ADCS 2004

Proceedings of the Ninth
Australasian Document Computing Symposium,

December 13, 2004

Edited by
Peter Bruza, Alistair Moffat, and Andrew Turpin

Proceedings of the Ninth Australasian Document Computing Symposium,
held at the University of Melbourne,
December 13, 2004.

Published by the
Department of Computer Science and Software Engineering,
The University of Melbourne,
Victoria 3010, Australia.

Editors:
Peter Bruza,
Alistair Moffat,
Andrew Turpin.

ISBN 0-9757172-0-0

http://www.cs.mu.oz.au/~alistair/adcs2004/

ADCS 2004

The Ninth Australasian Document Computing Symposium

Melbourne, Australia
13 December 2004

Chairs' Preface

These proceedings contain the papers of the Ninth Australasian Document Computing Symposium hosted by, and held within, the Department of Computer Science and Software Engineering at The University of Melbourne.

The keynote address, seven long papers, and five short papers cover a wide range of topics and activities, and reflect the breadth of interest of the Australasian research community in the area of document computing. One quarter of the papers submitted came from outside Australia.

The twelve research papers included here were selected from 21 submissions. Every paper had at least two peer reviews, and many of them three. Submitted papers were anonymously reviewed at their full length by experts in the area. Dual submissions were explicitly prohibited.

The members of the program committee and the extra reviewers deserve special thanks for their contribution to ADCS 2004. Reviewers not listed among the program committee include John Ducrou, Dawei Song, Xindong Wu, Falk Scholer, and Steven Garcia. We would also like to thank the University of Melbourne for their involvement in 2004 as sponsor and host of this event.

This symposium includes many formal presentations but perhaps its greatest benefit lies in the opportunity it provides for document computing practitioners to get together informally and to share ideas and enthusiasm. We look forward to meeting you in Melbourne.

Peter Bruza
Alistair Moffat
Andrew Turpin

15 November 2004

Symposium Chair

Alistair Moffat	University of Melbourne

Program Co-Chairs

Peter Bruza	DSTC
Andrew Turpin	University of Melbourne

Program Committee

Vo Ngoc Anh	University of Melbourne
Theresa Dirndorfer Anderson	UTS
Bob Colomb	University of Queensland
Nick Craswell	Microsoft Research
Peter Eklund	University of Wollongong
Tamas D. Gedeon	Australian National University
David Hawking	CSIRO
Judy Kay	University of Sydney
Raymond Lau	QUT and City University of Hong Kong
Rob McArthur	DSTC
Gitesh K. Raikundalia	Victoria University
James Thom	RMIT University
Ross Wilkinson	CSIRO
Hugh Williams	RMIT University

ADCS Advisory Committee

Peter Bruza	DSTC
David Hawking	CSIRO
Judy Kay	University of Sydney
Alistair Moffat	University of Melbourne
Ross Wilkinson	CSIRO
Justin Zobel	RMIT University

Contents

Invited Presentation

Long Papers (Fully Refereed)

Short Papers (Fully Refereed)

Personal Information Management: Stuff I've Seen and Beyond

Dr Susan Dumais
Adaptive Systems and Interaction Group,
Microsoft Research
http://research.microsoft.com/~sdumais

Abstract

Most information retrieval technologies are designed to facilitate information discovery. However, much knowledge work involves finding and re-using previously seen information in the context of ongoing work activities. We have developed a prototype system called Stuff I've Seen (SIS) to support information re-use. The system provides a unified index to information that a person has seen, regardless of whether the information was seen as an email, appointment, web page, document, or hand-written note. Because the information has been seen before, rich contextual cues and visualizations (including timelines and memory landmarks) can be used to present search results. The system also provides a test bed for exploring ideas about meta-data driven retrieval and about techniques for supporting information management in the richer context of ongoing work activities.

The speaker

Dr Susan Dumais is a Senior Researcher in the Adaptive Systems and Interaction Group at Microsoft Research where she works on algorithms and interfaces for improved information access and management. Prior to joining Microsoft Research in 1997, she was at Bellcore and Bell Labs for many years. She has published widely in the areas of human-computer interaction and information retrieval. Her current research focuses on personal information retrieval, user modeling, text categorization, and collaborative information retrieval. Previous research included well-known work on Latent Semantic Indexing (a statistical method for concept-based retrieval), combining search and navigation, individual differences, perceptual learning and attention, and organizational impacts of new technology.

Susan is Past-Chair of ACM's SIGIR group, and serves on the NRC Committee on Computing and Communications Research to Enable Better Use of Information Technology in Digital Government, and the NRC Board on Assessment of NIST Programs. She is on the editorial boards of: ACM:Transactions on Information Systems, ACM:Transactions on Human Computer Interaction, Human Computer Interaction, Information Processing and Management, Information Retrieval, Hypertext, Encyclopedia of Information Retrieval, and Annual Review of Information Science and Technology, and is actively involved on program committees for several conferences. She is an adjunct professor at the University of Washington in the Information School, and has been a visiting faculty member at Stevens Institute of Technology, New York University, and the University of Chicago.

Focused crawling in depression portal search: A feasibility study

Thanh Tin Tang
Department of Computer Science, ANU
ACT 0200, Australia
thanh.tang@cs.anu.edu.au

David Hawking
CSIRO ICT Centre
ACT 2601, Australia
david.hawking@csiro.au

Nick Craswell
Microsoft Research
CB3 0FB, UK
nickcr@microsoft.com

Ramesh S. Sankaranarayana
Department of Computer Science, ANU
ACT 0200, Australia
ramesh@cs.anu.edu.au

Abstract *Previous work on domain specific search services in the area of depressive illness has documented the significant human cost required to setup and maintain closed-crawl parameters. It also showed that domain coverage is much less than that of whole-of-web search engines. Here we report on the feasibility of techniques for achieving greater coverage at lower cost. We found that acceptably effective crawl parameters could be automatically derived from a DMOZ depression category list, with dramatic saving in effort. We also found evidence that focused crawling could be effective in this domain: relevant documents from diverse sources are extensively interlinked; many outgoing links from a constrained crawl based on DMOZ lead to additional relevant content; and we were able to achieve reasonable precision (88%) and recall (68%) using a J48-derived predictive classifier operating only on URL words, anchor text and text content adjacent to referring links. Future directions include implementing and evaluating a focused crawler. Furthermore, the quality of information in returned pages (measured in accordance with the evidence based medicine) is vital when searchers are consumers. Accordingly, automatic estimation of web site quality and its possible incorporation in a focused crawler is the subject of a separate concurrent study.*

Keywords focused crawler, hypertext classification, mental health, depression, domain-specific search.

1 Introduction

Depression is a major public health problem, being a leading cause of disease burden [13] and the leading risk factor for suicide. Recent research has demonstrated that high quality web-based depression information can improve public knowledge about depression and is associated with a reduction in depressive symptoms [6]. Thus, the Web is a potentially valuable resource for people with depression. However, a great deal of depression information on the Web is of poor quality when judged against the best available scientific evidence [8, 10]. It is thus important that consumers can locate depression information which is both relevant and of high quality.

Recently, in [15], we compared examples of two types of search tool which can be used for locating depression information: whole-of-Web search engines such as Google, and domain-specific (portal) search services which include only selected sites. We found that coverage of depression information was much greater in Google than in portals devoted to depression or health.

BluePages Search (BPS)[1] is a depression-specific search service offered as part of the BluePages depression information site. Its index was built by manually identifying and crawling areas on 207 Web servers containing depression information. It took about two weeks of intensive human effort to identify these areas (seed URLs) and define their extent by means of include and exclude patterns. Similar effort would be required at regular intervals to maintain coverage and accuracy. Despite this human effort, only about 17% of relevant pages returned by Google were contained in the BPS crawl.

One might conclude from this that the best way to provide depression-portal search would be to add the word 'depression' to all queries and forward them to a general search engine such as Google. However, in other experiments in [15] relating to quality of information in search results, we showed that substantial amounts of the additional relevant information returned by Google was of low quality and not in accord with best available scientific evidence. The operators of the BluePages portal (ANU's Centre for Mental Health Research) were keen to know if it would be feasible to provide a portal search service featuring:

1. increased coverage of high-quality depression information,

[1] `bluepages.anu.edu.au`

Proceedings of the 9th Australasian Document Computing Symposium, Melbourne, Australia, December 13, 2004.

2. reduced coverage of dubious, misleading or unhelpful information, and

3. significantly reduced human cost to maintain the service.

We have attempted to answer the questions in two parts. Here we attempt to determine whether it is feasible to reduce human effort by using a directory of depression sites maintained by others as a seedlist and using focused crawling techniques to avoid the need to define include and exclude rules. We also investigate whether the content of a constrained crawl links to significant amounts of additional depression content and whether it is possible to tell which links lead to depression content.

A separate project is under way to determine whether it is feasible to evaluate the quality of depression sites using automatic means. It will be reported elsewhere. If the outcomes of both projects are favourable, the end-result may be a focused crawler capable of preferentially crawling relevant content *from high quality sites.*

2 Focused crawling - related work

Focused crawlers, first described by de Bra et al. [2], for crawling a topic-focused set of Web pages, have been frequently studied [3, 1, 5, 9, 12].

A focused crawler seeks, acquires, indexes, and maintains pages on a specific set of topics that represent a relatively small portion of the Web. Focused crawlers require much smaller investment in hardware and network resources but may achieve high coverage at a rapid rate.

A focused crawler starts with a seed list which contains URLs that are relevant to the topic of interest, it crawls these URLs and then follows the links from these pages to identify the most promising links based on both the content of the source pages and the link structure of the web [3]. Several studies have used simple string matching of these features to decide if the next link is worth following [1, 5, 9]. Others used reinforcement learning to build domain-specific search engines from similar features. For example, McCallum et al. [11] used Naive Bayes classifiers to classify hyperlinks based on both the full text of the sources and anchor text on the links pointing to the targets.

A focused crawler should be able to decide if a page is worth visiting before actually visiting it. This raises the general problem of hypertext classification.

In traditional text classification, the classifier looks only at the text in each document when deciding what class should be assigned.

Hypertext classification is different because it tries to classify documents without the need for the content of the document itself. Instead, it uses link information. Chakrabati et al. [3] used the hypertext graph including in-neighbours (documents citing the target document) and out-neighbours (documents that target document cites) as input to some classifiers.

Our work also used link information. We tried to predict the relevance of uncrawled URLs using three features: anchor text, text around the link and URL words.

3 Resources

This section describes the resources used in our experiments: the BluePages search service; the data from our previous domain-specific search experiments; the DMOZ depression directory listing and the WEKA machine learning toolkit.

3.1 BluePages Search

BluePages Search (BPS) is a search service offered as part of the existing BluePages depression information site. Crawling, indexing and search were performed by CSIRO's Panoptic search engine[2].

The list of web sites that made up the BPS was manually identified from the Yahoo! Directory and from querying general search engines using the query term 'depression'. Each URL from this list was then examined to find out if it was relevant to depression before it was selected. The fencing of web site boundaries was a much bigger issue. A lot of human effort was needed to examine all the links in each web site to decide which links should be included and excluded. Areas of 207 web sites were selected. These areas sometimes included a whole web server, sometimes a subtree of a web server and sometimes only some individual pages. Newspaper articles (which tend to be archived after a short time), potentially distressing, offensive or destructive materials and dead links were excluded during the construction of the BPS index.

A simple example of seeds and boundaries is:

- $seed$ = `www.counselingdepression.com/`, and
- $include\_patterns$ = `www.counselingdepression.com`.

