstone: lessons from the first decade. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '07, pages 147–156, New York, NY, USA, 2007. ACM.

[15] Ian H. Witten, David Bainbridge and David M. Nichols. *How to Build a Digital Library, Second Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2009.

[16] I.H. Witten and D. Bainbridge. Creating digital library collections with Greenstone. *Library Hi Tech*, Volume 23, Number 4, pages 541–560, 2005.

[17] I.H. Witten, D. Bainbridge, R. Tansley, C.-Y. Huang and K. Don. StoneD: A bridge between Greenstone and DSpace. *D-Lib Magazine*, Volume 11, Number 9, September 2005.

An Ontology-based Mining Approach for User Search Intent Discovery

*Yan Shen, Yuefeng Li, Yue Xu, *Renato Iannella, Abdulmohsen Algarni*

Computer Science Discipline, Faculty of Science and Technology
Queensland University of Technology, Australia

*Semantic Identity

{*y12.shen,y2.li,yue.xu,r.iannella,a1.algarni*}*@qut.edu.au*

Xiaohui Tao

Department of Mathematics and Computing, Faculty of Sciences
The University of Southern Queensland, Australia

xtao@usq.edu.au

Abstract *Discovering proper search intents is a vital process to return desired results. It is constantly a hot research topic regarding information retrieval in recent years. Existing methods are mainly limited by utilizing context-based mining, query expansion, and user profiling techniques, which are still suffering from the issue of ambiguity in search queries. In this paper, we introduce a novel ontology-based approach in terms of a world knowledge base in order to construct personalized ontologies for identifying adequate concept levels for matching user search intents. An iterative mining algorithm is designed for evaluating potential intents level by level until meeting the best result. The propose-to-attempt approach is evaluated in a large volume RCV1 data set, and experimental results indicate a distinct improvement on top precision after compared with baseline models.*

Keywords Ontology mining, Search intent, LCSH, World knowledge

1 Introduction

For an effective search engine, retrieving desired information is one primary objective that motivates many researchers to invest more than several decades. To improve the existing search capabilities, a series of advanced algorithms and processes along with the solid experimental supports have been developed [3]. However, due to the increasing complexity of the internet, recent search engines suffer from an emerging issue - ambiguity [8] [10]. It is caused by the fact that the majority of present search techniques are highly sensitive on vocabulary [1] and lack of personalization [2]. As a result, many personalized methods by considering user profiles are studied to alleviate this problem. However, these methods are either expensive in extraction or inaccurate in description. In order to avoid the discussed drawbacks and enrich the inference capacity of Web personalization, more and more people [19] [11] have taken advantage of using ontologies. Note that ontologies play an important role on machine learning and Information Retrieval (IR) [4]. With an integration of specified concepts and semantic relations, many retrieval systems can perform higher accuracy and automated characteristics. The ontologies classify all the knowledge into a well-structured way, which facilitate users to assess information items.

The paper aims to take advantage of ontologies to obtain accurate user search intent. Search intent is a significant object that contains what user needs. It can be studied into two means: the specificity and exhaustivity intent. Specificity describes the focusing extent of a topic, i.e., user's interests have a narrow and focusing goal, whereas exhaustivity describes a different extent of a topic, i.e., general/wider scope of user's interests. However, in recent years, a hard question is how to discover and characterize user intent. One existing method is quantification, i.e. using relevance weight of a pattern [20], and then re-ranking ,which is a document-based technique.

Here, we propose a hierarchical concept level-finding method, which is a knowledge-based technique as an alternative solution. A novel ontology-based approach is introduced to discover user search intent. Library of Congress Subject Headings (LCSH), which has a widespread coverage in various knowledge domains, is applied as a world knowledge base for learning personalized ontologies. According to these

Proceedings of the 16th Australasian Document Computing Symposium, Canberra, Australia, 2 December 2011.

comprehensive ontologies, diverse information is allocated in a number of concept levels. These levels might be linked if relationships exist. We assume that an actual user search intent can be found from one of the levels. The higher a level, the broader extent it has. Conversely, extents are more specific while towards lower levels. The idea is similar to a zooming navigation. To define a certain search intent, an iterative mining algorithm is developed. The attempt-to-propose approach is evaluated experimentally by 100 topics in Reuters Corpus Volume 1 (RCV1). The results indicate the top precision performance is improved remarkably. This paper will help to design a search strategy to avoid the ambiguous troubles caused by typical search techniques and owns the potential value to construct an innovative zooming navigation.

The rest of the paper is organized as follows: Section 2, some significant related work is discussed; Section 3, a world knowledge ontology and its definitions are presented; Section 4, primary components of the proposed approach are fully described; Section 5, a number of scientific experiments are conduced, and the major experimental results are outlined and discussed for evaluation; Section 6, a conclusion is brought to summarize this work and specify potential work in future.

2 Related Work

This section is classified into two categories, one is about user information needs, the other is focusing on ontology-based techniques.

User Information Needs

For user information need acquisition, many efforts have been involved to improve the accuracy and effectiveness. Closely related to our work, user ontology consisting of both concepts and semantic relations is presented by Jiang and Tan [5]. Their goal is to represent and capture users' interests in target domain. Subsequently, a method, they called Spreading Activation Theory (SAT), is employed for providing personalized services. Li and Zhong [9] develop a term-based ontology leaning method for acquiring user information needs. More recently, Tao et al. [14] propose an ontology-based knowledge retrieval framework to capture user information needs by considering user knowledge background and user's local instance repository (user profile) with association roles and data mining techniques. Other work also realizes the importance of user information need, they treat user interest as implicit feedback and store in user profile. Trajkova and Gauch [16], and Liu et al. [10] learn a user's profile from her/his browsing history, whereas Sieg et al. [13] utilize ontological user profile on the basis of the user's interaction with a concept hierarchy which captures the domain knowledge, and Tao et al. [15] [14] require the user to specify a profile manually. In short, these work aim to enhance search performance through asking users explicit feedback, such as preferences, or collected implicit feedback, which are normally either expensive in extraction or inaccurate in description.

Ontology-Based Techniques

Ontology is a collection of concepts and their interrelationships, which provide an abstract view of an application domain. It is an explicit specification of a conceptualization. Over the recent years, people who are mentioned below have often held the hypothesis that ontology-based approaches should perform better than traditional ones on IR, since ontologies are more discriminative and arguably carry more "semantics". As a result, many research concentrate on how to use ontology techniques. Zhong [19] proposes a learning approach for task (or domain-specific) ontology, which employs various mining techniques and natural-language understanding methods. Li and Zhong [9] present an automatic ontology learning method, in which a class is called a compound concept, assembled by primitive classes that are the smallest concepts and cannot be divided any further. Liu and Singh [11] develop ConceptNet ontology and attempt to specify common sense knowledge. However, ConceptNet does not count expert knowledge. Navigli et al. [12] build an ontology called OntoLearn to mine the semantic relations among the concepts from Web documents. Gauch et al. [4] use ontology references based on the categorisation of online portals and propose to learn personalised ontology for users. Developed by King et al. [6], IntelliOnto is built based on the DDC (Dewey Decimal Classification) system and attempt to describe the background knowledge.

Unfortunately, the previous work on ontology learning covers only a small size of concepts, where mainly uses "Is-A" (super-class, or sub-class) relation in the knowledge backbone. They don't consider to mine and characterize knowledge in a concept level rather than domains. To extend these methods, the backbone of personalized ontologies is been determined to build a real hierarchical structure by applying information in a world knowledge repository.

3 LCSH: World Knowledge Base

World knowledge is the common-sense knowledge acquired by people based on their experience and education. It can be considered as an exhaustive repository to maintain the known knowledge by human being [18]. LCSH is an ideal world knowledge repre-

sentation because of a rich vocabulary is used to cover all subject areas. Meanwhile, wealthy semantic relations among terms are good at reveal precise relationships of subjects. In the LCSH, subject headings are basic semantic units for conveying domain knowledge and concepts, they have three main types of references: *Broader Term* (BT), *Narrower Term* (NT) and *Related Term* (RT). BT means a hypernym, is a more general term, e.g. "Pressure" is a generalization of "Blood Pressure"; NT means a hyponym, is a more specific term, e.g. "Economic Crisis" is a specialization of "Crisis". These two references are used in our model to indicate the $is-a$ relations among subjects in the world knowledge base. To facilitate in-levels ontology construction later in this paper. The references are firstly redefined to $ancestor$ and $descendant$ lexical relations in our approach respectively. $ancestor$ refers to the concept of BT, and $descendant$ refers to NT, more information can be found in Table 1. All the subjects are formalized as:

Type	Paraphrase	Example
Ancestor	is the general term for	"profession is the general term for scientist" $\Longrightarrow Ancestor(profession, scientist)$
Descendant	is-a	"scientist is a profession" $\Longrightarrow Descendant(scientist, profession)$

Table 1: Examples for redefined relations

Definition 1 (Subjects): Let $\mathbb{S}$ denote a set of subject headings in LCSH, a subject $s \in \mathbb{S}$ is formalized a triple $(label, ancestor, descendant)$, where

- $label$ is the heading of s in LCSH thesaurus;
- $accestor$ is a function regarding the subjects that are more general and located a higher level than s in the world knowledge base;
- $descendant$ is a function regarding the subjects that are more specific and located a lower level than s in the world knowledge base.

At this stage, there is only one relation $r = (is-a)$ considered by our approach. Thus, the world knowledge base can be formalized as:

Definition 2 (World Knowledge Base): A world knowledge base ontology is a directed acyclic graph structure defined as a pair $\Theta := (\mathbb{S}, r)$, consisting of

- $\mathbb{S}$ is a set of subjects in LCSH $\mathbb{S} = \{s_1, s_2, ..., s_n\}$;
- r is the semantic relation $r = (is-a)$ existing among the subjects in $\mathbb{S}$;

4 Proposed Approach

A quick overview of the proposed approach is illustrated in Figure 1. The paper first holds a hypothesis that a user search intent should exist somewhere in an ontology. It is treated as a user information need and represented by a range of concept extent. The intent could be general or specific. In order to minimum user burdens, a query is the only input for the proposed approach, it likes a real search activity. To cope with a user query, a subject-based search model is developed in order to retrieve matching results from a LCSH database. The function is similar as a keyword-based search except the type of returned results is a list of subject headings. Both "AND" and "OR" operators are employed at the same time. This process might increase information redundancy, however, it can extend a scope to cover potential user intents with restricted user information. Semantic relation extractions are conducted for all the matching subjects for learning personalized ontologies. After that, all terms appearing in the subjects are used to do a query expansion, and then find semantically similar matches rather than lexically dissimilar by taking the extracted relations into account. The related methods of learning personalized ontologies and in-levels ontology mining are mainly explained in the next two subsections.

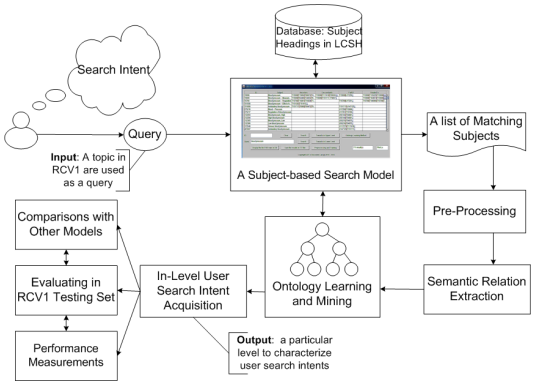


Figure 1: The general architecture design

4.1 Personalized Ontology Learning

Concept hierarchy is an essential subtask of ontology learning. In theory, it is a prerequisite hierarchy where a mount of nodes represent concepts in a domain, and related links are served as prerequisite relationships. For this paper, we create a special hierarchy format to satisfy our research purposes. The hierarchical backbone is drawn as Figure 2. One of the objectives is to make use of this hierarchy to allocate information into a well structure, which facilitate users to access information items. Another objective is to infer implicit knowledge by tracking internal relationships among subjects. The gathered implicit knowledge will be used to estimate whether a user search intent is characterized in a certain level.

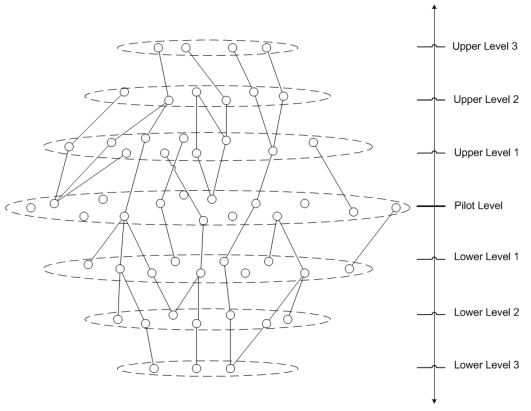


Figure 2: The backbone of in-levels hierarchy

After implementing the subject-based search model, a list of matching subjects can be obtained. Each of them is realized as a concept here and represented by a single node in Figure 2. To learn a personalized ontology, a concept's domain is confirmed by taking these subjects into account because all the subjects have their domains, which are considered as correlative information regarding a topic. While the subjects are all in the same abstract level, where is originally defined as "Pilot Level". Initially, a formalized definition for its domain is provided as:

Definition 3 (Domain for a Level): Let C_i denote a set of subjects $C_i := \{s_1, s_2, ..., s_h\}$ in a level l_i. We define its domain by $dom(l)_i := dom(s_1) \cup dom(s_2) \cup ... \cup dom(s_h)$, where $dom(s_h)$ contains all the terms in the label of subject s_h.

Dashed circulars in Figure 2 are utilized to indicate the domains of different levels. With respect to the ontology learning, we also formalize:

Definition 4 (Personalized Ontology): the personalized ontology for a topic in a 4-tuple $\Theta^p := (\mathbb{C}, L, DOM, e)$, where

- $\mathbb{C}$ is a super set of C including all subjects in levels $\mathbb{C} = \{s_1, s_2, ...s_h\}$;
- L is a set of levels consisting of a domain and subjects $L = \{l_1, l_2, ..., l_i\}$;
- $DOM := (dom(l_1), dom(l_2), ..., dom(l_e))$;
- e is the number of levels.

The semantic relations specified in the world knowledge base benefit our approach to acquire a set of new subjects in other levels. All the levels are classified into two directions, one is upper, the other is lower. The pilot level is selected as a benchmark in the hierarchy. Some subjects might just occur one time in the pilot because of no semantic relations. The major in-levels concept is similar to a knowledge generalization process. Indeed, a subject in a upper level covers a more general knowledge than the lower one. In other words, the knowledge in a level can be summarized by the knowledge in the next upper level. Eventually, all the knowledge in the world knowledge base will be summarized in philosophy. This is a main reason why the domains of upper levels are getting smaller when moving towards to the peak of the backbone in Figure 2, where looks like a shape of cone. However, why this happens as the same in lower levels? From a perspective of IR, the subject-based search model uses to return specific subjects based on keywords. The majority of matching subjects are usually lack of semantic relations in the pilot level to extend more knowledge. As a result, the number of subjects in lower levels are decreasing as well as their domains. Therefore, the shape becomes a inverse cone. Note that the backbone structure is not a formal tree, a node can has more than one parent or child.

4.2 In-Levels Ontology Mining Method

To prove the hypothesis mentioned earlier, an iterative ontology mining method is proposed in this section. It starts from the pilot level, and then builds a personalized ontology (the backbone of in-levels hierarchy) in order to find a suitable level for a search intent. The building process simply employs the $is-a$ relation to find all parents in an upper level or get all children from a lower level. For understanding precisely, an entire study is separated into two phases to explain the method in details. Each phrase involves several steps.

Phase 1: Represent feature in levels

There are two main objectives: 1) to decide subjects and their weights for the pilot level l_ρ; and 2) to represent the pilot level l_ρ as a feature vector F_ρ. Firstly, retrieve a number of matching subjects from the pilot level l_ρ after implementing the subject-based search model. Then, calculate a weight for all subjects $s \in C_\rho$ by using the following equation:

$$w(s) = \frac{|q \cap s|}{|s|} \qquad (1)$$

where $|q \cap s|$ denotes the number of terms appeared in both query q and subject s, $|s|$ denotes the total number of terms in subjects. Therefore, a set of subject weight pairs are obtained as $S(w) = \{< s_1, w_1 >, < s_2, w_2 > , ..., < s_n, w_n >\}$.

Secondly, expand the query to a set of terms by union all terms from the submitted query and matching subjects,and let $Q_\rho = \{t_1, t_2, ..., t_m\}$. For example, the submitted query is $query = \{t_1, t_2\}$, other subjects are $s_1 = \{t_1, t_2\}$, $s_2 = \{t_1, t_2, t_6\}$, and $s_3 =$

$\{t_1, t_2, t_5, t_8\}$. After that, $Q_\rho = query \cup s_1 \cup s_2 \cup s_3 = \{t_1, t_2, t_5, t_6, t_8\}$.

Thirdly, Calculate weights for all terms $t \in Q_\rho$ via using the following equation:

$$weight(t) = \sum_{t \in s, s \in C_\rho} \frac{w(s)}{|s|} \quad (2)$$

Then, we receive a set of term weight pairs as a feature vector $F_\rho = \{< t_1, w_1 >, < t_2, w_2 >, ..., < t_m, w_m >\}$ to represent this level, where $w_m = weight(t_m)$.

Phase 2: Determine the best level for user search intents

The goal is to determine the appropriate level for characterizing the user search intent according to a training set. Let D_t stand for a set of documents in the training set, which has $D_t = {D_t}^+ \cup {D_t}^-$. ${D_t}^+$ is a set of positive documents, and ${D_t}^-$ stands for negative ones, where t denotes a certain topic. All these documents have been initialized a value of either 0 or 1 by linguists. We calculate a weight for each document in the training set by using the feature F_ρ. Thus, rank D_t by using Ranking Algorithm provided as follows:

$$rank(d) = \sum_{t \in Q_\rho}^{n} weight(t) \quad (3)$$

Based on the ranked documents, a top-K precision $precision(l_\rho)$ can be calculated for the pilot level l_ρ by apply the equation below:

$$precision(l_\rho) = \frac{\sum_{i=1}^{K} f(d_i)}{K} \quad (4)$$

where $f(d_i) = 1$ if d_i is relevant, otherwise $f(d_i) = 0$.

Algorithm 1 discovering search intent (e.g.upper levels only)

Input:
D^t in the training set of RCV1; F_ρ; Parameter μ.
Output:
A suitable level to match a user intent

```
Let j = ρ, i = j;
Let i = i + 1; //ρ = ρ + 1, shift to the upper level;
Get Q_i and F_i;                    //refer to Phase 1;
Use F_i to rank D_t;                //see Eq.(3)
Get precision(l_i);                 //see Eq.(4)
if precision(l_i) < precision(l_j) then
    return l_j;
else
    if i − j > μ then
        return l_i
    end if
end if
j = i;
Go to 2;
```

The pilot level is possibly not the expected level resulting in shift to the upper level $l_{\rho+1}$ or lower levels in the hierarchy. Thereby, a new set of subjects in $l_{\rho+1}$ are returned by getting all subjects s that have a $is - a$ relationship with any subjects in the pilot level l_ρ. Repeat the above steps to rank the documents D_t by using the feature vector $F_{\rho+1}$ for level $l_{\rho+1}$, and then calculate the top-K precision $precision(l_{\rho+1})$. If $precision(l_\rho) > precision(l_{\rho+1})$, return l_ρ as the appropriate one as user search intent. Otherwise, go to step two and implement the same procedure in $l_{\rho+2}$ again. Algorithm 1 describes the idea that is looping for upper levels until meeting the most satisfactory level based on precision performance, where parameter μ is used to control the distance between the selected level l_i with the pilot level l_ρ. If a level is too far away with the pilot level, we assume that it is less significant to search intents. To save space, the paper omits the explanation for lower levels because its algorithm is quite similar as Algorithm 1. According to two phases above, we are able to gain a level with the best top-K precision among all the hierarchical levels. This level is considered as the output of user search intents from our proposed approach.

According to two phases above, we enable to gain a level with the best top-K precision among all the hierarchical levels. This level is considered as the output to match a user search intent.

5 Evaluation

In this session, it first states the data collections used for the subject-based search model and our experiments. Thus, a description is provided to explain our experimental design. Evaluated results are also presented after examining by diverse performance measurements in the concept hierarchy.

5.1 Data Collections

LCSH was chosen as the database for the subject-based search model development. The size is approximately 719 mega bytes stored in Microsoft Office Access. Initially, 20 tables were created to save different data. The data of topical, corporate, and geographic subjects (491,250 subjects in total) were extracted for building the ontology of world knowledge base. Meanwhile, there are five different references linking all these subjects in LCSH. The paper only adopted the references of BT and NT, and encoded as a semantic relation of $is - a$.

As a well-known evaluation methodology founded by IR research community, the Text Retrieval Conference (TREC)[1] Filtering Track is widely applied to evaluate the effectiveness of search applications. RCV1 was applied in our experiments because it's a crucial component of TREC-11 2002 Filtering Track, which

[1] http://trec.nist.gov/

		$pr@20$		$MeanAve.Pre.$		$F_1 - Measure$	
	# Subjects	Value	% Improve	Value	% Improve	Value	% Improve
Upper Lv.7	**25.96**	**0.204**	**21.19**	**0.228**	**0.07**	**0.281**	**-1.025**
Upper Lv.6	37.76	0.199	18.42	0.224	-1.43	0.279	-1.66
Upper Lv.5	54.04	0.193	14.39	0.225	-1.01	0.281	-1.02
Upper Lv.4	75.96	0.18	6.49	0.221	-2.69	0.278	-2.1
Upper Lv.3	114.16	0.183	8.53	0.223	-1.9	0.28	-1.35
Upper Lv.2	178.8	0.188	11.79	0.229	0.55	0.284	0.08
Upper Lv.1	365.16	0.18	7.03	0.231	1.5	0.287	1.15
Pilot Lv.	**2132.04**	**0.168**		**0.228**		**0.284**	
Lower Lv.1	370.04	0.17	1.19	0.228	0.39	0.284	0.21
Lower Lv.2	103.84	0.19	11.31	0.23	0.88	0.285	0.5
Lower Lv.3	32.52	0.174	3.54	0.222	-2.41	0.2798	-1.63

Table 2: First 50 topics performance measurement results

is a corpus that contains totally 806,791 documents. These documents were produced by Reuter's journalists between August 20, 1996 and August 19, 1997. All of them were marked in Extensible Markup Language (XML). They distributed into training and testing sets. Before adopting these XML files for our experiments, they have been tokenized into plain texts. Entirely, RCV1 covers 100 topics by two types: 1) the first 50 topics have been developed by assessors from the National Institute of Standards and Technology, and the assessors have made the relevant judgements manually; 2) the last 50 topics have been built automatically by machine learning instead of by human being for intersection topics. To minimize bias in experiments, the paper conducted a pre-processing for all the queries, subjects in LCSH, and XML files in RCV1 corpus. The pre-processing includes stop-words removal and stemming by applying Porter Stemmer algorithm. The development of the subject-based search model and our experiments were encoded by JAVA.

5.2 Measures & Baseline Model

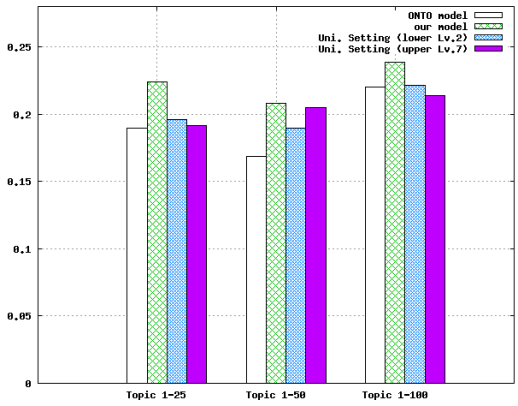


Figure 3: Top 20 precision results

In order to prove the accuracy and feasibility of our approach, the paper estimated all the levels by applying four state-of-the art measuring methods. There are top 20 precision based on the relevance judgement in RCV1 ($pr@20$), the precision averages at 11 standard recall levels ($11-points$), the *Mean Average Precision* (MAP), and the $F_1 - Measure$.

Top 20 precision is considered as the most important standard in the evaluation since a web searcher is mostly going to look at the top 20 documents [7]. In the domain or IR, precision is the percentage of retrieved documents that are relevant. Each document in RCV1 has already been judged the relevance by 0 and 1. Compared to these judgements, the top 20 precision can be computed.

$$pr@20 = \frac{|\{first20 ranked\ docs\} \cap \{relevant\ docs\}|}{20}$$

MAP is correlated with Average Precision ($Ave(p)$). $Ave(p)$ is the average of precision at each relevant document retrieved in the ranked sequence. Consisting of the $Ave(p)$, the equation of MAP is formed as:

$$MAP = \frac{1}{|Q|} \sum_{s=1}^{|Q|} \mathrm{Ave\,(p)}$$

where Q stands for the number of queries. *F1-measure* was first introduced by C. J. van Rijsbergen [17]. It combines recall and precision with an equal weight in the following form:

$$F_1 - Measure = \frac{2 \times precision \times recall}{(precision + recall)}$$

$11-points$ measure is also used to estimate the performance of retrieval models by averaging precisions at 11 standard recall levels (i.e. $recall = 0, 0.1, 0.2, ..., 1$).

A ontology model named ONTO model from Tao et al in 2010 [14] was selected as one baseline for evaluating. The ONTO model has already evaluated its outperformed capability after comparing with a number of other models. Hence, the related comparisons

are meaningful. Furthermore, two uniform level settings(parameter $\mu = 2\, and\, 7$) were selected as baseline models, which are upper level 7 and lower level 2 respectively.

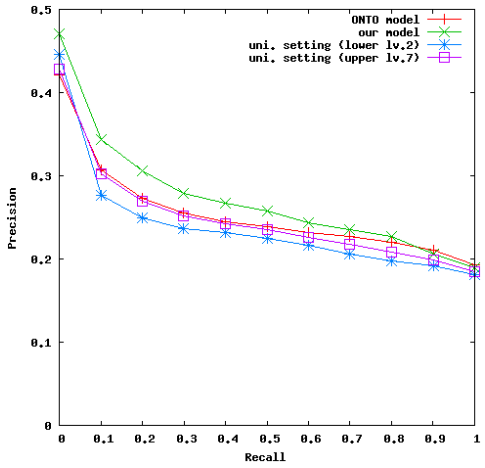


Figure 4: First 1-25 11-points performance

5.3 Results and Discussion

The performance of the experimental models was measured by the mentioned measurements for different levels. Table 2 includes all statistical results computed after implementing the first 50 topics in RCV1. To indicate the influences affected, $the\ percentage\ change\ in\ performance$ was used to compute the difference in Top 20 precision, MAP, and $F_1 - Measure$ results among the levels. It is formulated as:

$$\%Improve = \frac{Result_{level} - Result_{pilot}}{Result_{pilot}} \times 100$$

The lager $\%Improve$ value the more significant improvement achieved. We noticed that the number of subjects are decreasing while towards upper or lower levels. This is an evidence to prove that the proposed hierarchy is reasonable. Refer to subsection 4.1, we can picture a shape for upper levels as a cone. In contract, an inverse cone is for lower levels.

As shown in Table 2, the results of MAP and $F_1 - Measure$ are not improving dramatically. The main reason is that they both considered about the same recall computing over the RCV1 data sets. However, if only considering about precision, the upper level 7 has the best result on $pr@20$, which is 21.19% better than the baseline model. As a result, for the first 50 topics, the upper level 7 was determined as the best level to characterize user search intents. According to Figure 3, the $pr@20$ results are always leading the others. The results from the uniform settings are also superior to the ONTO model on first 25 and 50 topics. The results of $11 - points$ on first 25 and 50 topics are illustrated in Figure 4 and 5.

Limitations: Three main limitations exist in this work. The first one is: our investigation is mainly focusing on the usage of $is - a$ relations in LCSH. The other relations, including $used - for$ and $related - to$ are regardless within our approach. As a result, the maximum number of depth detected based on the constructed concept hierarchy is 28 but not 37 as specified in the LCSH specification [14]. Some of useful implicit knowledge might be not entirely discovered from world knowledge representation. The second limitation is caused naturally from the LCSH. In reality, user interests are usually changing all the time. The choice and form of headings are not necessarily current because the LCSH terms have evolved over time, but they can never be totally up to date. This might lead to misinterpretation of user search intents. The last one is about the dataset that applied for evaluation. It is a textual collection of news, but database used for searching is a subject collection of library headings. This might possibly influence experimental results.

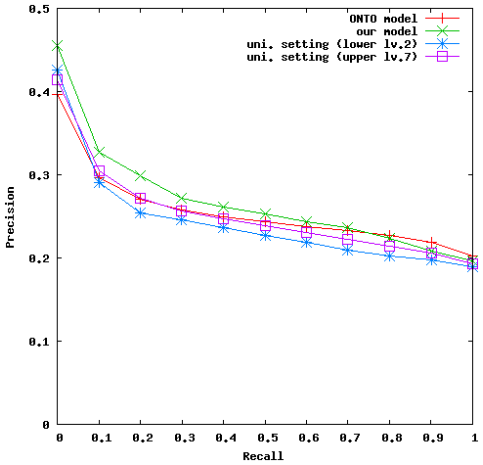


Figure 5: First 1-50 11-points performance

In sum, the proposed ontology-based approach is been proven successfully based on the experiments. The improvements are consistent and significant on the top 20 precision measure. The related results indicate the overall performance are better than the baseline model.

6 Conclusions

A novel ontology-based approach is introduced for user search intents discovery. It utilizes a subject-based search model to filer out irrelevant information, and then allocates matching results into a world knowledge

base - LCSH. A concept-based hierarchy is built by applying semantic relations from the world knowledge in order to characterize accurate user intents in an actual level. A huge test bed was utilized for a number of experiments. The experimental results demonstrate that our proposed approach is working and promising. It can enhance the search effectiveness in top precision. The major contribution of this paper is describing an alternative ontology-based method to discover user search intents except pattern mining. This research will significantly influence the development of personalized Web search services, and the related deliverables have potentials to contribute intelligent search navigation.

In future, we plan to investigate how to use the rest semantic relations in LCSH, including *equivalence* ($used - for$) and *associative* ($related - to$). These semantic relations can not only assist to revise subjects appeared in a level, but also navigate to user search intents more precisely. The implicit knowledge including search intents can be obtained effectively from a world knowledge base. Recent studies report that patten mining methods are effective strategies for relevance information acquisition in a short number of terms rather than content-based methods [8]. Since now our approach achieved the task to discover user search intents in a contain level. Another attempt is to discover a certain subject in the level.

References

[1] T. Berners-Lee, J. Hendler, O. Lassila et al. The semantic web. *Scientific american*, Volume 284, Number 5, pages 28–37, 2001.

[2] P.A. Chirita, C.S. Firan and W. Nejdl. Personalized query expansion for the web. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 7–14. ACM, 2007.

[3] Z. Dou, R. Song, J.R. Wen and X. Yuan. Evaluating the Effectiveness of Personalized Web Search. *IEEE Transactions on Knowledge and Data Engineering*, pages 1178–1190, 2008.

[4] S. Gauch, J. Chaffee and A. Pretschner. Ontology-based personalized search and browsing. *Web Intelligence and Agent Systems*, Volume 1, Number 3, pages 219–234, 2003.

[5] X. Jiang and A.H. Tan. Learning and inferencing in user ontology for personalized Semantic Web search. *Information Sciences*, Volume 179, Number 16, pages 2794–2808, 2009.

[6] J.D. King, Y. Li, X. Tao and R. Nayak. Mining world knowledge for analysis of search engine content. *Web Intelligence and Agent Systems*, Volume 5, Number 3, pages 233–253, 2007.

[7] H.V. Leighton and J. Srivastava. First 20 precision among world wide web search services(search engines). *Journal of the American Society for Information Science*, Volume 50, Number 10, pages 870–881, 1999.

[8] Y. Li, A. Algarni and N. Zhong. Mining positive and negative patterns for relevance feature discovery. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 753–762. ACM, 2010.

[9] Y. Li and N. Zhong. Mining ontology for automatically acquiring web user information needs. *IEEE Transactions on Knowledge and Data Engineering*, pages 554–568, 2006.

[10] F. Liu, C. Yu and W. Meng. Personalized web search for improving retrieval effectiveness. *IEEE Transactions on knowledge and data engineering*, Volume 16, Number 1, pages 28–40, 2004.