In this case, every link within this web site is included. In complicated cases, however, some areas should be included while others are excluded. For instance, examining `www.drada.org` would result in the following seed and boundaries:

- $seed$ = `www.drada.org/`
- $include\_patterns$ = `www.drada.org`
- $exclude\_patterns$ = `www.drada.org/facts/bipolar.html`, `www.drada.org/facts/bipolar_nih.html`, `www.drada.org/Store/bookreviews_`.

The above boundaries mean that everything within the web site should be crawled except for pages about bipolar depression and book reviews.

[2] `http://www.panopticsearch.com/`

3.2 Data from our previous work

In our previous work, we conducted a standard information retrieval experiment, running 101 'depression' queries against six engines of different types: two health portals, two depression-specific search engines, one general search engine and one general search engine where the word 'depression' was added to each query if not already present (GoogleD). We then pooled the results for each query and employed research assistants to judge them. We obtained 2778 judged URLs and 1575 relevant URLs from all the engines. We used these URLs as a base in the present work to estimate relevance.

We found that, over 101 queries, GoogleD returned more relevant results than those of the domain-specific engines. 621 relevant URLs were returned by BPS while 683 relevant results were retrieved by GoogleD. As GoogleD was the best performer in obtaining the most relevant results, we also used it as a base engine to compare with other collections in the present work.

3.3 DMOZ

DMOZ[3] is the Open Directory Project which is "the largest, most comprehensive human-edited directory of the Web. It is constructed and maintained by a vast, global community of volunteer editors"[4].

We started with the Depression directory[5] which contains documents and directories purportedly relevant to depressive disorder.

3.4 Weka

Weka[6] was developed at the University of Waikato in New Zealand [16]. It is a data mining package which contains machine learning algorithms. Weka provides tools for data pre-processing, classification, regression, clustering, association rules, and visualization. Weka was used in our experiments for the prediction of URL relevance using hypertext features. It was used because it provided many classifiers, was easy to use and served our purposes well in predicting URL relevance.

4 Experiment 1 - Usefulness of a DMOZ category as a seed list

A focused crawler needs a good seed list of relevant URLs as a starting point for the crawl. These URLs should span a variety of web site types so that the crawler can explore the Web in many different directions. Instead of using a manually created list, we attempted to derive a seed list from a publicly available directory - DMOZ. Because depression sites on the web are widely scattered, the diversity of content in DMOZ is expected to improve coverage. Using DMOZ allows us to leverage off the categorisation work being done by volunteer editors.

4.1 DMOZ seed generation

We started from the 'depression' directory on the DMOZ web site, namely `http://www.dmoz.org/Health/Mental_Health/Disorders/Mood/Depression/`. This directory is intended to contain links to relevant sites and subsites about depression. The directory, however, also had a small list of 12 within-site links to other directories, which may or may not be relevant to depression. We, therefore, only needed to do some minor boundary selection for these links to include relevant directories. For example, the following directories were included because they are related to depression and they are links from the depression directory: `dmoz.org/Health/Mental_Health/Disorders/Child_and_Adolescent/Childhood_Depression/`, and `dmoz.org/Health/Pharmacy/Drugs_and_Medications/Antidepressants/`. These links were selected simply because their URLs contain the term 'depression' (such as `childhood_depression`) or 'antidepressants'. The seed URLs, as a result, included the above links and all the links to depression-related sites and subsites from this directory.

Include patterns corresponding to the seed URLs were generated automatically. In general, the include pattern was the same as the URL, except that default page suffixes such as `index.htm` were removed. Thus, if the URL referenced the default page of a server or web directory, the whole server or whole directory was included. If the link was to an individual page, only that page was included.

The manual effort required to identify the seed URLs and define their extent varied greatly between BPS and DMOZ. While it took about two weeks of intensive effort in the BPS case, it only required about one hour's work for DMOZ.

4.2 Comparison of the DMOZ collection and the BPS collection

This experiment aimed to find out if a constrained crawl from the low-cost DMOZ seed list can lead to domain coverage comparable to that of the manually configured BPS.

After identifying the DMOZ seed list and include patterns as described above, we used the Panoptic crawler to build our DMOZ collection. We then ran the 101 queries from our previous study and obtained 779 results for DMOZ.

We attempted to judge the relevance of these results using the 1575 known relevant URLs (see Section 3.2) and to compare the DMOZ results with those of the BPS collection.

Table 1 shows that 186 out of 227 judged URLs (a pleasing 81%) from the DMOZ collection were relevant. However, the percentage of judged results (30%)

[3] `http://www.dmoz.org`
[4] `http://www.dmoz.org/about.html`
[5] `http://www.dmoz.org/Health/Mental_Health/Disorders/Mood/Depression/`
[6] `http://www.cs.waikato.ac.nz/~ml/weka/`

Table 1: Comparison of relevant URLs in DMOZ and BPS results of running 101 queries.

	URLs	judged URLs	relevant URLs
BPS	683	683	621
DMOZ	779	227	186

was too low to allow us to validly conclude that DMOZ was a good collection.

Since we no longer had access to the services of the judges from the original study we attempted to confirm that a reasonable proportion of the unjudged documents were relevant to the general topic of depression by sampling URLs and judging them ourselves.

We randomly selected 2 lists of 50 non-overlapped URLs among the unjudged results and made relevance judgments on these. In the first list, we obtained 35 relevant results and in the second list, 34 URLs were relevant. Because there was close agreement between the proportion relevant in each list we were confident that we could extrapolate the results to give a reasonable estimate of the total number of relevant pages returned.

Extrapolation suggests 381 relevant URLs for the unjudged DMOZ set. Hence, in total we might be able to obtain 567 (186 + 381) relevant URLs from the DMOZ set. This number was not as high as that of BPS, but it was relatively high (72% relevant URLs in DMOZ set compared to 91% of these in BPS). Therefore, we could conclude that the DMOZ list is an acceptably good, low-maintenance starting point for a focused crawl.

5 Experiments 2A-2C - Additional link-accessible relevant information

Although some focused crawlers can look a few links ahead to predict relevant links at some distance from the currently crawled URLs [7], the immediate outgoing links are of most immediate interest.

We performed three experiments to gauge how much additional relevant information is accessible one link away from the existing crawled content. If no additional relevant content is linked to from pages in the original crawl, the prospects of successful focused crawling are very low. Figure 1 shows an illustration of the one-link-away set of URLs from the DMOZ crawl.

The first experiment (2A) involved testing if outgoing links from the BPS collection were relevant while the second (2B) compared the outgoing link sets of BPS and DMOZ to see if DMOZ was really a good place to lead a focused crawler to additional relevant content. The last experiment (2C) attempted to find out if URLs relevant to a particular topic linked to each other.

5.1 Experiment 2A: Outgoing links from the BPS collection

The data used for this experiment included:

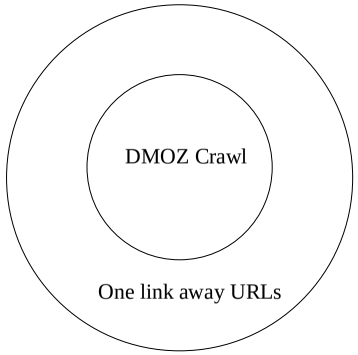

Figure 1: Illustration of one link away collection from the DMOZ crawl.

- the BPS index,
- the BPS outgoing link set containing all URLs linked to by BPS URLs, and
- 2 sets of judged-relevant URLs: BPS relevant and all relevant.

Our previous work concluded that BPS didn't retrieve as many relevant documents as GoogleD because of its small coverage of sites. We wanted to find out if focused crawling techniques have the potential to raise BPS performance by crawling one step away from BPS. Among 954 relevant pages retrieved by all engines except for BPS, BPS failed to index 775 pages. The extended crawl yielded 196 of these 775 pages or 25.3%.

In other words, an unrestricted crawler starting from the original BPS crawl would be able to reach an additional 25.3% of the known relevant pages, in only a single step from the existing pages. In fact, the true number of additional relevant pages is likely to be higher because of the large number of unjudged pages.

It is unclear whether the additional relevant content in the extended BPS crawl would enable more relevant documents to be retrieved than in the case of GoogleD. Retrieval performance depends upon the effectiveness of the ranking algorithm as well as on coverage.

5.2 Experiment 2B: Comparison of out-going links between BPS and DMOZ

This experiment compared the out-going link sets of BPS and DMOZ to find out if the DMOZ seed list could be used instead of the BPS seed list to guide a focused crawler to relevant areas of the web. The following data were used:

- 2 sets of out-going links from the BPS and DMOZ collections, and
- 2 sets of all judged URLs and judged-relevant URLs.

Table 2: Comparison of relevant out-going link URLs for BPS and DMOZ.

Collection	size	judged	relevant
outgoing BPS	49,370	248	196
outgoing DMOZ	122,985	203	158

From our previous work, we obtained 2778 judged URLs which were used here as a base to compare relevance. Table 2 shows that even though the outgoing link collection of DMOZ was more than double the size of that of BPS, more outgoing BPS pages were judged. Among the judged pages, BPS and DMOZ had 196 and 158 relevant pages respectively in their outgoing link sets. Although DMOZ had less known relevant pages than BPS, the proportion of relevant pages versus judged pages were quite similar for both engines(78% for DMOZ and 79% for BPS). This result together with the size of each outgoing link collection implied that (1) The DMOZ outgoing link set contained quite a large number of relevant URLs which could potentially be accessed by a focused crawler, and (2) The DMOZ seed list could lead to much better coverage than the BPS seed list.

5.3 Experiment 2C: Linking patterns between relevant pages

We performed a very similar experiment to the experiment described in Section 5.1, with the purpose of finding out if relevant URLs on the same topic are linked to each other. Instead of using the whole BPS collection of 12,177 documents as the seed list, we only chose the 621 known relevant URLs. The following data were used:

- the BPS known relevant URLs,
- the BPS outgoing link set from the above, containing all URLs linked to by BPS known relevant URLs, and
- judged-relevant URLs from our previous work.

The outgoing link collection of the BPS known relevant URLs contained 5623 URLs. Of these, 158 were known relevant. This was a very high number compared to the 196 known relevant URLs obtained from the much bigger set of all outgoing link URLs (containing above 40,000 URLs) in the previous experiment. It is likely from this experiment that relevant pages tend to link to each other. This is good evidence supporting the feasibity of the focused crawling approach.

6 Experiment 3 - Hypertext classification

After downloading the content of the seed URLs and extracting links from them, a focused crawler needs to decide what links to follow and in what order based on the information it has available. We used hypertext classification for this purpose.

6.1 Collection of URLs for training and testing

For both BPS and DMOZ crawls, we collected all immediate outgoing URLs satisfying the following two conditions (1) known relevant or known irrelevant URLs and (2) the URLs pointing to each of these URLs were also relevant. We collected 295 relevant and 251 irrelevant URLs for our classification experiment.

6.2 Features

Several papers in the field used the content of crawled URLs, anchor text, URL structure and other link graph information to predict the relevance of the next unvisited URLs [1, 5, 9]. Instead of looking at the content of the whole document pointing to the target URL, Chakrabarti [4] used 50 characters before and after a link and suggested that this method was more effective. Our work was somewhat related to all of the above. We used the following features to predict the relevance of the target URL.

- anchor text on the source pages: all the text appearing on the links to the target page from the source pages,
- text around the link: 50 characters before and 50 characters after the link to the target page from the source pages[7], and
- URL words: words appearing in the URL of the target page.

We accumulated all words for each of these features to form 3 vocabularies where all stop words were eliminated. URL words separated by a comma, a full stop, a special character and a slash were parsed and treated as individual words. URL extensions such as `.html`, `.asp`, `.htm`, `.php` were also eliminated. The end result showed 1,774 distinct words in the anchor text vocabulary, 874 distinct words in the URL vocabulary, and 1103 distinct words in the content vocabulary.