[11] H. Liu and P. Singh. Commonsense reasoning in and over natural language. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 293–306. Springer, 2004.

[12] R. Navigli, P. Velardi and A. Gangemi. Ontology learning and its application to automated terminology translation. *IEEE Intelligent Systems*, Volume 18, Number 1, pages 22–31, 2003.

[13] A. Sieg, B. Mobasher and R. Burke. Web search personalization with ontological user profiles. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 525–534. ACM, 2007.

[14] X. Tao, Y. Li and N. Zhong. A personalized ontology model for web information gathering. *IEEE Transactions on Knowledge and Data Engineering*, 2010.

[15] X. Tao, Y. Li, N. Zhong and R. Nayak. Ontology mining for personalizedweb information gathering. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 351–358. IEEE Computer Society, 2007.

[16] J. Trajkova and S. Gauch. Improving ontology-based user profiles. In *Proceedings of RIAO*, Volume 4, pages 380–389. Citeseer, 2004.

[17] CJ Van Rijsbergen. Information retrieval, chapter 7. *Butterworths, London*, Volume 2, pages 111–143, 1979.

[18] L.A. Zadeh. Web intelligence and world knowledge-the concept of web iq (wiq). In *Fuzzy Information, 2004. Processing NAFIPS'04. IEEE Annual Meeting of the*, Volume 1, pages 1–3. IEEE.

[19] N. Zhong. Representation and construction of ontologies for Web intelligence. *International Journal of Foundations of Computer Science*, Volume 13, Number 4, pages 555–570, 2002.

[20] X. Zhou, S.T. Wu, Y. Li, Y. Xu, R.Y.K. Lau and P.D. Bruza. Utilizing search intent in topic ontology-based user profile for web mining. In *IEEE/WIC/ACM International Conference on Web Intelligence, 2006. WI 2006*, pages 558–564, 2006.

The Interplay of Information Retrieval and Query by Singing with Words

Iman Amini
School of Computer Science
RMIT
Melbourne, Australia
iman.amini@rmit.edu.au

Alexandra L. Uitdenbogerd
School of Computer Science
RMIT
Melbourne, Australia
sandrau@rmit.edu.au

Mark Sanderson
School of Computer Science
RMIT
Melbourne, Australia
mark.sanderson@rmit.edu.au

Abstract *Speech recognition can be used in music retrieval systems to identify the words in users' sung queries. Our aim was to determine which of several techniques is most suitable for retrieving songs given a sung query with words. We used Sphinx for speech recognition, and tested several retrieval techniques on the output of the recognition system. The most effective retrieval technique was a combination of Edit Distance and Okapi, which persistently retrieved the correct song at the top one ranked results given that the queries were at least 50% correct. However, techniques performed differently when the queries were split into four buckets with varying level of correctness in the range of 0 to 73%.*

Keywords Pattern Matching, Ranking, Speech Recognition, Music Information Retrieval.

1 Introduction

Music Information Retrieval (MIR) involves finding music using a user query to a system. MIR systems differ in what information they use and how they use it as evidence to find music, songs or lyrics. Sung queries carry two types of relevant information, verbal and melodic. The type of MIR investigated in this paper focuses on the former, that is, words sung by the user to find the relevant lyric. A system that handles such queries can be divided into two sub-systems, namely, speech recognition and lyric matching.

Systems based on sung lyrics can fail due to poor "speech" recognition, or suboptimal matching techniques. Table 1 is an example that depicts the typical limits of speech recognition systems. The first column shows recognition output for the original input text given in the second column.

Proceedings of the 16th Australasian Document Computing Symposium, Canberra, Australia, 2 December 2011.

Acoustic characteristics of a singing voice vary from those of speech. Words are often pronounced differently, vowels are sustained longer, and some words are sung with varying pitch. Indeed, words sung by sopranos at high pitch lose clarity due to the overlap of pitch and vowel frequencies [6]. These factors can degrade the accuracy of the speech recognition. Others have worked on solutions to these problems by applying speaker adaptation techniques [13, 20].

In this paper, however, we focus on the lyric matching techniques for sung queries. We test and combine five techniques that can be categorized into three groups: word-based matching, edit distance, and phonetic matching.

Croft [2] earlier observed that ranking algorithms that have similar effectiveness can have a very low overlap in their result sets. We test 5 different matching techniques and combine our best word-based and phoneme- or character-based techniques. Results are promising, with combinations of Okapi similarity measure on words and an edit distance formulation (Okapi-Edit distance) and SAPS-L giving the best results.

The rest of this paper, describes related research work, followed by details of the range of lyric matching techniques used. Next, the retrieval experiment and results are outlined, before the paper concludes.

Transcription

The term transcription generally refers to two distinct set of outputs, both generated from a sung query.

1. Textual transcription, denotes the conversion of audio query to words.
2. Musical transcription, refers to the conversion of audio query to musical notes.

Both types of transcription involve analysis of frequency spectrograms generated from the audio data [13]. In the case of musical transcription, some

Recognized Segment	Original Segment
me been raining	it's been raining
gold who mind do fool around	don't fool around don't fool around
every say the a	every single day
let there be sang	let there be spring
morning light and we sang chan	morning light and we sang here

Table 1: Original lyrics misheard by the Speech Recognition generated by the PocketSphinx toolkit

of the frequencies have a direct correspondence to the pitch of notes that are sung, while others are related to such audio features as the timbre of the instrument or voice. A transcription algorithm for symbolic notes converts frequencies into its corresponding musical notes.

On the other hand, in order to produce a *textual* transcription of the sung query, speech recognition system uses a probabilistic language model such as n-gram or finite state grammar, to produce a sequence of uttered words.

2 Related Work

There are several main areas of prior work relevant to this paper: speech recognition – particularly for singing, speech retrieval, misheard lyric matching, and music information retrieval using sung text. We restrict our survey of speech recognition to work that is specifically about singing, and provide some examples of speech retrieval work.

Many researchers [11, 16], have used MLLR (Maximum Likelihood Linear Regression) to carry out speaker adaptation. In order to avoid customization to a specific user Suzuki et al. [20] used MLLR with 6 male singers employed to sing 127 choruses for training data. They experimented on singing recognition using a normal Finite State Automaton (FSA) and a modified FSA for a Japanese language grammar. Using the modified version of FSA which exploits a constraint in the Japanese grammar, they improved word recognition correctness from 83.2% to 86.0% and retrieval accuracy from 82% to 85% as compared to the normal FSA.

Mesaros and Virtanen [13] trained and configured their speech recognition to minimize the chance of errors in decoding the speech data, considering models that are specific to gender and singer. For retrieval they used a simple count of the words occurring in both the query and songs, achieving a top-ranked match for 57% of the queries for their dataset. Wang et al [21], in contrast, used a dynamic programming-based alignment of syllables for the same purpose. Their system used both melody and lyrics information. They first determined whether the user had sung or hummed by comparing the number of distinct phones decoded in the user's acoustic model, which was expected to be less for humming than for singing. A related but simpler problem that was studied is the task of audio and lyric alignment [3].

Phonetic matching is the task of retrieving words that are similar in pronunciation but not necessarily in spelling. There has been a number of studies on phonetic matching to improve name retrieval [1, 17], however, the problem of phonetic matching using sung queries has rarely been tackled. Zobel and Dart [24] showed that a phonetic-based edit distance (Editex) and a competing technique known as Ipadist were more effective than various commonly used alternatives when matching surname queries to a data set of 30,000 distinct surnames. Prior to the above experiment, Zobel and Dart [23] had proved that q-grams and the minimal edit distance were superior. While most of the phonetic approaches to string matching convert the strings into some canonical forms to provide the grounds to match phonetically similar strings, Syllable Alignment Pattern Searching (SAPS) is different in that it segments two phonetic strings into syllables to calculate their similarity score [4].

Ring and Uitdenbogerd [18] investigated the problem of misheard lyrics and found that standard edit distance was as effective as more sophisticated phonetic variants. Xu et al. [22] used a confusion matrix to solve the same task for lyrics in Japanese. Hirjee and Brown [5], trained a probabilistic model on examples of actual misheard lyrics and developed a phoneme similarity scoring matrix. This probabilistic method significantly outperformed all other techniques including SAPS-L and Phoneme Edit Distance, finding 5-8% more correct lyrics within the first five top results than the previous best method, Phoneme Edit Distance. This model was based on phonetic confusion data constructed from pairs of original misheard and correct lyrics found on misheard lyrics websites. For any given pair of phonemes a and b, the model generates a log-odds score outputting the likelihood of a being (mis)heard as b.

Speech retrieval is a similar problem to retrieval of music via sung queries with words, in that both involve retrieving audio by matching a representation of the words that the query and target document contain. Ng et al. [14, 15] found that word-based retrieval was consistently more effective, but, given reliable speech recognition, phoneme-based approaches had similar effectiveness, and had the advantage of handling out-of-vocabulary terms. They used n-grams of varying lengths, finding that 3-grams and 4-grams improved retrieval.

3 Matching

Here we describe the three classes of matching techniques used.

3.1 Word-based matching

Two forms of word-based matching were used in the experiments.

Our baseline technique counts the number of common words in the lyric and the query excluding the repeated words in the lyrics. For instance, if the word "school" happens to occur twice in the query and three times in the lyric, the algorithm will score two points [13]. We refer to this technique as *Word Count*.

We also tested *Okapi BM25* [7], which is a probabilistic similarity measure that exploits factors such as term frequency, and is known to be very effective for text retrieval. The formula and constant values used for our experiments are shown in equation 1.

$$BM25(Q,D) = \sum_{(t \in Q)} w_t . \frac{(k_1+1) f_{d,t}}{K + f_{d,t}} . \frac{(k_3+1) f_{q,t}}{k_3 + f_{q,t}} \quad (1)$$

where $K = k_1.((1-b) + \frac{b.L_d}{AL})$
Q query
D document
w_t is the Robertson-Spark Jones weight
$f_{d,t}$ and $f_{q,t}$ are the number of occurrences of term t in the document and query, respectively
L_d and AL are the document length and average document length
k_1,k_3 and b are parameters that are determined empirically
k_1 = 1.2, b = 0.75 and k_3 = 0

k_3 is often set to 0 because of the short nature of queries which often implies that words in the queries do not occur more than once.

3.2 Edit Distance

Two forms of edit distance were used in the experiments. The Levenshtein *edit distance* between two strings can be described as the minimum number of edit operations (e.g. insertions, deletions and substitutions) that are needed to transform the first string into the second one. Equation (2) demonstrates the recurrence relation for minimal Levenshtein edit distance, in which function $r(s_i, t_j)$ returns 0 in case of a two identical characters and 1 otherwise.

$$\begin{aligned} edit(0,0) &= 0 \\ edit(i,0) &= i \\ edit(0,j) &= j \\ edit(i,j) &= min[edit(i-1,j)+1, \\ &\quad edit(i,j-1)+1, \\ &\quad edit(i-1,j-1)+r(s_i,t_j)] \end{aligned} \quad (2)$$

Code:	0	1	2	3	4
Letters:	aeiouy	bp	ckq	dt	lr
Code:	5	6	7	8	9
Letters:	mn	gj	fpv	sxz	csz

Table 2: Editex letter groups

We also used *Editex* [24], which can be described as a phonetic version of the Edit Distance. Table 2 shows the letter groupings used in the matching process. The recurrence relation of Editex was modified to be local rather than global in its alignment of strings as shown in equation (3), due to the different length between query and song being matched. Function $d(a,b)$ is a redefined version of $r(a,b)$ used in Edit Distance, and is defined in more detail by Zobel and Dart [24].

$$\begin{aligned} edit(0,0) &= 0 \\ edit(i,0) &= 0 \\ edit(0,j) &= edit(0,j-1) + d(t_{j-1}, t_j) \\ edit(i,j) &= min[edit(i-1,j) + d(s_{i-1}, s_i), \\ &\quad edit(i,j-1) + d(t_{j-1}, t_j), \\ &\quad edit(i-1,j-1) + r(s_i, t_j)] \end{aligned} \quad (3)$$

3.3 Phonetic matching

The phonetic matching technique *SAPS* consists of three steps, described in Gong and Chan [4]. The modified version of SAPS, known as SAPS-L used in [18] and this paper does not perform global alignment on lyrics to avoid substitution penalties caused by the big length difference between lyrics and queries. SAPS-L slightly varies in all the three phases:

1. Preprocessing: SAPS-L transforms the plain text into a canonical form in order to obtain an accurate syllable segmentation. For example, it maps "tjV" (where V is any vowel) to "chV" and "ph" to "f".

2. Segmentation: The strings are segmented into "syllables". For example, "dancing" is segmented thus: "DanSing".

3. Alignment and Similarity Calculation: Alignment is similar to edit distance, but with more complex scoring. Table 3 are the parameters defining the scores for substitutions and gaps, s_n representing a substitution and g_n a gap. The initial condition set for the local alignment of the lyrics and global alignment of the query is given in equation 4. For further details, see Ring and Uitdenbogerd [18].

$$\begin{aligned} M[i,j] &= max \\ M[0,0] &= 0 \\ M[i,0] &= 0 \\ M[0,j] &= M[0,j-1] + g(S_2[j], -) \end{aligned} \quad (4)$$

Constant	Meaning	Value
s_1	match within strings	1
s_2	mismatch within strings	-1
s_3	start of a syllable	-4
s_4	match starting syllables	6
s_5	mismatch starting syllables	-2
g_1	gap not at start of syllable	-1
g_2	gap at start of syllable	-3

Table 3: Parameters defining the scores for substitutions and gaps

4 Experiment

Our aim was to test several matching techniques for effectiveness when used on words extracted from sung queries. In particular, we compared word-based techniques with character or phoneme-based matching, as well as combinations of both approaches, since this can often lead to better results [8, 9, 19]. We used the Sphinx decoder [10] for speech recognition, with input data as described in Section 4.1.

For the retrieval effectiveness evaluation, we assumed that a retrieved song was relevant if it was the same song as the query. For our collection this meant that there was only one relevant answer for each query. As such, we used reciprocal rank and success@n measures of effectiveness.

We explore two variables associated with the problem: query length and speech recognition correctness. Query length was achieved by truncating to a specific number of words, selecting the first 5, 10, and 15 query terms. Three queries had a length in the range 11–14. Creating speech recognition correctness as the second continuous variable involved splitting queries into four buckets with varying level of correctness. Queries were sorted in increasing order of correctness and grouped into different buckets (levels of correctness were 0%, 20%, 37%, 73%). Note that the overal overage of correctness being 32.5% for all the buckets is demonstrated in Table 5. For the 50 queries, two buckets contained 13 queries, two contained 12 queries. This allowed us to study the behaviour of techniques when queries are entirely incorrect (0%) and relatively correct (20-37-73%). Correctness was calculated based on exact matching of the words, therefore, we expected phoneme/character based techniques to show marginal success for the correctness being as low as 0, due to the existence of similar sounding words with different spellings.

4.1 Data Collection

Approximately three hours of training data was gathered from a speech and a singing database. The speech data used for training was about two hours in length and did not have any specific context. It was collected from the CMU Assignment package[1]. The singing collection consisted of one hour of singing in English from a variety of musical genres by two authors of the paper: one beginner male and one experienced female. Our singing collection from the male singer was about 40 minutes long, consisting of 27 songs and the rest of the singing material sung by the female singer consisted of 16 songs, approximately 20 minutes long. Some singing files were recorded using the Audacity software version 1.2.6 and configured to the setting which matched the speech data while others were recorded using Cubase.

4.2 Manipulating the Singing and Lyrics Database

The recordings were either made at 16,000 Hz sample rate in mono, or converted to that format as per the requirements for the speech recognition system. The recorded songs were segmented into 30 seconds chunks including silence gaps and Mel Frequency Cepstral Coefficients (MFCC) – the most salient features for training the speech recognition system [12] were extracted using the sphinx_fe program.

We gathered a lyrics database with 2,359 songs in English from the authors of [18] and we extend it to include the 43 lyrics used in our training set. The database was annotated and all the punctuation removed except the apostrophes, which had led to some inconsistencies during the early experiment, when removed. For instance, "I'll" has a different pronunciation to "Ill", Finally, all the words were reduced to lower case, and extra spaces removed.

The dictionary contains 2,262 unique words made up of 40 distinct phones and the filler dictionary has two symbols representing the silence gap which occurs at the beginning and in the middle, <s >, or at the end of every song $</s>$. We used sphinx web toolkit [2] to automatically generate the language model which constructs two n-gram models by default: unigram and bigram.

4.3 Speech Recognition Results

For the training purpose we sliced all the singing data to segments with length average of approximately 30 seconds. From the collection of 30 seconds tracks, we manually selected fifty queries. Depending on the length of the original track, we select 0 to 6 segments from each song, each representing a query. We calculated correctness for the truncated version of our query set, containing 10 terms for each query, and the original set with 15 terms on average, per query. Accuracy and error rates are shown in Table 4 and 5. Correctness was calculated using the formula:

$$Correctness = \frac{N-D-S}{N}$$

where N is the total number of words in the original lyrics, D is the number of deletions and S is the total

[1] available at `http://www.speech.cs.cmu.edu/15-492/assignments/hw1/hw1_data.zip`

[2] `http://www.speech.cs.cmu.edu/tools/lmtool-new.html`

number of substitutions. Accuracy was calculated as follows:

$Accuracy = \frac{N-D-S-I}{N}$
where I is the number of insertions.

Total words	Percent correct	Error	Accuracy
1155	40.78%	79.22%	20.78%
Insertions	Deletions	Substitutions	
231	34	650	

Table 4: Sentence and word accuracy generated for original query set

Total words	Percent correct	Error	Accuracy
493	32.5%	70.91%	29.92%
Insertions	Deletions	Substitutions	
19	15	314	

Table 5: Sentence and word accuracy generated for truncated query set

The results are fairly poor due to the small training set, and limited control over how the singing data was used compared to the speech data. However, we were interested in how retrieval techniques work across a range of recognition accuracies, and a low starting point allows this to be explored over a wider range.

4.4 Retrieval Results

In this section we analyze the retrieval results in detail, compare and evaluate the effect of different variables on the result sets. We measure the effectiveness of the results using success at 10 and to make a better distinction between the 4 top techniques we use reciprocal rank which shows the rate of success at 1 as well.

Initially we compare 5 techniques using the queries made up of the first 10 words in the raw speech recognition output. Table 6 shows that the best technique overall is SAPS-L finding the best match in top 10 results for 58% of the queries, while both word-based techniques (Word Count and Okapi) performed poorly. The t-test calculated on these results shows that the only technique to be statistically significantly worse than any other is the baseline Word Count with p value of 0.0041. We believe that a larger test collection can potentially reveal statistical difference amongst the top 4 techniques.

Word Count	SAPS-L	Editex	Okapi	Edit Distance
25.25%	58%	51.75%	37.5%	56.25%

Table 6: Percentage of success @ 10

Figure 1 shows that effectiveness increases with query length. Editex and SAPS-L appeared to benefit most from the longer queries with an increase of 12%.

Figure 2 shows the difference in retrieval effectiveness for queries binned into the four buckets depending on recognition accuracy for queries of ten words. Next we combine some of the word based and character based techniques to find the best

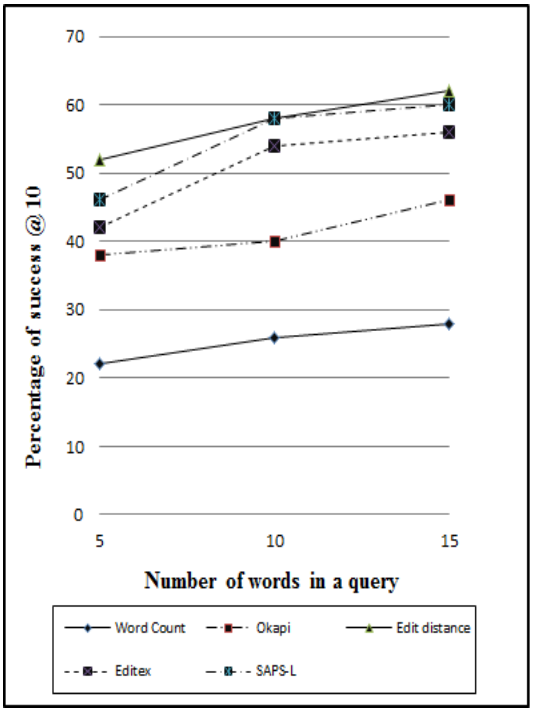


Figure 1: Trend in retrieval effectiveness with varying query length

Okapi-Edit Distance	Okapi-Editex
58%	56%

Table 7: Percentage of success @ 10 for combination of techniques

combination, using the same set of queries used for the previous experiment demonstrated in Figure 2 and results are shown in Table 7. Combination of techniques involved ranking all the lyrics twice for the two ranking techniques being combined against the given set of queries as shown in equation 5. We have tried different parameters to weight each ranking score and selected the most promising ones that led to better retrieval results. In the case of Okapi, the higher score implies more similarity, and for distance based rankings, higher similarities get lower scores.

$$\begin{aligned} \text{okapi-edit distance} &= 0.5(O) - 2(ED) \\ \text{okapi-editex} &= O - 2(EX) \end{aligned} \quad (5)$$

where
O : score generated by Okapi
ED : score generated by Edit Distance
EX : score generated by Editex

Combining Okapi and edit distance and SAPS-L appear to be the best techniques, however, no single technique is robust against varying speech recognition accuracy. Moreover, combinations of different pair of techniques did not lead to any improvement over the al-

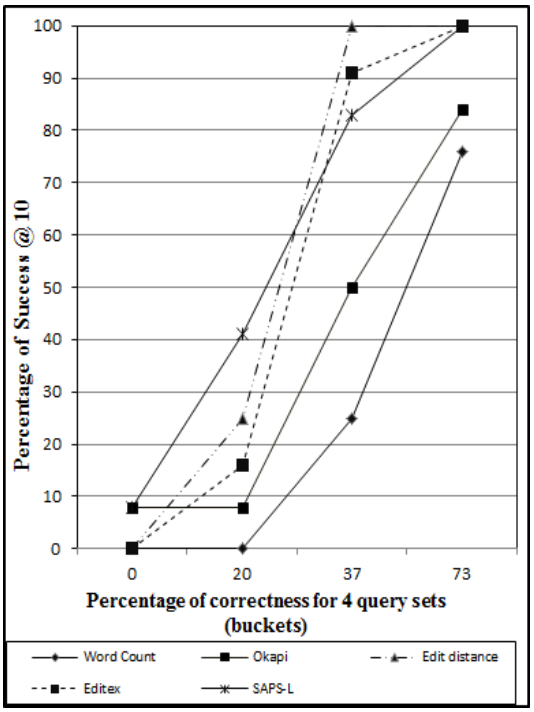


Figure 2: Trend of the retrieval techniques queried against four different sets of queries

ready combined techniques. In order to make better distinction amongst the top four techniques, we calculate reciprocal ranks for each of the queries. We intended to discover which technique is reliable and for what point of correctness. Figure 3 shows reciprocal ranks for each of the 50 queries plotted along the y axis with varying level of correctness along the x axis. Note that correctness was calculated for each individual query. We can see from the figure that SAPS-L performs 100% accurately for correctness level ranging from 80 to 100%. The combination of Edit Distance and Okapi, has the most desirable confidence point, finding the best match at top one ranked results, given that queries are at least 50% accurate. However, the marginal success of some techniques at 0% level of correctness, "Okapi" in particular, were questionable. Table 8 shows the query that retrieves the correct match, despite being highly inaccurate. This query has one word overlap with the target lyric which has a rare occurrence in the test collection (exploited by the Okapi), and is repeated many times in the original lyric, therefore giving a high score to the target lyric. The wrong position of the correct word, however, doesn't increase the correctness, which is 0%.

5 Conclusion

In this paper we attempted to learn ways to improve the effectiveness of query by singing with words, which uses speech recognition to train and recognize user queries. We have explored and tested some of the common matching techniques that have been tested

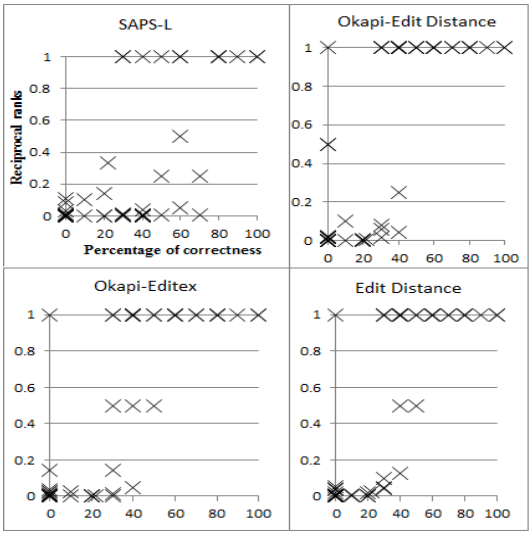


Figure 3: Reciprocal ranks for top 4 techniques

in the context of previously published lyric-based retrieval.

Despite the faulty output produced by the speech recognition, some matching techniques performed reasonably well, although none of the four top techniques were statistically superior to the others. However, the simple "Bag of Words" technique previously used in Mesaros and Virtanen [13] as a proof of concept proves to be persistently and statistically inferior to the other six techniques.

Original:	happy	birthday	sarah	chan	happy	birthday	sarah	chan	happy	happy
Hypothesis:	on	to	have	died	sarah	chan	blind	and	we	sang

Table 8: Speech recognition hypothesis for a noisy query

There is much scope for future work. There are many alternative matching techniques that could be effective for matching transcribed lyrics. It would also be useful to compare recognition output that is naturally at a higher level of accuracy and to classify singing data based on the characteristics of singer's voice.

References

[1] R. Bhagat and E. Hovy. Phonetic models for generating spelling variants. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 1570–1575. Morgan Kaufmann Publishers Inc., 2007.

[2] W. Bruce Croft. Combining approaches to ir (invited talk). In *DELOS Workshop: Information Seeking, Searching and Querying in Digital Libraries*, 2000.

[3] H. Fujihara and M. Goto. Three techniques for improving automatic synchronization between music and lyric. In *In proceedings of IEEE International Conference on Audio, Speech and Signal Processing*, pages 69–72, Las Vegas, USA, 2008.

[4] R. Gong and Tony K.Y.Chan. Syllable alignment: A novel model for phonetic string search. In *IEICE Transaction on Information and Systems*, pages 332–329, 2006.

[5] H. Hirjee and D. G.Brown. Solving misheard lyric search queries using a probabilistic model of speech sounds. *11th International Society for Music Information Retrieval Conference ISMIR*, pages 147–152, 2010.

[6] E. Joliveau, J. Smith, and J. Wolfe. Vocal tract resonances in singing: The soprano voice. *Journal of the Acoustical Society of America*, 116(4):2434–2439, October 2004.

[7] K. Spärck Jones, Steve Walker, and Stephen E. Robertson. A probabilistic model of information retrieval: development and comparative experiments - part 1. *Inf. Process. Manage.*, 36(6):779–808, 2000.

[8] B. Kantor, P. Cool, and Quatrain. Combining evidence for information retrieval. In Harman, editor, *Text Retrieval Conference(TREC)*, pages 35–44. National Institute of Standards and Technology Special, 1993.

[9] J. Lee. Combining multiple evidence from different properties of weighting schemes. In *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 180–188, 1995.

[10] K.F Lee, H.W. Hon, and R. Reddy. An overview of the sphinx speech recognition system. In *IEEE Transactions on Acoustics, Speech and Signal Processing*, pages 35–45, 1990.

[11] C. J. Leggetter and P. C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer Speech and Language*, pages 171–185, 1995.

[12] B. Logan et al. Mel frequency cepstral coefficients for music modeling. In *International Symposium on Music Information Retrieval*, volume 28, page 5. Citeseer, 2000.

[13] A. Mesaros and Tuomas Virtanen. Automatic recognition of lyrics in singing. *EURASIP J. Audio, Speech and Music Processing*, 2010.

[14] Corinna Ng, R. Wilkinson, and J. Zobel. Experiments in spoken document retrieval using phoneme n-grams. *Speech Communication*, 32(1-2):61–77, 2000.

[15] Corinna Ng and J. Zobel. Speech retrieval using phonemes with error correction. In *SIGIR*, pages 365–366, 1998.

[16] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applicatoin in speech recognition. In Murray Hill, editor, *In Proceedings of the IEEE, Vol. 77, No.2*, pages 257–284, 1989.

[17] H. Raghavan and J. Allan. Matching inconsistently spelled names in automatic speech recognizer output for information retrieval. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 451–458. Association for Computational Linguistics, 2005.

[18] N. Ring and Alexandra L. Uitdenbogerd. Finding 'lucy in disguise': the misheard lyric matching problem. In *Prodeedings of AIRS 2009*, pages 157–167, 2009.

[19] Joseph A. Shaw and Edward A. Fox. Combination of multiple searches. In *TREC*, pages 35–44, 1994.

[20] M. Suzuki, Toru Hosoya, Akinori Ito, and Shozo Makino. Music information retrieval from a singing voice using lyrics and melody information. *EURASIP J. Adv. Sig. Proc.*, pages 151–151, 2007.

[21] Chung-Che Wang, Jyh-Shing Roger Jang, and Wennen Wang. An improved query by singing/humming system using melody and lyrics information. In *11th International Society for Music Information Retrieval Conference*, pages 45–50, 2010.

[22] X. Xu, M. Naito, T. Kato, and K. Kawai. Robust and fast lyric search based on phonetic confusion matrix. In K. Hirata and G. Tzanetakis, editors, *Proc. 10th International Society ofor Music Information Retrieval Conference*, pages 417–422, Kobe, Japan, October 2009. ISMIR.

[23] J. Zobel and Philip W. Dart. Finding approximate matches in large lexicons. *Softw., Pract. Exper.*, 25(3):331–345, 1995.

[24] J. Zobel and Philip W. Dart. Phonetic string matching: Lessons from information retrieval. In *SIGIR*, pages 166–172, 1996.

Click Log Based Evaluation of Link Discovery

David Alexander
Dept. of Computer Science
University of Otago
Dunedin, New Zealand
dalexand@cs.otago.ac.nz

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

Alistair Knott
Dept. of Computer Science
University of Otago
Dunedin, New Zealand
alik@cs.otago.ac.nz

Abstract *We introduce a set of new metrics for hyperlink quality. These metrics are based on users' interactions with hyperlinks as recorded in click logs. Using a year-long click log, we assess the INEX 2008 link discovery (Link-the-Wiki) runs and find that our metrics rank them differently from the existing metrics (INEX automatic and manual assessment), and that runs tend to perform well according to either our metrics or the existing ones, but not both. We conclude that user behaviour is influenced by more factors than are assessed in automatic and manual assessment, and that future link discovery strategies should take this into account. We also suggest ways in which our assessment method may someday replace automatic and manual assessment, and explain how this would benefit the quality of large-scale hypertext collections such as Wikipedia.*

Keywords Information Retrieval, Hypertext

1 Introduction

Link discovery is the automatic generation of new hyperlinks in existing hypertext. In recent years, it has seen substantial development within the information retrieval community. This is likely due to the similarity between link discovery and search: both tasks involve retrieving documents that are *relevant* to an information need, be it expressed in a *search query* or in the *anchor* (the clickable text) of a hyperlink. By making the assumption that topical relevance solely determines hyperlink quality, one can split the task of link discovery neatly into the two subtasks of *selecting appropriate anchors* and *finding relevant targets* for those anchors.

We challenge this assumption. While topical relevance is often useful in hyperlinks, it is not the only useful quality, nor do existing evaluation methodologies necessarily measure it in the most appropriate way. We propose that instead of making subjective judgements of relevance, we should instead measure the usefulness of hyperlinks directly by analysing recorded user behaviour. We do this using click logs, which are readily obtainable and can contain vast quantities of data. In doing so, we assume nothing about the nature of useful hyperlinks — only about the nature of users' interactions with such links.

Proceedings of the 16th Australasian Document Computing Symposium, Canberra, Australia, 2 December 2011.

In this article, we introduce a set of metrics for hyperlink quality that are calculated from the data in a click log. We compare these to the two most popular existing methods of assessment — *automatic assessment* and *manual assessment* — which are explained below.