For purposes of illustration , Table 3 shows the features extracted from each of six links to the same URL.

Assume that we would like to predict `www.ndmda.org` for its relevance to depression and that we have six already-crawled pages pointing to it from our crawled collection. From each of the pages, features are extracted in the form of anchor text words and the words within a range of a maximum of 50 characters before and after the link pointing to `www.ndmda.org`. There is no content around the link from `www.noondaydemon.com/patresources.html` to the target URL because that URL contains only stop words and/or numbers which have been stripped off. The URL words for the target URL after being parsed contains: `ndmda,org`.

[7] We first extracted the 50-character string and then eliminated markup and stopwords, sometimes leaving only a few words.

Table 3: Features for www.ndmda.org after removing stop words and numbers.

Target URL: www.ndmda.org URL words: ndmda, org		
source URL	**anchor text**	**content around the link**
www.paxil.com/depression/dp_sym.html	ndmda, org	depression, bipolar, support, alliance,american, psychiatric
www.healthyplace.com/communities/ depression/living/my_experience5.asp	depression, bipolar, support, aliance	support, group, affliated, highly, recommend
www.healthyplace.com/Communities/ Depression/nimh/suicide_5.asp	national, depressive, manic, depressive, association	
www.noondaydemon.com/ pat_resources.html	national, depressive, manic, depressive, association	ncwa, ndmda
www.paxil.com/depression/dp_ln.html	ndmda, org	depression, bipolar, support, alliance,american, psychiatric
www.emufarm.org/ cmbell/ depress/deplink.html	national, depressive, manic, depressive, association	organisations

Table 4: Algorithms used from Weka.

Classifier	**Description**
IBK	k-nearest neighbors.
ZeroR	Zero rule. Predicts the majority class. Used as a baseline.
NaiveBayes	Statistical method. Assumes independence of attributes. Uses conditional probability and Bayes rule.
Complement Naive Bayes	Class for building and using a Complement class Naive Bayes classifier.
J48	C4.5 algorithm. A decision tree learner with pruning.
Bagging	Class for bagging a classifier to reduce variance.
AdaBoostM1	Class for boosting a nominal class classifier using the Adaboost M1 method.

6.3 Classifiers

We compared a range of classification algorithms provided by Weka [16]. (See Table 4.)

When training and testing the collection, we used a stratified cross-validation method, i.e. using 10-fold cross validation where one tenth of the collection was used for training and the rest was used for testing and the operation was repeated 10 times. The results were then averaged and a confusion matrix was drawn to find accuracy, precision and recall.

6.4 Input data

We treated the three vocabularies containing all features independently from each other. We computed term frequency and inverse document frequency ($tf.idf$) for each feature attached to each of the URLs specified in Section 6.1 using the following formula [14].

$$tf.idf = tf(t,d) * \log(n/df(t))$$

where t is a term, d is a document, $tf(t,d)$ is the frequency of t in d, n is total number of documents and $df(t)$ is the number of documents containing t.

By this means we obtained a list of URLs, each associated with the $tf.idf$s for all terms in the 3 vocabularies. A learning algorithm was then run in Weka to learn and predict if these URLs were relevant or irrelevant. We also used boosting and bagging algorithms to boost the performance of different classifiers.

6.5 Measures

We used three measures to analyse how a classifier performed in categorizing all the URLs. We denoted true positive and true negative for the relevant and irrelevant URLs that were correctly predicted by the classifier respectively. Similarly, false positive and false negative were used for irrelevant and relevant URLs that were incorrectly predicted respectively. The three measures are listed below.

- Accuracy: shows how accurately URLs are classified into correct categories.
  $accuracy = \frac{true\ positive + true\ negative}{all\ URLs}$
- Precision: shows the proportion of correctly relevant URLs out of all the URLs that were predicted as relevant.
  $precision = \frac{true\ positive}{true\ positive + false\ positive}$

- Recall: shows the proportion of relevant URLs that were correctly predicted out of all the relevant URLs in the collection.
  $recall = \frac{true\ positive}{true\ positive + false\ negative}$

Although accuracy is an important measure, a focused crawler would be more interested in following the links from the predicted relevant set to crawl other potentially relevant pages. Thus, precision and recall are better measures.

6.6 Results and discussion

The results of some representative classifiers are shown in Table 5. ZeroR represented a realistic performance "floor" as it classified all URLs into the largest category i.e relevant. As expected, it was the least accurate. Naive Bayes and J48 performed best. Naive Bayes was slightly better than J48 on recall but the latter was much better in obtaining higher accuracy and precision. Out of 228 URLs that J48 predicted as relevant, 201 were correct (88.15%). However, out of the 264 URLs predicted as relevant by Naive Bayes, only 206 (78.03%) were correct. Overall, the J48 algorithm was the best performer among all the classifers used.

We found that bagging did not improve the classification result while boosting showed some improvement for recall (from 64.74% to 68.13%) when the J48 algorithm was used.

We also performed other experiments where only one set of features or any combination of two sets of features were used. In all cases, we observed that the accuracy, precision and recall were all worse than when all three sets of features were combined.

Our best results, as detailed in Table 5, showed that a focused crawler starting from a set of relevant URLs, and using J48 in predicting future URLs, could obtain a precision of 88% and a recall of 68% using the features mentioned in Section 6.2.

We wished to compare these performance levels with the state of the art, but were unable to find in the literature any applicable results relating to the topic of depression. We therefore decided to compare our predictive classifier with a more conventional content classifier for the same topic.

We built a 'content classifier' for 'depression', using only the content of the target documents instead of the features being used in our experiment. The best accuracies obtained from the two classification systems were very similar, 78% for the content classifier and 77.8% for the predictive version. Content classification showed slightly worse precision but better recall.

We concluded from this comparison that hypertext classification is quite effective in predicting the relevance of uncrawled URLs. This is quite pleasing as a lot of unnecessary crawling can be avoided.

Finally we explored two variant methods for feature selection. We found that generating features using stemmed words caused a reduction in performance, as did reducing the feature set using a feature selection method.

7 Conclusions and future work

Weeks of human effort were required to set up the current BPS depression portal search service and considerable ongoing effort is needed to maintain its coverage and accuracy. Our investigations of the viability of a focused crawling alternative have resulted in three key findings.

First, web pages on the topic of depression are strongly interlinked despite the heterogeneity of the sources. This confirms previous findings in the literature for other topic domains and provides a good foundation for focused crawling in the depression domain. The one-link away extensions to the closed BPS and DMOZ crawls contained many relevant pages.

Second, although somewhat inferior to the expensively constructed BPS alternative, the DMOZ depression category features a diversity of sources and seems to provide a seed list of adequate quality for a focused crawl in the depression domain. This is very good news for the maintainability of the portal search because of the very considerable labour savings. Other DMOZ categories may provide good starting points for other domain-specific search services.

Third, predictive classification of outgoing links into relevant and irrelevant categories using source-page features such as anchor text, content around the link and URL words of the target pages, achieved very promising results. With the J48 decision-tree algorithm, as implemented by Weka, we obtained high accuracy, high precision and relatively high recall.

Given the promise of the approach, there is obvious follow-up work to be done on designing and building a domain-specific search portal using focused crawling techniques. In particular, it may be beneficial to rank the URLs classified as relevant in the order of degree of relevance so that a focused crawler can decide on visiting priorities. Also, appropriate data structures are needed to hold accumulated information for unvisited URLs (i.e. anchor text and nearby content for each referring link.) This information needs to be updated as additional links to the same target are encountered.

Another important question will be how to persuade Weka to output a classifier that can be easily plugged-in into the focused crawler's architecture. Since the best performing classifier in these trials was a decision tree, this may be easier than otherwise.

Once a focused crawler is constructed, it will be necessary to determine how to use it operationally. We envisage operating without any include or exclude rules but will need to decide on appropriate stopping conditions. If none of the outgoing links are classified as likely to lead to relevant content, should the crawl stop, or should some unpromising links be followed? And with what restrictions?

Table 5: Classification Results.

Classifier	Accuracy (%)	Precision (%)	Recall(%)
IBk	54.76	80	21.69
ZeroR	54.02	54.02	100
Complement Naive Bayes	71.06	77.51	65.42
Naive Bayes	73.07	78.03	69.83
J48	77.83	88.15	68.13

Because of the requirements of the depression portal operators site quality must be taken into account in building the portal search service. Ideally, the focused crawler should take site quality into account when deciding whether to follow an outgoing link, but this may or may not be feasible. Another more expensive alternative would be to crawl using relevance as the sole criterion and to filter the results based on quality.

Site quality estimation is the subject of a separate study, yet to be completed. In the meantime, it seems fairly clear from our experiments that it will be possible to increase coverage of the depression domain for dramatically lower cost by starting from a DMOZ category list and using a focused crawler.

Verifying whether techniques found useful in this project also extend to other domains is an obvious future step. Other health-related areas are the most likely candidates because of the focus on quality of information in those areas.

Acknowledgments

We gratefully acknowledge the contribution of Kathy Griffiths and Helen Christensen in providing expert input about the depression domain and about BluePages, and of John Lloyd and Eric McCreath for their advice on machine learning techniques.

References

[1] C. C. Aggarwal, F. Al-Garawi and P. S. Yu. On the design of a learning crawler for topical resource discovery. *ACM Trans. Inf. Syst.*, Volume 19, Number 3, pages 286–309, 2001.

[2] P. De Bra, G. Houben, Y. Kornatzky and R. Post. Information retrieval in distributed hypertexts. In *Proceedings of the 4th RIAO Conference*, pages 481–491, New York, 1994.

[3] S. Chakrabarti, M. Berg and B. Dom. Focused crawling: A new approach to topic-specific web resource discovery. In *Proceeding of the 8th International World Wide Web Conference (WWW8)*, 1999.

[4] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson and J. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proceedings of the seventh international conference on World Wide Web 7*, pages 65–74. Elsevier Science Publishers B. V., 1998.

[5] J. Cho, H. Garcia-Molina and L. Page. Efficient crawling through url ordering. In *Proceeding of the Seventh World Wide Web Conference*, 1998.

[6] H. Christensen, K. M. Griffiths and A. F. Jorm. Delivering Interventions for Depression by Using the Internet: Randomised Controlled Trial. *British Medical Journal*, Volume 328, Number 7434, pages 265–0, 2004.

[7] M. Diligenti, F. M. Coetzee, S. Lawrence, C. L. Giles and M. Gori. Focused crawling using context graphs. In *Proceeding of the 26th VLDB Conference*, Cairo, Egypt, 2000.

[8] Berland G, Elliott M, Morales L, Algazy J, Kravitz R, Broder M, Kanouse D, Munoz J, Puyol J, Lara M, Watkins K, Yang H and McGlynn E. Health Information on the Internet: Accessibility, Quality, and Readability in English and Spanish. *The Journal of the American Medical Association*, Volume 285, Number 20, pages 2612–2621, 2001.

[9] M. Hersovici, M. Jacovi, Y. S. Maarek, D. Pelleg, M. Shtalhaima and S. Ura. The shark-search algorithm. an application: tailored web site mapping. In *Proceeding of the Seventh World Wide Web Conference*, 1998.

[10] Griffiths K and Christensen H. The quality and accessibility of australian depression sites on the world wide web. *The Medical Journal of Australia*, Volume 176, pages S97–S104, 2002.

[11] A. McCallum, K. Nigam, J. Rennie and K. Seymore. Building domain-specific search engines with machine learning technique. In *Proceedings of AAAI Spring Symposium on Intelligents Engine in Cyberspace*, 1999.