Automatic assessment involves establishing a ground-truth for relevant hyperlinks based on the existing hyperlinks in the corpus. A ranked list of hyperlinks produced through link discovery is compared to the set of links in this ground-truth, and the link discovery strategy under assessment is scored according to the degree of similarity observed. In producing this list of links, the link discovery algorithm cannot refer to the ground-truth; instead, it refers to an *orphaned* version of the hypertext document to be linked: that is, one that has had its incoming and outgoing links removed. Other documents in the corpus are available to the algorithm in unorphaned form. Manual assessment also uses a ground-truth but establishes it from the subjective judgements of human assessors.

Our goal is to use click logs to develop an assessment method that can replace automatic and manual assessment while retaining the advantages of both. This could ultimately enable link discovery to enter mainstream use, where link discovery could be kept in check by rigorous and automatic user evaluations generated from a live click log, and thus highly experimental link discovery strategies could be used without fear of low precision. To determine the possible implications of such a replacement, we investigate the following research questions in this article:

1. Are click-based assessments capable of distinguishing topically relevant from topically non-relevant hyperlinks?
2. How does the use of click-based assessment affect the ranking of existing link discovery strategies from the INEX evaluation forum?

We compare the results of automatic and manual assessment with the results of click-based assessment, both on a per-hyperlink basis and across the entire test

corpus (a set of Wikipedia articles). We examine the ranking that each assessment method gives to a particular set of link discovery algorithms, and find that some perform well under our assessment method, some perform well under the existing assessment method, but few perform well under both. The consequence of this is that neither topical relevance nor the structure of existing Wikipedia hypertext appear to determine user behaviour.

2 Prior Work

In this section, we outline the qualities of the existing assessment methods, so that in the next section we may show how our method improves upon them.

2.1 Development of assessment methods

The INEX Link-the-Wiki track ran annually from 2007 to 2010. It assessed link discovery algorithms on the Wikipedia encyclopaedia, by providing *topics* (Wikipedia articles that had been "orphaned" by having their links removed) to be re-linked to the rest of the corpus. The algorithms produced ranked lists of links for each topic. Automatic assessment simply compares these links to the pre-orphan links.

It was through Link-the-Wiki that manual assessment was first introduced. Huang et al. [5] performed an experiment at INEX 2008 in which they added the pre-orphan links (i.e. the automatic assessment ground-truth) to the pool of links to be assessed, and then asked participants to manually assess the resulting pool. They found that many of the pre-orphan links were assessed as non-relevant by the manual assessors, suggesting that automatic assessment based on these links was not accurate.

These evaluation methods can be seen as the beginning of a progression: first, a criterion for hyperlink quality (namely relevance) is specified, and in automatic assessment a particular source (Wikipedia) is assumed to exhibit this quality due to *implicit* norms among its authors; then, with manual assessment, assessors attempt to judge this criterion *explicitly*. Our click log based method is intended as a third step in this progression, whereby the measurement of hyperlink quality is not based on *any* assumptions or judgements, but is instead carried out through direct observation of users.

2.2 Quality of assessment data

The principle of automatic assessment is that it harnesses a vast quantity of data of assumed high quality. When automatic assessment is used, we can expect that new documents added to a corpus will be hyperlinked with quality approximately as good as that of the existing documents. However, the research of Huang et al. [5], showing that Wikipedia contains many links assessed as non-relevant by assessors, calls this quality into question.

By contrast, manual assessment allows assessors to determine hyperlink relevance however they see fit. This would allow them to take into account arbitrarily complex use cases of the hyperlinks — if it were not extremely time-consuming for them to do so. Instead, manual assessment relies on the assessor's intuitions and is prone to inconsistency.

This inconsistency has been observed in *ad hoc* information retrieval in several studies, including that of Sanderson et al. [9] who note that one's criteria for relevance develop gradually as more documents are assessed — and we believe that similar results can be expected in link discovery.

The data used in manual assessment is also often incomplete, but this problem can be mitigated in several ways, for example by using metrics such as *bpref* (Buckley & Voorhees [2]), which ignores unassessed results.

2.3 Relevance

Relevance, as the term is used in the evaluation of search engines, means that search results are of use to a person with a stated information need. In link discovery, it means that the *topic* of the target document is appropriate for the anchor text as it appears in the source document.[1]

This entails a very simple relationship between the source and target documents: it is only the strength of the connection between *those two documents* that determines the strength of the link. Indeed, link discovery algorithms based on this formulation of relevance need only to perform two tasks: (1) finding phrases in the new document that refer to the topics of existing documents; and (2) deciding which such phrases represent *topically relevant* hyperlinks. The second of these tasks is often implemented with a simple document similarity metric, or even left out altogether (as in the highly successful *Structural Threshold* algorithm of Itakura & Clarke [6]).

These tasks are analogous to the tasks of selecting and ranking results for a search engine. However, search engines are not used in the same way as freeform hypertext, nor should they be evaluated in the same way. Search engines use hyperlinks to point to specific results, whereas freeform hypertext uses them to provide context. This context can take many forms, some of which fit directly into the categories of *non-relevant* hyperlinks identified by Huang et al. [5] (such as links to Wikipedia articles listing general events in specific periods of history).

3 Our approach

A hypertext browsing session does not necessarily have the same structure as a search engine session. Though

[1] This distinction exists because users of hypertext do not specify *information needs* explicitly (as search engine users do with queries). This means that users' information needs must be assumed to relate to the documents they view.

users often arrive at a hypertext with a certain information need, there may be no single document that satisfies it completely; instead, multiple documents, and even the structure of the links between them, can provide valuable information to the user. This raises the question of what unit of information retrieval activity (e.g. the *session*, the *document*, the *passage*, and so on) should be measured.

Our goal is to evaluate individual hyperlinks, rather than entire hypertexts, because the granularity of per-link assessments is ideal for directing improvements to the hypertext. We therefore favour a bottom-up approach to assessment: instead of assigning click based scores to entire sessions (and using these to calculate scores for links), we directly assign a score to each link based on behaviour recorded from users interacting with that link.

3.1 Click log based metrics

One of the advantages of our approach is that it directly measures the effectiveness of hyperlinks in real-life user interactions. This involves first recording data from users and then interpreting that data. There are several ways of gathering data on user behaviour, but we believe that *click log analysis* is the best way because it allows for a continuous stream of data, unencumbered by the bias of explicit judgements, to be gathered without human intervention or the disruption of users.

The raw data of a click log consists of a sequence of page-views (with timestamps) for each user. No detailed information is recorded about users' interaction with their browsers, such as the specific anchors that are clicked to generate each page-view.

The data for our metrics, which are detailed below, is collected as follows. First, the requests recorded in a click log (the log we use in this work is described in Section 4.1) are grouped by user, and each user's entire history of requests is grouped into *sessions* such that no two consecutive requests within a session have a gap of more than 30 minutes between them. Then, page-views are associated with the links that appear to have caused them, if any.[2]

We introduce four metrics for calculating scores from this data. Each of our metrics builds on the previous, encapsulating more information about user behaviour.

The various metrics are all based on the idea of users implicitly "voting" for the quality of links. We assume that users have a limited number of votes to cast (because their time is limited) and therefore a vote for one link is cast at the expense of votes for lower-quality links. We also assume that even when users' votes are individually binary, as more votes are collected they will tend towards a proportionate representation of each link's quality.[3] We need not understand what *causes* each link to be worthy of inclusion: it may sometimes be strict relevance; other times, the context that it provides to the user; and perhaps occasionally, sheer curiosity on the part of users.

Our simplest metric is based on the assumption that high-quality links will be popular:

1. **Click-Voting (CV)** counts every click on a link as a vote in support of that link. It is assumed that users will be able to estimate the quality of links before clicking on them, and will therefore be more likely to click on high-quality links. $C(l)$ denotes the set of clicks recorded for the link l.

$$U_{\mathrm{CV}}(l) = |C(l)|$$

However, CV does not account for the problem whereby links that occur on popular pages receive unduly high numbers of votes. This causes the scores of links on the most frequently visited pages to dwarf all others. We address this problem by introducing another factor:

2. **Proportional Click-Voting (pCV)** measures the number of votes for each link as a proportion of the number of "voters" — i.e. visitors to the page containing the link. $V(l)$ denotes the set of page-views recorded for the source document of link l, whether or not they result in a click for that link.

$$U_{\mathrm{pCV}}(l) = |C(l)| \cdot \frac{1}{|V(l)|}$$

pCV is the probability that a visitor to the source page of a link will click that link, assuming each visitor clicks exactly one link. However, this probability is only an accurate measure of quality if the page visitor examines every link to decide whether to click it, and therefore the decision regarding a link's quality is made with full knowledge of the link's existence.

This is rendered unlikely by the phenomenon of *position bias*, in which links occurring early in a document are highly likely to be clicked regardless of the quality of links occurring later. This is assumed to be because users do not always view the later links. It can be expressed as a diminishing probability of links being clicked the further down the page they appear. We introduce a further factor to address this problem:

3. **Biased Proportional Click-Voting (bpCV)** corrects for position bias by estimating the probability that the link would be clicked if position bias did not exist. The model we use to calculate the correction factor, denoted by $B(l)$, is described in Section 3.2.

$$U_{\mathrm{bpCV}}(l) = |C(l)| \cdot \frac{1}{|V(l)|} \cdot B(l)$$

[2] This is determined for each page-view by finding the most recent previously viewed page in the same session that contains a link to the requested page.

[3] Specifically, the votes should approximate the *average* perceived quality over all users, since different users may find links to be more or less useful depending on their needs.

The normalisations in pCV and bpCV account for *false negatives* — i.e. links that are unused for reasons other than low quality. However, users cannot necessarily determine the quality of a link until after clicking it, which means that the use of a link does not necessarily imply that its quality is high. We address this by observing the behaviour of the user *after* clicking the link:

4. **Normalised Reading Time (nRT)** counts users' votes proportionally to their individually measured information gain upon clicking a link. This is represented by the amount of time that the user spends on the resulting page, measured as the number of seconds between the page request in question and the one that follows it.[4] This metric is normalised in the same way as above: i.e. by the number of source page visitors and the position bias. $R(c)$ denotes the reading time recorded for the page-view resulting immediately from the link-click c.

$$U_{\mathrm{nRT}}(l) = \sum_{c \in C(l)} R(c) \cdot \frac{1}{|V(l)|} \cdot B(l)$$

We prefer Normalised Reading Time as a click-based metric, because it is an indicator of not only the number of users who found a link useful, but of *how much* useful information was gained. We believe that the time spent reading a page can be expected to correlate with the user's information gain since the assimilation of information is a time-consuming process which users are unlikely to perform on information that is not useful to them.

3.2 Position bias

While the other constituents of the click-based metrics can be calculated straightforwardly, the position bias is a quantity that must be estimated using a model of user behaviour.

Position bias can be modelled in numerous ways: we use the "cascade model" introduced by Craswell et al. [3], which states that the probability of a given link being clicked is dependent on both the user's perception of the relevance of the link and the probability that *none of the previous links were clicked* before reaching it. It assumes that the user views the links in a document from top to bottom, and that once a link is clicked, the user does not return to click further links.

If c_i is the event of clicking link l_i, and v_i is the event of viewing link l_i, the equation given by Craswell et al. for the cascade model shows how to calculate $P(c_i)$ given $P(c_i|v_i)$ (assuming the reader is already on the page containing l_i). We have $P(c_i)$ empirically — it is equivalent to the pCV metric — so we rearrange the equation to give $P(c_i|v_i)$, i.e. the probability of the link being clicked if position bias did not exist. To calculate this, we multiply the value of pCV with the following correction factor, derived from the equation in Craswell et al.:[5]

$$B(l_i) = \prod_{j=0}^{i-1} (1 - P(c_j) \cdot B(l_j))^{-1}$$

4 Experiments

In this section we present the results of two experiments.

The first experiment is a "sanity check" that establishes that the click-based metrics are sensitive to topical relevance (as well as to other qualities), and therefore are in general agreement with systems that measure topical relevance.

The second experiment determines the practical implications of using a click-based assessment method rather than manual assessment by performing INEX-style assessments on the runs submitted to INEX 2008. One set of assessments is produced for each of our four metrics, with each metric providing a score for each link. According to INEX tradition, Mean Average Precision scores are calculated for each run from the per-link scores. We also use the per-link scores to calculate Normalised Discounted Cumulative Gain scores. We then compare the rank-order of runs under these click-based assessments to the rank-order given by INEX manual assessment.

Before presenting the results of these experiments, we describe the data sources and our handling of them.

4.1 Data source

To gather data for click-based scores, we use the 2008 click log of the University of Otago student web proxy, which records all student web access through campus computers. Our anonymising process involves removing non-Wikipedia requests and personal information such as usernames, but allows us to separate the requests of the 17,635 unique users recorded in the log.

Although the recorded usage (and therefore our evaluation) pertains to a constantly changing Wikipedia, the INEX assessments against which we compare our evaluations are based on a static snapshot of Wikipedia. This snapshot was taken in 2005,[6] and included the 659,388 articles that Wikipedia contained at the time.[7] However, the 2,595,572 Wikipedia requests in the click log cover only 451,979 unique articles.

[4] This cannot be calculated for the last page-view in each session, since no page-view follows it. Instead, the reading time for the last page-view is estimated (pessimistically) as the average reading time across the session.

[5] If multiple anchors with the same target occur in a document, we use the first anchor to calculate this factor.

[6] For further information on the Wikipedia collection and how it is used in INEX assessments, consult the following papers from 2006 — the year in which the collection was introduced — Denoyer & Gallinari [4] and Malik et al. [8].

[7] The 2005 Wikipedia snapshot was used because it was the official document collection for INEX 2008. We chose INEX 2008 because it was run at approximately the same time as our click log was recorded.

Name	Significance	Significance (adj.)
CV	p = 0.000	p = 0.001
pCV	p = 0.004	p = 0.009
bpCV	p = 0.000	p = 0.000
nRT	p = 0.000	p = 0.000

Table 1: Results of *t*-tests comparing click-based scores for manually-judged relevant and non-relevant links. (Experiment 1)

Because of the limited scope of the click log, many hyperlinks in Wikipedia have no recorded clicks in the log. There are 24,094,179 hyperlinks in all of the requested Wikipedia articles, but only 473,510 of these have any recorded clicks. Of these, only 311 also have manual assessments. For each distinct hyperlink that is clicked, a set of click-based scores can be calculated from our four metrics.

In short, there are 473,510 distinct sets of click-based scores — one for each hyperlink — that contribute to the calculation of the performance metrics in our second experiment, but only 311 sets of click-based scores are used in the first experiment (because it only applies to the intersection of manual and click-based assessments). However, this number of assessable links is sufficient for our current purpose: to examine our new evaluation method and compare its results to those of the existing methods. In the Further Work section, we suggest sources of further click log data and ways in which the existing data could be extended to cover more links.

4.2 Experiment 1: compatibility

The first of our two research questions concerns the extent to which click-based metrics can distinguish between topically relevant and non-relevant links. We acknowledge that manual assessment, despite its problems, is the most accurate way of determining the relevance (although not necessarily the overall quality) of hyperlinks. Therefore, we use INEX manual assessment as the benchmark for relevance in the first experiment.[8]

In this experiment, we use our click-based metrics to calculate scores for individual hyperlinks. We hypothesise that these scores will be higher for hyperlinks that are judged relevant by assessors than for those that are judged non-relevant, because we believe users will be inclined to click topically relevant links more often than others. We test this by taking the set of hyperlinks for which both click log data and manual assessments are available, and splitting it into two subsets — the *relevant* set and the *non-relevant* set — according to the manual assessments. We find that the average click-based score in the *relevant* set is indeed higher than in the *non-relevant* set, regardless of which of our four click-based metrics is used.

Because of the small sample size for this experiment (311 hyperlinks), we considered the possibility that the difference in click-based scores was due to chance. However, unpaired one-tailed *t*-tests (the results of which are shown in Table 1) show that the differences are significant to the 1% level.

The use of four separate click-based metrics necessitates a familywise adjustment to the *p*-values in these *t*-tests. Popular adjustments such as the Bonferroni correction are too conservative for our purposes because they assume that the experiments are statistically independent, which is not true in our case since each click-based metric is based upon the previous one. Instead, we use a correction suggested by Blakesley [1, pp. 37–38], which accounts for statistical dependence by using correlation coefficients to calculate the adjusted *p*-value.

These results suggest that our metrics are in general agreement with manual assessment: they can distinguish relevant links from non-relevant links as accurately as manual assessors can, while also providing a graded measure of hyperlink quality that manual assessors cannot provide.

4.3 Treatment of INEX Runs

Because the INEX runs specify multiple targets (a maximum of 5) for each anchor, the entire list of anchor/target pairs for a given document is not intended to be treated as a ranked result list: instead, the anchors are ranked, and the targets for each anchor are ranked separately. To reflect this, our click-based score for each anchor is calculated as the mean click-based score over all of the targets of that anchor. The list of anchors is then treated as a ranked list.

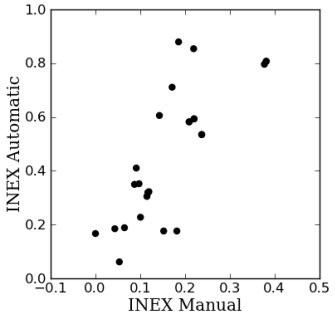
No matching is performed between the anchor positions (in the document) specified by INEX runs and the anchor positions in Wikipedia: instead, the quality of a link is calculated based on all clicks that have the same source and target documents as that link, regardless of anchor text. This is equivalent to INEX *file-to-file* evaluation, which is ideal for our purposes because it avoids the problems of matching anchors in different versions of documents when the click log does not record the necessary information to do so.

Source/target pairs found in a run but not in Wikipedia are considered "unassessed", and removed from the ranked list before assessment.[9] Source-target pairs that exist, but are never clicked, are given a click-based score of zero for all four metrics.[10]

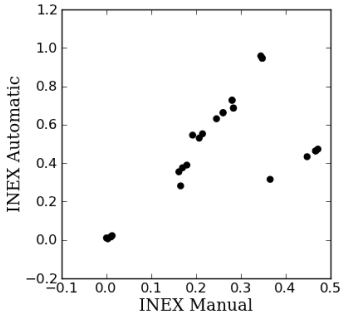
[8] We do not use automatic assessment for this experiment because the set of links that it would judge non-relevant is identical to the set of links for which click log data is unavailable.

[9] To determine whether this removal would bias the results, we performed the same processing on the INEX manual assessments and compared the resulting ranking with the original ranking. The Spearman's rank correlation coefficient between the two rankings is 0.853, which we consider high enough to indicate a negligible difference.

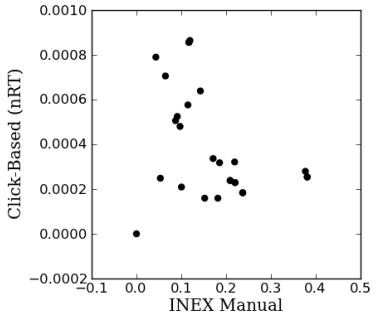
[10] All of the links that we assess occur in documents that are visited at least once in the log. This means that although the click



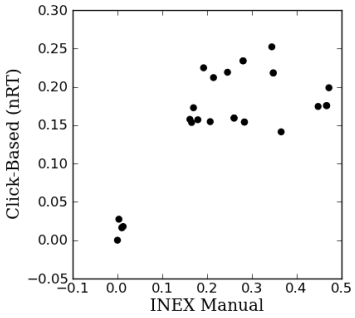
(a) MAP scores for runs under automatic and manual assessment.



(b) nDCG scores for runs under automatic and manual assessment.



(c) MAP scores for runs under INEX and click-based assessment, using nRT metric.



(d) nDCG scores for runs under INEX and click-based assessment, using nRT metric.

Figure 1: Scatter plots showing the difference between the automatic/manual comparison and the manual/click-based comparison. (Experiment 2)

Of the 32 runs submitted to INEX 2008, not every run includes all 50 assessed topics. To avoid the unfairness that would occur if we used incomplete topics in our evaluation, we remove 4 of the runs[11] (because they include too few topics) and only assess according to a particular set of 22 topics,[12] chosen so that: first, each of the remaining runs includes all of the topics in the set; and second, each of the topics in the set includes links for which both manual and click-based assessments were available.

4.4 Experiment 2: correlation

We use two performance metrics to aggregate the per-link click-based scores (and relevance judgements, for the INEX assessments) across the entire set of topics used. These metrics are Mean Average Precision (MAP) and Normalised Discounted Cumulative Gain (nDCG). We use MAP because it is well understood in the information retrieval community, and nDCG because it is designed specifically for use with graded judgements such as ours. To use MAP with graded judgements, we modify the standard formula for average precision as follows (where L is the set of links specified by a run for a given topic):

$$\mathrm{AvgP} = \frac{1}{|L|} \sum_{l \in L} \frac{\text{Total of all scores down to } \mathrm{rank}(l)}{\mathrm{rank}(l)}$$

The formula for nDCG is used without modification, and is defined as follows. (iDCG is the *ideal DCG*: the DCG that a run would score for a given topic if it returned all possible relevant/useful links in the best possible order.)

$$\mathrm{DCG} = \mathrm{rel}(l_1) + \sum_{l \in L, l \neq l_1} \frac{\mathrm{rel}(l)}{\log_2 \mathrm{rank}(l)}$$

$$\mathrm{nDCG} = \frac{\mathrm{DCG}}{\mathrm{iDCG}}$$

Because several institutions participated in the Link-the-Wiki track in 2008, we are able to evaluate a wide variety of link discovery strategies using our

count of a link may be zero, none of the normalisations of our click-based metrics are ever undefined due to division by zero.

[11] A fifth run is also removed due to improper specification of anchors.

[12] The list of topics is as follows: *Air conditioning*, *Anne Rice*, *Boston Celtics*, *Boston Tea Party*, *Chamonix*, *Data mining*, *De Stijl*, *Greenhouse gas*, *Information retrieval*, *Love*, *Mainz*, *Mouse*, *Natural gas*, *Near and far field*, *New Zealand Labour Party*, *Personal name*, *Ratan Tata*, *Search engine*, *Ski jumping*, *Studio Ghibli*, *Symphony*, and *Togo Heihachiro*.

	Auto	Manual	CV	pCV	bpCV	nRT
Auto	1.00					
Manual	0.74	1.00				
CV	0.49	-0.03	1.00			
pCV	0.24	-0.27	0.88	1.00		
bpCV	0.09	-0.34	0.59	0.75	1.00	
nRT	0.05	-0.36	0.58	0.75	0.99	1.00

(a) Rank correlation matrix for MAP scores under different assessment methods.

	Auto	Manual	CV	pCV	bpCV	nRT
Auto	1.00					
Manual	0.61	1.00				
CV	0.72	0.39	1.00			
pCV	0.72	0.39	1.00	1.00		
bpCV	0.72	0.39	1.00	1.00	1.00	
nRT	0.77	0.57	0.89	0.89	0.89	1.00

(b) Rank correlation matrix for nDCG scores under different assessment methods.

Table 2: Correlation matrices for assessment metrics. (Experiment 2)

evaluation method. This includes the University of Amsterdam's algorithm, based on *link likelihood ratio*, and QUT's algorithm, based on *title matching*.

Most notably, however, it includes a number of implementations of the Itakura & Clarke [6] algorithm: from the University of Waterloo, the University of Otago, and Lycos. This algorithm was particularly popular in 2008 because of its high performance in 2007.

The purpose of our experiment is to examine the relationship between the existing assessment methods and our own. Because any change in assessment methodology is prone to causing changes in the rank order of runs, we compare the nature of these changes from automatic to manual assessment with those from manual to click-based assessment, rather than examining the latter in isolation.

The scatter plots in Figures 1a and 1b show the relationship between automatic and manual assessment (under MAP and nDCG, respectively). This relationship is mostly linear, albeit with a prominent group of outliers that perform well under manual assessment but not as well under automatic assessment when measured using nDCG.

Figures 1c and 1d show the relationship between manual and click-based assessment. At first glance, it appears that there is no correlation. However, closer inspection reveals a linear correlation up to a certain point, after which two lines appear, each encompassing runs that perform well under one assessment method but not the other. This effect is particularly noticeable under MAP, where the top-right quadrant of the scatter plot is conspicuously sparse.

This suggests that the relationship between high performance under the two assessment methods is not positive or negative, but is mutually exclusive in the existing set of link discovery algorithms. This does not mean that an algorithm cannot satisfy the criteria of both — only that no current algorithm does so.

We analyse the results presented in these scatter plots using Spearman's rank correlation coefficient.[13] Tables 2a and 2b show the correlation matrices for all assessment methods (automatic, manual and the four click-based metrics) under MAP and nDCG. Predictably, the click-based metrics correlate well with each other, especially under nDCG. Also, automatic assessment correlates well with manual assessment, but neither correlates as highly with the click-based metrics as the click-based metrics do with each other. Surprisingly, automatic assessment correlates better with click-based assessment than with manual assessment.[14]

5 Discussion

The differences in rank order of runs between INEX manual assessment and click-based assessment are not surprising, given that click-based assessment uses large quantities of user data rather than the judgement of a few assessors.

However, the implications of our click-based method range further than simply ranking link discovery runs. Using click logs as a data source enables assessment to be performed in ways that were not possible in past, such as by automatically checking the quality of links as they are added to and removed from an online corpus such as Wikipedia, and adjusting linking over time.

Furthermore, our work suggests that the established order of link discovery strategies is not final, but rather that it may be affected by criteria of hyperlink quality other than topical relevance. This is an important result because link discovery has recently come to be regarded as a solved problem, owing to the excellent performance of the Itakura & Clarke [6] algorithm under automatic and manual assessment.

Since this algorithm selects new links based on their probability of occurring in the existing hypertext, it disregards context and simply mimics the surface structure of the hypertext. Itakura et al. [7] acknowledge that it is surprising that their algorithm should perform so well given its simplicity. Our introduction of the click-based metrics is promising for the continuation of link discovery as an open field of research.

There are also practical advantages to using a click log based method rather than manual assessment. Click

[13] We choose Spearman's coefficient (rather than Pearson's coefficient) because it is non-parametric: i.e. it disregards the curvilinearity of the line and only takes into account the rank order of the data points. It has previously been established in information retrieval that the rank order of performance scores such as MAP and nDCG is more important that their absolute values.

[14] This appears to be a result of the very low-scoring cluster visible in the bottom-left corners of the nDCG graphs.

logs provide large quantities of data essentially for free, whereas manual assessment involves a high cost even to assess a small number of documents. Manual assessment typically uses a small number of assessors each performing a large number of assessments, whereas click logs record a large number of users each performing small tasks. According to the findings of Sanderson et al. [9] (that an assessor's assessment criteria tend to "shift" over time), manual assessment can also be expected to be less reliable than click log based assessment. Although it is too early to say that our method can replace manual assessment, it is clearly desirable that it be made able to do so. The results of our first experiment, showing that our metrics are sensitive to topical relevance, suggest that this may be possible.

6 Further work

The simplest improvement to our work would be to find larger and more diverse click logs to use. The click log that we use here does not have very wide coverage of Wikipedia articles, and it omits some information that could reasonably have been recorded, such as the *referrer* of each request. The best way to implement a click-based assessment system would be to use the click logs of the website that was being assessed (e.g. Wikipedia). However, even a large proxy log such as that of an ISP would be useful.

Also, by making some assumptions about what kinds of links are likely to share similar click-based scores, it would be possible to use a relatively sparse set of click-based judgements — such as those that we have generated from the University of Otago click log — to assess an entire corpus. This could be done simply by using known click-based scores to estimate the click-based scores of similar links (for example, links that shared the same anchor text and target document). However, this technique would have to be used carefully to avoid diluting the specificity of our metrics to the point where they was no longer dependent on users' information needs, as inferred from the context of links.

The availability of good click log data would also be improved by using "live" click logs. Click logs are typically produced by web servers and proxy servers as part of their normal operation, and they are therefore constantly being updated. However, research into click logs tends to be done on portions that are recorded over fixed time periods (for example, our click log is from 2008). If an assessment methodology similar to the one used here were instead used on a continuously updating click log — perhaps through a program installed on the web server for the website under assessment — its assessments could be used to automatically prune low-quality hyperlinks, and add new hyperlinks that were similar to existing high-quality ones. In conjunction with an automated link discovery system (which would also be applied continuously) this would automate much of the time-consuming task of hyperlinking new documents, while avoiding the low precision that has previously made automated link discovery impractical for live websites.

7 Conclusion

In this article, we have introduced a series of metrics of hyperlink quality which are based on user behaviour. These metrics can be used to calculate per-link scores (which could help hypertext authors decide which links to remove) or they can be used to evaluate entire hypertext collections according to standard information retrieval performance measures such as Mean Average Precision (MAP) and Normalised Discounted Cumulative Gain (nDCG).

We have justified these metrics theoretically, and verified that they produce similar relevance classifications to a baseline set of results from INEX manual assessment. We have also examined the relationship between INEX assessment (both automatic and manual) and our click-based metrics when applied to INEX runs. We have seen that runs from the currently available set diverge into specialised subsets — one for each assessment method — as their performance improves. This suggests that our criteria should be considered alongside the existing criteria when link discovery strategies are developed, and that the task of meeting these criteria presents substantial opportunities for further work in link discovery.

References

[1] R.E. Blakesley. *Parametric control of familywise error rates with dependent P-values*. Ph.D. thesis, University of Pittsburgh, 2008.

[2] C. Buckley and E.M. Voorhees. Retrieval evaluation with incomplete information. In *SIGIR*, pages 25–32. ACM, 2004.

[3] N. Craswell, O. Zoeter, M. Taylor and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM*, pages 87–94. ACM, 2008.

[4] L. Denoyer and P. Gallinari. The wikipedia xml corpus. In *ACM SIGIR Forum*, Volume 40, pages 64–69. ACM, 2006.

[5] W.C. Huang, A. Trotman and S. Geva. The importance of manual assessment in link discovery. In *SIGIR*, pages 698–699. ACM, 2009.

[6] K. Itakura and C. Clarke. University of Waterloo at INEX2007: Adhoc and Link-the-Wiki tracks. *Focused Access to XML Documents*, pages 417–425, 2008.

[7] K.Y. Itakura, C.L.A. Clarke, S. Geva, A. Trotman and W.C. Huang. Topical and Structural Linkage in Wikipedia. *ECIR*, 2011.

[8] S. Malik, A. Trotman, M. Lalmas and N. Fuhr. Overview of inex 2006. *Comparative Evaluation of XML Information Retrieval Systems*, pages 1–11, 2007.

[9] M. Sanderson, F. Scholer and A. Turpin. Relatively Relevant: Assessor Shift in Document Judgements. *ADCS 2010*, pages 60, 2010.

Efficient sorting of search results by string attributes

Nicholas Sherlock
Department of Computer Science
University of Otago
Otago 9054 New Zealand
nsherloc@cs.otago.ac.nz

Andrew Trotman
Department of Computer Science
University of Otago
Otago 9054 New Zealand
andrew@cs.otago.ac.nz

Abstract *It is sometimes required to order search results using textual document attributes such as titles. This is problematic for performance because of the memory required to store these long text strings at indexing and search time. We create a method for compressing strings which may be used for approximate ordering of search results on textual attributes. We create a metric for analyzing its performance. We then use this metric to show that, for document collections containing tens of millions of documents, we can sort document titles using 64-bits of storage per title to within 100 positions of error per document.*

Keywords Information Retrieval, Web Documents, Digital Libraries

1 Introduction

A document search engine typically takes a search query provided by the user and generates a list of documents which contain some or all of the terms in the query. This list of documents is sorted in a particular order, defined by a "ranking function". Common ranking functions assign an importance to each term in the search query using some metric, then for each term in the query, apply that importance to the number of occurences of the term in each document to give each document a score. The list of documents is then sorted using this score to present documents to the user with the highest scores first.

There are several such ranking functions that are of interest in search over online discussion forums. For example, the online forum software "phpBB" offers users the ability to order their results by document attributes such as post time, author name, forum name, topic title, and post title. In the case of text fields like names and titles, this is problematic, as it requires the search engine to have these fields available in text form for comparison sorting at search time. This consumes a large amount of memory. For example, in a collection of 14.8 million forum posts, simply storing the post title for each post requires 500 megabytes of memory.

Proceedings of the 16th Australasian Document Computing Symposium, Canberra, Australia, 2 December 2011.

In addition, the search engine must allocate memory to an index structure which allows it to efficiently retrieve those post titles by document index, which, using a simplistic scheme with 4 bytes required per document offset, would require an additional 50 megabytes of storage.

For search terms which occur in many documents, most of the memory allocated to storing text fields like post titles must be examined during result list sorting. As 550 megabytes is vastly larger than the cache memory available inside the CPU, sorting the list of documents by post title requires the CPU to load that data from main memory, which adds substantial latency to query processing and competes for memory bandwidth with other processes running on the same system.

In this paper, we examine several methods which can be used to reduce the memory requirements for storing document attributes in a form suitable for sorting. We examine the tradeoffs which can be made between sorting accuracy and the memory required for storage at search time. We then demonstrate the accuracy of our methods by applying them to a corpus of 14.8 million discussion posts taken from an online discussion forum called "Chicken Smoothie", and 22 million posts from the online discussion forum at "Ancestry.com".

2 Ranking

First, let us examine the way that the search engine calculates ranking scores for documents. For each search term in the query, an index over the document collection is consulted. This index maps the term onto a list of document IDs which contain that term, along with extra information about that term's appearance in each document (for example, a count of the number of times that term appears in the document). This information is passed to a "ranking function", which uses it to generate a ranking score for each document.

The scores generated by the ranking function for each term in the search query are combined together in a structure called the "accumulator". The accumulator can be represented as an array of integers of a fixed size, one integer for each document in the collection. Every element in the accumulator is initialised to zero at the beginning of the search. Then, the value returned from

the ranking function for each term in the search query is typically added (using simple arithmetic addition) to the value already present in the accumulator for that document, to give a new score for the document.

At the end of the search process, the search engine's accumulator is sorted in descending order, which brings the documents which the ranking function scored the highest (which are hoped to be the "most relevant" documents to the user's information need, based on their search query) to the front of the result list.

2.1 Pregenerated ranking

If a ranking function's value does not depend on the content of the user's query, then instead of computing it when a document is located at search time, it can be computed when the document index is first created (at "indexing time"). This pregenerated ranking can be stored as a file (a "pregen") consisting of an array of integers, one for each document in the collection. When the search engine starts up, it can read each of its pregen files into memory. Then at search time, rather than performing any computations to calculate the ranking score for a document, the search engine can simply read the precalculated value stored in the pregen and store that value directly into its accumulator to be sorted. This technique allows the pregenerated ranking to take advantage of the search engine's existing implementation of accumulator sorting. For example, the search engine may already implement special optimisations like partial sorting of accumulators for the case where only the top-K sorted documents are required[5].

If a ranking function is particularly expensive to compute (compared to reading a precomputed value from memory), speed gains can be made at search time, at the expense of the extra memory required to store the pregens. For example, one ranking function which is expensive to compute, but independent of the user's query, is Google's PageRank algorithm[4]. This ranking function essentially assigns a score to web documents based on how many incoming links they have.

If we could compute a compact pregenerated ranking for each of the document attributes that our users want to sort on, we could improve the speed at search time (by sorting the smaller pregen instead of the longer original strings) and reduce the memory required to store those text attributes. For example, if we could fit the 14.8 million post titles in the Chicken Smoothie collection into 64-bit integers, the pregenerated ranking would only require 110 megabytes of memory to store. That would allow the search engine to save 440 megabytes of memory that would otherwise be required to store complete post titles and an index structure for those titles.

3 Generating pregens by sorting

One way in which a pregenerated ranking on a textual document attribute can be generated is by sorting. At indexing time, the text attributes for each document are extracted and stored in temporary memory. When the indexing of the document collection is complete, the document attributes are sorted using a string comparison function. Each document's position in the list of sorted attributes then directly becomes its pregenerated ranking score. In this way, we generate a maximally-compact set of ranking scores for the document collection (having exactly as many distinct values as there are distinct attributes in the collection), which also perfectly encodes the relative ordering of the documents in the sorted list of attribute text. If each integer in the pregen is 32 bits large, approximately 4.3 billion distinct documents can be perfectly ranked using this method (2^{32}).

This method has several drawbacks. Because every attribute of the documents which must be sorted needs to be available at the end of indexing time, we must either allocate enough memory to store those values (e.g. 550MB for post titles in the Chicken Smoothie corpus), or avoid allocating memory by storing those values in some sort of disk structure (requiring at least 1 gigabyte of additional disk I/O, as we must both write the values to disk and read them in again).

Another drawback is that each value in the generated ranking is dependent on the attributes of every document in the collection. If a later change to the document collection results in the first post in the ranking being deleted (say, "aardvarks and apples"), every pregenerated ranking function for the collection is now invalid and must be recomputed.

This is also a problem in distributed search, where a document collection may be split into several chunks, with each chunk being indexed by a separate computer. The pregens created by each index will be incompatible, since they are based on different, incomplete fragments of the total document collection. If, during searching, a search engine consults several such distributed indexes, and retrieves a list of documents from each with corresponding scores from their pregens, it will be unable to merge those results using just the values of the pregens to create a list which is sorted by the ordering over the complete collection.

A better method for generating pregens would be able to compute a ranking score for a document attribute immediately upon reading it (which would eliminate the requirement to store those values until the end of indexing), and the computed value would be independent of all other documents in the collection (which would provide consistency between different distributed fragments of the complete document collection).

4 Approximate pregens

The immediate issue with a method that directly computes one ranking score from one attribute (examining no other attributes) is that the pigeonhole princi-

ple shows that the resulting pregen cannot, in general, perfectly rank the documents, as the previous approach could. The average post title's length in the Chicken Smoothie corpus is 35 bytes, and we hope to reduce that to an 8-byte integer for each document. Even in the best case where the resulting values are evenly distributed, each post title (in the universe of possible post titles) would alias with 1.1×10^{65} different titles (i.e. $\frac{2^{35\times8}}{2^{8\times8}}$), destroying the information about the relative ordering of those titles.

In order for this approximate pregen method to be effective, we hope for two things to be true. First, that there is significantly less information in the post titles than their raw byte counts would suggest, and secondly that the error in the resulting ranking is sufficiently small that our users will not notice the difference compared to a perfect ranking.

5 Baseline

The accuracy of an approximate pregenerated ranking can be measured against a baseline which is considered "perfectly ranked". This perfect ranking corresponds to the user's expectations for document ordering in the search results, which allows users to locate the documents that they are interested in in the results.

The Chicken Smoothie corpus consists of 14.8 million forum posts, which are overwhelmingly written in the English language. Topics in the Chicken Smoothie's "International Forum", which is intended to be a forum for posts written in non-English languages, comprise only 0.1% of the total number of topics in the corpus. Because of this language bias, it is reasonable for the perfect ranking to ignore the relative order of non-English characters, as most users are not aware of, and do not take advantage of, the "correct" ranking for these characters. We will also consider that accented characters are equivalent to non-accented characters for the purpose of sorting. The ordering rules for accented characters are different even among common languages such as French and German, so in a multilingual collection like Chicken Smoothie, there is no one collating sequence for accented characters which is useful for all users. We consider that uppercase letters are equivalent to lowercase letters, as this difference does not play a significant role in English text. For this baseline, we consider the relative ordering of punctuation characters to be significant, and require it to follow the same ordering as in ASCII. We will call this baseline the "standard baseline".

We also consider the performance of our pregen ranking against a more limited baseline. This baseline contains only the ASCII alphanumeric characters and the space character, ignoring all punctuation marks and non-English characters. The rationale of this baseline is that users are unaware of the correct ordering of ASCII punctuation characters or other special characters, and so are not able to take advantage of that ordering when reading the search results. We will call this baseline the "restricted baseline".

The baseline is defined by a comparison function which, when applied to the attributes of a pair of documents, either declares that the two documents are considered equivalent for the purposes of ranking (that is, their characters do not differ in a way that the baseline considers significant), or identifies that one of the documents must occur before the other in the final ranking. This comparison function is the collating sequence of the document collection.

6 Character encoding

The Chicken Smoothie corpus is Unicode text encoded using "UTF-8", which encodes the majority of the characters used in English text into single bytes whose encoding matches ASCII, and encodes more complex characters, such as characters with accents and characters from other languages, into multi-byte sequences. As the discussion in the Chicken Smoothie forum is primarily in the English language, Unicode characters outside the ASCII range are rare (about 0.1% of the byte total). Since ASCII is a 7-bit encoding (128 distinct codepoints), we can expect to perfectly encode 35-byte titles into about 30.6 bytes ($\log_{256}(128^{35})$ bytes). That falls far short of the 8-byte target.

6.1 Standard baseline

Since the standard baseline does not distinguish between uppercase and lowercase characters, we can reduce the number of ASCII codepoints that we encode by 26, by lowercasing the text before we process it. We also consider the fact that a title of a document may not contain any ASCII control codes, since document titles in our collection are single-lines of text (and so contain no newline control characters), and cannot contain tab characters because of web browser limitations. This allows the number of codepoints to be reduced by a further 32 characters. Since the baseline considers the relative order of non-ASCII characters to be unimportant, we merge all those characters into one encoded codepoint, which sorts after any ASCII codepoint. This character encoding has 90 codepoints in total. Using this encoding, we can encode 35-byte titles to text which users consider "perfectly ranked" (compared to the standard baseline) in $\log_{256}(90^{35}) = 28.4$ bytes. We call this simple 90-codepoint encoding the "printable ASCII" text encoding.

We developed a base-40 character encoding, which has distinct codepoints for alphanumeric characters and the space character, but merges all punctuation and non-ASCII characters into the remaining 3 codepoints. Those 3 codepoints are chosen to preserve the relative ordering between the groups of alphanumeric characters and the groups of punctuation characters. The first codepoint is allocated to those punctuation characters occurring before the character '0', the

second codepoint for those characters occurring before the character 'a', and the last codepoint is allocated to the characters which occur after the letter 'z'. The performance of this character encoding depends on punctuation characters being rare enough in the collection that the relative ordering within the groups of punctuation marks which are conflated do not significantly perturb the ranking.

6.2 Restricted baseline

The restricted baseline only contains lowercase alphanumeric characters and the space character. As with the standard baseline, text is converted to lowercase before processing, and characters with accents are converted to their unaccented equivalents. A trivial base-37 encoding can encode every character in the restricted baseline to a distinct codepoint. Base-36 encodes every character in the baseline except space (and so is useful in the extreme circumstance where the encoded string terminates before the first word of input is completed).

We also developed a base-32 encoding, which halves the number of codepoints allocated to storing numeric characters (every second digit from ASCII is conflated with the digit before it). This encoding trades off minor inaccuracy in the ordering of the (relatively rare) numeric characters in exchange for more available precision for encoding more-common characters. Bases which are powers of two are attractive for encoding text, because many of the arithmetic operations required for encoding strings can be reduced to simple bit shifts.

7 Encoded string compression

The characters encoded by our various encoding schemes must be combined together to give an integer for sorting. A simple method of achieving this is "radix encoding". Radix encoding treats the output as an integer of the same base as the number of possible symbols in an encoded character, and adds each encoded character to the output as a digit in that base, with the earliest characters in the string becoming the most significant digits of the encoded integer. If the final encoded character can not completely fit in the encoded integer, it is scaled down.

If the number of symbols used to represent encoded characters is r, and the size of the encoded integer is b bits, we have the following psuedocode for a radix encoder of a string:

$\mathit{chars\ in\ output} := \lfloor \log_r 2^b \rfloor$
$r_{\mathit{last}} := \frac{2^b}{r^{\mathit{chars\ in\ output}}}$
$\mathit{output} := 0$
for $i := 1$ **to** *chars in output* **do**
    $\mathit{output} := \mathit{output} \times r + \mathit{encode}(\mathit{read}())$
od
$\mathit{output} := \mathit{output} \times r_{\mathit{last}} + \frac{\mathit{encode}(\mathit{read}())}{r-1} \times (r_{\mathit{last}} - 1)$

The "read" operation retrieves the next character from the input string, and the "encode" operation applies the chosen character encoding (characters which have no representation in the character encoding are ignored and the next character is read instead). The radix encoder assumes that every symbol is equally likely, so assigns equal-length encodings to each symbol. When the actual probabilities of the symbols being encoded are known, it is possible to choose an encoding which uses shorter strings of bits to represent common symbols, and longer strings of bits to represent infrequent symbols. This allows space characters and vowels, which are the most common characters in our collection of English text, to have short encodings, while the uncommon letters Q and Z have longer encodings.

The most well-known variable-length coding scheme is probably Huffman coding[2]. Huffman coding selects a bit-string representation for each input symbol such that the average length of encoded strings which follow the expected symbol probability distribution is minimized. A similar coding scheme, Hu-Tucker coding[1], is additionally able to preserve the lexicographic ordering of the resulting encoded strings, and so would be suitable for pregen encoding. However, as both Huffman coding and Hu-Tucker coding assign codes for input symbols which are an integer number of bits long, they only accurately model input symbol probabilities which are negative powers of two. Input symbol distributions which deviate from that pattern are encoded slightly less efficiently than theoretically possible.

There are coding schemes capable of assigning representations to input symbols which are effectively a fractional number of bits long, while also preserving the lexicographic ordering of input strings. "Arithmetic encoding"[6] is one such scheme. The goal of the arithmetic encoder is to construct an interval which represents the string of symbols being encoded, then to arbitrarily choose an integer which lies in this interval. This integer represents the encoded string.

For example, consider an encoder which encodes strings of base-4 characters (a, b, c, d) with relative probabilities (4, 2, 1, 1) into a 6-bit integer. The initial interval is the full range of the output integer, $[0..64)$. This interval is subdivided into 4 segments, one segment for each of the possible first symbols of the string, according to the relative probabilities of the characters. The first encoded character is therefore represented by one of the intervals $[0..32)$, $[32..48)$, $[48..56)$, $[56..64)$. Imagine that the first character is an "a". The current interval now becomes $[0..32)$. To encode the next character, the interval is subdivided again, into $[0..16]$, $[16..24)$, $[24..28)$, $[28..32)$. If the next character in the string is a "c", the current interval now becomes $[24..28)$. At this point, there is not enough precision left in the output integer for each range to be distinct: $[24..26)$, $[26..27)$, $[27..27.5)$,

$[27.5..28)$. If the next character is an "a" or "b", it will be unambiguously encoded, but "c" and "d" are no longer distinct.

Each time an interval is chosen which lies entirely within the smaller half of the current interval, it is the same as choosing a zero bit as the next-most significant bit of the output integer (because the value of the midpoint of the interval, represented by a 1 bit in the output integer, did not need to be added). Conversely, choosing an interval in the upper half of the current interval outputs a 1 bit. Small subintervals, which represent less-likely strings, require more bits to specify as the interval must be bisected more times in order to accurately specify their position in the initial interval.

This encoding scheme is attractive as a pregen string compressor, as the resulting integer preserves the relative ordering between the input symbols—if the symbol "b" occurs after the symbol "a" in the character encoding being used, then all arithmetic-encoded strings which begin with "b" will be larger than those which begin with "a".

We have chosen to use a simple unigram model for English text, which assigns a relative probability to each possible symbol in the encoded characterset. We derived these probabilities from the Chicken Smoothie post title corpus. The measured symbol frequencies for the ASCII printable encoding are shown in figure 1. It is likely that a more advanced English character probability model (such as a bigram or higher order model) would be more effective at compressing text. For example, Moffat and Turpin[3] suggest that an order-2 model could encode English text at around 2.5 bits per character, which would allow strings of about 26 characters to be stored in a 64-bit pregen value. We hope to investigate this in the future.

8 Metric

In order to measure the performance of the approximate pregenerated rankings against the baselines, a metric must be defined. We have chosen to use the Kendall rank correlation coefficient (τ). This metric compares the relative rank order of every pair of documents in the two rankings, and assigns a correlation score in the range $[-1.0..1.0]$. Kendall's τ is computed for a list of n documents which has no ties with the expression:

$$\tau = \frac{n_c - n_d}{\frac{1}{2}n(n-1)}$$

Where n_c is the number of concordant pairs, and n_d is the number of discordant pairs. A concordant pair is a pair of documents n the baseline (B_a, B_b) having $B_a < B_b$, where that ordering is preserved by the pregen ($P_a < P_b$). A discordant pair is the opposite ($B_a < B_b$, but $P_a > P_b$). If $B_a = B_b$ or $P_a = P_b$, then the pair is said to be "tied", and is neither concordant or discordant. The "tau-b" variant of Kendall's Tau which we use also includes a correction factor for these tied pairs (this is not shown in the expression above).

A score of 1.0 indicates perfect agreement in the rankings, a score of -1.0 indicates perfect disagreement in the rankings (one ranking is the reverse of the other), and a score of 0.0 indicates rankings which are not correlated with each other (random ranking). We are interested in how close we can bring this ratio to 1.0.

The Tau value for simple string truncation on the standard baseline is already 0.999 for a 64-bit pregen, so simple intuition about how close Tau is to 1.0 will not be enough to understand the pregen's behaviour. For that purpose, we will compare our pregen's performance against that of a "reasonable pregen". The ordering of a "reasonable pregen" behaves the same way as a simple truncation of the input strings. More precisely, if a document D_a ranks before document D_b in the baseline ranking, the pregenerated ranking must not rank D_a after D_b, it can either rank D_a before D_b, or put D_a and D_b at the same position in the ranking. Additionally, if two documents D_a and D_b rank at an equal position in the baseline ranking, they must also rank at an equal position in the pregenerated ranking. The radix and arithmetic string coding methods can be considered to be "reasonable pregens" by this definition if the character encoding being used does not misorder input characters compared to the ordering in the baseline. This is true, for example, of the base-37 encoding on the restricted baseline, or the printable ASCII encoding on the standard baseline. The base-37 encoding is not a reasonable encoding on the standard baseline, since it entirely discards characters that the baseline considers significant.

Given these preconditions, consider a reasonable pregenerated ranking which conflates every group of size x of documents in a ranking of n distinct documents. That means that, for example, for $x = 2$, a baseline ranking of $(1, 2, 3, 4, 5, 6, 7, 8)$ would become $((1, 2), (3, 4), (5, 6), (7, 8))$ in the pregenerated ranking, with the documents in each conflated group having the same pregen value as each other. The number of pairs of documents within a group of size x, g_n is given by:

$$g_n = \frac{1}{2}x(x-1)$$

Within each conflated group in the pregen, we consider that the documents in that conflation will be, on average, randomly ordered in the search engine's output. That means that within those groups, the number of concordant pairs and discordant pairs will be equal, and so the number of discordant pairs in the group, g_d, is half of the group size:

$$g_d = \frac{1}{2}(\frac{1}{2}x(x-1))$$

Since there are $\frac{n}{x}$ of these groups in the total list of documents, the total number of discordant pairs in the ranking n_d is given by:

1907	Space
223, 1, 1, 1, 1, 29, 25, 155, 157, 60, 5, 48, 78, 136, 29	Punctuation
16, 50, 16, 17, 5, 4, 1, 5, 7, 8	Digits 0 - 9
41, 30, 1, 3, 1, 190, 1, 29, 1, 28, 11, 4, 1	Punctuation
1048, 134, 549, 502, 1204, 205, 306, 384, 669, 21, 88, 515, 333	Letters a - m
737, 854, 481, 12, 858, 643, 850, 199, 100, 315, 38, 225, 8,	Letters n - z
28, 13, 29, 93	Punctuation
15	Unicode

Figure 1: Relative symbol frequencies for the printable ASCII character encoding on Chicken Smoothie titles

$$n_d = \frac{n}{x} g_d$$

Because this pregen behaves like a truncation, we know that these are the only discordant pairs in the ranking, so n_c is computed by subtracting the number of discordant pairs from the total number of pairs in the pregen of n documents:

$$n_c = \frac{1}{2} n(n-1) - n_d$$

Substituting these values into Kendall's Tau gives the expression:

$$\tau = \frac{(\frac{1}{2}n(n-1) - (\frac{n}{x}\frac{1}{2}\frac{1}{2}x(x-1))) - (\frac{n}{x}\frac{1}{2}\frac{1}{2}x(x-1))}{\frac{1}{2}n(n-1)}$$

Combining the two occurences of the expression for n_d gives:

$$\tau = \frac{\frac{1}{2}n(n-1) - \frac{n}{x}(\frac{1}{2}x(x-1))}{\frac{1}{2}n(n-1)}$$

Performing the division simplifies the expression to:

$$\tau = 1 - \frac{\frac{n}{x}(\frac{1}{2}x(x-1))}{\frac{1}{2}n(n-1)}$$

Cancelling terms in the numerator and denominator gives the value of Kendall's Tau for a pregenerated ranking of n distinct values which conflates each group of x elements:

$$\tau = 1 - \frac{x-1}{n-1}$$

We can reverse this expression to derive the value of x for a given τ for this theoretical ranker:

$$-\tau + 1 = \frac{x-1}{n-1}$$

$$(-\tau + 1)(n-1) = x - 1$$

$$(1-\tau)(n-1) + 1 = x$$

x is a useful value to examine, because we can consider its relationship to the number of search results which the search engine presents on a single page. If those numbers are of similar magnitude, our search engine could resort just the documents that appear on a given search result page using their complete titles, and achieve a high overall ranking accuracy.

9 Results

The performance of each encoding scheme is graphed against the standard and restricted baselines of the Chicken Smoothie and Ancestry.com post title collections in figure 2, and the performance on the Chicken Smoothie post titles is displayed in table 1. The performance is measured using the conflation group size metric we defined in the previous section. In the standard baseline, we have included the "strtrunc" encoding, which is a simple truncation of the lowercased input. The graphs also show the "optimal" encoding, which is the conflation score expected if the input document titles are evenly distributed, and the pregen value is computed by sorting all of the documents in the collection and then numbering them in sequential order.

Where arithmetic string encoding has been used, it always outperforms the radix string encoding on the same character encoding, on our test data. This is true even for the Ancestry.com document collection, where the character probability model has been left unchanged from that computed for the Chicken Smoothie collection, suggesting that this unigram character model is generally applicable to English text.

The Ancestry.com post title dataset shows an unusual spike in performance around the 56-bit mark for the ASCII-printables pregen on the standard baseline. This is due to a large number of distinct post titles in this collection which happen to share a long prefix. This prefix is the string "Looking for " (as in "Looking for John Smith"), which occurs in over 200,000 documents in the Ancestry.com collection. The sharp improvement in performance corresponds to the position at which the pregen ranking begins to encode the characters that appear after this long prefix. The next-best performing pregen, base-40, can only encode 12 characters in 64-bits, which happens to coincide with the length of this prefix, so it conflates the ordering of these 200,000+ documents. Further difficulties are encountered with documents having the prefix "Looking for info", which is two characters longer, outstripping even the compression capability of the arithmetic-encoded ASCII printables encoding.

The 64-bit pregens on the Chicken Smoothie post title collection achieve a conflation group size of 64 documents on the standard baseline, and 52 documents

Scheme	Bits	Average conflations Standard baseline
Strtrunc	8	10771
	16	4109
	24	2837
	32	2649
	64	586
Base40	8	5640
	16	3147
	24	1505
	32	1023
	64	877
Asciiprintables	8	7779
	16	2686
	24	2224
	32	352
	64	111
Asciiprintablesarith	8	4680
	16	1697
	24	474
	32	232
	64	64

Scheme	Bits	Average conflations Restricted baseline
Base37	8	5411
	16	2589
	24	793
	32	259
	64	121
Base37arith	8	4400
	16	1486
	24	310
	32	176
	64	52
Base36	8	6059
	16	3278
	24	1513
	32	1453
	64	1325
Base32	8	4754
	16	2508
	24	518
	32	240
	64	67
Base32arith	8	4365
	16	1456
	24	308
	32	179
	64	55

Table 1: Pregen performance on Chicken Smoothie post titles for various encodings and pregen bit sizes

on the restricted baseline, both using an arithmetic encoding of the baseline's character set. On the Ancestry.com post title collection, the accuracy is 134 documents on the standard baseline, and 104 documents on the restricted baseline.

Not shown on the graphs is the performance of a pregen ranking for sorting Chicken Smoothie posts by author name. As author names are much shorter than document titles on average, an accuracy of 3 documents is achieved on the standard baseline, and approximately 1 document on the restricted baseline.

10 Conclusion

We have demonstrated that pregenerated rankings of text attributes such as post titles can achieve a result that is, on average, sorted to within 134 documents on average, across two very different English language collections. Similar performance was achieved on both the restricted and standard baselines, demonstrating that we can preserve the ordering of punctuation characters in the final output without any major additional ranking error. We have developed a metric which allows the ranking performance of pregens to be understood. We have identified a situation in which this pregenerated ranking method performs poorly, which is the presence of long common prefixes between distinct document titles, but we expect that the application of a bigram probability model for English text will solve this problem by improving our text compression performance. We hope to quantify that improvement in the future.

References

[1] T. C. Hu and A. C. Tucker. Optimal computer search trees and variable-length alphabetical codes. *SIAM Journal on Applied Mathematics*, Volume 21, Number 4, pages pp. 514–532, 1971.

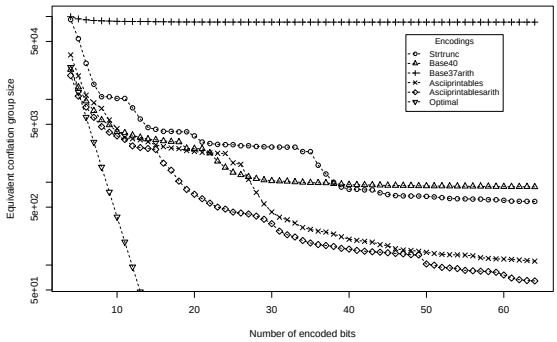
[2] D.A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, Volume 40, Number 9, pages 1098 –1101, sept. 1952.

[3] A. Moffat and A. Turpin. *Compression and coding algorithms*. Kluwer international series in engineering and computer science. Kluwer Academic Publishers, 2002.

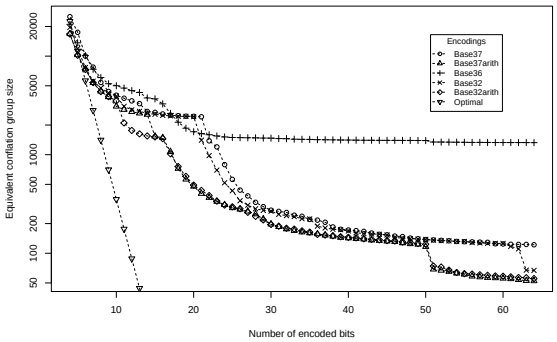
[4] Lawrence Page, Sergey Brin, Rajeev Motwani and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[5] Andrew Trotman, Xiang-Fei Jia and Shlomo Geva. Fast and effective focused retrieval. In Shlomo Geva, Jaap Kamps and Andrew Trotman (editors), *Focused Retrieval and Evaluation*, Volume 6203 of *Lecture Notes in Computer Science*, pages 229–241. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-14556-8

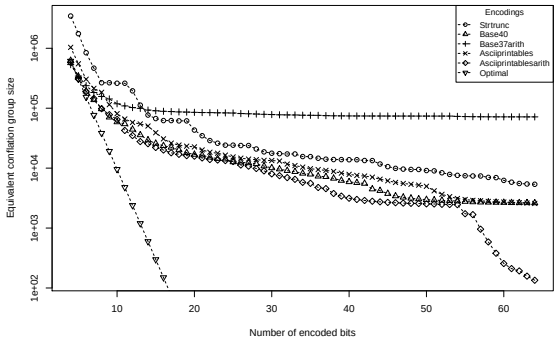
[6] Ian H. Witten, Radford M. Neal and John G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, Volume 30, pages 520–540, June 1987.



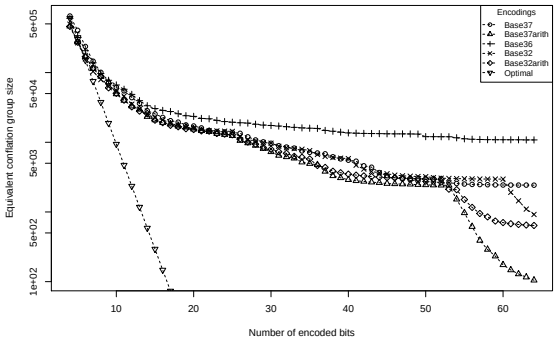
(a) Pregen ranking performance against standard baseline on Chicken Smoothie post titles



(b) Pregen ranking performance against restricted baseline on Chicken Smoothie post titles



(c) Pregen ranking performance against standard baseline on Ancestry.com post titles



(d) Pregen ranking performance against restricted baseline on Ancestry.com post titles

Figure 2: Pregen performance on sorting document titles

Examining document model residuals to provide feedback during Information Retrieval evaluation

Laurence A. F. Park

School of Computing and Mathematics
University of Western Sydney, Australia

lapark@scm.uws.edu.au

Abstract *Evaluation of document models for text based Information retrieval is crucial for developing document models that are appropriate for specific domains. Unfortunately, current document model evaluation methods for text retrieval provide no feedback, except for an evaluation score. To improve a model, we must use trial and error. In this article, we examine how we can provide feedback in the document model evaluation process, by providing a method of computing relevance score residuals and document model residuals for a given document-query set. Document model residuals provide us with an indication of where the document model is accurate and where it is not. We derive a simple method of computing the document model residuals using ridge regression. We also provide an analysis of the residuals of two document models, and show how we can use the correlation of document statistics to the residuals to provide statistically significant improvements to the precision of the model.*

Keywords Information Retrieval, Evaluation, Residuals, Optimisation

1 Introduction

The goal of a text based information retrieval system is to locate text documents, that are relevant to a users query, from a document collection. Research into text based retrieval systems is important since text documents are used in many domains to keep records and store information.

By examining the recent TREC tracks[1], we find uses for text retrieval in the legal and medical domains, general search for the Web and specialised search for homepage finding, blogs, and microblogs. There are many applications for text based information retrieval, and each application is defined by the document collections used and the probability of each query being issued. A specific domain (such as medicine) will require search on a specific document collection, and have a specific distribution over the set of possible queries.

[1] `http://trec.nist.gov/tracks.html`

Proceedings of the 16th Australasian Document Computing Symposium, Canberra, Australia, 2 December 2011.

To evaluate a retrieval system for a specific domain, we should obtain the document collection being used, all possible queries that can be issued, and the probability of each query being issued. The system would be evaluated as the expected evaluation score from a randomly sampled query. Using this form of evaluation, we can obtain the document model weights Δ that best suits the data and queries, by optimising to obtain the best expected evaluation score:

$$\Delta = \arg\max_{D} \sum_{\vec{q}} \text{eval}\left(\vec{r}_q, \vec{s}_q\right) P(q)$$

$$\text{such that } \vec{s}_q = \text{relevance}\left(D, \vec{q}\right)$$

where D is a matrix containing document models weights as its rows, $\vec{q}$ is a query vector containing the query weights for query q, $\vec{r}_q$ is the vector of manual relevance judgements for query q, $\vec{s}_q$ is the vector of document model relevance scores for query q, $P(q)$ is the probability of sampling query q, eval is the evaluation function, and relevance is the document model function used to compute the relevance of the document matrix to the query vector. Unfortunately, this form of evaluation is impossible, since it is unlikely that we would have a database of every possible query, and it would take a huge effort to obtain all the manual relevance judgements for all possible queries on each document in the collection.

The current approach to building an appropriate text retrieval system for a particular domain is to approximate to the expected evaluation score using a small random sample of queries from the domain. Obtaining a small random sample of queries means that we only need a small number of manual relevance judgements, reducing the work required for evaluation. Unfortunately, document models have a large number of parameters (one for each unique term in the document collection), therefore, we are unable to optimise D using a small sample. Doing so would produce a model with zero degrees of freedom and hence overfit the document model to the sample query set.

Therefore, document models are constructed as functions of the document collection and the term frequencies within the documents. By doing this, we are able to build models that generalise well to new query sets. The document models are constructed to approximate the ideal document model weights in Δ. So far, the most popular document models [3, 8, 9, 6] have been constructed based on the distribution

of relevant and irrelevant terms, and by observing statistics such as document model lengths and term counts.

Once these document models are built, we must still ask how can we evaluate them? The current approach to evaluation is to compute the sample mean evaluation score over the sample set of queries. This allows us to compare document models for accuracy, but gives us no insight as to where the document models are performing well and where they are not. We also cannot evaluate how well each model approximates the ideal document model weights Δ for the domain.