[12] F. Menczer, G. Pant and P. Srinivasan. Evaluating topic-driven web crawlers. In *Proceeding of the 24th Annual Intl. ACM SIGIR Conf. On Research and Development in Information Retrieval*, 2001.

[13] C. J. L. Murray and A. D. Lopez (editors). *The Global Burden of Disease and Injury Series*. Harvard University Press, Cambridge MA, 1996.

[14] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. Technical report, 1987.

[15] T.T. Tang, N. Craswell, D. Hawking, K. M. Griffiths and H. Christensen. Quality and relevance of domain-specific search: A case study in mental health. *To appear in the Journal of Information Retrieval - Special Issues*, 2004.

[16] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, 1999.

On the Effectiveness of Relevance Profiling

David J Harper
Smart Web Technologies Centre
The Robert Gordon University
Aberdeen AB25 1HG UK
d.harper@rgu.ac.uk

David Lee
Smart Web Technologies Centre
The Robert Gordon University
Aberdeen AB25 1HG UK
david.lee@smartweb.rgu.ac.uk

Abstract

Relevance profiling is a general process for within-document retrieval. Given a query, a profile of retrieval status values is computed by sliding a fixed sized window across a document. In this paper, we report a series of bench experiments on relevance profiling, using an existing electronic book, and its associated book index. The book index is the source of queries and relevance judgements for the experiments. Three weighting functions based on a language modelling approach are investigated, and we demonstrate that the well-known query generation model outperforms one based on the Kullback-Leibler divergence, and one based on simple term frequency. The relevance profiling process proved highly effective in retrieving relevant pages within the electronic book, and exhibits stable performance over a range of sliding window sizes. The experimental study provides evidence for the effectiveness of relevance profiling for within-document retrieval, with the caveat that the experiment was conducted with a particular electronic book.

Keywords

relevance profiling; within-document retrieval; language modelling; information retrieval experimentation.

1 Background

Increasingly, long electronic documents are being published and delivered using web and other technologies, and users need to find information within these long documents. Various approaches have been proposed for within-document retrieval, including passage retrieval [1], and user interfaces supporting content-based browsing of documents [2]. We have developed a tool, ProfileSkim, for within-document retrieval based on the concept of relevance profiling [3], and we reported on a comprehensive user-centred evaluation of this tool [4] [5]. ProfileSkim integrates passage retrieval and content-based document browsing. The key concept underpinning the tool is relevance profiling, in which a profile of retrieval status values is computed across a document in response to a query. Within the user interface, an interactive bar graph provides an overview of this profile, and through interaction with the graph the user can select and browse in situ potentially relevant passages within the document. The reader is referred to [3] for a description of, and detailed rationale for, relevance profiling and the ProfileSkim user interface.

In the user evaluation study [4] [5], we compared the performance of ProfileSkim against a tool based on the sequential "Find" command delivered with most browsers and word processing packages. We concluded that:

- Relevance profiling**,** as implemented and presented by the ProfileSkim, was effective in enabling users to identify relevant passages of a document;
- Using ProfileSkim, users were able to select and browse relevant passages more efficiently, because only the best matching passages needed to be explored; and
- Users found the tool satisfying to use for within-document retrieval, because of the overview provided by the relevance profile.

That study was based on a simulated work task, namely recreating part of the (back of book) index for an electronic book. On the basis of the results, we tentatively concluded that relevance profiling in combination with a set of pre-defined index entries, might prove to be an effective replacement for the typical intellectually derived book index.

In this paper, we wish to investigate the underlying relevance profiling mechanism, through controlled bench experiments. Specifically, we want to investigate different forms of the weighting function used for relevance profiling, and different settings for parameters of the process. Additionally, we wanted to investigate whether relevance profiling could provide an alternative to the intellectually generated book index. Consequently, we based our experiments on an electronic book, with an existing book index, that formed the basis of the experimental corpus.

Proceedings of the 9th Australasian Document Computing Symposium, Melbourne, Australia, December 13, 2004.

2 Relevance Profiling and Weighting Functions

Relevance profiling is a process whereby a profile of retrieval status values is computed across a document in response to a query. This profile is presented to the user in the form of a bar graph, and by interacting with this bar graph the user can identify, and navigate to, relevant sections of a document. In this paper each section, and hence bar in the graph, corresponds to a page of the document (electronic book).

The process is sketched in on object-oriented pseudo-code in Figure 1. Effectively, a retrieval status value (RSV) is computed for each word position in the document. The relevance profiling procedure operates by sliding a window of fixed size across the document (lines 7-18, Figure 1), and at each word position a window weighting function is applied (line 12). The weighting function is a function of the set of words in the document, query and the window, starting at the given position and of the given size. The window weights are aggregated for each page of the document, and the maximum weight achieved by a window starting in the page is returned (lines 13-16). Note, that a filter is applied so that only pages containing at least one term from the query are retrieved, i.e. assigned a score (line 10).

```
 1 process relevanceProfiling
          ( Document, Query, weightingFunction,
          WindowSize, scoresArray )
 2 begin
 3   for Page = 1 to Document.countPages()
 4   begin
 5     scoresArray[ Page ]= minimumWindowScore
 6   end
 7   for windowPosition = 1 to Document.length()
 8     begin
 9       PageInDoc= Document.pageOf( windowPosition )
10       if (Document.containsTermOf (Query, PageInDoc)
11         begin
12           windowWeight = weightingFunction
                   ( Document, Query, windowPosition,
                   WindowSize )
13           if (windowWeight > scoresArray [ PageInDoc ])
14           begin
15             scoresArray [ PageInDoc ]= windowWeight
16           end
17       end
18   end
19 end
```

Figure 1: Pseudo-code describing the relevance profiling process

The main purpose of the experimental study is to investigate the behaviour of the relevance profiling process, and specifically the effect on retrieval effectiveness of the choice of window size and weighting function. Before describing the experimental study, we introduce the various weighting functions that are investigated.

The weighting functions are based on a language modelling approach in which we model a document, as a probability distribution over the terms of the vocabulary, and similarly for each (sliding) window. We define the document model to be the probability distribution *P(t|D)*, where for every term *t* in the vocabulary (in our case, of the document), the model gives the probability that we would observe term *t* if we randomly sampled from the document *D*. Similarly, we define the window model, *P(t|W)* for window *W*.

We will define three weighting functions (line 12, Figure 1), which we refer to as the query generation model (GEN), the Kullback-Leibler model (KL), and a simple term frequency model (FREQ).

2.1 Query Generation Model (GEN)

We adapt the language modelling approach proposed for document retrieval in [6] [7] for relevance profiling. Language modelling is used to construct a statistical model for a text window, and based on this model; we compute the window RSV as the probability of generating a query (denoted *Q*). We model the distribution of terms (actually stemmed words) over a text window, as a mixture of the text window and document term distributions as follows:

$$\mathrm{P}(Q \mid W) = \prod_{t_i \in Q} p_{mix}(t_i \mid W) \qquad \textbf{(1)}$$

where: $p_{mix}(t_i \mid W) = \lambda * p(t_i \mid W) + (1-\lambda) * p(t_i \mid D)$

The estimates are smoothed by the document word statistics using the mixing parameter, λ [a]. The individual word probabilities are estimated in the obvious way using maximum likelihood estimators:

$$p(t_i \mid W) = n_{iW} / n_W \quad p(t_i \mid D) = n_{iD} / n_D \qquad \textbf{(2)}$$

where n_{iW} (n_{iD}), and n_W (n_D), are the number of term occurrences of term *i* in the window (document), and total term occurrences in the window (document) respectively.

Equivalently, we can rank windows by taking the log of both sides of (1), yielding:

$$\log \mathrm{P}(Q \mid W) = \sum_{t_i \in Q} \log p_{mix}(t_i \mid win) \qquad \textbf{(3)}$$

[a] We have investigated the best value for the mixing parameter empirically, and similar results were obtained in the range 0.8 through 0.999. In the experiments reported, we use 0.8.

2.2 Kullback-Leibler Model (KL)

The Kullback-Leibler divergence has been used extensively in applications of language modelling in information retrieval [8]. In this approach, we compute the Kullback-Leibler divergence between the window model and the document model, and we restrict the computation to the query terms, as follows:

$$KL(W \parallel D) = \sum_{t_i \in Q} p(t_i \mid W) \log\left(p(t_i \mid W) \big/ p(t_i \mid D)\right) \quad \textbf{(4)}$$

We use the strength of the divergence as a measure of window relevance with respect to a query. Intuitively, the larger the divergence of the window model from the (common) document model, the more likely the window is relevant to the query. Unlike the GEN model, the KL term weights have both a representation component (outside the *log*), and a discriminative component (the *log* term).

We are unable to use maximum likelihood estimates for the component probabilities, as we must avoid estimating zero for $p(t_i|W)$. We use the estimation rule attributed to Jefferys [9, p. 127] to estimate the parameters as follows:

$$p(t_i \mid W) = \left(n_{iW} + 0.5\right) \big/ \left(n_W + 1.0\right)$$
$$p(t_i \mid D) = \left(n_{iD} + 0.5\right) \big/ \left(n_D + 1.0\right) \quad \textbf{(5)}$$

2.3 Term Frequency Weighting Model (FREQ)

In this approach, we simply weight the window by summing the total occurrences of all query terms within the window. This approach can be given a language modelling interpretation, which will be useful in our later analysis. Using the query generation approach, we compute the window RSV as the probability of generating **any term** of the query as follows:

$$\mathrm{P}(any\ Q\ term \mid W) = \sum_{t_i \in Q} p(t_i \mid W) \quad \textbf{(6)}$$

This is equivalent to summing all the term occurrences in a window over the query terms.

3 Experimental Study

3.1 Research Questions

In this study we investigate three main research questions, namely:

- Is there an optimal window size for the sliding window, and is this optimal size dependent on the particular weighting function used?
- What is the relative effectiveness of the three weighting functions we have described?
- Could relevance profiling be used as an aid in generating a book index, through intellectual effort by a human, or indeed as a complete replacement for the book index?

In respect of window size, we conjecture that smaller window sizes will tend to be precision-enhancing as they are less likely to discover unrelated term occurrences, and that large window sizes will tend to be recall enhancing. We investigate a range of window sizes from a few sentences in length (50 words) up to 2-3 paragraphs or half a page (250 words).

For the weighting functions, we conjecture that the simple term frequency weighting (FREQ) will perform worst, given that it exploits less of the available information than the other functions. Both the query generation approach and the Kullback-Leibler approach have proved effective in conventional document retrieval. However, in this application, we believe that comparatively small size of sample text (namely the samples of window text) is likely to be a major limiting factor, and specifically in estimating probabilities. The simpler query generation model may prove more effective due to the smoothing provided by the mixture approach.

3.2 Corpus

We wish to investigate relevance profiling for within-document retrieval, and especially for long documents. And, in order to measure relative effectiveness, we need to establish a so-called "ground truth" for the experiments. We used van Rijsbergen's classic textbook, "Information Retrieval" [9], which is arguably long (191 pages, 60000 words). It is available in an electronic format, and it possesses a comprehensive book index. We had used this corpus successfully in previous end-user experiments [4] [5]. Here, we use the corpus in a similar way to a conventional test collection: the pages are the units of retrieval, the queries are provided by the book index entries, and the relevance assessments are the relevant pages associated with each index entry. We note that the book index was created by the book's author, and it is possible that the indexing is not exhaustive. Consequently, the absolute performance of the techniques we investigate is likely to be higher than we report. However, the main purpose of the experiments is to compare techniques, and in this respect all are evaluated against the same "ground truth".

Table 1 summarises the main characteristic of the corpus.