We are able to compute the ideal document model by finding the document model that provides the best precision for a given query set, using a set of manual relevance judgements of each query to each document. But, due to the small sample of queries we use, there are many solutions to this problem, and hence it is not obvious which solution we should be comparing our document model to.

In this article, we investigate how we can examine where document models are performing well and where they can be improved, by computing the ideal document model with smallest deviation to the provided document model. Once we have the ideal model, we can analyse the document model residuals. The document model residuals provide us with a means of estimating the accuracy of our document model, and also provide us with feedback as to where the document model deficiencies are for the given document domain. We provide the following contributions:

- a simple method for computing relevance score residuals
- a method of using ridge regression for identifying the ideal document model, and hence the document model residuals
- an analysis of the residuals of the term frequency and BM25 document models using the CRAN document collection.

This article proceeds as follows: Section 2 presents the current form of evaluation for text based Information retrieval and how it can be improved. Section 3 introduces the document model residuals and how we can compute them. Section 4 provides a small example of how to compute the document model residuals. Section 5 contains the analysis of residuals from a document collection. Finally, section 6 presents the work that is related to the content of this article.

2 Document model evaluation

A document model is a representation of a document for use in information retrieval. The most common document models are functions that take a document and statistics from the document collection, to compute weights representing the relevance of each term to the document in question.

There are three main classes of document models for text retrieval: vector space models, probabilistic models and language models. Each of the document models use the same linear function to compute document relevance scores:

$$s_d = \sum_{q \in Q} m_{\text{model}}(d, t, C) w_{q,t}$$

where s_d is the document relevance score, $m_{\text{model}}(d, t, C)$ is the weight of term t in document d computed using a document model and statistics from the document collection C, and $w_{q,t}$ is the weight of term t in query q.

The vector space document model [6] generally follows the TF-IDF (term frequency - inverse document frequency) form:

$$m_{\text{VSM}}(d, t, C) = f_{d,t} \log\left(\frac{N}{f_t}\right) \tag{1}$$

where $f_{d,t}$ is the frequency of term t in document d, f_t is the number of documents containing term t, and N is the number of documents.

A popular probabilistic document model is the BM25 model [3], which has the form:

$$m_{\text{BM25}}(d, t, C) = \frac{f_{d,t}(k_1 + 1)}{f_{d,t} + k_1(1 - b + b l_d/\bar{l})} \times \log\left(\frac{N - f_t + 0.5}{f_t + 0.5}\right) \tag{2}$$

where l_d is the length of document d, $\bar{l}$ is the average document length, and k_1 and b are parameters.

The language model document model function [8] has the form:

$$m_{\text{LM}}(d, t, C) = \log\left(\alpha P(t|d) + (1 - \alpha) P(t|C)\right)$$

where $P(t|d)$ is the probability of term t being sampled from document d, $P(t|C)$ is the probability of term t being sampled from the document collection C, and α is the smoothing parameter.

To evaluate the ranking produced from the document model, we must have a set of relevance judgements for the query. Relevance judgements are values that are assigned to each document for each query, reflecting the relevance of the associated document to the associated query. Relevance judgements may be binary (1 meaning relevant, 0 meaning irrelevant), or ordinal (0 meaning irrelevant, and 1, 2, 3, 4 and 5 being levels of relevance, where 5 implies the greatest relevance). We compare the document model relevance scores to the relevance judgements using an evaluation function. The evaluation function compares the ranking of documents (induced by the relevance scores) to the relevance judgements. A perfect score is awarded to document models that score documents of greater relevance over those of less relevance (or in the binary case, score all relevant documents greater than the set of irrelevant documents). A popular evaluation function for binary relevance judgements is average precision (AP) [4]. Average precision provides us with a score between 0 and 1 inclusive, where 1 is given for a perfectly ranked list.

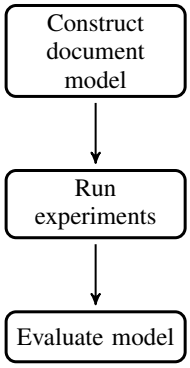


Figure 1: Current method of evaluating document models. There is no method of identifying if the document model can be improved, based on the outcome of the evaluation.

The process of document model evaluation in previous retrieval experiments has stopped here. Once the document model is evaluated, we obtain an evaluation score for the model, and can then compare it to other scores from other document models. This one way process is shown in Figure 1. There is currently no method of examining where the differences in document models lie, or where a model is performing poorly. In the next section, we will examine how we can assess document model deficiencies and their suitability for a given collection, using document model residuals.

3 Evaluation feedback using residuals

The construction of a document model is a regression problem, where we are mapping a set of document and query attributes to a real scalar value that reflects the relevance of the associated document to the associated query. Therefore, we are able to assess the goodness of fit of a document model by examining its residuals. Document model residuals show us how each element of the document model is affecting the accuracy of the document model. By examining the document model residuals, we obtain feedback on how the document model can be improved, and hence close the loop on Information retrieval evaluation (shown in Figure 2).

We saw in the previous section that many of the document models are linear functions; this linearity is one of the factors that leads to fast query times and so is preferred over non-linear models. The linear model can be given as:

$$\vec{s}_d = \vec{d}Q^T \quad \text{or} \quad \vec{s}_q = D\vec{q}^T$$

where $\vec{s}_d$ is the vector of relevance scores for document d predicted for all queries, and $\vec{s}_q$ is the vector of relevance scores for all documents predicted for queries q, $\vec{d}$ is the vector of document weights that are computed using the document model, D is the matrix containing vectors $\vec{d}$ as its rows, $\vec{q}$ is a vector of query weights, Q is a row matrix of query vectors $\vec{q}$, and Q^T is the transposed Q.

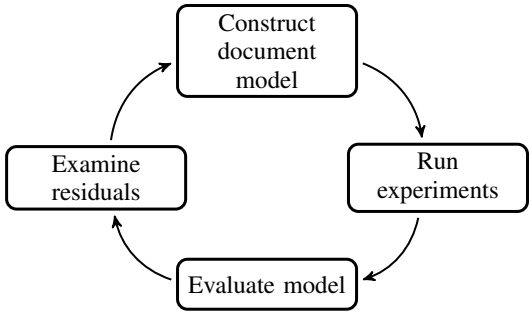


Figure 2: Suggested document model improvement cycle for information retrieval. Using the evaluation results, we are able to construct the document model residuals and examine where the model can be improved.

Using this linear form of retrieval, we can examine two forms of residuals: *relevance score residuals* and *document weight residuals*. Relevance score residuals are the difference between the ideal relevance scores and the relevance scores predicted by the document model. Document weight residuals are the difference between the document weights computed using the document model and the document weights that would lead to the ideal relevance scores.

3.1 Relevance score residuals

The best document model weights Δ for a given document-query set pair can be shown as:

$$\vec{s}_q = \Delta\vec{q}^T \tag{3}$$

where the document score vector $\vec{s}_q$ is the solution to:

$$\vec{s}_q = \arg\max_{\vec{\theta}_q} \left(\text{eval}\left(\vec{r}_q, \vec{\theta}_q\right)\right)$$

Our document model, used to compute the values in D are an attempt to achieve the ideal document model values in Δ, and are therefore an approximation to Δ. The difference between D and Δ is the set of document model residuals E. Therefore, we can write equation 3 as:

$$\begin{aligned} \vec{s}_q &= (D + E)\,\vec{q}^T \\ &= D\vec{q}^T + E\vec{q}^T \end{aligned} \tag{4}$$

The first term in 4 is the set of scores generated by our document model, while the second term is the set of scores that need to be added to our model's scores to provide the best document ranking:

$$\vec{s}_q = \vec{s}_{D,q} + \vec{s}_{E,q}$$

where $\vec{s}_{D,q}$ is the set of scores using our document model, being an approximation to $\vec{s}_q$, and $\vec{s}_{E,q}$ is the difference in $\vec{s}_q$ and $\vec{s}_{D,q}$, or the relevance score residuals. In order to identify the set of relevance score residuals, we should choose $\vec{s}_{E,q}$ to maximise the evaluation metric being used. We can show this as:

$$\vec{s}_{E,q} = \arg\max_{\vec{s}_{\Xi,q}} \left(\text{eval}\left(\vec{r}_q, \vec{s}_{D,q} + \vec{s}_{\Xi,q}\right)\right) \quad (5)$$

There are many solutions to this optimisation problem, since any vector that we add to $\vec{s}_{D,q}$ to give the correct ranking is a solution. Therefore, we add the extra constraint that we want the relevance score residuals that are a solution to equation 5, and cause the smallest deviation to the document model scores $\vec{s}_{D,q}$:

$$\vec{s}_{E,q} = \arg\max_{\vec{s}_{\Xi,q}} \left(\text{eval}\left(\vec{r}_q, \vec{s}_{D,q} + \vec{s}_{\Xi,q}\right) - \gamma \|\vec{s}_{\Xi,q}\|_2\right) \quad (6)$$

where γ is a positive constant that controls the residual minimisation, and $\|\cdot\|_2$ is the l_2 vector norm.

To solve the optimisation problem in equation 6, we need to obtain the gradient of the function being optimised. Unfortunately, the function contains $\text{eval}(\cdot,\cdot)$. If we examine the set of evaluation functions for information retrieval, we find that they are all disjoint. As the score vector transitions through the vector space, the evaluation score will not change until the ranks of the score vector elements change. Therefore, the evaluation function is a series split of levels, not a smooth function. This means that the gradient of the evaluation metrics are not continuous and hence we are unable to solve the maximisation by examining the evaluation function gradient.

Rather than propose a complex method to solve the optimisation problem in equation 6, we have chosen to add a further constraint to make the problem simple to solve. We add the constraint that the elements of $\vec{s}_{\Xi,q}$ must be non-negative, giving us the optimisation problem:

$$\vec{s}_{E,q} = \arg\max_{\vec{s}_{\Xi,q}} \left(\text{eval}\left(\vec{r}_q, \vec{s}_{D,q} + \vec{s}_{\Xi,q}\right) - \gamma \|\vec{s}_{\Xi,q}\|_2\right)$$
$$\text{such that } s_{\Xi,q,d} \geq 0$$

where $s_{\Xi,q,d}$ are the elements of $\vec{s}_{\Xi,q}$.

When examining the case of binary relevance judgements and γ is small, we find that the solution to this problem is the vector that increases all relevant document scores to be at least slightly greater than the score of each irrelevant document. The algorithm to compute this vector is provided in Algorithm 1.

Algorithm 1 Compute relevance score residuals $\vec{s}_{E,q}$ where ϵ is a small positive real value.

1: $\vec{s}_q = \vec{s}_{D,q}$
2: Find in $\vec{s}_{D,q}$, the greatest score assigned to an irrelevant document $s_{i,q}$.
3: **for** each relevant document score $s_{r,q}$ in $\vec{s}_q$ **do**
4: **if** $s_{r,q} \leq s_{i,q}$ **then**
5: $s_{r,q} = s_{i,q} + \epsilon$
6: **end if**
7: **end for**
8: $\vec{s}_{E,q} = \vec{s}_q - \vec{s}_{D,q}$

3.2 Document model residuals

The document model residuals are the difference in the document weights computed using the document model, and the document weights of the ideal document model. We can extend on the optimisation used to compute the relevance score residuals in equation 6, to compute the document model residuals. We do this by replacing the relevance scores with the document model weights used to compute the relevance scores:

$$E = \arg\max_{\Xi} \left(\text{eval}\left(\vec{r}_q, D\vec{q}^T + \Xi\vec{q}^T\right) - \gamma\|\Xi\vec{q}^T\|_2\right) \quad (7)$$

Again, this is a complex optimisation since the $\text{eval}(\cdot,\cdot)$ is not likely to have a continuous gradient. Instead of solving the problem in equation 7, we have broken the problem into two stages, making the optimisation problem easier to compute. The first stage involves computing the relevance score residuals (examined in section 3.1), the second stage involves computing the document model residuals from the relevance score residuals.

The ideal document model scores are those that provide the best precision for the document-query set. Computing the ideal document model weights, given the ideal document scores, is a regression problem:

$$\vec{s}_\delta = \vec{\delta}Q^T$$

where $\vec{\delta}$ is the vector of ideal document weights for document d, Q is the matrix of query vectors, and $\vec{s}_\delta$ is the set of ideal document scores for document d. The document scores can be written as:

$$\begin{aligned}\vec{s}_\delta &= \vec{d}Q^T + \vec{e}Q^T \\ &= \vec{s}_{d,Q} + \vec{s}_{e,Q}\end{aligned}$$

where $\vec{d}$ is the vector of weights computed using our document model, $\vec{s}_{d,Q}$ is the set of scores of document d, $\vec{e}$ is the vector of document model residuals, and $\vec{s}_{e,Q}$ is the difference between the ideal document model scores and our document model scores.

We know the values in Q, $\vec{d}$, $\vec{s}_\delta$ and $\vec{s}_{d,Q}$. From these, we can easily calculate $\vec{s}_{e,Q}$ and so can compute the vector of document model residuals $\vec{e}$ using linear regression:

$$\vec{e} = \arg\min_{\vec{\eta}} \|\vec{s}_{e,Q} - \vec{\eta}Q^T\|_2$$

But, note that since we usually have a small number of queries, and a large vector space, the regression is computed with zero degrees of freedom, hence the vector $\vec{e}$ will overfit the document-query set, and not generalise well to other queries. To increase the generalisation of $\vec{e}$, we regularise the regression, and compute the ideal document weights using ridge regression [2]:

$$\vec{e} = \arg\min_{\vec{\eta}} \left(\|\vec{s}_{e,Q} - \vec{\eta}Q^T\|_2 - \lambda\|\vec{\eta}\|_2\right) \quad (8)$$

where λ is a real positive value that controls the level of regularisation.

4 Residual computation example

In this example, we have used a document model to compute the term weights for each document and stored the weights in the matrix D, where each row of D represents the term weights for a specific document in the document collection. We have also constructed the query matrix Q containing the weights associated to each query. The query weights are binary, showing the absence or presence of a term:

$$D = \begin{bmatrix} 1 & 1 & 2 & 2 & 0 \\ 1 & 2 & 5 & 2 & 0 \\ 0 & 1 & 1 & 2 & 1 \\ 2 & 0 & 4 & 1 & 2 \\ 4 & 0 & 5 & 1 & 1 \end{bmatrix} \quad Q = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

We compute the predicted relevance scores as the inner product of each document vector with each query vector, given as $\hat{S} = DQ^T$. The contents of $\hat{S}$ is shown below with the relevance judgement matrix R:

$$\hat{S} = \begin{bmatrix} 3 & 3 & 5 & 3 \\ 4 & 3 & 8 & 6 \\ 4 & 2 & 3 & 1 \\ 3 & 3 & 7 & 6 \\ 2 & 5 & 10 & 9 \end{bmatrix} \quad R = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The mean average precision (MAP) of this document model on this document-query set is computed using $\hat{S}$ and R as 0.675.

The goal of each document model is to compute a relevance score for each document, so that the documents judged relevant are scored greater than those judged irrelevant. To compute the relevance score residuals, we must first compute the ideal set of relevance scores, that have little variation on the predicted relevance scores $\hat{S}$. We compute these scores using the first seven lines of Algorithm 1. This gives us the ideal relevance score matrix:

$$S = \begin{bmatrix} 3 & 3 & 5 & 3 \\ 4 & 3 & 10+\epsilon & 9+\epsilon \\ 4 & 2 & 3 & 9+\epsilon \\ 3 & 3+\epsilon & 7 & 6 \\ 2 & 5 & 10 & 9 \end{bmatrix}$$

We compute the relevance score residuals S_E as the difference between the ideal relevance scores S and the relevance scores predicted using the document model $\hat{S}$ (using the last line of Algorithm 1):

$$S_E = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 2+\epsilon & 3+\epsilon \\ 0 & 0 & 0 & 8+\epsilon \\ 0 & \epsilon & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Using the relevance score residual matrix and the query matrix, we can compute the document model residuals E using ridge regression. If we use a small value of λ (in this case $\lambda = 10^{-10}$), we obtain the set of document weight residuals that provide perfect precision:

$$E = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0.5 & 2+10^{-6} & -1 & 0.5 \\ 8+10^{-6} & 4 & 0 & -8+10^{-6} & 4 \\ 10^{-6} & 0 & -10^{-6} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

By adding these residuals to our document weights in D and applying the queries, we obtain a mean average precision of 1. Unfortunately, by using a small value of λ, we usually over fit the document-query set, meaning that the residuals are a reflection of the data rather than the document model. To obtain more informative residuals, we should use a larger value of λ. The larger the value of lambda, the greater the generalisability of the residuals to other query sets, but we obtain a smaller increase in precision. For this example, we have chosen to examine the document weight residuals for $\lambda = 0.1$, and obtain:

$$E = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0.885 & 0.372 & 1.910 & -0.782 & 0.372 \\ 5.140 & 2.619 & 1.533 & -5.501 & 2.619 \\ 10^{-6} & 0 & -10^{-6} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

By adding this set of document weight residuals to the document weight matrix D and applying the queries, we obtain a mean average precision of 0.854.

By examining the document weight residuals, we see that the largest residuals exist on the third row, which is associated to the third document. This may indicate that the current document model is not sufficient for the third document. We can also see that the largest residuals exist in the first and fourth columns (associated to the first and fourth terms). This is an indicator that the term weighting within the document model may need adjusting to suit these terms.

5 Examining document model residuals

In the previous sections, we provided a framework for computing document model residuals. In this section we will compute the residuals of the raw document model, where the weights of the model are the term frequencies, and the BM25 document model, where the weights of the document model are the BM25 term weights. After computing the residuals, we will examine the correlation between the residuals and statistics of the document collection to demonstrate how we can adjust the document models to provide a better fit to the document collection. We will perform this analysis using the CRAN document collection[2], which contains 1398 documents, but most importantly 225 long queries, and manual relevance judgements for all documents on each query. Note that residuals can only be found on terms that are within queries and documents that have relevance judgements. Therefore this set will provide us with many residuals.

5.1 Raw term frequency residuals

First, we will examine the raw term frequencies and demonstrate how a well known form of term weight can

[2] `ftp://ftp.cs.cornell.edu/pub/smart/`

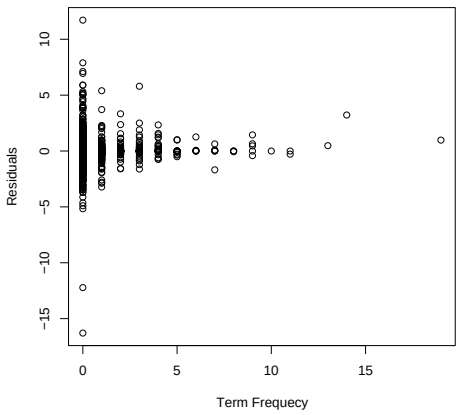


Figure 3: Residuals of the term frequencies for all documents compared to the term frequency values.

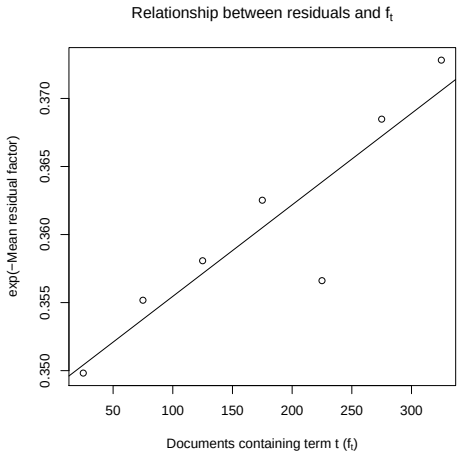


Figure 4: The correlation between f_t (the number of documents the term appears in) and the exponential of the negative mean residual factor associated to the f_t value. The line shows the least squares fit to the points.

be observed using the residuals. To compute the residuals, we construct the matrix D containing the raw term frequencies $f_{d,t}$ and the matrix Q containing binary values; 1 if the associated term appeared in the query, and 0 if the term did not appear in the query. We then obtained the set of relevance score residuals, with the constraint that the residuals were not negative, using Algorithm 1. Once we obtained the set of relevance score residuals, we computed the document model residuals using ridge regression from equation 8 with $\lambda = 1$. The plot of the residuals of each term frequency is shown in Figure 3.

The first observation of Figure 3 is that the residuals are almost symmetric about zero for each term frequency, and the that variance of the residuals decreases as the term frequency value increases. This shows that we need to modify the values of small term frequencies to achieve δ, the ideal model for the collection, but larger term frequencies need little modification. We can also see that there are many large residuals associated to the term frequency zero; this reflects the importance of having non-zero weights associated to term frequencies of zero.

We will now examine the correlation of the document model residuals to the term document count f_t. To do this, we examined the mean residual for values of f_t between 0 and 350, where the mean is computed for f_t using bins of width 50. We found no obvious relationship between the document model residuals and f_t. Therefore, we also examined the document model residual factor, computed as $(e_{d,t} + f_{d,t})/f_{d,t}$. A plot comparing the mean document model residual factor and f_t is shown in Figure 4.

Using this plot in Figure 4, we computed the linear relationship:

$$\exp\left(-\frac{e_{d,t} + f_{d,t}}{f_{d,t}}\right) = 0.35 + 6.72 \times 10^{-5} f_t$$

which suggests the improved document model:

$$f_{d,t} \log\left(\frac{1}{a + b f_t}\right) \tag{9}$$

where a and b are parameters determined by the document collection; in this case $a = 0.35$ and $b = 6.72 \times 10^{-5}$. This form of document model is very similar to the VSM document model (shown in equation 1) which provides an improvement over the raw term frequency model. Therefore, by examining the document model residuals, we have shown the importance of weighting by the log of the inverse term weight f_t.

To examine the effect of the modified term frequency document model, we constructed the matrix D containing the values computed using equation 9, and compared the increase in each query's AP to the AP of the raw term frequency model. A histogram showing the distribution of the change in AP for each query is provided in Figure 5.

We can see from this histogram that an increase in AP was provided to a large set of queries (172 queries), and a decrease in AP was provided to a small set of queries (45 queries). The paired Wilcoxon signed rank test on the AP produced from the raw term frequencies compared to the modified term frequencies provided a p value of less that 10^{-16}, showing that the modification produced by observing the residuals provided a significant increase in AP. The mean AP was increased from 0.2253 to 0.2316.

5.2 BM25 residuals

We will now examine the BM25 document model and try to identify if there is any correlation of the document weight residuals to the document collection statistics. We construct the matrix D using the BM25 document weights from equation 2 and the matrix Q containing binary values reflecting the appearance of the associated term in the query.

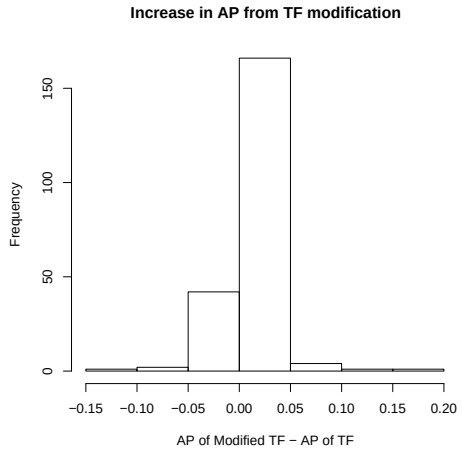


Figure 5: Distribution of the increase in AP after modifying the raw term frequencies with the results from the residual analysis.

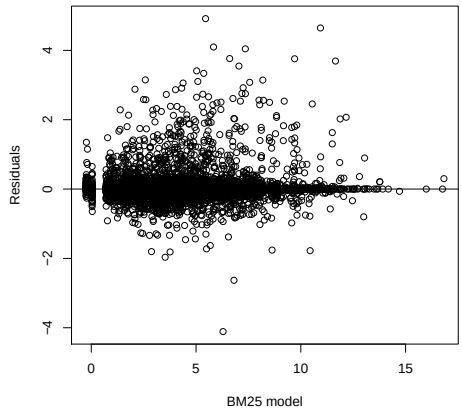


Figure 6: Residuals of the BM25 document weights for all documents compared to the model weights.

We again obtained the set of relevance score residuals using Algorithm 1. We computed the document model residuals using ridge regression from equation 8 with $\lambda = 1$. The plot of the residuals of each BM25 term weight is shown in Figure 6.

We can see from Figure 6 that there is small variance in the residuals for small and large BM25 weights, but the variation increases near the region where the BM25 weight is 5. This shows that the mid-range BM25 weights are not as accurate at the small and large values, hence improvements can be made. We can see that there is also a set of residuals where the weight is 0 (implying the term frequency is 0). The residuals at zero are not as large as found in the term frequency plot, but still present, implying that well calculated non-zero weights for zero valued

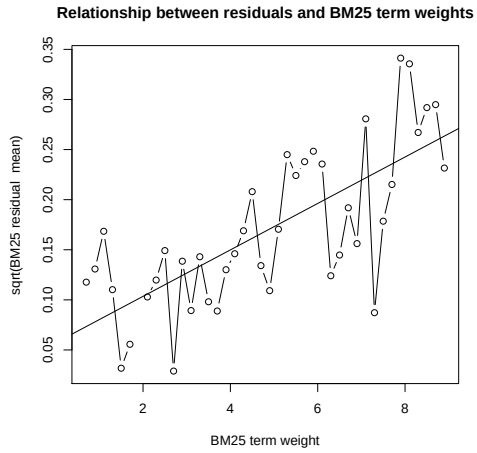


Figure 7: The correlation between the BM25 term weights and the square root of the mean residual of the associated term weight. The line shows the least squares fit to the points.

term frequencies would improve the accuracy of the document model.

We will now examine the average residual error compared to the BM25 weight. The BM25 weight is a continuous value, therefore, to compute the average residual, we gathered all BM25 weights into 0.2 width bins and computed the average residual associated to each weight in the bin. The correlation observed is shown in Figure 7. The line fit to the plot in Figure 7 depicts the relationship:

$$\sqrt{e_{d,t}} = 0.026 m_{\text{BM25}}(d,t,C) + 0.038$$

which suggests the improved document model:

$$m_{\text{BM25}}(d,t,C) + (a + b m_{\text{BM25}}(d,t,C))^2 \qquad (10)$$

where a and b are parameters determined by the document collection; in this case $a = 0.038$ and $b = 0.026$. This residual analysis has suggested that we should use a quadratic form for our document model. To examine the improvement given by this model, we constructed the matrix D containing the values from equation 10 and compared the increase in each query's AP to the AP of BM25. A histogram showing the distribution of the change in AP for each query is provided in Figure 8.

We can see from this histogram that an increase in AP was provided to a set of queries (51 queries), and a decrease in AP was provided to a smaller set of queries (32 queries). The paired Wilcoxon signed rank test on the AP produced from the BM25 weights compared to the modified BM25 weights provided a p value of 0.0240 showing that the modification produced by observing the residuals provided a significant increase in AP. Given all of this, we found that the mean AP decreased from 0.3620 to 0.3617. By examining the histogram in Figure 8 we can see that the decrease would be due to the one query that caused the small bar at -0.06 to -0.05.

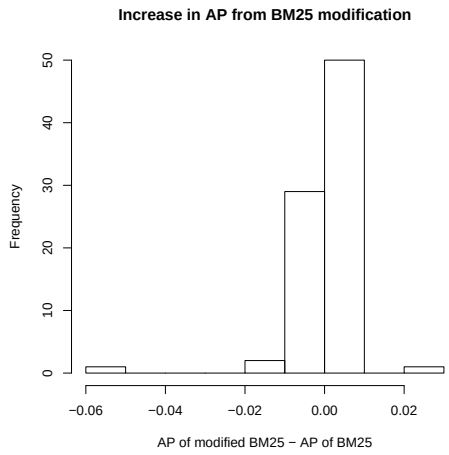


Figure 8: Distribution of the increase in AP after modifying BM25 with the results from the residual analysis.

Note that even though we have derived new document models, we do not suggest that these should be used in practice. These models were obtained by examining the residuals on a single document set using a small number of queries. Further analysis such as cross validation should be performed in order to better examine the effects of modified document models and to determine the value of λ that should be used. We included this analysis simply to demonstrate how the residuals could be used in practice.

6 Related Work

There are similarities between the residual computation we have proposed and the work done by the learning to rank community [5, 1, 7]. Learning to rank is the process of using machine learning methods to a construct document models for specific documents-query sets. When using learning to rank methods, we obtain a document model that has high precision, but the models are usually complex and so we are unsure why the model provides high precision.

The work we have proposed is not for automatically constructing high precision document models, but for examining where our existing document models work well and where they are deficient. By using the methods presented in this article, we will gain a better understanding of what is required to obtain high precision for text document sets from any domain.

7 Conclusion

Information retrieval evaluation is needed in order to determine the document models that are most suitable for retrieval tasks. The existing evaluation process provides a score to each document model, allowing us to compare models, but unfortunately, there is no method of determining where the document models are performing well and where they are deficient.

In this article, we presented a framework for computing document model residuals. By examining the residuals of a document model, we can discover the difference between the document model weights and the weights of the ideal document model. We can also use the document model residuals to examine correlation between document collection statistics, and hence improve the model.

We included an analysis of the document model residuals for the raw term frequency model and the BM25 document model. We showed that we were able to find correlation between the residuals and statistics, such as the term document count and the within document term frequency.

By computing and analysing the document model residuals, we are able to close the gap in the evaluation cycle, and reduce the black box evaluation that is currently performed. This will allow us to develop accurate domain specific document models more intelligently.

References

[1] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 89–96, New York, NY, USA, 2005. ACM.

[2] Trevor Hastie, Robert Tibshirani and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2nd edition, 2001.

[3] K. Sparck Jones, S. Walker and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments, part 2. *Information Processing and Management*, Volume 36, Number 6, pages 809—840, 2000.

[4] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze. *Introduction to Information Retrieval*, Chapter 8: Evaluation in information retrieval. Cambridge University Press, 2008.

[5] Ramesh Nallapati. Discriminative models for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 64–71, New York, NY, USA, 2004. ACM.

[6] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, Volume 24, Number 5, pages 513 – 523, 1988.

[7] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 391–398, New York, NY, USA, 2007. ACM.

[8] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342, New York, NY, USA, 2001. ACM Press.

[9] Justin Zobel and Alistair Moffat. Exploring the similarity space. *ACM SIGIR Forum*, Volume 32, Number 1, pages 18–34, Spring 1998.

Six Degrees of *Cohesion* in Patent Citation Network

Kelly Y. Itakura
Queensland University of Technology
Brisbane, 4000, Australia
kelly.itakura@qut.edu.au

Shlomo Geva
Queensland University of Technology
Brisbane, 4000, Australia
s.geva@qut.edu.au

Abstract *While the phrase "six degrees of separation" is widely used to characterize a variety of human-derived networks, in this study we show that in patent citation network, related patents are connected with an average distance of 6, whereas an average distance for a random pair of nodes in the graph is approximately 15. We use this information to improve the recall level in prior-art retrieval in the setting of blind relevance feedback without any textual knowledge.*

Keywords Patent Retrieval, Link Discovery, Social Networks, Information Retrieval, Digital Libraries, Document Management,

1 Introduction

Many human-derived network is characterized by the so-called "six degrees of separation" popularized by Milgram's small world experiment [6]. While most instances involve connecting two human beings directly such as sending letters in the case of Milgram, some find applications that indirectly involve human beings. For example, Elmacioglu [2] showed that the six-degree rule also applies in the co-authorship relationships in computer science academic publications.

Finding prior arts for a patent application is a time consuming process. A standard search engine with keyword matching oftentimes does not work well because of the applicants' attempt to obfuscate the application to increase the chance of acceptance. One of CLEF 2010 IP[1] tasks is to retrieve prior art for a patent application. The multi-lingual nature of the patent corpus makes it more difficult to find a language-independent method. Even if the overall strategy is the same, there is still a need to come up with language-specific rules.

The contribution of this paper is to study the patent-citation network and use the citation-link information alone to create a good working set of patents with a high recall value. Returning all patents in the corpus will achieve the recall rate of 100%, however, we achieve the recall rate of 75% by returning only those patents within the distance of 12. From here, we can utilize other meta-information and contextual information to narrow down the final set of relevant patents. We additionally found that unlike the co-authorship graph of Elmacioglu, a random pairs of patents are connected 15 degrees apart, if ever, while prior-arts patents are connected 6 degrees apart.