In the main, we conducted the experiments using the set of multi-term queries as all the weighting func-

tions are monotonic with respect to single term queries. We note that a high proportion of the multi-word queries are in fact phrasal queries rather than sets of words.

Number of pages:	191		
Number of words:	60035		
Index Entry Type	**No. of Entries[b] (queries)**	**Total relevant pages**	**Average #rels/entry**
single word	46	144	3.1
multi-word	232	766	3.3

Table 1: Details of corpus: "Information Retrieval" textbook by van Rijsbergen [9]

3.3 Experiment Design

The experiments were conducted as follows. Each query is used to generate a relevance profile for the document. Then, we simply rank the pages according to the retrieval status value, and evaluate the ranking based on the relevance judgements provided by the book index. In fact, this is not an ideal way of evaluating relevance profiling, which is designed to identify relevant groups (or clumps) of pages, and not simply individual pages. It would have been better to assess the effectiveness in identifying the same groups of pages identified in the book index. Given the availability of measures and tools for evaluating ranked output, we opted to do page ranking.

3.4 Measures

We used a range of single-valued measures averaged over the set of queries to assess effectiveness, namely: non-interpolated precision, R-precision (R-prec.), and three variations of the F-measure. R-precision is the precision achieved at rank R, where R is the number of relevant documents for a given query. This measure combines elements of both precision, proportion of retrieved R documents that are relevant, and recall, proportion of R relevant documents retrieved. The F-measure is simply the complement to the E-measure [9], and is given by:

$$F = R\,P/\left(\alpha R + (1-\alpha)P\right) \qquad \textbf{(7)}$$

where R[c] is recall, P is precision, and α enables one to bias the measure in favour of precision (α -> 1) or recall (α ->0). We use three variants with α set to 0.8 (precision-oriented), 0.5 (balanced harmonic mean of R and P), and 0.2 (recall-oriented).

To compute F-values for each query, we compute precision and recall at each different score in the ranking (note, not at each rank position), and then compute the optimal values F-values for the different settings.

Statistical significance testing was performed using the sign test, and p-values less than 0.05 were considered significant.

4 Experiments

We investigate each research question in turn, and present and discuss the results. Given the possible interaction between window size and weighting function, we decided to simply explore the window size initially with the query generation model (GEN), and we later confirmed the validity of the findings for the other two weighting functions.

4.1 Window Size Experiments

In the first set of experiments, we investigated effect of the size of the sliding window on the performance achievable with the query generation model (GEN). The size of the window was varied from 50 words through 200 words in steps of 25, and for 250 words. The experiment was run with two sets of queries, those for which the queries contained more than one word (MULTI), and those comprising a single word (SINGLE). In the context of document skimming, we think it likely that multi-word queries will be more usual that single word queries. However, we were also interested in the performance achievable with single word queries, as they form a significant proportion of entries in our book index. If ProfileSkim were used as a replacement for the book index then it would have to provide good performance for this type of query.

We present the window size results in Tables 2 and 3, where the **bold figures** are the best values achieved for each measure, and the *italicised figures* are values close to the best.

The results for the multi-term queries (Table 2) show that for F-measures, the results are broadly comparable over a wide range of window sizes from 50 through 250. Interestingly, the recall-oriented F (α=0.2) results are better than the other F results, indicating that relevance profiling may be a recall-oriented device. For the (averaged) non-interpolated precision and for R-Precision, the results are better for the smaller window sizes (50, 75), and significantly so compared with the results for window sizes 100 and up.

[b] There were a number of entries that we not used in the experiments, namely the 'see also' entries.

[c] This is a different use of 'R' and we trust the use is clear from the context.

Window size	**Prec.**	**R-prec.**	**F (α=0.8)**	**F (α=0.5)**	**F (α=0.2)**
50	*0.656*	0.559	0.699	0.695	*0.755*
75	**0.662**	**0.579**	*0.704*	*0.702*	**0.757**
100	0.648	0.541	*0.706*	*0.699*	*0.754*
125	0.650	0.551	*0.705*	*0.700*	**0.757**
150	0.645	0.541	**0.709**	**0.703**	**0.757**
175	0.647	0.546	*0.705*	*0.699*	**0.757**
200	0.643	0.539	0.701	0.695	0.752
250	0.628	0.514	0.691	0.686	0.749

Table 2: Averaged effectiveness of query generation weighting (GEN) for 232 multi-term queries for different window sizes

The performance for the single term queries (Table 3) is reduced by 5-10% compared with the multi-term queries, except for the R-Precision measure. Interestingly, for all measures, the maximum effectiveness is achieved at a window size of 200.

Window size	**Prec.**	**R-prec.**	**F (α=0.8)**	**F (α=0.5)**	F **(α=0.2)**
50	0.602	0.559	0.604	0.604	0.650
75	*0.618*	0.573	0.607	0.609	0.654
100	0.609	0.566	0.638	0.630	0.664
125	0.607	0.568	0.626	0.625	0.664
150	0.606	0.567	0.642	0.634	*0.672*
175	*0.620*	*0.593*	0.642	0.634	*0.672*
200	**0.621**	**0.598**	**0.660**	**0.647**	**0.678**
250	*0.617*	**0.598**	**0.654**	*0.642*	*0.673*

Table 3: Averaged effectiveness of query generation weighting (GEN) for 46 single term queries for different window sizes

The results of the window size experiments demonstrate the robust nature of the relevance profiling process, in that high and broadly comparable levels of performance are achieved across a range of window sizes. However, statistically speaking, small window sizes (50, 75) yield significantly better results for non-interpolated precision and R-precision, for multi-term queries. Nevertheless, it is clear that relevance profiling works well for windows of a few sentences in length up to half a page. For single term queries, the best effectiveness is achieved for larger window sizes (200, 250). A possible explanation for this is that, in order to establish relevance for a short query, we may require a larger sample of text

Contrary to expectations, small window sizes did not seem to be precision-enhancing, nor large window sizes recall-enhancing. If one looks in detail at the relevance profiling process (Figure 1), one can observe both precision- and recall-enhancing devices. By insisting that pages can only achieve a score if at least one query term is actually present in the page, we are effectively boosting precision. On the other hand, the fact that any window beginning in a page can contribute an RSV to that page is a recall-enhancing device. We would argue that these devices operate to mitigate the effects of window size, and particularly the possible deleterious effect of large window sizes on precision.

4.2 Weighting Function Experiments

In this second set of experiments, we investigate the comparative effectiveness of the three weighting functions described in section 2.

In Table 4, we report the best levels of effectiveness achieved for each weighting function, where the window size is chosen optimally for each function (and in some cases measure).

	Standard		
Function	GEN	KL	FREQ
Window Size	75	50	75
Average Precision	**0.662**	0.575	0.536
Average R-precision	**0.579**	0.460	0.430
F (α=0.8)	**0.704**	0.630	0.542
F (α=0.5)	**0.702**	0.626	0.550
F (α=0.2)	**0.757**	0.698	0.636

Table 4: Averaged effectiveness of three weighting functions for 232 multi-term queries (using optimal window setting for each function)

	With coordination filtering		
Function	GEN	KL	FREQ
Window Size (unless stated otherwise)	200	75	75
Average Precision	**0.607**	*0.595*	*0.598*
Average R-precision	**0.544 (75)**	*0.533*	*0.539*
F (α=0.8)	**0.677**	*0.669 (100)*	0.646
F (α=0.5)	**0.661**	*0.654 (100)*	0.638
F (α=0.2)	**0.696**	*0.690 (200)*	*0.685 (200)*

Table 5: Averaged effectiveness of three weighting functions with co-ordination level filtering for 232 multi-term queries (using optimal window setting for each function)

Let us consider the 'standard' results first. Clearly, the performance of the query generation model (GEN) exceeds that of the other two weighting functions by a large and statistically significant margin. It is approximately 10%-15% better than the Kullback-Leibler model, and above 20% better than the term frequency weighting model. Although KL weighting outperforms FREQ, it is not significantly better. It would appear that the ostensibly simple query generation model is better than the arguably more information-rich KL model, and we surmised earlier the problem of estimating parameters from small samples may be important in relevance profiling.

If we look at the individual term weights in the KL model (eqn 4), it is clear that within the *log* term, $p(t|D) << p(t|W)$, and thus the contribution of $p(t|D)$ dominates the expression. Given the nature of a book index, we believe that many terms in the index will be of similar frequency, and thus the $P(t|D)$ will be both small and relatively constant for query terms. Consequently, the term outside the *log*, namely $p(t|W)$ will provide the overall "shape" of the weighting function, and this is the same basis for the term frequency weighting (eqn 6). Hence, the performance of KL and FREQ should be, and indeed are, broadly similar.

How then do we explain why the performance of GEN is so much better than the other two weighting functions? We think it is the treatment of the non-occurrence of query terms in a window that is the distinguishing hallmark of GEN. In GEN, if a query term does not appear in the window, then the weighting function penalises this harshly. Consider that in the probability mixture (eqn 1), the contribution from the document model will always be very small. Whereas, in the other two functions the weighting is more or less linear in the number of term occurrences. To test this conjecture, we ran a third set of experiments, in which we added an additional filter, which we dubbed the coordination level filter. In these experiments we insist that all query terms are present in a window (or "coordinated" as it used to be known), prior to a retrieval status value being calculated for that window.

The results of this third set of experiments are presented in Table 5. In some cases, there was a different optimal window size for some measures, and this is included in brackets with the measure value. We expected that co-ordination filtering will reduce the number of pages retrieved, and hence likely to depress recall, and possibly boost precision. For GEN, performance was reduced by about 5-10% for all measures. However, the co-ordination filtering boosted the performance of both KL and FREQ to comparable levels to the filtered GEN performance. This provides strong evidence that it is the treatment of the non-occurrence of query terms which is the key to the performance of the unfiltered (or standard) query generation model. It may be that the performance of the KL model could be improved by experimenting with the parameter estimation rules. Interestingly, incorporating co-ordination filtering, results in generally larger optimal window sizes for GEN and to some extent KL.

4.3 Relevance Profiling for Book Indexing

Here, we draw together the evidence that relevance profiling might be used as an aid in generating a book index, namely through intellectual effort by a human, or indeed as a complete replacement for the book index.

First, the absolute effectiveness achieved when using the query generation model was very high, for both the multi-term and single term sets of queries. For the multi-term queries (Table 3), it achieved optimal F-values of 0.703 (F, α=0.5), which effectively means that at this optimum point, on average both Precision and Recall achieved levels around 70%. Moreover, the R-precision value of 0.58 is quite high, and means if a user inspects down to the rank corresponding the exact number of relevant pages for each query, then on average 58% of these relevant pages will be retrieved.

High recall is clearly important in the context of book indexing. We also computed the number of queries that achieved recall of 100% at a cutoff of 20 retrieved pages. For GEN (window size = 75), 175 of the 232 multi-term queries achieved maximum recall, and only 9 of the 232 queries failed to retrieve any relevant documents. Further, GEN (window size = 75) retrieves 587 of the possible 766 relevant pages for the multi-term queries, and 100 of the possible 144 relevant pages for the single term queries. It seems reasonable to conclude that a tool such as ProfileSkim, which implements relevance profiling, would prove a useful aid to a human indexer. Indeed, given a set of index entries, relevance profiling might even replace the traditional book index.

It is worth noting that careful inspection of the existing book index showed conclusively that the index is not completely exhaustive. In particular, there is inconsistent treatment of page ranges, where frequently not all relevant pages in a range are included in the index. Relevance profiling was able to reliably identify many of these unidentified "relevant" pages.