[1] http://www.ir-facility.org/clef-ip

Proceedings of the 16th Australasian Document Computing Symposium, Canberra, Australia, 2 December 2011.

2 Previous Works

In CLEF 2009 and 2010, few participants exploited citation networks to improve the performance of their prior art retrieval system. Magdy and Jones [5] used citation network to retrieve relevant patents and found that even though MAP is increased dramatically, recall was very low. Lopez and Romary [3, 4] limited the search space by using in-text citations extracted using manual rules combined with other meta-data. Their final working set has 75% recall. Outside of the CLEF campaign, Tiwana and Horowtz [8] used cation network to retrieve prior arts but the experimental results are not reported. Breschi and Lissoni [1] computed the probability of a patent citation based on co-citation networks of patent applicants. Our method achieves a high recall of 75% to extract citations without the hassle of parsing the context of patent documents.

3 Experiments

In CLEF 2010 IP, we were provided with a corpus, topics, training topics and the corresponding qrels. The corpus contains 3,118,088 patent documents of size 117GB in total. The corpus is of multi-lingual nature [2], but since our study focuses on links alone, the multilingual nature is irrelevant. Patent documents are marked up into different fields. In this paper, we only use citation and priority claim fields noted by `patcit` and `priority-claim`. Citation information can also be found in the main context of patent documents; however, it needs to go through extensive parsing such as done by Lopez and Romary [3, 4] in order to be extracted because of the lack of formatting unlike academic papers. More information of the dataset can be found at CLEF 2010 IP overview paper [7].

We built a citation network by extracting all patents that appear in the citation and priority claim fields. We ignored the direction of the graph. There are 6,526,748

[2] English, French, and German

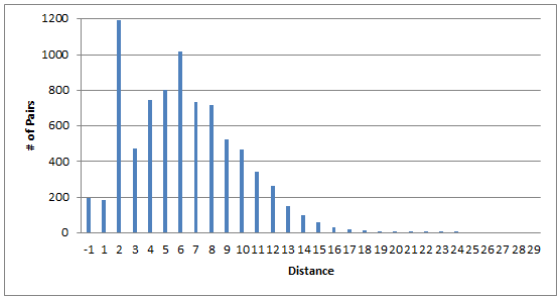


Figure 1: Distribution of Distances Between Relevant Patents

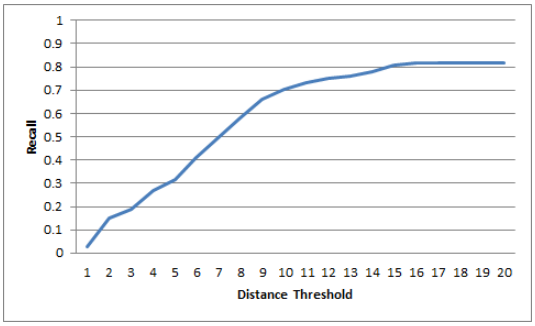


Figure 2: Recalls for Varying Distance Threshold

nodes in total, much of them not appearing in the corpus. The average distance between any pair of nodes was estimated as follows. We randomly picked 1,000 nodes, and for each node, we computed the average path length by breadth-first traversal. An average of the 1,000 average path lengths was 14.52. We call this average path length of the graph. Note that the average length ignores the case when there is no path.

The qrels provided contained 1,768 relevant patents (prior arts candidate) in total for 300 topics. Thus an average topic contains only 5.89 relevant patents. Because the qrels were automatically created based on the citation fields in the topic patents [7], those fields were removed when provided to participants. Thus there is no link in the citation graph from a topic patent to any relevant patent in the qrels.

Next, we computed the average distance between any pairs of relevant patents for each topic. Figure 1 shows the distribution of distance between relevant patents. The length of -1 shows that there is no path, and included in the computation of the average length as the distance of -1. This happens in 2.41% of the cases. The average path length is 6.19 with a median of 6.

In order to take advantage of the knowledge of an average distance between relevant nodes, we use the scenario of a link-based relevance feedback. Suppose that we have one patent that we think is highly relevant. Our goal is to extract connected patents to improve a recall level. Our extraction mechanism is purely link-based and does not involve text parsing.

For each topic, and for each relevant patent, we extracted all patents that are connected to it with varying distance thresholds. Due to time constraint, our data set is limited to 8 topics with 59 relevant patents. Figure 2 shows that the recall level plateaus around 80% at a threshold of 15, which is close to the average path length of the graph at 14.52. However, at this threshold, almost all patents in the corpus was retreived, and it seems that the threshold of 12 still gives a high recall rate of 75% while returning 26.8% of the patents in the corpus. Thus using the knowledge of average distance, we can achieve the same recall rate as that of Lopez and Romary [3, 4], without the hassle of in-context parsing. Since the corpus is multi-lingual, in-context parsing requires multi-lingual knowledge. In addition, their citation network would be much larger than ours given the same corpus because of additional in-context citations, in addition to the citation fields we employed. From this working set, we can use textual information to rank the relevant patents.

4 Conclusions

We showed that relevant patents are connected with an average distance of 6, mostly ranging from 1 to 13. This is slightly less than half of the average path length of a connected pairs of random patents, relevant or not. These numbers can be used to create a threshold to retrieve additional patents in the case of blind-relevance feedback. We achieve the same recall rate as that of Lopez and Romary [3, 4], with much simpler method, however, our scenario assumes that we have already retrieved one relevant patent. Both Lopez and Romary and our work require filtering process after retrieving this initial set of candidates.

Future work involves clustering all patents such that they are within the distance of 1 to 13 of each other, naming each cluster with the most popular category, and then matching the topic category against the clusters. Eventually, it would require some form of text processing, but we hope our method reduces its need because unlike in the case of this experiment which involved only three languages, patents are written by many other languages.

References

[1] Stefano Breschi and Francesco Lissoni. Knowledge networks from patent data. In Henk Moed, Wolfgang Glnzel and Ulrich Schmoch (editors), *Handbook of Quantitative Science and Technology Research*, pages 613–643. Springer Netherlands, 2005.

[2] Ergin Elmacioglu. On six degrees of separation in dblp-db and more. *ACM SIGMOD Record*, Volume 34, pages 33–40, 2005.

[3] Patrice Lopez and Laurent Romary. Multiple retrieval models and regression models for prior art search. In *CLEF2009 Working Notes*, pages –, Corfu, Greece, September 2009.

[4] Patrice Lopez and Laurent Romary. Experiments with citation mining and key-term extraction for prior art search. In *CLEF2010 Notebook Papers*, pages –, Padua, Italy, September 2010.

[5] Walid Magdy and Gareth J.F. Jones. Applying the kiss principle for the clef- ip 2010 prior art candidate patent search task. In *CLEF2010 Notebook Papers*, pages –, Padua, Italy, September 2010.

[6] S. Milgram. The small-world problem. *Psychology Today*, Volume 1, pages 61–67, 1967.

[7] Florina Piroi and John Tait. Clef-ip 2010: Retrieval experiments in the intellectual property domain. In *CLEF2010 Notebook Papers*, pages –, Padua, Italy, September 2010.

[8] Shahzad Tiwana and Ellis Horowitz. Findcite: automatically finding prior art patents. In *Proceeding of the 2nd international workshop on Patent information retrieval*, PaIR '09, pages 37–40, New York, NY, USA, 2009. ACM.

Harvesting Domain-Specific Terms using Wikipedia

Su Nam Kim
Department of Computer Science and Software Engineering
The University of Melbourne
VIC, Australia
snkim@csse.unimelb.edu.au

Lawrence Cavedon
School of CS and IT
RMIT University
VIC, Australia
lawrence.cavedon@rmit.edu.au

Timothy Baldwin
Department of Computer Science and Software Engineering
The University of Melbourne
VIC, Australia
tb@ldwin.net

Abstract *We present a simple but effective method of automatically extracting domain-specific terms using Wikipedia as training data (i.e. self-supervised learning). Our first goal is to show, using human judgments, that Wikipedia categories are domain-specific and thus can replace manually annotated terms. Second, we show that identifying such terms using harvested Wikipedia categories and entities as seeds is reliable when compared to the use of dictionary terms. Our technique facilitates the construction of large semantic resources in multiple domains without requiring manually annotated training data.*

Keywords Domain-specific Terms, Self Training, Wikipedia

1 Introduction

The identification and use of **domain-specific terms**—i.e., terms that have significance or special meaning in a domain—have been leveraged in a range of NLP and related tasks (e.g., query expansion and cross-lingual text categorization [7], and named entity recognition [2]). However, construction of knowledge bases of such terms (e.g. *Agrivoc*, *UMLS*) has generally been manual, requiring high cost and time. Even hand-crafted resources often have limitations in quality, are typically small, and have the problem of needing to be maintained as new words are added or become significant within the domain.

Most previous work on automatically identifying and categorizing domain-specific terms has used supervised techniques (e.g. [1, 7, 5, 3]). This has the disadvantage of requiring a significant amount of hand-crafted training data; this in itself can require a sizeable investment for any new domain. Much less work has used unsupervised approaches (e.g. [6, 4, 8]).

Our primary aim is to define an unsupervised method to acquire domain-specific terms, allowing us to harvest a large number of domain-specific terms, and making the technique more practical. To achieve this, we leverage the Wikipedia folksonomy, i.e. a large set of user-contributed web documents hand-tagged with categories. Following Milne et al. [5], we contend that **Wikipedia categories** are domain-specific, and thus, Wikipedia categories and entries can replace manually-annotated domain-specific terms to enable self-supervised learning. Wikipedia has previously been shown to be effective training data for self-supervised learning in information extraction tasks, as reported by Wu and Weld [9]. Further, we can use the harvested domain-specific terms and subcategories as seeds/a training corpus to expand the set of domain-specific terms.

Vivaldi and Rodríguez [8] have previously made use of Wikipedia categories, page content and the category hierarchy to retrieve Wikipedia entries as domain-specific terms. Despite similarity between the approaches, our main goal is to evaluate the usability of automatically extracted Wikipedia terms as a self-training corpus. Also, as shown in previous work, supervised methods (based on contextual semantic similarity) generally outperform unsupervised ones. Thus, we focus on extracting a smaller number of high quality Wikipedia terms for training supervised methods, rather than aiming at large-scale unsupervised term extraction. Another contribution of this paper is to evaluate the domain-specificity of Wikipedia. Despite simple overlap checking between Wikipedia

Proceedings of the 16th Australasian Document Computing Symposium, Canberra, Australia, 2 December 2011.

and *Agrivoc* by Milne and Witten [5], it is not clear how well Wikipedia categories correlate with domain-specificity, especially in terms of the category hierarchy. Thus, we test this using human annotators.

One challenge with Wikipedia is that categories become highly specialised and fragmented, and paradoxically less domain-specific [5], as we move further away from the top of the category hierarchy. To combat this, we restrict our selection of subdomains to the top-N categories of the hierarchy, to maximize utility and coverage. To validate the usability of Wikipedia information, we used human annotators to verify the domain-specificity of selected subcategories. Second, to build an inventory of domain-specific terms, we project the task as a classification task. That is, we classify target terms with their domain-specificity relative to the target domain using context-based semantic similarity. To do so, we use extra features from Wikipedia; in particular, we use Wikipedia category entries plus the words contained in the document's first-sentence to describe entries. In evaluation, we test the performance when using of first-sentence words from Wikipedia articles by comparing to the use of snippets.

2 Wikipedia Categories and Domain-Specificity

2.1 Building a Category Hierarchy

Milne and Witten [5] demonstrated that Wikipedia entries contain a significant number of domain-specific terms found in *Agrivoc*, a hand-created domain resource. The authors also showed that the overlap between Wikipedia entries and *Agrivoc* increases with higher degree of domain-specificity. However, due to different degrees of domain-specificity of categories, their technique needs to be modified to construct a proper category hierarchy due to cyclic links. Further, in lower levels of the category hierarchy, the degree of domain-specificity decreases. We thus model domain-specificity in terms of the levels of hierarchy in the category tree. As an illustration of the domain-specificity of Wikipedia entries from specific categories, see the Wikipedia category hierarchy for *Internet* and its subcategories in Figure 1.[1]

In the category hierarchy, nodes indicate categories, and links are between categories. The category hierarchy was mined using breadth-first search, starting from the root category, *Internet*, then recursively adding subcategories linked from the current category set. During this process, we found two issues: (1) a category may have multiple super-categories, and (2) a category may have no super-category. The first problem was addressed by not linking super-categories of a node that already had a super-category in the current category set: e.g. *Cat_C* is linked as a super-category for *Cat_A* but *Cat_B* is not. For the second problem, we observed that when a category with no super-category is linked to sub-categories, these sub-categories generally have an alternate super-category; hence, we can disconnect the original sub-category: e.g., if we disconnect the link between *Cat_E* and *Cat_D*, *Cat_D* is still in the hierarchy. By iterating this process, we constructed a tree with 28 hierarchical levels and 478,332 unique categories, starting with *Internet* as the root.

Level	$[0,1)$	$[1,2)$	$[2,3)$	$[3,4)$	4
L1	0	1	0	19	3
L2	0	0	28	114	0
L3	2	10	78	193	0
L4	3	32	196	165	0
All	5	43	302	491	3

Table 1: Scores of sub-categories for degree of domain-specificity with respect to the *Internet* domain

In the Wikipedia category hierarchy, we found two issues related to the measure of domain-specificity. First, even if categories seem related, they may be placed at different levels: e.g. *Web 1.0* and *Web 2.0* are placed at different levels. Currently, this problem is addressed manually by the Wikipedia community and we do not discuss it here. Second, the degree of domain-specificity may vary even at the same level: e.g. *Web service specification* and *Acid tests* are both located under *Web standards* but receive different domain-specificity annotations from the human annotators. We decided to consider categories down to 5 levels in the tree as domain-specific: this was selected empirically. We leave as future work the relation between tree-depth and domain-specificity.

2.2 Verifying Domain-Specificity of Category

To verify that Wikipedia sub-categories are domain-specific w.r.t. a target domain, we asked three human annotators to score the domain-specificity between a given sub-category and the main category, i.e., *Internet*. The scores are between 0 and 4, where 0 means no relatedness and 4 means domain relatedness with high confidence. We assume that if the score of a sub-category is at least 2, then the sub-category is domain-specific relative to *Internet*. Prior to the actual annotation, the human annotators were trained using other data. In our case, we trained them over the domain *Disease* and its sub-categories. Table 2.2 shows the distribution of average scores over *all* selected sub-categories (*LAll*) and selected sub-categories at each level (*L1-4*). 58.53% of sub-categories have scores of at least 3, and 94.31% of sub-categories have scores of at least 2. The agreement between judges was $R = 0.52$ using Spearman rank correlation. We conclude that identifying domain-specific sub-categories using our method is reliable, though with decreasing reliability with increasing depth in the hierarchy.

[1] Constructed using the Nov 2009 version of Wikipedia.

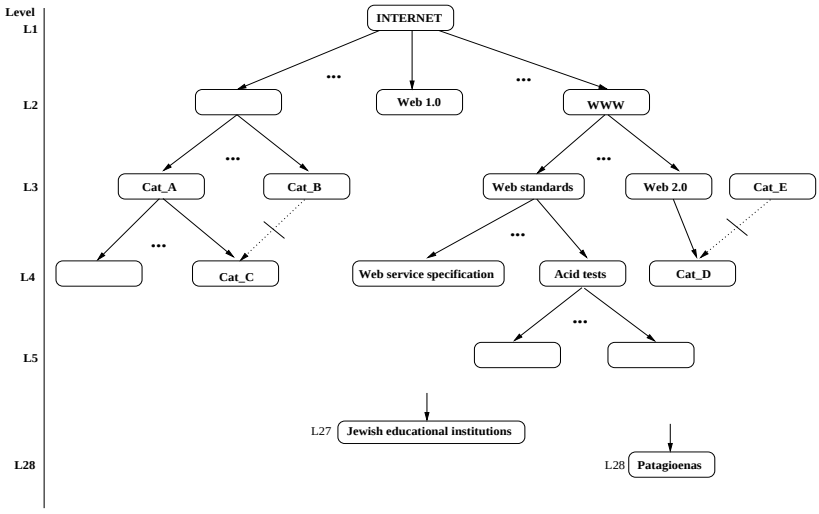


Figure 1: An example of the Wikipedia category hierarchy for *Internet*

3 Domain-Specific Term Identification

3.1 Contextual Semantic Similarity

Our main focus here is to determine whether the use of Wikipedia categories and entries can replace manually-crafted training data for building an inventory of domain-specific terms. We also investigate whether the first sentence of Wikipedia entries can be used as reliable context in identifying domain specificity, much like snippets. To test this second hypothesis, we use a semantic similarity metric using context words in an N-word window, as domain-specific terms often share similar co-occurring words.

To measure semantic similarity among domain-specific terms and targets, snippets are often used as context (e.g. [7]) due to their ease of acquisition and their relatively large size. However, retrieving snippets for large data sets can be time-consuming; snippets may also contain significant noise. On the other hand, we observed that dictionary definitions of words often contain frequent words in the context; however, the coverage of existing dictionaries is relatively low. As an alternative, we follow Kazama and Torisawa [2], who showed that the first sentence of a Wikipedia page is descriptive and can be used as a definition for a Wikipedia entry. For example, the definition of *Microsoft FrontPage* is *A WYSIWYG HTML editor and web site administration tool from Microsoft....* Words such as *HTML, editor, web* often co-occur with context words related to *DreamWeaver*.

3.2 Experiments and Results

In evaluating domain-specific term identification, our interests are in measuring: (1) the quality of Wikipedia terms as training data by comparing them to the use of hand-crafted dictionary terms; and (2) the utility of first-sentence words from Wikipedia in comparison to the use of snippets.

To determine the domain-specificity of categories, we first constructed a category hierarchy starting at *Internet*, and chose to explore categories found down to the 5th level (selected categories), resulting in 19,460 entries over 844 categories. To collect gold-standard data, we took a portion of Wikipedia entries which were already identified as being domain-specific to *Internet*: i.e., we used Wikipedia entries containing up to 3 *Internet*-related document categories. Further, we chose to use only Wikipedia entries that were composed exclusively of ASCII characters and numerals (but not numerals only) as our data, to extract clean snippets as instances. We separately collected an equivalent number of Wikipedia entries from *non-Internet* categories as test and training data using the same selection criteria. We restricted all *non-Internet* entries to have only one category, and every such term to belong to a different category. Finally, we used 20% and 80% of terms for test and training data, respectively. For the test data, we had two human annotators verify the domain-specificity of each entry to *Internet* or *non-Internet*; the inter-annotator agreement was $\kappa = 0.97$.

For features for the test data, we used snippets as a source of context words, since the target words do not necessarily occur in Wikipedia. As features of Wikipedia terms in the training data, we tested snippets and first sentence words from Wikipedia. To collect snippets, we used the top-1000 retrieved snippets based on querying the terms in the test and training data using `Yahoo!BOSS`,[2] then filtered out stop words. The use of first-sentence words as training data significantly reduces the time required to collect

[2] `http://developer.yahoo.com/search/boss/`

Feature I/N	Value	Frequency F5/F1	F10/F2	F15/F3
Training Source : Dictionaries				
D/S	DF	56.73%	57.35%	55.23%
	TF	**70.65%**	70.45%	67.75%
D/F	DF	60.87%	51.19%	49.48%
	TF	**77.23%**	77.02%	56.63%
D/C	DF	52.43%	57.56%	55.85%
	TF	73.29%	**73.65%**	73.29%
Training Source : Wikipedia				
S/S	DF	84.99%	**85.14%**	83.00%
	TF	77.69%	74.73%	71.85%
F/F	DF	70.08%	52.28%	51.24%
	TF	**85.20%**	70.34%	64.18%
C/C	DF	71.33%	73.29%	77.17%
	TF	**84.78%**	84.42%	84.11%

Table 2: **Accuracy of Term Identification**: *I/N* in the Feature column indicates of **I**nternet and **N**on-Internet terms, where *D,S,F,C* indicate **D**ictionary definition, **S**nippets, **F**irst-sentence words, and **C**omplete Wikipedia pages, resp.; *DF/TF* refers to the feature representation (document or term frequency); *FN* (e.g., *F1* and *F5*) indicate that the minimum term frequency is N.

context words. For comparison, we also tested use of the complete Wikipedia page as context. Note that due to the availability of snippets and first sentences from Wikipedia pages, the test data consists of 975 *Internet* terms and 977 *non-Internet* terms; the training data is made up of 3,906 terms for each of *Internet* and *non-Internet*.

For comparison, we collected 4,369 dictionary glossaries for *Internet* training data from 4 different sources.[3] After filtering duplicated terms and 19 terms that were also in the set of test terms, we obtained 2,972 terms as our hand-crafted *Internet* training data. As features for dictionary terms, we used dictionary definitions. For *non-Internet* training data, we used the same Wikipedia terms.

Table 2 shows the performance of our domain-specific term identification experiments using a linear-kernel support vector machine with various inputs. For comparison, a majority-class baseline (50.05%) and accuracies using the dictionaries are shown, since there are no other comparator systems due to the paucity of comparable unsupervised approaches and the unavailability of manually-crafted training data.

Compared with best performance using dictionaries (77.23%), our method achieved 85.20% accuracy at its best. This suggests that our method performs better than existing dictionaries at identifying domain-specific terms. The errors occur for the following reasons: (1) our training data contains noise; (2) the domain-specificity of the training data varies; (3) the size of the training data is relatively small; and (4) the semantic similarity metric can introduce error. Likewise, errors introduced in the approach using dictionaries are observed to be similar. We also note that the lower performance using dictionary terms is due to the smaller number of training terms; dictionary terms also seem to contain only more general words that occur frequently in *Internet*. We further found that using first-sentence words as context performs as well as using snippets — *85.20%* for first-sentence words vs. *85.14%* for snippets — and even outperformed the use of complete Wikipedia pages. Considering that acquiring snippets for large training terms is time-consuming, using first-sentence words not only produced the higher accuracy but also has the benefit of efficiency.

[3] `www.webopedia.com/Top_Category.asp`, `www.cnet.com/Resources/Info/Glossary/index.html`, `www.sharpened.net/glossary/`, `www.matisse.net/files/glossary.html`

4 Conclusions

In this paper, we explored techniques for extracting domain-specific terms from Wikipedia, and used these as seed/training data to predict the domain-specificity of unseen terms. We also showed that first-sentence words from Wikipedia entries are a useful feature for measuring semantic similarity among terms. In evaluation, we manually verified the domain-specificity of selected sub-categories and demonstrated the technique by identifying domain-specific terms over the Wikipedia *Internet* category. We believe that this method can be used to efficiently harvest large collections of domain-specific terms in a range of domains, significantly reducing time and cost for this task.

References

[1] Patrick Drouin. Detection of domain specific terminology using corpora comparison. In *Proceedings of the fourth international Conference on Language Resources and Evaluation*, pages 79–82, Lisbon, Portugal, 2004.

[2] Jun'ichi Kazama and Kentaro Torisawa. Exploiting Wikipedia as external knowledge for named entity recognition. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 698–707, 2007.

[3] Mitsuhiro Kida, Masatsugu Tonoike, Takehito Utsuro and Satoshi Sato. Domain classification of technical terms using the web. *Systems and Computers*, Volume 38, Number 14, pages 2470–2482, 2007.

[4] Su Nam Kim, Timothy Baldwin and Min-Yen Kan. Extracting domain-specific words — a statistical approach. In *Proceedings of the Australasian Language Technology Workshop 2009 (ALTW 2009)*, pages 94–98, Sydney, Australia, 2009.

[5] David Milne, Olena Medelyan and Ian H. Witten. Mining domain-specific thesauri from wikipedia : A case study.

In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 442–448, Washington, USA, 2006.

[6] Youngja Park, Siddharth Patwardhan, Karhik Visweswariah and Stephen C. Gates. An empirical analysis of word error rate and keyword error rate. In *Proceedings of International Conference on Spoken Language Processing*, Brisbane, Australia, 2008.

[7] Leonardo Rigutini, B. Liu and Marco Magnini. An em based training algorithm for cross-language text categorization. In *Proceedings of the Web Intelligence Conference (WI)*, pages 529–535, Compiegne, France, 2005.

[8] Jorge Vivaldi and Horacio Rodríguez. Finding domain terms using wikipedia. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*, Valetta, Malta, 2010.

[9] Fei Wu and Daniel S. Weld. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the ACL (ACL 2010)*, pages 118–127, Uppsala, Sweden, 2010.

Tensor Query Expansion: A cognitively motivated relevance model

Mike Symonds, Peter D. Bruza, Laurianne Sitbon, Ian Turner
Faculty of Science and Technology,
Queensland University of Technology, Brisbane, Qld, Australia
m.symonds@student.qut.edu.au, (p.bruza, l.sitbon, i.turner)@qut.edu.au

Abstract *In information retrieval, a user's query is often not a complete representation of their real information need. The user's information need is a cognitive construction, however the use of cognitive models to perform query expansion have had little study.*

In this paper, we present a cognitively motivated query expansion technique that uses semantic features for use in ad hoc retrieval. This model is evaluated against a state-of-the-art query expansion technique. The results show our approach provides significant improvements in retrieval effectiveness for the TREC data sets tested.

Keywords Information storage and retrieval, query expansion, natural language processing, tensors

1 Introduction

Information retrieval researchers have known that a user's query is typically an imprecise description of the user's real information need ever since the Cranfield experiments in document retrieval in the 1960's. This is all the more relevant today with web queries, which are commonly between two and three words in length. For these reasons there has been, and still is, a strong interest in the use of query expansion techniques. These techniques augment the original query, creating what is hoped to be a more accurate representation of the user's real information need and has been shown to consistently increase retrieval effectiveness [14].

Current state-of-the-art query expansion techniques are often based on word statistics found within documents and ignore information about term dependencies that are inherent to natural language [7, 20]. However, there is growing evidence that suggests query expansion techniques that use term dependency information, such as word proximity and word co-occurrences, can provide more effective query expansion over these traditional approaches [9, 10]. Very few of these dependency based techniques are motivated from a cognitive perspective, from which the user's real information need is created [11].

In this paper we present a formal query expansion approach, we call *tensor query expansion* (TQE), based on a cognitively motivated model of word meaning. We hypothesise that as this approach is more intuitively linked to the cognitive construct of a user's real information need, a word meaning model could be used within the query expansion process to make query representations more like the user's real information need. Our results demonstrate significant improvements in retrieval effectiveness, and support the argument that the user's real information need can be successfully modelled using a cognitive model of word meaning.

Proceedings of the 16th Australasian Document Computing Symposium, Canberra, Australia, 2 December 2011.

2 Related Work

The main areas of research that provide a theoretical framework for our approach include: (i) the use of query expansion techniques to improve the representation of a user's information need, (ii) linguistic theories of word meaning, and (iii) the use of semantic spaces to model word meaning.

2.1 Query Expansion Techniques

State-of-the-art document retrieval models, such as those from the language modelling group, are framed within probabilistic settings, with documents and queries represented as statistical distributions.

The language modelling framework does not have a natural extension for query expansion. Ad hoc approaches have been applied with some success [1, 3]. However, more formal techniques, including Zhai and Lafferty's model-based feedback and Lavrenko and Croft's relevance models are often used [20, 7]. The unigram relevance model is often used as a state-of-the-art benchmark for research into query expansion techniques [10].

The relevance modelling approach involves estimating the probability of observing a word w given some relevant evidence for a particular information need, represented as query Q. The relevance model is a multinomial distribution in which the conditional probability is computed as:

$$P(w|Q) = \int_D P(w|D)P(D|Q). \qquad (1)$$

By assuming that most of the relevant information comes from the set of relevant documents for a query

Q, the conditional probability estimate of equation (1) can be rewritten as:

$$P(w|R) \approx \frac{\sum_{D \in \mathcal{R}_Q} P(w|D)P(Q|D)P(D)}{\sum_w \sum_{D \in \mathcal{R}_Q} P(w|D)P(Q|D)P(D)}, \quad (2)$$

where $\mathcal{R}_Q$ is the set of documents (pseudo) relevant to query Q. To simplify the estimation, there is an assumption that $P(D)$ is uniform over this set of documents. By using a set of the most probable terms from this distribution the query representation is updated, often through linear interpolation with the original query model, and used in re-ranking the documents, as shown:

$$P(w|Q) = \lambda P_o(w|Q) + (1 - \lambda)P(w|R), \quad (3)$$

where λ is the feedback interpolation coefficient that determines the mix with the original estimate $P_o(w|Q)$. In the unigram case, the relevance model estimates are often based on the Dirichlet smoothed term likelihood scores, expressed as:

$$P(w|D) = \frac{df_w + \mu \frac{cf_w}{|C|}}{|D| + \mu}, \quad (4)$$

where df_w is the document frequency of term w, cf_w is the collection frequency of term w, $|C|$ is the word count in the collection, $|D|$ is the word count of the document and μ is the Dirichlet smoothing parameter. Within this paper, this instance of the relevance model is referred to as RM3.

Our approach works within the relevance modelling framework and replaces $P(w|R)$ in equation (3) with an analogous estimate produced by a formal model of word meaning. The retrieval effectiveness of our cognitively motivated relevance model is compared with the unigram based relevance model on a number of newswire data sets. This comparison is centered around the use of word meanings within our cognitive approach.

2.2 Word Meaning

The structuralist theories of linguistics, championed by Ferdinand de Saussure (1916), stated that meaning arose from the relationships between words and provided a relatively clean linguistic framework, free of psychology, sociology and anthropology [4]. Based on these ideas, word meaning was created by two types of relationships: (i) syntagmatic and (ii) paradigmatic associations.

A syntagmatic association exists between two words if they co-occur more frequently than expected from chance. Some common examples may include "coffee-drink" and "sun-hot" [13]. These associations can also have varying strengths. Consider the example sentence "A dog bit the mailman". The term "dog" would likely have a stronger syntagmatic association with "bit" than "mailman", based on the fact that the word "bit" would likely co-occur with "dog" more often.

A paradigmatic association exists between two words if they can substitute for one another in a sentence without affecting the grammaticality or acceptability of the sentence. Some common examples may include "drink-eat" and "quick-fast" [13]. In the example sentence, "A dog bit the mailman", the word "bit" could be replaced with "chased", hence "bit" and "chased" could be said to have a paradigmatic association.

2.3 Semantic Space Models

Linked to structuralist ideas of linguistics, researchers have argued that relationships between words can be modelled by comparing the distributions of words within text [16]. A popular approach to representing these word distributions is to collect word occurrence frequencies and place them in high-dimensional *context* vectors [18]. This approach allows techniques from linear algebra to be used to model relationships between objects, including semantic associations when a word space is developed.

There have been a number of successful psychologically relevant semantic space models that learn semantic associations directly from text, including HAL (Hyperspace Analogue to Language [8]) and LSA (Latent Semantic Analysis [6]). More recent models demonstrate that encoding structural information into the semantic space improves performance on a number of cognitive tasks [5, 15, 17] and can help address weaknesses raised by the lack of structural information in models like LSA [12].

Of these more recent models, the tensor encoding (TE) model [17] provides measures of syntagmatic and paradigmatic associations between words. We argue that this strong connection to structural linguistic theory creates a solid foundation for modelling the creation of word meanings, that likely form part of the user's cognitive process when developing their real information need, for use in an information retrieval task.

3 The Tensor Encoding model

The TE model creates a semantic space based on tensor representations of words. These tensor representations are built by encoding the word order and word co-occurrence information found in natural language. The tensor order created within the space depends on the size of the tuples encoded in the vocabulary binding process.