5 Conclusions and Future Work

In this paper we have reported a series of bench experiments of relevance profiling. The experiments were conducted with an electronic book and its associated book index. We investigated three alternative

weighting functions, and various parameters of the relevance profiling process, most notably window size.

The major findings of the experiments are:

- relevance profiling is a robust process yielding acceptably high levels of performance over a range of window sizes from 75-200 words in length;
- the query generation model (GEN) is significantly more effective than the Kullback-Leibler (KL) and the term frequency (FREQ) models;
- a key factor in effectiveness of query generation model is its treatment of non-occurrences of query terms in the sliding window, which we attribute to the smoothing provided by the probability mixture model; and
- the absolute performance of relevance profiling indicates that it might be usefully employed as an aid to human book indexing, and indeed may provide a viable alternative to the book index (providing that a good set of index entries can be generated).

We would emphasise that these experiments have been conducted with a single book, and associated index, and it would be highly desirable to repeat the experiments with a number of such books. Nevertheless, we believe that the general conclusions would likely be confirmed by such experiments, albeit the absolute levels of effectiveness may differ, depending on the exhaustivity of the (human) indexing.

The results reported here suggest a number of possible avenues for future work. First, given the evident, and indeed widely recognised, importance of parameter estimation and smoothing in language modelling applications, and given the specialised nature of relevance profiling, further experiments should be conducted on both GEN and KL using different parameter estimation rules or smoothing approaches. Second, all the weighting functions we investigated assumed that query terms are distributed independently. Given that, in general, this is unlikely to be the case, and especially for phrasal queries, it would be desirable to investigate models based on term dependence. However, we would suggest only first-order models be investigated given the paucity of sample data.

Finally, it would be interesting to explore how to generate "useful" queries automatically, where by useful we mean the query identifies coherent chucks of relevant text. We note that such queries would not necessarily be just phrasal. This would pave the way for both automating the process of book indexing, and perhaps more tantalizingly, enabling query-less retrieval within documents.

Acknowledgements

This work was done within the Smart Web Technologies Centre that was established through a Scottish Higher Education Funding Council (SHEFC) Research Development Grant (No. HR01007). David Lee was supported by a Summer Studentship provided by The Robert Gordon University. We thank the following colleagues for their feedback on this work: Ian Pirie, Ivan Koychev, Bicheng Liu, Ralf Bierig and Murat Yakici. I also thank Margery Murray for her invaluable assistance in preparing the manuscript. Finally, we would like to thank the anonymous referees for their insightful comments on the submitted paper.

6 References

[1] Kaszkiel, M. and Zobel, J.: Passage Retrieval Revisited. In: *Proceedings of the Twentieth International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Philadelphia, July 1997. ACM Press, 178-185, 1997.

[2] Hearst, M. A.: TileBars: visualization of term distribution information in full text information access. *Proc. CHI'95*, 56-66, 1995.

[3] Harper, D. J., Coulthard, S., and Sun, Y. A language modelling approach to relevance profiling for document browsing. In *Proceedings of the Joint Conference on Digital Libraries*, pages 76-83, Oregon, USA, July 2002.

[4] Harper, D.J., Koychev, I. and Sun, Y. Query-based document skimming: A user-centred evaluation of relevance profiling. In: *Proceedings of 25th European Conference on Information Retrieval.* Lecture Notes in Computer Science, Springer-Verlag, Berlin, pp. 377-392, 2003.

[5] Harper, D.J., Koychev, I., Sun, Y. and Pirie, I. Within-Document Retrieval: A User-Centred Evaluation of Relevance Profiling. In: *Information Retrieval,* Kluwer Academic Publishers, The Netherlands, 7, 265-290, 2004.

[6] Ponte, J. and Croft, W. B.: A language modeling approach to information retrieval. In: *Proceedings of the 1998 ACM SIGIR Conference on Research and Development in Information Retrieval*, 275-281, 1998.

[7] Song, F. and Croft, W.B.: A general language model for information retrieval. In: *Proceedings of the 1999 ACM SIGIR Conference on Research and Development in Information Retrieval*, 279-280, 1999

[8] Croft, W.B. and Lafferty, J. (Eds.): Language modeling for information retrieval. Kluwer Academic Publishers, The Netherlands, 2003.

[9] Van Rijsbergen, K. *Information Retrieval*. Butterworths, London, 1979.

Optimal Structure Weighted Retrieval

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

Abstract *Improving ranking functions for structured information retrieval has received much attention since the inception of XML. Weighting document structures is one method providing significant improvement – but how good can these improvements be?*

Optimal structure weighted retrieval occurs when each query is processed using the optimal set of weights for that query. Optimal retrieval for a set of queries occurs when a set of weights optimized for that set of queries is used. Measuring mean average precision for each of these will give a performance upper bound for document structure weighted retrieval.

In this investigation a near optimal set of weights is learned for TREC WSJ collection topics 101-200 using a genetic algorithm. Weights are learned for vector space inner product, naïve probability and BM25 ranking functions and a performance upper bound is calculated.

The upper bound using a different set of weights for each query, gives mean average precision improvements of about 15% for BM25 and naïve probability; about 30% for inner product. This suggests structure weighting might be useful for relevance feedback. Optimal weights for the set of queries shows improvements of about 5% for naïve probability and inner product, but of only about 1% for BM25; suggesting this technique is not as effective for ad hoc retrieval.

Keywords Information Retrieval.

1. Introduction

As markup languages such as SGML [13] and XML [3] became more popular, many data providers switched to offering their data in these formats. The two big information retrieval collections, TREC [11] and INEX [5], are both available in XML.

This availability of structured documents raises questions about relevance ranking – how, exactly, can the document structure be used to improve whole document ranking? One approach is to weight term occurrences based on where in the document they lie. A term found in an abstract is perhaps of greater significance than the same term found in the body text of the same document, fulfilling the principle of summarization. This was the suggestion of Fuller *et al.* [7].

In structure-weighted retrieval each document structure (or tagged element) is given a weight. Each occurrence of each search term is weighted according to the structure in which it occurs. Document term frequencies are then replaced by linear weighted term frequencies based on term structure occurrences. In this way whole documents are ranked using the structural information present in the documents.

There are several ways to choose the weights. Proposals include trial and error [28], simulated annealing [2], genetic algorithms [26], and asking the user to supply them as part of the query [6].

Regardless of how the weights are chosen, there remains a question central to all ranking functions: performance. Ordinarily this is easily measured. Any standard test collection is obtained, the mean average precision (MAP) is computed over all the queries in the collection and compared to that of other ranking functions. The highest performing function is considered "best" for that collection. A test such as the *t*-test is often used to show the significance of this difference.

With structure weighting, exactly the same approach is used, only the ranking function doesn't change, only the structure weights change. In effect, the process is to optimize an existing function to a given set of structures. But how good is this optimization in isolation? What is the expected performance gain using just this approach? How does it compare to using a different ranking function?

Imagine there is an oracle that knows, in advance and for every query, the optimal set of weights to use. In optimal structure weighted retrieval, a query is received from a user, given to the oracle that returns the weights, those weights are loaded into the retrieval engine and the query processed. In essence, the oracle ensures every query is answered optimally.

In this investigation the performance gain of structure weighted retrieval is measured in exactly this way. The TREC Wall Street Journal collection and TREC topics 101-200 are used. The oracle is

Proceedings of the 9th Australasian Document Computing Symposium, Melbourne, Australia, December 13, 2004.

simulated by a genetic algorithm [12], which learns, for each topic, a near optimal weight set. Experiments were conducted using vector space inner product [20], naïve probability [10] and the BM25 [19] ranking functions. An approximation to the performance upper bound is found.

Results for unweighted retrieval show BM25 outperforms naïve probability, which outperforms inner product. Using the oracle, the mean gain is approximately 15% for BM25 and naïve probability, and 30% for inner product (which does not change this order). When the oracle was tasked to learn a single set of weights to apply to all queries, the improvements were much smaller. This suggests document structure weighting might be better suited to relevance feedback than to *ad hoc* retrieval.

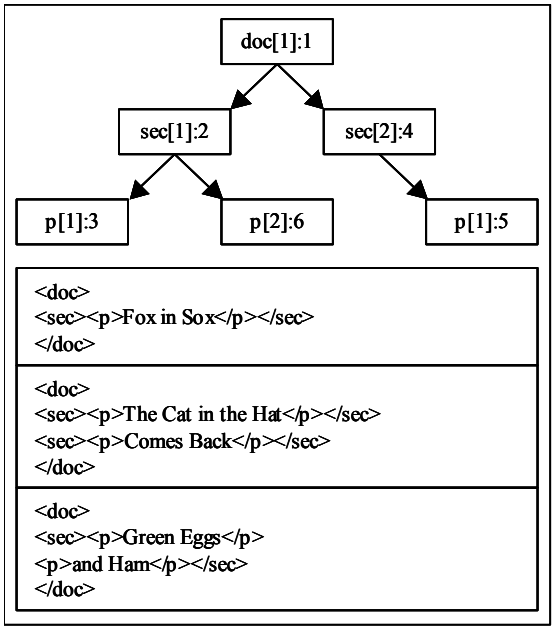

Figure 1: Three documents and the corpus tree for those documents. Instances are shown in square braces, e.g., p[2] represents the second paragraph. Identifiers, issued on an as encountered basis, are shown after the colon.

2. Structured information retrieval

In an inverted file information retrieval system there is one dictionary, and each dictionary term points to one inverted list of postings. The postings are usually represented as $\{<d_1, f_1><d_2, f_2>, \ldots, <d_n, f_n>\}$ where d_n is a document number and f_n is the frequency of the given term in the given document.

For structured information retrieval it is also necessary to know where in the document terms are found, so the postings must be annotated with this. Exactly how the postings are annotated does not matter, so long as it is possible to know, for each posting, in which structure that posting is found.

Several encoding techniques have been suggested. Each different tag in the DTD could have a separate inverted list [21; 22]. The path to the posting could be stored as a path directly in the indexes [4; 25]. Or a tree representing the structure of the collection could be built, labeled, and used.

To build the tree, first each document tree is built; then these are superimposed to form a single tree. This corpus tree includes every path through every document, but is unlikely to match the structure of any one document. Three documents and the corpus tree they form are shown in Figure 1.

Nodes in the tree can be encoded in several ways. If the tree is considered to be a k-way virtual tree (where some nodes may not exist) and each node is labeled with an identifier reflecting this, paths upwards through the tree can be computed directly from the identifiers. These identifiers can then be stored directly in the postings [16; 24].

Alternatively, each node in a tree of M nodes can be given an ordinal identifier from 1 to M. Then for each posting a bitstring of length M is constructed with a 1 bit for every node above the given postings and a 0 for all others. These bitstrings are then stored with each posting [17].

The corpus tree can be built dynamically during single pass indexing [15; 26]. As each new path in each new document is encountered it is added to the corpus tree and given a unique ordinal identifier. These identifiers are then stored with each posting, $\{<d_1, p_1, f_1><d_2, p_2, f_2,>, \ldots, <d_n, p_n, f_n>\}$ where p_n is the corpus tree node identifier.

With each of these approaches it's possible to determine which terms occur where in which document and how many times. For the experiments herein it doesn't matter how the indexes are stored so long as that information is available.

3. Ranking

3.1. HTML

Tag weighting schemes for HTML have received much attention. The document weight is computed from not only the term occurrences, but from the tags in which the term occurs.

One approach [14] is to give each tag a weight. Then for ranking, compute two values: term frequency within the document, tf_{id}, and the product of the tag weights for each tag in which the term is found. These are multiplied to give a weighting for the term in the document. This approach averages the tag weights over the parts of the document covered by the term. A document containing a term twice in <TITLE> and once in <B> is weighted identically to a document containing the same term once in <TITLE> and twice in <B>.

To overcome this averaging, each occurrence of each term can be multiplied by an element weight [2; 18]. Term frequency tf_{id} for a given term, i, in a given document, d, is replaced by a structure weighted term frequency, ctf_{id}, computed as

$$ctf_{id} = \sum_{p=1}^{n} \left(C_p \times tf_{ipd} \right) \quad (1)$$

where p is a given element, C_p is a weight for that element, and tf_{ipd} is the number of occurrences of term i in structure p of document d.

Different approaches vary in which elements (and other heuristics) are used for ranking. Some, for example, include the number of incoming web page links.

3.2. SGML and XML

There are two approaches to searching SGML and XML, retrieval of elements and retrieval of whole documents. The annual INEX workshop focuses on element retrieval for which there are a wide number of approaches [5]. This investigation focuses on whole document retrieval (as seen in digital academic libraries), for which element retrieval techniques are not appropriate.

For whole document retrieval, Kotsakis [15] used the weighted term frequency approach from equation (1) with vector space ranking and Trotman [26] did the same for naïve probabilistic and BM25 ranking.

3.3. Structured ranking equations

3.3.1. Vector space inner product

The vector space inner product weight, w_{dq}, is computed as the inner product of the document vector, w_{id}, and the query vector, w_{iq}. The structure-weighted variant is

$$w_{dq} = \sum_{i=1}^{t} \left(w_{id} \times w_{iq} \right) \tag{2}$$

where

$$w_{iq} = tf_{iq} \times IIDF_i \tag{3}$$

and tf_{iq} is the number of occurrences of term i in query q and

$$w_{id} = ctf_{id} \times IIDF_i \tag{4}$$

and

$$IIDF_i = \log_2 \frac{N+1}{n_i} \tag{5}$$

where N is the number of documents in the collection and n_i is the number of occurrences of term i in the collection.

3.3.2. Naïve probability model

The weight of the document with respect to a query, w_{dq}, is given by

$$w_{dq} = \sum_{i=1}^{t} \left(\left(C + PIDF_i \right) \times \left(L + \left(1 - L \right) \times \frac{ctf_{id}}{m_d} \right) \right) \tag{6}$$

where

$$PIDF = \log_2 \frac{N - n_i + 1}{n_i} \tag{7}$$

where $C = 1.0$, $L = 0.3$ and m_d is the term frequency of the most frequent term in document d.

3.3.3. Okapi BM25

For Okapi BM25 ranking,

$$w_{dq} = \left(\sum_{i=1}^{t} BIDF \times \frac{(k_1 + 1) \times ctf_{id}}{K + ctf_{id}} \times \frac{(k_3 + 1) \times tf_{iq}}{k_3 + tf_{iq}} \right) \tag{8}$$

where

$$BIDF = \log \frac{N - n_i + 0.5}{n_i + 0.5} \tag{9}$$

and

$$K = k_1 \times \left(\left(1 - b \right) + \frac{b \times T_d}{T_{av}} \right) \tag{10}$$

where $k_1 = 1.2$, $k_3 = 7$, $b = 0.75$, T_d is the number of terms in the document, and T_{av} is the average document length measured in terms (the same unit of measure as T_d).

If all weights are set equal to 1, no structure is preferred over any other and unstructured retrieval results. If a structure weight is set to 0, that structure will not be included in ranking. Other choices of weights set the relative importance of each structure to the others. There remains, however, the problem of choosing the weights.

4. Choosing document structure weights

One way to choose weights is to ask the user to supply them as part of the query. Graphic query languages supporting this have already been proposed [1], as have text based query languages [6]. Although these languages allow the user to customize the weights for the each query, it is not obvious how to choose good weights. Tests on HTML documents demonstrate a decrease in precision for almost all points of recall when a human subject is asked to do this [18].

Manual approaches to choosing collection-wide weights have also been tried. By setting all weights equal and varying one weight it is possible to measure the influence of each element in isolation. A set of weights for all the elements is then chosen based on how each element performs in isolation [28]. This technique, however, is unable to identify co-dependence between element.

To get an optimal set of weights a fully automated machine learning approach is needed. Gradient-based

optimization has successfully been used [18]. Recently genetic algorithms have also been show to produce good weights for HTML [14] and for XML [26]. These investigations have shown that a set of weights learned on one set of topics is effective for improving another.

4.1. The genetic algorithms approach

Genetic algorithms [12] have been shown to be a robust optimization mechanism in many fields [8]. In information retrieval they have been used to build indexes [9], for relevance feedback [27], and for choosing document structure weights.

Each node in the corpus tree is given a unique ordinal node identifier as it is created. Consequently, once indexing has completed, the nodes will be labeled 1 to *M*. Weights for each node can, therefore, be represented as an array of length *M*, one array position for each weight.

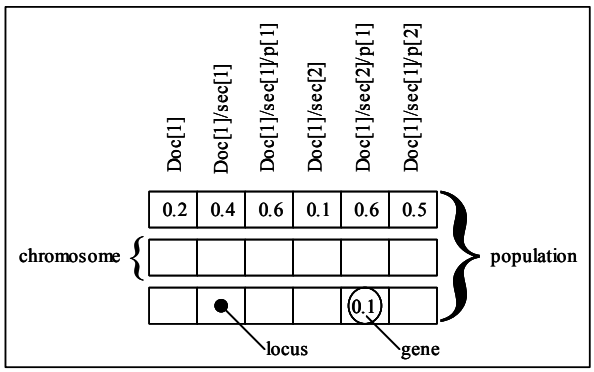

Figure 2: The chromosomal encoding of a set of weights forms a genetic algorithms individual. A set of individuals is a population.

The array is analogous to an individual chromosome in a genetic algorithm simulation. The array positions are analogous to loci and the values in the array to genes. A population is a set of arrays. This analogy is shown in Figure 2.

The genetic operations are a direct manipulation of these arrays.

For reproduction an individual is selected with fitness proportionate selection.

In mutation, an individual is chosen using fitness proportionate selection. From that individual a random locus is chosen. A random gene is then selected and placed at that locus.

In one-point crossover, two individuals are chosen using fitness proportionate selection. A random locus is chosen. Two new individuals are then created with the genes after the locus switched.

The simulation is run until a satisfactory result is found or for a fixed number of generations.

5. Experimental design

Two experiments were conducted; both used the Wall Street Journal collection (1987-1992) from TREC disks 1 and 2. Topics 101 to 200 were used, with topics having fewer than 5 judgements being discarded (121, 175, 178 and 181). Topics were converted into queries by extracting terms from the description field and stopping common words. Stemming was not used.

The collection was indexed with a structured information retrieval system using a corpus tree and with postings annotated with pointers into the tree.

A population of 50 individuals was chosen at random. The probability of mutation was 0.2, of crossover was 0.5, and of reproduction was 0.3 (other ratios were not tested). Genes took a value between 0 and 1. The initial population was seeded with an individual with weights all 1; the equivalent of unweighted retrieval. Each simulation ran for 25 generations. The experiments were conducted 50 times to reduce the chance of error.

The purpose of these experiments is to compute an upper bound for structure weighted retrieval (in isolation of other factors). This is done by deliberately over fitting to a training set and then reporting the result as the optimum – it is the best that can be seen on the training set (in this case the entire set), and so can reasonably be reported as optimal. Although it is possible the optimal set of weights will not be found in one run of the genetic algorithm (it might become stuck at a local maximum), repeating the experiment 50 times decreases this inherent error. As no proof of optimality of the weighs is given, the results herein must be considered near-optimal and not optimal.

5.1. Experiment 1: The oracle

Each time a query is issued, the oracle is consulted for the optimal set of structure weights. These weights are then loaded into the retrieval engine and the query resolved. Such an oracle cannot exist, but the effect of having one can be simulated.

In experiment 1, weights are learned for each topic individually. From the 50 runs, weights from the one run showing the largest improvement in average precision are chosen as the best weights. These weights can reasonably be expected to be near optimal.

Using the near optimal weights with each query is an approximation of the oracle. For each query, the near optimal weights are loaded into the retrieval engine and the query resolved against the document collection. The average precision for this query is near the optimal average precision possible for this query, using this technique.

This experiment was conducted three times, once for each of inner product, naïve probability and BM25 (equations (2), (6) and (8) respectively).

5.2. Experiment 2: One answer fits all

If the oracle could be consulted only once, it would be to find the set of weights that would maximize MAP

over all queries. This is simulated in experiment 2 where one set of weights is learned using all queries.

This experiment was conducted three times, once for each of inner product, naïve probability and BM25 (equations (2), (6) and (8) respectively).

6. Results

The results of experiment 1 are compared to those of experiment 2 and to unweighted retrieval to get an approximation to how good document structured retrieval can be.

6.1. Experiment 1

Table 1 presents the results from Experiment 1. Using the oracle results in mean average prevision improvements varying between less than one percent and several hundred percent. The mean improvement is calculated as the sum of improvements divided by the number of queries. This is not the improvement in the mean average precision, which is presented in Table 2.

No queries showed a decrease in precision. In Figure 3 the improvements are plotted from greatest to least improvement and for each ranking function. Only a small number of queries receive large improvements, most receive small improvements. Average precision improvements of greater than 5% were seen in 78% of queries using inner product, 57% using naïve probability and 67% using BM25.

Without using structure weighting, BM25 outperforms naïve probability, which outperforms inner product. When the oracle is used with each of the three functions, they still perform in the same order; BM25 is better than naïve probability is better than inner product.

Unexpectedly, different queries showed different improvements using different ranking functions. In Figure 4 the topics are shown sorted in order of most to least improvement for inner product. The lines representing BM25 and naïve probability show improvement spikes. From visual inspection, the queries showing large improvements appear to be different for each ranking function.

The differences in Figure 4 can be quantized using the Pearson correlation. Pearson values near 1 show a positive correlation; those near -1 show a negative correlation, while values close to 0 show no correlation. From Table 3; there is no cross function correlation in how much each topic is improved.

Method	Max	Mean	Min
I. Prod	1081.88%	71.83%	0.77%
N. Prob	356.75%	28.39%	0.04%
BM25	848.18%	35.04%	0.04%

Table 1: Average precision improvements seen in experiment 1.

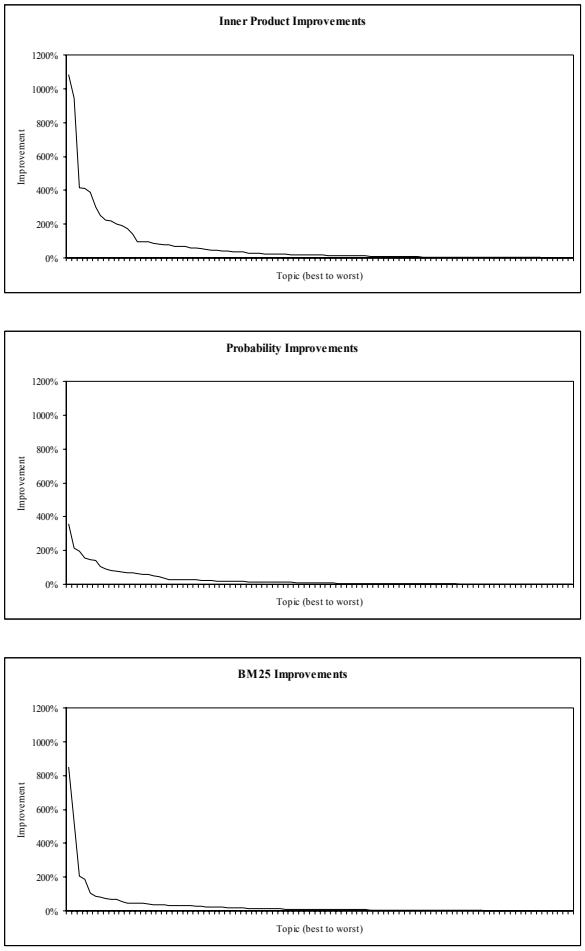

Figure 3: Improvements in precision vary greatly, but are mostly small.