To gain a better understanding of how the TE model creates tensors that formally capture this word-order and co-occurrence information, consider the second order TE model created for the example sentence, "*A dog bit the mailman*", where *A* and *the* are considered to be stop words (noisy, low information terms that are not included in the vocabulary). Each term in the vocabulary is first assigned an environment vector, which corresponds to the unit vector for the term's id value:

Term-id	Term	Environment vector
1	dog	$\boldsymbol{e}_{dog} = (1 \quad 0 \quad 0)^T$
2	bit	$\boldsymbol{e}_{bit} = (0 \quad 1 \quad 0)^T$
3	mailman	$\boldsymbol{e}_{mailman} = (0 \quad 0 \quad 1)^T$

A *memory tensor* for each term is built by summing the proximity-scaled Kronecker products of the environment vectors within a sliding context window over the text. For the second order TE model, memory matrices are created by the binding process defined by:

$$\boldsymbol{M}_w = \sum_{k \in CW}^{k \prec w} (R - d_k + 1).\boldsymbol{e}_k \otimes \boldsymbol{e}_w^T + \sum_{k \in CW}^{k \succ w} (R - d_k + 1).\boldsymbol{e}_w \otimes \boldsymbol{e}_k^T, \quad (5)$$

where w is the target term, k is a non-stop word found within the sliding context window (CW), $k \prec w$ indicates that term k appears before term w in the context window, $k \succ w$ indicates that term k appears after term w, R is the radius of the sliding context window, and d_k is the distance between term k and term w. Note, stop words are not bound, but they are included when determining the window boundaries. Consider the memory matrices created for the vocabulary terms using a sliding context window with radius 2.

Binding Step 1: $\overbrace{A_s \quad [dog] \quad bit \quad the_s} \quad mailman$

$$\boldsymbol{M}_{dog} = 2 \times \boldsymbol{e}_{dog} \otimes \boldsymbol{e}_{bit}^T = 2 \times \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} (0 \quad 1 \quad 0) = \begin{pmatrix} 0 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Binding Step 2: $\overbrace{A_s \quad dog \quad [bit] \quad the_s \quad mailman}$

$$\boldsymbol{M}_{bit} = 2 \times \boldsymbol{e}_{dog} \otimes \boldsymbol{e}_{bit}^T + \boldsymbol{e}_{bit} \otimes \boldsymbol{e}_{mailman}^T = 2 \times \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} (0\,1\,0) + \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} (0\,0\,1) = \begin{pmatrix} 0 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}.$$

Binding Step 3: $A_s \quad dog \quad \overbrace{bit \quad the_s \quad [mailman]}$

$$\boldsymbol{M}_{mailman} = \boldsymbol{e}_{bit} \otimes \boldsymbol{e}_{mailman}^T = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} (0 \quad 0 \quad 1) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}.$$

The resulting pattern is that all non-zero elements are situated on the row or column corresponding to the target term's term-id. If this vocabulary building process was performed over the entire corpus the general form of a *memory matrix* would be similar to:

$$\boldsymbol{M}_w = \begin{pmatrix} 0, \quad \ldots, 0, \quad f_{1w}, \quad 0, \quad \ldots, 0 \\ \cdots \\ 0, \quad \ldots, 0, \quad f_{(w-1)w}, \quad 0, \quad \ldots, 0 \\ f_{w1}, \ldots, f_{w(w-1)}, f_{ww}, f_{w(w+1)}, \ldots, f_{wN} \\ 0, \quad \ldots, 0, \quad f_{(w+1)w}, \quad 0, \quad \ldots, 0 \\ \cdots \\ 0, \quad \ldots, 0, \quad f_{Nw}, \quad 0, \quad \ldots, 0 \end{pmatrix},$$

where f_{iw} is the value in row i column w of the matrix which represents the proximity scaled co-occurrence frequencies of term i before term w, f_{wj} is the value in row w column j of the matrix that represents the proximity scaled co-occurrence of term j after term w, and N is the number of unique terms in the vocabulary.

It is worth noting that the illustrative example provided was for the second order implementation of the TE model, and hence the representations are matrices. The TE model can be extended to form higher order tensors that hold n-tuple information by modifying equation (5) to sum the Kronecker products of n-tuples. For this research we chose to use the second order TE model. Using higher order TE models is left for future work.

The sparse memory matrices of the second order TE model provide opportunities for efficient construction, as outlined in [17]. They also allow for efficient evaluation measures to be developed to achieve the goal of modelling word meaning.

3.1 Computing Word Meaning

Based on the structuralist theories of linguistics the TE model aims to extract the strength of syntagmatic and paradigmatic associations between terms to allow it to construct word meaning.

3.1.1 Syntagmatic associations:

Due to the unique structure of the memory matrices, it was shown in [17] that the strength of syntagmatic associations between a sequence of priming terms $Q = (q_1, \ldots, q_p)$ and any vocabulary term w, can be efficiently calculated by measuring the cosine of the angle θ between the memory matrices for Q ($\boldsymbol{M_Q}$) and w ($\boldsymbol{M_w}$):

$$s_{\text{syn}}(Q, w) = \cos\theta = \frac{< \boldsymbol{M_Q}, \boldsymbol{M_w} >}{||\boldsymbol{M_Q}||_F ||\boldsymbol{M_w}||_F}, \quad (6)$$

where

$$< \boldsymbol{M_Q}, \boldsymbol{M_w} >= \sum_{\substack{j=1 \\ w \in Q}}^{N} s_w^2 f_{jw}^2 + \sum_{\substack{j=1 \\ j \neq w \\ w \in Q}}^{N} s_w^2 f_{wj}^2 + \sum_{\substack{i=q_1 \\ i \neq w}}^{q_m} (s_i^2 f_{wi}^2 + s_i^2 f_{iw}^2),$$

and

$$||\boldsymbol{M_Q}||_F = \sqrt{\sum_{i=q_1}^{q_m} \left[\sum_{j=1}^{N} s_i^2 f_{ji}^2 + \sum_{\substack{j=1 \\ j \neq i}}^{N} s_i^2 f_{ij}^2 \right]},$$

and

$$||\boldsymbol{M_w}||_F = \sqrt{\sum_{j=1}^{N} f_{jw}^2 + \sum_{\substack{j=1 \\ j \neq w}}^{N} f_{wj}^2},$$

and where the memory matrix for Q was constructed by summing the memory matrices of the individual terms

in the sequence $\boldsymbol{M_Q} = \boldsymbol{M_{q_1}} + \ldots + \boldsymbol{M_{q_p}}$, $q_1, \ldots, q_m$ are the list of m unique priming terms found in Q having $m \leq p$, s_i is the number of times term q_i appears in Q, f_{ab} is the co-occurrence frequency of term a appearing before term b in the vocabulary, f_{ba} is the co-occurrence frequency of term a appearing after term b.

This measure of syntagmatic association was shown to have linear time complexity and be effective at predicting words that are most likely to precede or succeed another word in text [17].

3.1.2 Paradigmatic associations:

Having explicit 2-tuple co-occurrence information stored in the memory matrices means that the TE model allows probabilistic measures to be used in addition to geometric measures. This means that information theoretic measures, like mutual information, could be used if they provided the best performance for a given task. However, for measuring the strength of paradigmatic associations between a sequence of priming terms $Q = (q_1, \ldots, q_p)$ and a vocabulary term w, the following measure was shown in [17] to perform better on a synonym judgement task than other recent semantic space models encoding structural information:

$$s_{\text{par}}(Q, w) = \frac{1}{Z_{\text{par}}} \sum_{j=q_1}^{q_p} \sum_{i=1}^{N} \frac{f_{ij} f_{iw} + f_{ji} f_{wi}}{f_j f_w}, \quad (7)$$

where f_j is the vocabulary frequency of term j, f_{ji} is the ordered co-occurrence frequency of term j before term i, N is the size of the vocabulary, and Z_{par} normalizes the scores, such that $\sum_{w \in V} s_{\text{par}}(Q, w) = 1$.

As we plan to combine the syntagmatic and paradigmatic measures for the task of query expansion, we chose to modify equation (8) to extract more pure paradigmatic associations, and hence reduce the scores of terms that co-occur with query terms (indicating syntagmatic relations). The resulting paradigmatic feature function was defined as:

$$s_{\text{par1}}(Q, w) = \frac{1}{Z_{\text{par1}}} \sum_{j=q_1}^{q_p} \sum_{i=1}^{N} \frac{f_{\overline{ij}} \cdot f_{\overline{iw}}}{\max(f_{\overline{ij}}, f_{\overline{iw}}, f_{\overline{wj}})^2}, \quad (8)$$

where $f_{\overline{ij}} = (f_{ji} + f_{ij})$, $f_{\overline{iw}} = (f_{wi} + f_{iw})$, $f_{\overline{wj}} = (f_{jw} + f_{wj})$, N is the size of the vocabulary, max() returns the maximum argument value, and Z_{par1} normalizes the scores, such that $\sum_{w \in V} s_{\text{par1}}(Q, w) = 1$.

4 Tensor Query Expansion

Research into models of memory have demonstrated that the type of semantic information that is most useful to perform a given task varies [5, 17]. For example, on a synonym judgement task, paradigmatic associations are most helpful, while on a task estimating the most common pre-ceding or post-ceding term for a target term, the syntagmatic associations are most useful.

For the task of query expansion, we assume that the underlying word meanings that form the user's information need are likely formed by a mix of both syntagmatic and paradigmatic associations. This assumption will be tested by comparing the retrieval effectiveness achieved when different mixes of syntagmatic and paradigmatic associations are used to form estimates for query models within the relevance modelling framework on an ad hoc retrieval task.

As outlined in section 2.1, relevance models provide a formal method for query expansion within the language modelling framework. Equation (2) shows the relevance model process includes estimating the probability $P(w|R)$, of observing a word w based on relevant evidence, often (pseudo) relevant documents, for a particular query Q using a multinomial distribution. Our aim will be to create an analogous distribution to estimate $P(w|R)$. These estimates will be based on word meanings, more specifically the combination of syntagmatic and paradigmatic associations found with the vocabulary created by the TE model on a set of (pseudo) relevant documents. We call this query expansion technique, *tensor query expansion* (TQE).

To formally estimate the conditional probability we use a Markov random field, similar to that used in [10]. Let an undirected graph G contain nodes that represent random variables, and the edges define the independence semantics between the random variables. Within the graph, a random variable is independent of its non-neighbours given observed values of its neighbours.

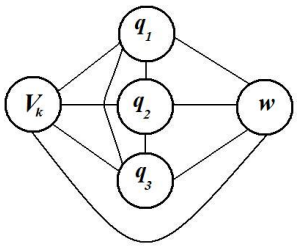


Figure 1: Example of the TQE graphical model for a three term query.

Figure 1 shows a graph G that consists of query nodes q_i, expansion term node w, and a vocabulary node V_k. Term w is constrained to exist within the vocabulary V_k, which is built from a set of k documents considered (pseudo) relevant to Q. We parameterize the graph based on clique sets to provide more flexibility in encoding useful features over cliques in the graph. The joint distribution over the random variables in G is defined by:

$$P_{G,\Gamma}(Q, w, V_k) = \frac{1}{Z_\Gamma} \prod_{c \in cl(G)} \varphi(c; \Gamma), \quad (9)$$

where $Q = q_1, \ldots, q_p$, $cl(G)$ is the set of cliques in G, each $\varphi(.; \Gamma)$ is a non-negative *potential function* over clique configurations parameterized by Γ,

and $Z_\Gamma = \sum_{Q,w} \prod_{c \in cl(G)} \varphi(c;\Gamma)$ normalizes the distribution. The joint distribution is uniquely defined by the graph G, potential functions φ and the parameter Γ. Using the fact that the logarithm of products is equal to the sum of logarithms, the simplified form of the joint distribution becomes:

$$\log P_{G,\Gamma}(Q, w, V_k) = \frac{1}{Z_\Gamma} \sum_{c \in cl(G)} \log \varphi(c;\Gamma), \quad (10)$$

where the potential functions are commonly parameterized as:

$$\varphi(c;\Gamma) = \exp[\gamma_c f(c)], \quad (11)$$

with $f(c)$ being some real-valued *feature function* over clique values and γ_c is the weight given to that particular feature function. Substituting equation (11) into equation (10) gives:

$$\log P_{G,\Gamma}(Q, w, V_k) = \frac{1}{Z_\Gamma} \sum_{c \in cl(G)} \gamma_c f(c). \quad (12)$$

After G is constructed, we can compute the conditional probability of an expansion term w given Q, as:

$$P_{G,\Gamma}(w|Q) = \frac{P_{G,\Gamma}(Q, w, V_k)}{\sum_{w \in V_k} P_{G,\Gamma}(Q, w, V_k)}, \quad (13)$$

where V_k is the universe of all possible vocabulary terms and w is a possible expansion term.

By using equation (12) and equation (13) with constant terms removed, a rank equivalent form for the conditional probability can be written as:

$$P_{G,\Gamma}(w|Q) \propto \sum_{c \in cl(G)} \gamma_c f(c), \quad (14)$$

where a constraint of $\sum_{c \in cl(G)} \gamma_c = 1$ is applied for ease of training.

4.1 Model Parameterization

The conditional probability expressed in equation (14), provides a formal method for combining feature functions, designed to extract various types of vocabulary term dependencies, mapped via cliques in the graph. For the graph shown in figure 1, a number of useful clique sets capturing dependencies are summarised in table 1.

Since it is not our goal to find optimal feature functions, but to demonstrate the use of a Markov random field to formally combine feature functions that model syntagmatic and paradigmatic associations, we focus on evaluating estimates over the clique sets relevant to the syntagmatic and paradigmatic measures.

To enable a more balanced comparison of the influence of each feature we first convert the syntagmatic measure to a distribution by normalising the scores, such that the feature function in equation (6) becomes:

Set	Description
T_{par}	Set of cliques containing the vocabulary node and exactly one query term node and the expansion term (w) node.
T_{syn}	Set of cliques containing the vocabulary node and exactly one query term node and the expansion term (w) node, with query term node and expansion term node connected by an edge.

Table 1: Summary of TQE clique sets to be used.

$$s_{\text{syn1}}(Q, w) = \frac{1}{Z_{\text{syn1}}} s_{\text{syn}}(Q, w), \quad (15)$$

where $Z_{\text{syn1}} = \sum_{w \in V_k} s_{\text{syn}}(Q, w)$ normalises the scores. Using the T_{syn} and T_{par} clique sets, and our feature functions $s_{\text{syn1}}(Q, w)$ and $s_{\text{par1}}(Q, w)$, equation (14) becomes:

$$P_{G,\Gamma}(w|Q) \propto \gamma_{T_{\text{syn}}} s_{\text{syn1}}(Q, w) + \gamma_{T_{\text{par}}} s_{\text{par1}}(Q, w), \quad (16)$$

where $\gamma_{T_{\text{syn}}}, \gamma_{T_{\text{par}}} \in [0, 1]$ and $\gamma_{T_{\text{syn}}} + \gamma_{T_{\text{par}}} = 1$. By normalising the distribution and replacing $\gamma_{T_{\text{syn}}}$ and $\gamma_{T_{\text{par}}}$ with a single interpolation parameter, γ, the rank equivalent estimate in equation (16) can be rewritten as:

$$P_{G,\Gamma}(w|Q) = \frac{1}{Z_\Gamma} \left[\gamma s_{\text{syn1}}(Q, w) + (1 - \gamma) s_{\text{par1}}(Q, w)\right], \quad (17)$$

where $\gamma \in [0, 1]$, mixes the amount of syntagmatic and paradigmatic features used in the estimation, and $Z_\Gamma = \sum_{w \in V_k} [\gamma s_{\text{syn1}}(Q, w) + (1 - \gamma) s_{\text{par1}}(Q, w)]$, is used to normalise the distribution.

As the estimate in equation (17) is considered analogous to the estimate $P(w|R)$ used in the relevance modelling framework, we argue that our $P_{G,\Gamma}(w|Q)$ can replace the $P(w|R)$ in equation (3) giving us a cognitively motivated method of updating the query model within the language modelling framework. It is worth noting that one of the major differences between the unigram based relevance model and our approach is that our estimates are based on the vocabulary measures, not the document statistics. Using the relevance models feedback interpolated form shown in equation (3), the final conditional probability becomes:

$$P(w|Q) = \lambda P_o(w|Q) + (1 - \lambda) P_{G,\Gamma}(w|Q). \quad (18)$$

Our cognitively motivated relevance model needs to also be considered in terms of the computational costs of extracting these semantic features from the (pseudo) relevant document set.

5 Computational Complexity

The TQE technique uses two semantic features that measure the strength of syntagmatic and paradigmatic associations. The creation of the memory matrices in equation (5) provides a formalism for capturing the

co-occurrences and encoding word order. However, the original TE model research [17] demonstrated that the word order and co-occurrence information is efficiently captured within low dimension storage vectors (SV) due to the unique structure of the memory matrices.

The dimensionality of the storage vectors required is based on the size of the vocabulary created and the radius of the context window used in the vocabulary binding process.

For example, on a synonym judgement task using a vocabulary of 134,000 terms, the TE model's best performance was achieved using the paradigmatic measure, a context window of radius one and storage vectors of 1,000 dimensions [17]. This supports previous research [15] that showed paradigmatic associations are most effectively modelled when a very small context window is used. When this is considered alongside the fact that the vocabulary size created from the set of top 30 (pseudo) relevant documents in our ad hoc retrieval experiments was less than 10,000, a storage vector of 50 dimensions is chosen to model the paradigmatic associations within our TQE approach.

Considering the worst case time complexity of the paradigmatic feature in equation (7) is $T(n) = O(\frac{D^2_{SV_{\text{par}}}}{4}.|Q|)$, where $D_{SV_{\text{par}}}$ is the dimensionality of the storage vector (set to 50), and $|Q|$ is the length of the query, keeping the dimensionality of the storage vector small is important.

However, for tasks relying on syntagmatic associations, past research [19] has shown that using larger context windows leads to better performance. For this reason we chose to create a separate semantic space to model syntagmatic associations. This semantic space was built using a context window of radius 150 and storage vectors with 500 dimensions. Based on these dimensions the memory footprint of the storage vectors to build the two semantic spaces used by the TQE technique would be at most 21 MBytes ($550 \times 10,000$ integers), assuming a four byte integer.

The original TE model research [17] showed that the worst case time complexity of the syntagmatic feature, in equation (6) was $T(n) = O(\frac{D_{SV_{\text{syn}}}}{2}.|Q|)$, where $D_{SV_{\text{syn}}}$ is the dimensionality of the syntagmatic storage vector (set to 500 in the TQE approach).

6 Experimental Results

Evaluation of the TQE approach was performed on the TREC data sets outlined in table 2. The AP and WSJ data sets were chosen as they were likely to contain very different content from each other, and hence should form different strength semantic associations for the same queries.

A common approach for evaluating query expansion approaches is through the measure of average retrieval effectiveness and robustness on ad hoc retrieval tasks using pseudo-relevance feedback [10].

Name	Description	# Docs	Topics
AP	Assoc. Press 88-90	242,918	train: 1-150 test: 151-200
WSJ	Wall Street Journal 87-92	173,252	train: 1-150 test: 151-200

Table 2: Overview of TREC collections and topics

The experiments in this research were carried out using a modified version of the Lemur Toolkit[1]. All collections were stopped with the default 418 word Indri stop list and stemmed using a Porter stemmer. In all experiments, only the title component of the topics were used to construct the initial queries.

6.1 Ad Hoc Retrieval

The TQE approach was compared to a baseline unigram language model (noFB) and a unigram relevance model (RM3). The following ad hoc retrieval experiments use a manual train/test split, as outlined in the *Topics* column of table 2.

For RM3 the Dirichlet smoothing parameter, μ was trained and for the TQE approach, the mixing parameter γ was trained. Both TQE and RM3 were evaluated using 30 feedback documents and 30 expansion terms on all data sets. The mean average precision (MAP) for the top ranked 1000 documents are reported in table 3.

Test Set	noFB	RM3	TQE
AP (151-200)	0.2112	0.2495^{α}	$0.2683^{\alpha\beta}$
WSJ (151-200)	0.3244	0.3546^{α}	$0.3831^{\alpha\beta}$

Table 3: Mean average precision (MAP) scores for the unigram language model (noFB), unigram relevance model (RM3) and cognitively motivated relevance model (TQE). The superscripts α and β indicate statistically significant improvements using a two-tailed paired t-test ($p < 0.05$) over noFB and RM3 respectively.

The ad hoc retrieval results suggest that the TQE approach can significantly improve the average precision results when compared to both RM3 and the baseline language model (noFB) on the WSJ and AP data sets. To gain better insight into how the TQE retrieval effectiveness for each query compares to that of the unigram relevance model a robustness analysis is worthwhile.

6.2 Robustness

Robustness includes considering the ranges of relative increase/decrease in average precision and the number of queries that were improved/hurt, with respect to RM3. Figure 2 illustrates the robustness of the TQE average precision scores reported in table 3 on the AP and WSJ data sets. In each graph in figure 2 the test

[1]The Lemur toolkit for language modelling and information retrieval: http://www.lemurproject.org

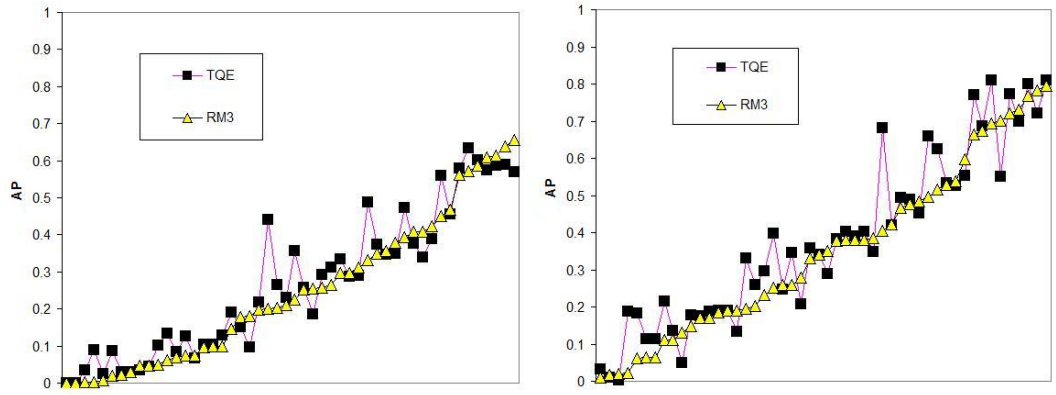


Figure 2: Robustness comparison of RM3 and TQE on the AP (left) and WSJ (right) test topics.

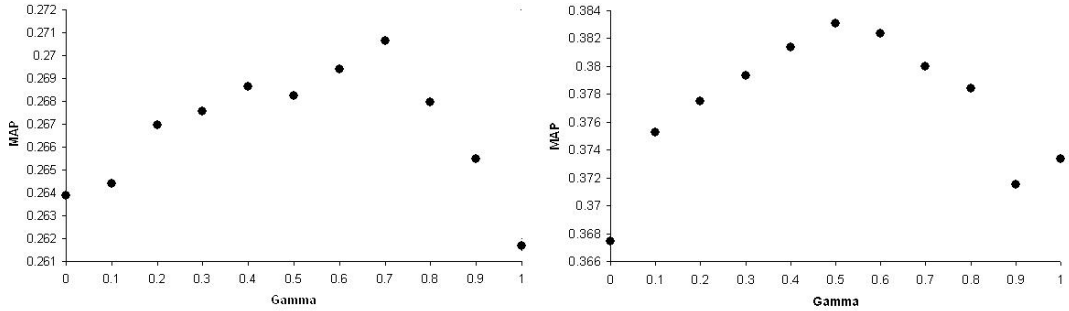


Figure 3: Effect of γ on MAP for the TQE approach on the AP (left) and WSJ (right) test data sets.

topic data is ordered from left to right in ascending MAP score achieved by the RM3 approach.

These graphs suggest that the TQE approach does not outperform RM3 on every test topic. The topics which TQE boosted the average precision score the most over RM3 were TREC topic 159: *Electric Car Development* (on AP) and TREC topic 156: *Efforts to enact Gun Control Legislation* (on WSJ). The TQE approach hurt the AP score the most for TREC topics 172: *The Effectiveness of Medical Products and Related Programs Utilized in the Cessation of Smoking* (on AP), and 157: *Causes and treatments of multiple sclerosis (MS)* (on WSJ).

It was noted that even though TREC topic 157 was hurt the most on the WSJ data set, this same topic had the second largest percent increase in average precision over RM3 on the AP data set. This indicates the importance of the TQE vocabulary in determining the expansion terms. An analogy can be made with the results you may receive if you asked two people, one who had read only the Associated Press news articles and the other solely the Wall Street Journal articles, what type of articles they believed a query like: *Causes and treatments of multiple sclerosis (MS)* should return. The difference in their responses may be attributed to the fact that the WSJ corpus is not as likely to contain articles on multiple sclerosis, and hence may not produce as strong a group of semantic associations for the query terms, especially the more informative terms, such as sclerosis.

This example also highlights the fact that the TQE approach does not account for the information value of specific query terms. Measures, such as *inverse document frequency* (idf), have been shown to improve the performance of information retrieval measures. Testing whether measures like idf can boost retrieval effectiveness of the TQE approach is left for future work.

6.3 Parameter sensitivity

The retrieval effectiveness for various gamma values in equation (17) are shown in figure 3. This graph illustrates that the optimal performance is achieved when both associations are used to create the estimates. This supports our assumption that the task of query expansion is benefited by both syntagmatic and paradigmatic associations. It also leads us to conclude that the TE model of word meaning can be used to model cognitive aspects of the user's real information need that assists the information retrieval process.

7 Conclusions and Future Work

The focus of this paper has been to present the TQE approach, a query expansion technique set in the relevance

modelling framework that is underpinned by a cognitively motivated model of word meaning. The TQE approach formally builds an efficient semantic space that augments the query model using vocabulary based semantic features. The TQE approach was able to significantly outperform a unigram relevance model on two TREC newswire data sets.

The findings of this research also suggest that word order and co-occurrence information stored in the TE model can effectively model the types of word meanings that may underpin the user's real information need. We believe our vocabulary based approach, along with strong linguistic and cognitive motivation, adds weight to the growing number of successful query expansion approaches using term dependencies [10, 9].

An evaluation of TQE on larger document collections and in comparison to some of the other dependency based approaches is the next step in comparing and contrasting the key features of the TQE approach.

An area for future work includes extending the binding operation within the TE model, that underpins the TQE approach, to capture information relating to 3-tuples. Recent research on creating distributed memory models using higher order tensors has shown useful property associations may be captured when 3-tuples and 3-grams are considered [2].

References

[1] Jing Bai, Dawei Song, Peter Bruza, Jian-Yun Nie and Guihong Cao. Query expansion using term relationships in language models for information retrieval. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 688–695, New York, NY, USA, 2005. ACM.

[2] Marco Baroni and Alessandro Lenci. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, Volume 36, pages 673–721, 2010.

[3] P. D. Bruza and D. Song. Inferring query models by computing information flow. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 260–269, New York, NY, USA, 2002. ACM.

[4] Norman N. Holland. *The Critical I*. Columbia University Press, New York, USA, 1992.

[5] Michael N. Jones and Douglas J. K. Mewhort. Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, Volume 114, pages 1–37, 2007.

[6] T. K. Landauer and S. T. Dumais. A solution to plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, Volume 104, pages 211–240, 1997.

[7] Victor Lavrenko and W. Bruce Croft. Relevance-based language models. *In Proceedings of the 24th Annual ACM Conference of Research and Development in Information Retrieval*, pages 120–127, 2001.

[8] K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior research methods, instruments and computers*, Volume 28, pages 203–208, 1996.

[9] Yuanhua Lv and ChengXiang Zhai. Positional relevance model for pseudo-relevance feedback. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 579–586, New York, NY, USA, 2010. ACM.

[10] Donald Metzler and W. Bruce Croft. Latent concept expansion using markov random fields. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 311–318, New York, NY, USA, 2007. ACM.

[11] Stefano Mizzaro. How many relevances in information retrieval? *Interacting With Computers*, Volume 10, pages 305–322, 1998.

[12] Charles A. Perfetti. The limits of co-occurrence: Tools and theories in language research. *Discourse Processes*, Volume 25, pages 363–377, 1998.

[13] Reinhard Rapp. The computation of word associations: comparing syntagmatic and paradigmatic approaches. In *Proceedings of the 19th international conference on Computational linguistics - Volume 1*, pages 1–7, Morristown, NJ, USA, 2002. ACL.

[14] J. Rocchio. *Relevance Feedback in Information Retrieval*, pages 313–323. Prentice-Hall, 1971.

[15] Magnus Sahlgren, Anders Holst and Pentti Kanerva. Permutations as a means to encode order in word space. In V. Sloutsky, B. Love and K. Mcrae (editors), *Proceedings of the 30th Annual Conference of the Cognitive Science Society*, pages 1300–1305. Cognitive Science Society, Austin, TX, 2008.

[16] Hinrich Schütze. Word space. In *Advances in Neural Information Processing Systems 5*, pages 895–902. Morgan Kaufmann, 1993.

[17] Michael Symonds, Peter Bruza, Laurianne Sitbon and Ian Turner. Modelling word meaning using efficient tensor representations. In *Proceedings of the 25th Pacific Asia Conference on Language, Information and Computation (PACLIC25)*. In Press, 2011.

[18] Peter D. Turney and Patrick Pantel. From frequency to meaning: vector space models of semantics. *Journal of Artificial Intelligence Research*, Volume 37, pages 141–188, January 2010.

[19] Jinxi Xu and W. Bruce Croft. Query expansion using local and global document analysis. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 4–11, New York, NY, USA, 1996. ACM.

[20] Chengxiang Zhai and John Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 403–410, New York, NY, USA, 2001. ACM.

Fast Search Engine Vocabulary Lookup

Xiang-Fei Jia
Computer Science
University of Otago
Dunedin, New Zealand
fei@cs.otago.ac.nz

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

Jason Holdsworth
Information Technology
James Cook University
Cairns, Australia
jason.holdsworth@jcu.edu.au

Abstract *The search engine vocabulary is normally stored in alphabetical order so that it can be searched with a binary search. If the vocabulary is large, it can be represented as a 2-level B-tree and only the root of the tree is held in memory. The leaves are retrieved from disk only when required at runtime. In this paper, we investigate and address issues associated with the 2-level B-tree structure for vocabulary lookup. Several compression algorithms are proposed and compared. the proposed algorithms not only performs well when the leaves are stored on disk, but also when the whole vocabulary is stored in memory.*

Keywords Inverted Files, Vocabulary Compression.

1 Introduction

Inverted files [16, 15] are the most widely used index structures in Information Retrieval (IR). An inverted file typically contains two parts, a vocabulary of unique terms extracted from a document collection and a list of postings (normally a pair of ⟨document number, term frequency⟩) for each of the vocabulary terms. Due to the large size of the postings, various techniques have been developed to improve the processing of them, including better compression algorithms [12], impact ordering [9, 2] and pruning [2, 13, 5]. With the applied techniques, the postings can be efficiently loaded and processed. What becomes of interest is the time taken to search through the vocabulary in order to locate the postings for the terms.

The vocabulary is normally stored in alphabetical order so that it can be searched with a binary search. If the vocabulary is large, it can be represented as a 2-level B-tree and only the root of the tree is held in memory. The leaves are retrieved from disk only when required at runtime. Three steps are required to locate the postings for the terms; (1) A binary search of the root gives the disk location of the leaf. (2) One disk-seek and one disk-read load the leaf into memory. (3) A search through the leaf gives the location of the postings.

Proceedings of the 16th Australasian Document Computing Symposium, Canberra, Australia, 2 December 2011.

There are three efficiency issues related to the 2-level B-tree structure. First, disks are an order of magnitude slower than main memory and it can take a long time to read the leaves. Second, The size of the leaves has a big impact on the performance of disk I/O. If the size is large, it can take a long time to read. However, large leaves might benefit from disk caching provided by operating systems [4]. On the other hand, small leaves are fast to read, but cannot benefit much from the caching. The last issue is how the leaves should be compressed. Algorithms with better compression ratios might be slow to decompress at runtime, due to the complexity of the algorithms.

In this paper, we investigate and address issues associated with the 2-level B-tree structure for vocabulary lookup. A number of compression algorithms are proposed and compared. As shown in the results, the proposed algorithms not only performs well when the leaves are stored on disk, but also when the whole vocabulary is stored in memory. The *embedfixed* algorithm provided the best trade-off between compression ratio and fast lookup.

2 In-memory Algorithms

Manning et al. [8] discuss storage efficiency for *in-memory* vocabulary compression and provide three data structures. This section gives a brief discussion of those structures.

In *fixed*, vocabulary terms are sorted in alphabetical order and stored in fixed-width blocks as shown in Figure 1(a). The example shown in the figure allocates 24 bytes for each term, followed by an 8-byte postings pointer (the location of the postings for the term). Terms are null-terminated (shown as "ø") for fast string comparison. This fixed structure is clearly not very space efficient, since the average length of a term is expected to be small. Also terms longer than 24 characters have to be truncated. In order to alleviate these problems it is better to store terms in variable-width structures.

The *string* structure stores the vocabulary terms as a long string of characters as shown in Figure 1(b). The vocab pointers are used to identify the beginning of each term and the end of each term is specified by the next vocab pointer. As strings are not null-terminated

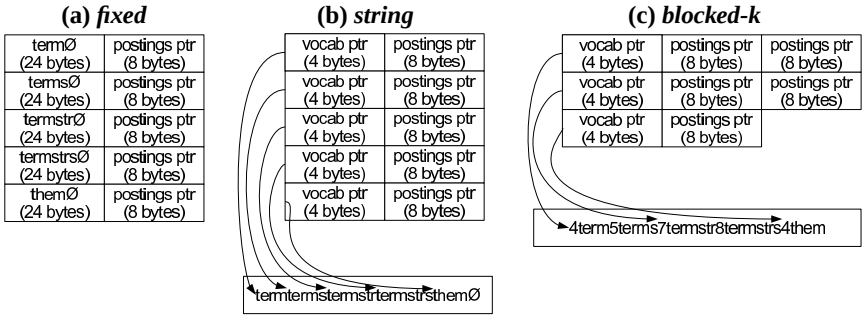


Figure 1: The in-memory vocabulary structures from Manning et al. [8]. In *blocked-k*, k has a value of 2.

special string comparison using string length has to be used.

The *blocked-k* structure further extends *string* by grouping terms into blocks of size k and assigning a vocab pointer to each group. An example of *blocked* with $k = 2$ is shown in Figure 1(c). An exception is that the last block has only one term since there are only five terms in the illustrated example.

The *blocked-k* structure provides better compression ratio by reducing the number of vocab pointers required. However it takes $O(log_2(n/k) + k)$ to search instead of $O(log_2 n)$ for *fixed* and *dictionary-as-string*, where k is the blocking factor and n is the total number of terms in the vocabulary.

3 Fast Lookups with 2-level B-tree

For large data collections and systems with limited resources, it is not feasible to store the whole vocabulary in memory. In this section we discuss how to efficiently compress the vocabulary in terms of both storage space and speed of search using a 2-level B-tree. The root of the tree is stored in memory. The leaves are stored on disk and retrieved only when required.

3.1 Prior Algorithms

It is easy to convert Manning's structures into 2-level B-tree structures and he suggests this. The header structure (the root of the 2-level B-tree) is constructed to allow fast lookup using binary search in order to locate the leaf. The leaf structure is built with the consideration of disk properties so that leaves can be stored efficiently on disk and read quickly from disk.

Two fields are required for each entry in the header structure. The first field stores the first term stored in each leaf. The header terms are null-terminated for fast string comparison. The second field stores the disk offset to the leaf on disk. The size of the leaf pointer is dependent on the vocabulary file size and might be either 4 or 8 bytes. Throughout our discussion, we use 8 bytes for the leaf pointer. The *leaf count* entry at the beginning of the header stores the number of leaves. The left part of Figure 2(a) shows a typical header. On average, it takes $O(log_2(lc))$ to locate the leaf in the header, where lc is the number of leaves.

In order to optimise disk I/O, a leaf should be a whole number of physical disk sectors. The sector is the smallest unit a disk can read/write. If leaves are sector aligned, the time taken to read a leaf should be minimised. During construction of the leaf, as many terms as possible are inserted into the leaf and the remaining bytes padded with zeros.

In order to convert the original *fixed* structure into a 2-level B-tree, each entry in the structure should be given a specific size so that terms do not cross sector boundaries. As shown in Figure 1(a) in our experiments, each entry is assigned 32 bytes, 24 bytes for storing the term and the other 8 bytes for the postings pointer. The size of a disk sector is normally 512 bytes, which is evenly divisible by 32. There is no need to store a header, instead the header can be constructed at startup time.

Figure 2(a) and Figure 2(b) show the original *string* and *blocked* represented as a 2-level B-tree. The header on the left in the Figure 2(a) is that describe above. The leaf structures are identical to the orignal structures encoded using *string* and *block-k* respectively, the *leaf length* entry is added to the beginning of the structure and stores the number of entries in the leaf.

3.2 New Algorithms

Our new *embed* algorithm is based on the assumption that similar data types should be grouped together as CPUs are good at caching and fast at reading data of a fixed length. The data types used in the leaf structure are word-aligned as CPUs read/write data in units of words. A vocab pointer is assigned to each term stored in the leaf and the stored terms are not compressed, thus no de-serialisation is required. As shown in Figure 2(c), *embed* leaf first stores the postings pointers, followed by the vocab pointers, then null-terminated strings and finally the number of entries in the leaf. The header structure of *embed* is the same as the header structure in 2-level *string*. The leaf structure of *embed* is similar to 2-level *string* with the exceptions that (1) elements are re-ordered for faster CPU processing and (2) terms are

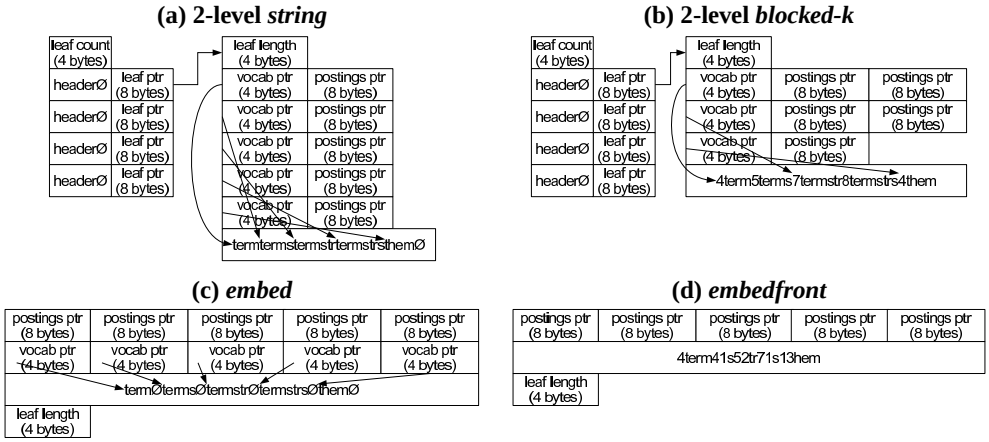


Figure 2: (a) The header and leaf structures for 2-level *string*. (b) The leaf structure for 2-level *blocked-k*, (c) The leaf structure for *embed*. (d) The leaf structure for *embedfixed*. The header structure of 2-level *string* is also used in 2-level *blocked-k*, *embed* and *embedfixed*.

null-terminated so that standard intrinsic string comparison can be performed.

The search time of *embed* is $O(log_2(lc))$, where lc is the number of leaves in the header, plus $O(log_2(ll))$, where ll is the number of entries in the leaf. The performance of *embed* is efficient as it allows binary search in the header and leaf structures to locate the term (no de-serialisation is required). However, *embed* is not space efficient. Terms inside each leaf are not compressed. Two new algorithms are introduced to overcome this problem, *embedfront* and *embedfixed*.

The *embedfront* algorithm uses front coding [14, p. 122] to compress terms in each leaf. Front coding takes advantage of the fact that adjacent terms in sorted alphabetical order tend to share a common prefix. Two integers (we use one-byte integers) are used; one to track the size of the common prefix between the current term and the previous one, and another is the size of the suffix for the current term. As shown in Figure 2(d), "term,terms,termstr,termstrs,them" is front encoded as "4term41s52tr71s13hem". The first term has a value of zero for the common prefix count and there is no need to store this.

Since there is no way to access the terms directly without de-serialisation, there is no need to store the vocab pointers for each term. The header structure of *embedfront* is identical to the header structure in 2-level *string*.

The *embedfront* algorithm not only compresses terms more effectively but also does not need vocab pointers. However, the search time in the leaf requires a linear search of $O(log_2(ll))$ or a de-serialisation and a binary search of $O(ll + log_2(ll))$.

3.2.1 Embedfixed

The *embed* algorithm allows fast lookup, but provides no compression of terms. On the other hand, *embedfront* uses front coding to compress the terms stored in leaves, but de-serialisation is required for lookup. An intermediate solution, e.g. *embedfixed*, is proposed to provide a trade-off between *embed* and *embedfront*.

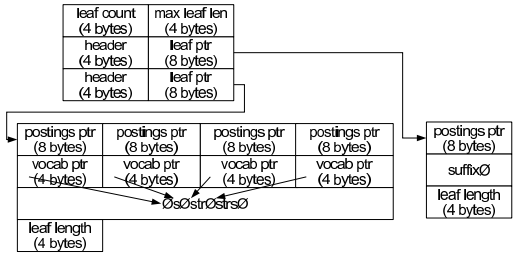


Figure 3: The header and leaf structures for *embedfixed*.

The *embedfixed* algorithm uses a simple but effective coding method for compressing terms and allows fast lookup without de-serialisation. Instead of allocating leaves as multiples of disk sectors and squeezing as many terms as possible into each leaf, a leaf in *embedfixed* only contains terms with the same common prefix. The leaf size thus depends on the number of common prefix characters and how many terms share that common prefix. Additionally and uniquely, only the suffixes of the terms are stored in the leaf and are null-terminated for fast string comparison. Essentially, our algorithm is a form of Trie [6, Section 6.3, p. 412].

The construction of the header is also different from the previous algorithms. Instead of storing the whole term (of varying lengths) in the header, only the characters of the common prefix are stored, and all common prefixes are of the same length. There are several advantages to constructing the header structure this way; (1) For the same number of leaves, the structure has a smaller footprint compared with the previous header structure since only partial terms are stored. (2) During lookup, a shorter string comparison is performed to locate the leaf containing the term. (3) If the partial terms

are treated as integers, integer comparison can be used for fast lookup instead of standard string comparison.

Essentially, *embedfixed* splits terms into two parts, the common prefix and the rest of the term (the suffix). For example, the term "termstr" with common prefix "term" will be split as "term" and "strø". The common prefix "term" will be stored in the header and the suffix "strø" will be stored in the leaf.

Figure 3 shows the complete structure of *embedfixed*. In the header, 4 bytes are allocated for the common prefixes and 8 bytes for the leaf pointers. The *leaf count* and *max leaf len* entries at the beginning of the header structure store the number of leaves and the size of the largest leaf respectively. The *max leaf len* entry is used to allocate one fix-sized buffer for reading leaves throughout the execution of the search engine. In the leaf, the 8-byte postings pointers are stored first, followed by the 4-byte vocab pointers, then the null-terminated suffixes and finally the number of entries in the leaf. The vocab pointers are used to avoid de-serialisation of the suffix string. With the illustrated terms of "term,terms,termstr,termstrs,them", the header will have common prefix "term" and the terms are compressed as "øsøstrøstrsø" in the leaf. The last term "them" does not share a common prefix with "term" and has to be stored in a different leaf (not illustrated).

There is a case that a leaf may contain only one term. This happens when the current term does not share a common prefix with the next term. For example, the terms "termstr,worm" have no common prefix. The first four characters of the term "termstr" will be stored in the header since the header has a length of 4 byte for the common prefix. The suffix "strø" will be stored in the leaf by itself. In this case, the vocab pointer is no longer needed. The *leaf length* entry is still required to indicate that it is a leaf containing a single term.

4 Experiments

We conducted all our experiments on a system with dual quad-core Intel Xeon E5410 2.3 GHz, DDR2 PC5300 8 GB main memory, Seagate 7200 RPM 500 GB hard drive and running Linux with kernel version 2.6.30. The hard drive has a model number of ST3500320AS and 512 byte sectors [11].

The collections used were the INEX 2009 Wikipedia collection [10] and the uni-gram corpus from the Web 1T 5-gram version 1 [3]. The Wikipedia collection has about 2.6 million documents, 2,348,343,176 total words and 11,393,924 unique words. The Web 1T corpus was generated from about 1 trillion terms of text from public web pages. It contains English word n-grams (uni-grams to 5-gram) and their frequency counts. Throughout the experiments, only the uni-grams was used. There are 13,588,391 words in the uni-gram data set.

The Million Query Track queries from TREC 2007 [1] were used for throughput evaluation. The track has a total of ten thousand queries with a total of 41,671 terms. The average query length is about 4 terms.

Instead of using a real search engine[1], a simulation program was written and used for the evaluation. The advantage of using a simulation is that it is easy to implement and evaluate the algorithms. At the same time, the simulation program can provide the correct performance since it mimics how a real search engine process queries for vocabulary lookup. Through out the experiments, only the vocabulary of the inverted files was built as the postings are not touched in the experiments.

The program has two executables, *build-dictionaries* and *search-dictionaries*. The *build-dictionaries* executable uses the parser from the search engine described in [13, 5] and takes a number of parameters including (1) which vocabulary structure to build, (2) the number of disk sectors for the leaf size and (3) the value of k for the *blocked-k* structure.

In order to minimise the interference from the Linux kernel during the evaluation, one of the CPU cores is configured off the management of the kernel and the *search-dictionaries* process was assigned to that particular core using the CPU affinity system calls [7, p. 172]. *search-dictionaries* also takes a number of parameters, including (1) which vocabulary to search, (2) whether the nodes are stored on disk or the whole vocabulary is loaded into memory, (3) the number of sectors for the leaf node size, (4) the value of k for the *blocked-k* structure and (5) which query file to use for batch mode processing.

5 Results

Four sets of experiments were conducted to test the 2-level vocabulary algorithms of *fixed*, *string*, *blocked-k* with a value of 2, 4 and 8, *embed*, *embedfront* and *embedfixed*. Unless specified, the number of characters stored in the header are 4 bytes long for *embedfixed*. All search results were average over twenty runs. The *gettimeofday()* function was used for the timing.

5.1 Experiment One

The first set of the experiments examined the storage space of the proposed 2-level structures with various leaf sizes. The leaf sizes were 1, 2, 4, 8, 16 and 32 sectors and the corresponding length in bytes are 512, 1024, 2048, 4096, 8192 and 16384 (the disk has a sector size of 512 bytes).

Figure 4(a) shows the results. In both collections, *fixed* and *embedfixed* showed static storage space as the size of the leaf did not depend on the number of sectors. *fixed* required about 349 MB to store the Wikipedia collection and about 337 MB for the unigram data set.

[1] But the ATIRE search engine currently under development at University of Otago now uses essentially the *embedfixed* algorithm and so it has been tested in a search engine.

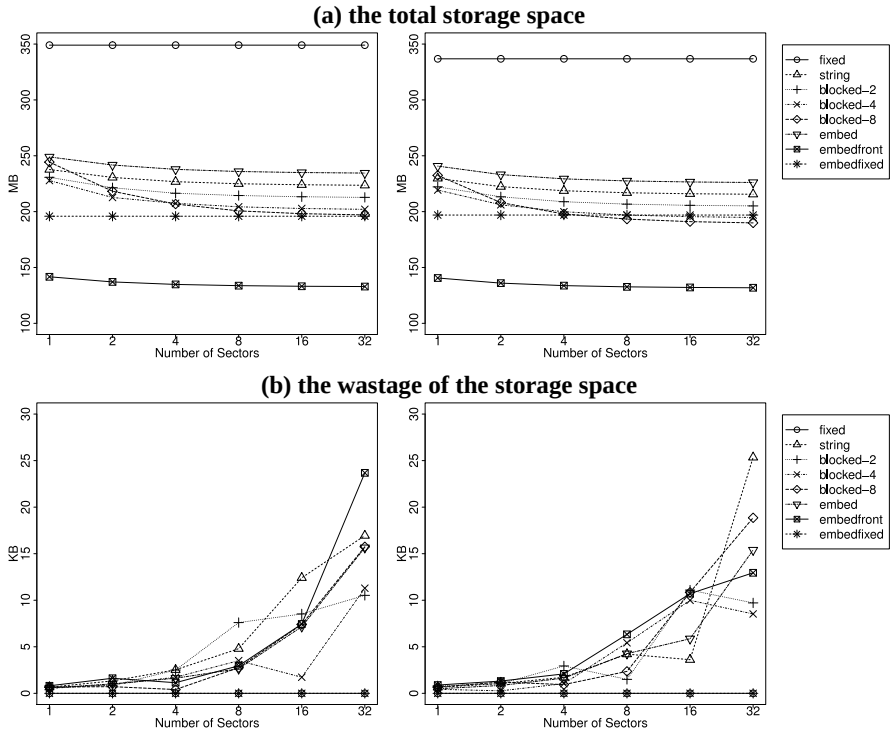


Figure 4: The total storage space and the wastage. The left figures shows the results using the Wikipedia collection [10] and the right figures shows the results using the uni-gram data set in the Web 1T corpus [3].

embedfixed used about 196 MB and 197 MB respectively in the Wikipedia collection and the uni-gram data set.

The storage space steadily decreased as the leaf size increased for *string*, *blocked-2*, *blocked-4*, *blocked-8*, *embed* and *embedfront* in both collections. When the leaf size increased from 1 to 8 sectors, *blocked-8* had the biggest drop in the storage required from 244 MB to 201 MB in the Wikipedia collection and from 232 MB to 193 MB in the uni-gram dataset.

Overall, *embedfront* was the most storage efficient and used about 62% and 60% less storage than *fixed* in the corresponding collections. In most cases, *embedfixed* performed better than the other algorithms except that it used about 60 MB and 50MB more storage than *embedfront* in the corresponding collections.

There are two reasons why *embedfront* used less storage space than *embedfixed*. First, the 4-byte vocab pointer is not used in *embedfront*. For a leaf stored with 100 terms, 400 bytes are saved (100 pointers of 4 bytes each). Second, *embedfront* computes the common prefix dynamically between each adjacent terms, while the common prefix in *embedfixed* is preset. For example, for “term,terms,termstr,termstrs”, *embedfront* computes the common prefixes of “term”, “terms” and ”termstr”, while shorter common prefix of “term” is used for *embedfixed*. For terms sharing long common prefixes, *embedfront* provides better compression.

Figure 4(b) shows the wasted storage space due to internal fragmentation in leaves. Apart from *fixed* and *embedfixed*, as the size of the leaf increased more storage space was wasted. However, the wastage is small compared with the total storage space.

There is an apparent contradiction when comparing Figure 4(a) and Figure 4(b). As the the number of sectors increased, the wastage increased, however the total storage space decreased for *string*, *blocked-k*, *embed* and *embedfront*. A common factor for the reduction of the storage space is a reduction in the number of leaves and hence the size of the header is smaller. Further more, *block-k* also benefitted from combining two or more odd sized leaves into even sized leaves and *embedfront* from better compression of terms stored in larger leaves.

5.2 Experiment Two

The second set of the experiments examined the search performance when the header of the vocabulary structures was loaded into memory and only the required leaves were retrieved from disk. The simulation program did not cache I/O, but depended on the Linux kernel to do so. Before each run of the experiments, the disk cache was flushed.

Figure 5(a) shows the I/O time, taken to read the required leaves from disk. The I/O time for *embefixed* is approximately constant regardless of the number of sectors to store each leaf. For other algorithms, the

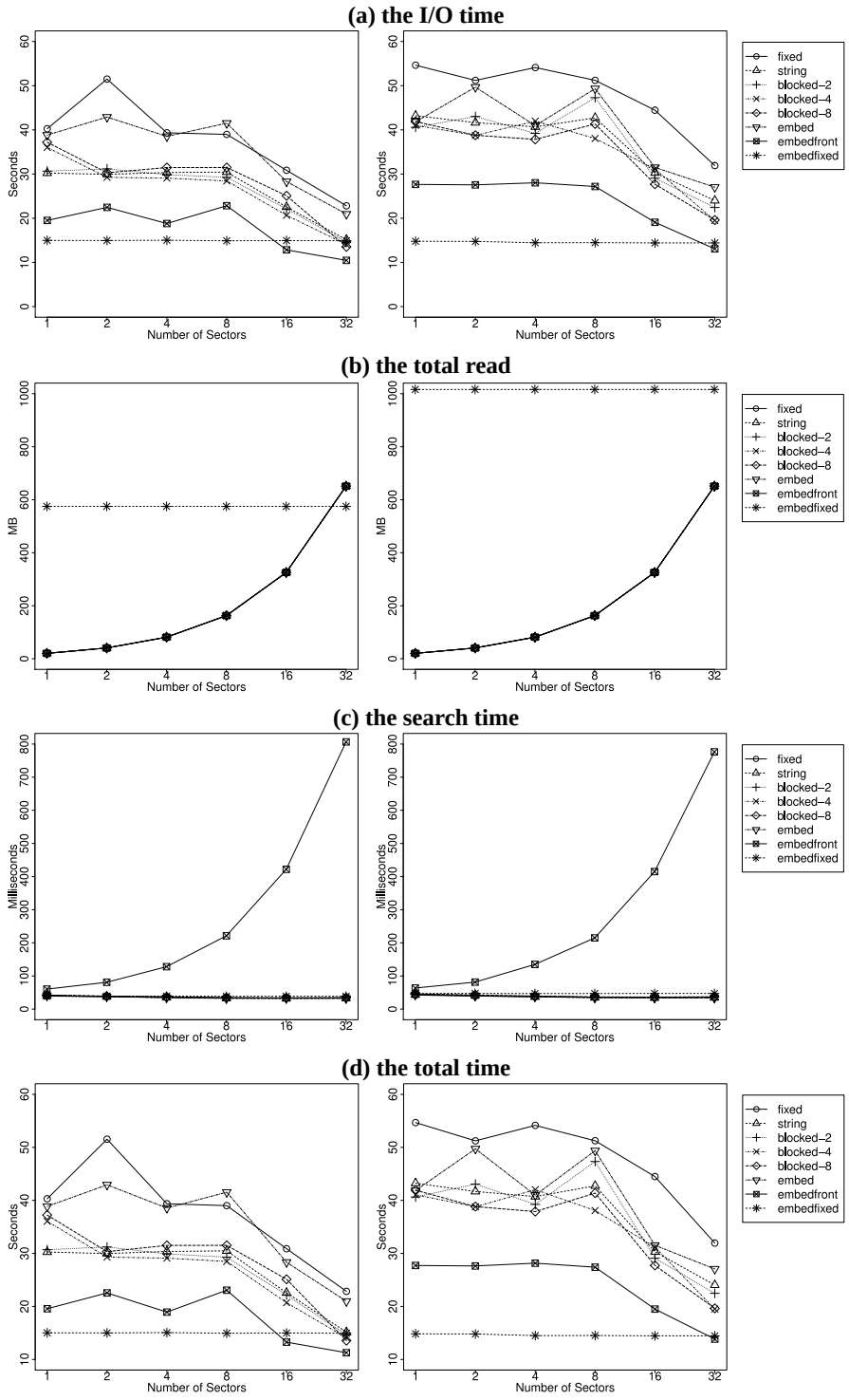


Figure 5: The performance of the algorithms when leaves were stored on disk. The left figures shows the results using the Wikipedia collection [10] and the right figures shows the results using the uni-gram data set in the Web 1T corpus [3].

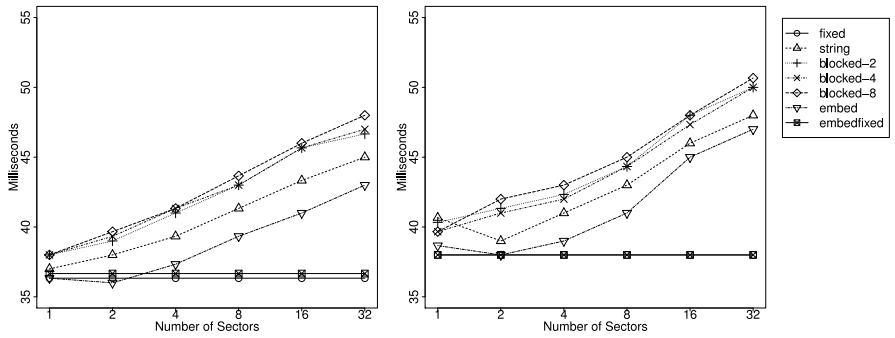


Figure 6: In-memory total time without *embedfront*. The left figures shows the results for the Wikipedia collection [10] and the right figures shows the results for the uni-gram data set in the Web 1T corpus [3].

I/O time decreased with the increasing number of sectors. For both collections, *embedfixed* performed the best, *embedfront* took more time than *embedfixed* but performed better than other algorithms, and *fixed* performed the worst.

The performance difference between *fixed*, *string*, *blocked-k*, *embed* and *embedfront* showed the impact of the total file size on disk I/O. The smaller the file size, the less time taken to read from disk.

The *embedfixed* algorithm performed the best, even though it had a larger file size than *embedfront*. This is due to the fact that *embedfixed* accessed a smaller number of leaves than *embedfront* and the leaves were larger. A large leaf can contain more terms than a small leaf and be better cached by the operating system. For example in the Wikipedia collection, *embedfront* had 280,399 leaves when the leaf was 2 sectors. On the other hand, *embedfixed* had 314,871 leaves but only 169,187 of them contained more than one term. 169,187 large blocks is smaller than 280,399 blocks, and they were read and cached earlier in the experiments. Figure 5(b) shows the total number of bytes read from disk. When the leaf had a size of 2 sectors, *embedfixed* read 560 MB and 950 MB more than *embedfront* respectively in the collections.

Figure 5(c) shows the search time. As the leaf size increased from 1 to 32 sectors, the search time taken by *embedfront* grew exponentially on the log scale of the number of sectors (linearly in linear scale). The performance gap between *embedfront* and the other algorithms was caused by the de-serialisation in order to access the individual terms stored in leaves during the lookup.

As shown in Figure 5(d), the total time was dominated by the I/O time. Figure 5(d) is almost identical to Figure 5(a) because the CPU time was order of magnitude lower than the I/O time. Overall, *embedfixed* showed the best performance.

5.3 Experiment Three

The third set of the experiments examined the search performance when the whole Vocabulary was loaded into memory. The results show the same pattern as shown in Figure 5(c). *embedfront* grew linearly as the leaf size increased. Figure 6 shows a detailed plot of the results from Figure 5(c), but without *embedfront*. In both collections, The time to search for *string*, *blocked-2*, *blocked-4*, *blocked-8* and *embed* increased as the leaf size increased, while *fixed* and *embedfixed* showed static performance.

5.4 Experiment Four

The last set of the experiments explored various lengths of the common prefix in the header for *embedfixed*. Figure 7(a) shows the required storage space as the size of the common prefix increased from 1 to 10 bytes. In both collections, the required storage space decreased from 1 to 4, and increased from 7 to 10. The *embedfixed* algorithm used the least storage space when when the common prefix had a length of 4.

Figure 7(b) shows the search performance when the whole vocabulary was loaded into main memory. In the Wikipedia collection, *embedfixed* showed the best performance when the common prefix had a length of 5 or 6. While in the uni-gram data set, *embedfixed* had the best performance when the common prefix had a length of 8. This is due to how terms are distributed among the header and leaf structures. For a header with long common prefix, more terms are squeezed into the header and more characters have to be compared in order to locate the leaf. For a header with shorter common prefixes, fewer terms are squeezed into the header and less characters are compared, however it takes longer to locate the term in the leaf.

6 Conclusion and Future Work

In this paper, we have investigated and addressed the related issues associated with the 2-level B-tree structure for storing the vocabulary of a search engine. We have conducted experiments on a number of algorithms, including 2-level *fixed*, *string*, *blocked-k*, *embed*, *embedfront* and *embedfixed*. Our new *embedfixed* algorithm provides a simple but effective encoding method

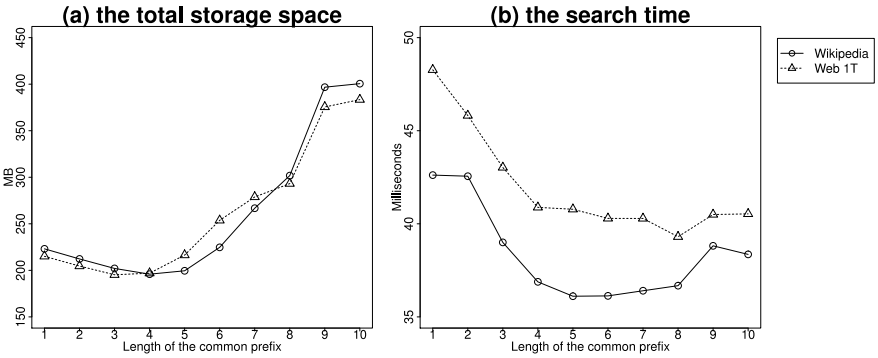


Figure 7: The left figure shows the storage space with different lengths of the common prefix. The right figure shows the total search time with different lengths of the common prefix.

by storing terms with the same common prefix in the same leaf, and the shared common prefix in the header structure. The new encoding method allows fast lookup without de-serialisation, with an average search time of $O(log_2(lc))$, where ls is the number of leaves in the header, plus $O(log_2(ll))$, where ll is the number of entries in the leaf.

In terms of compression ratio, *embedfront* showed the best performance in both the Wikipedia collection and the uni-gram data set. The *embedfixed* algorithm used about 60 MB and 50 MB more storage space than *embedfront* in the corresponding collections. While in terms of vocabulary lookup performance, *embedfixed* showed the best performance in both sets of the experiments when only the header structure was loaded into memory and when the whole vocabulary was loaded into memory.

However, the performance of vocabulary lookup is dependent on how disk data is cached in memory. An extreme case is that systems do not cache disk data at all. For such systems, the algorithmes with smaller leaf sizes will perform the best since less data is read from disk. In this case, the *embedfront* algorithm with a leaf size of 1 sector should be used. The opposite extreme is that the whole vocabulary can be loaded into memory at startup time, thus disk I/O is eliminated during lookup. In this case, the *embedfixed* algorithm should be used. In future experiments, we will explore the performance of the algorithms on systems with limited resources, for example smart phones.

Acknowledgements

Thanks Dylan Jenkinson for his early contribution to this work.

References

[1] James Allan, Ben Carterette, Javed Aslam, Virgil Pavlu, Blagovest Dachev and Evangelos Kanoulas. Million query track 2007 overview. In *TREC*, 2008.

[2] Vo Ngoc Anh, Owen de Kretser and Alistair Moffat. Vector-space ranking with effective early termination. pages 35–42, 2001.

[3] Thorsten Brants and Alex Franz. Web 1t 5-gram version 1. Linguistic Data Consortium, 2006.

[4] Xiang-fei Jia, Andrew Trotman, Richard O'Keefe and Zhiyi Huang. Application-specific disk I/O optimisation for a search engine. In *PDCAT '08*, pages 399–404, Washington, DC, USA, 2008. IEEE Computer Society.

[5] Xiangfei Jia, David Alexander, Vaughn Wood and Andrew Trotman. University of Otago at INEX 2010. In *INEX* [5], pages 250–268.

[6] Donald E. Knuth. *The Art of Computer Programming*. Addison-Wesley, 1997.

[7] Robert Love. *Linux System Programming*. O'Reilly Media, 2007.

[8] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[9] Alistair Moffat, Justin Zobel and Ron Sacks-Davis. Memory efficient ranking. *Inf. Process. Manage.*, Volume 30, Number 6, pages 733–744, 1994.

[10] Ralf Schenkel, Fabian Suchanek and Gjergji Kasneci. YAWN: A semantically annotated Wikipedia XML corpus. March 2007.

[11] Seagate. Barracuda 7200.11 serial ATA, January 2009.

[12] Andrew Trotman. Compressing inverted files. *Inf. Retr.*, Volume 6, Number 1, pages 5–19, 2003.

[13] Andrew Trotman, Xiang-Fei Jia and Shlomo Geva. Fast and effective focused retrieval. In *Focused Retrieval and Evaluation*, pages 229–241. Springer Berlin, 2010.

[14] Ian H. Witten, Timothy C. Bell and Alistair Moffat. *Managing Gigabytes: Compressing and Indexing Documents and Images*. John Wiley & Sons, Inc., New York, NY, USA, 1994.

[15] Justin Zobel and Alistair Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, Volume 38, Number 2, pages 6, 2006.

[16] Justin Zobel, Alistair Moffat and Kotagiri Ramamohanarao. Inverted files versus signature files for text indexing. *ACM Trans. Database Syst.*, Volume 23, Number 4, pages 453–490, 1998.