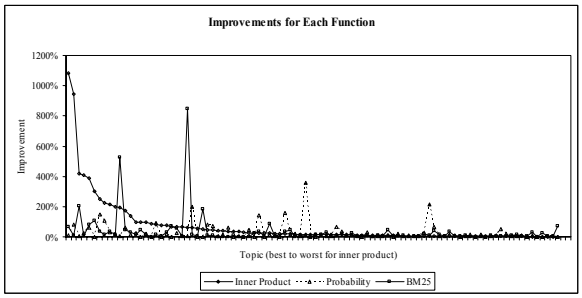

Figure 4: Different topics show different improvements with different ranking functions.

Method	I. Prod	N. Prob	BM25
Unweighted	0.15674	0.17804	0.24178
Experiment 1			
The Oracle	0.20329	0.20560	0.27750
Improvement	29.70%	15.48%	14.77%
P (*t*-test)	0.00	0.00	0.00
Experiment 2			
One Answer Fits All	0.16399	0.18991	0.24332
Improvement	4.63%	6.67%	0.64%
P (*t*-test)	0.00	0.00	0.04

Table 2: Mean average precision of experiments 1 & 2. Significance (P) is computed with a 2-tailed *t*-test.

Exp 1	I. Prod	N. Prob	BM25
I. Prod	1.00	0.09	0.14
N. Prob	0.09	1.00	-0.07
BM25	0.14	-0.07	1.00

Table 3: Pearson correlation showing improvements seen in one function do not correlate to those seen in another when weights are learned for each topic.

Using this result, its possible to ask a new question. What if there existed a super-oracle who knew the best function to use and the best set of weights to use with that function? In this case the MAP increases to 0.29046, this is a 20% improvement on unweighted BM25, a just under 4.7% improvement on the oracle result for BM25.

6.2. Experiment 2

Results for experiment 2 are shown in Table 2. When the oracle returns a single set of weights suitable for all queries, the improvement is substantially less than when weights are returned for each topic. This is exactly as expected, one general purpose set of weights can not be expected to perform as well as a set of special purpose weights tailored to each query.

Improvements for inner product and naïve probability are about 5%. Those for BM25 show a less than 1% improvement. Even so, BM25 outperforms naïve probability, which outperforms inner product. The same order as when no weighting was used.

Exp 2	I. Prod	N. Prob	BM25
I. Prod	1.00	-0.08	-0.13
N. Prob	-0.08	1.00	0.05
BM25	-0.13	0.05	1.00

Table 4: Pearson correlation showing improvements seen in one function do not correlate to those seen in another when one set of weights is used for all topics.

Exp 1 / Exp 2	Pearson
I. Prod	0.58
N. Prob	0.87
BM25	0.01

Table 5: Pearson correlation computed between experiment 1 and experiment 2. Those topics showing improvements are correlated across experiments for inner product and naïve probability, but not for BM25.

The Pearson correlation is shown in Table 4 – again there is no cross function correlation between how much each topic is improved. When the results from experiment 1 and experiment 2 are correlated, a strong correlation is seen for naïve probability, less so for inner product, and none for BM25, as shown in Table 5. BM25 most likely shows no correlation because the improvements in experiment 2 are themselves not significant.

BM25 showed close to no improvement across these queries. This may be because BM25 was developed for this collection. The tuning parameters, k_1, k_3 and b, are used to customize the function to a given set of documents, and are already optimized for TREC.

7. Discussion and future work

The documents in the TREC WSJ collection are marked up using about 20 unique tags. The documents are short, and the markup is sparse. The queries used in these experiments are derived from topic descriptions, they, too, are short. These two factors may account for why the near optimal improvements are small.

Should all the query terms occur in only one element (in the case of WSJ, the TEXT element), each document receives a constant weight resulting in a linear scaling of the ranking score; which is order preserving. Should only a few relevant documents additionally contain terms elsewhere (e.g. the title or HL element), only the order of those documents will be adjusted relative to the others; only a small change in MAP will be observed.

This could be verified by using longer queries. By increasing the number of terms in the query the chance of hitting terms outside the TEXT element will increase so the number of documents changing order will increase and the MAP will reflect this change.

Alternatively, a collection of documents with longer elements could be used. The INEX collection [5] contains long documents broken into title abstract, sections and subsections and may be appropriate. It does, however, contain 192 unique tags, many of which are unlikely to be useful for ranking (e.g. citation identifiers). Such tags would be removed, the content being preserved, before such experiments are conducted.

The TREC binary relevance judgments may also be the source of the small MAP improvement. If the ranking function has successfully identified the relevant documents, and ordered those at the top of the results list, any amount of re-ordering to place "more" relevant documents first will not be observed. This could be verified using a document collection that has non-binary relevance judgments (such as the cystic fibrosis collection [23])

The difference seen between experiment 2 and experiment 1 is substantial. When one set of weights is learned for a set of topics the improvement is small, when weights are learned for each query the improvement is larger. This suggests the latter technique might be useful, if it could be utilised in a retrieval engine. One way to improve MAP given a set of known relevant documents is relevance feedback. Such improvements are expected to be smaller than those observed herein as in relevance feedback only a subset of relevant documents are identified so fewer documents are available for learning; over-fitting to this subset may also occur.

8. Conclusions

An oracle that can return the optimal set of document structure weights to use for structure weighted information retrieval is theorized. Using the oracle gives the performance upper bound for this technique. For the TREC WSJ collection, an approximation to the oracle is constructed and tested and an approximation to the upper bound is computed.

When the oracle was not used, BM25 was shown to outperform naïve probability, which outperformed inner product. When the oracle was consulted for each query, mean average precision showed an improvement of 15% for naïve probability and BM25, and of 30% for inner product. Even so, using the oracle did not change the order of the functions; BM25 was still better then naïve probability which is still better than inner product.

Only small improvements are seen in most queries, while very few queries show large improvements. The queries that showed large improvements were different for each ranking function.

When the oracle was consulted once for a single set of weights to use for every query, smaller improvements were seen. Inner product and naïve probability showed a 5% improvement whereas BM25 showed a 0.6% improvement. Again, the order of goodness of the functions did not change. BM25 was better than naïve probability which was better than inner product.

These results suggest document structure weighted retrieval is better suited to relevance feedback than to *ad hoc* retrieval.

References

[1] Baeza-Yates, R., Navarro, G., & Vegas, J. (1998). A model and a visual query language for structured text. In *Proceedings of the String Processing and Information Retrieval: A South American Symposium*, (pp. 7-13).

[2] Boyan, J., Freitag, D., & Joachims, T. (1996). A machine learning architecture for optimizing web search engines. In *Proceedings of the AAAI Workshop on Internet-Based Information Systems*.

[3] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F., & Cowan, J. (2003). Extensible markup language (XML) 1.1 W3C proposed recommendation. The World Wide Web Consortium. Available: http://www.w3.org/TR/2003/PR-xml11-20031105/.

[4] Fuhr, N., & Gövert, N. (2002). Index compression vs. Retrieval time of inverted files for XML documents. In *Proceedings of the 11th ACM International Conference on Information and Knowledge Management*.

[5] Fuhr, N., Gövert, N., Kazai, G., & Lalmas, M. (2002). INEX: Initiative for the evaluation of XML retrieval. In *Proceedings of the ACM SIGIR 2000 Workshop on XML and Information Retrieval*.

[6] Fuhr, N., & Großjohann, K. (2000). XIRQL an extension of XQL for information retrieval. In *Proceedings of the ACM SIGIR 2000 Workshop on XML and Information Retrieval*.

[7] Fuller, M., Mackie, E., Sacks-Davis, R., & Wilkinson, R. (1993). Structured answers for a large structured document collection. In *Proceedings of the 16th ACM SIGIR Conference on Information Retrieval*, (pp. 204-213).

[8] Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*: Addison-Wesley.

[9] Gordon, M. (1988). Probabilistic and genetic algorithms in document retrieval. *Communications of the ACM,* 31(10), 1208-1218.

[10] Harman, D. (1992). Ranking algorithms. In W. B. Frakes & R. Baeza-Yates (Eds.), *Information retrieval: Data structures and algorithms* (pp. 363-392). Englewood Cliffs, New Jersey, USA: Prentice Hall.

[11] Harman, D. (1993). Overview of the first TREC conference. In *Proceedings of the 16th ACM SIGIR Conference on Information Retrieval*, (pp. 36-47).

[12] Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

[13] ISO8879:1986. (1986). *Information processing - text and office systems - standard generalised markup language (SGML).*

[14] Kim, Y.-H., Kim, S., Eom, J.-H., & Zhang, B.-T. (2000). SCAI experiments on TREC-9. In *Proceedings of the 9th Text REtrieval Conference (TREC-9)*, (pp. 392-399).

[15] Kotsakis, E. (2002). Structured information retrieval in XML documents. In *Proceedings of the ACM Symposium on Applied Computing*, (pp. 663-667).

[16] Lee, Y. K., Yoo, S.-J., Yoon, K., & Berra, P. B. (1996). Index structures for structured documents. In *Proceedings of the 1st ACM International Conference on Digital Libraries*, (pp. 91-99).

[17] Meuss, H., & Strohmaier, C. (1999). Improving index structures for structured document retrieval. In *Proceedings of the 21st Annual Colloquium on IR Research (IRSG'99)*.

[18] Rapela, J. (2001). Automatically combining ranking heuristics for HTML documents. In *Proceedings of the 3rd International Workshop on Web Information and Data Management*, (pp. 61-67).

[19] Robertson, S. E., Walker, S., Beaulieu, M. M., Gatford, M., & Payne, A. (1995). Okapi at TREC-4. In *Proceedings of the 4th Text REtrieval Conference (TREC-4)*, (pp. 73-96).

[20] Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM,* 18(11), 613-620.

[21] Schlieder, T., & Meuss, H. (2000). Result ranking for structured queries against XML documents. In *Proceedings of the DELOS Workshop on Information Seeking, Searching and Querying in Digital Libraries*.

[22] Schlieder, T., & Meuss, H. (2002). Querying and ranking XML documents. *Journal of the American Society for Information Science and Technology,* 53(6), 489-503.

[23] Shaw, W. M., Wood, J. B., Wood, R. E., & Tibbo, H. R. (1991). The cystic fibrosis database: Content and research opportunities. *Library and Information Science Research,* 13, 347-366.

[24] Shin, D., Jang, H., & Jin, H. (1998). BUS: An effective indexing and retrieval scheme in structured documents. In *Proceedings of the 3rd ACM International Conference on Digital libraries*, (pp. 235-243).

[25] Thom, J. A., Zobel, J., & Grima, B. (1995). *Design of indexes for structured documents* (CITRI/TR-95- 8). Melbourne, Australia: Department of Computer Science, RMIT.

[26] Trotman, A. (2005). Choosing document structure weights. *Information Processing & Management,* 41(2), 243-264.

[27] Vrajitoru, D. (2000). Large population or many generations for genetic algorithms? Implications in information retrieval. In F. Crestani & G. Pasi (Eds.), *Soft computing in information retrieval. Techniques and applications* (pp. 199-222): Physica-Verlag.

[28] Wilkinson, R. (1994). Effective retrieval of structured documents. In *Proceedings of the 17th ACM SIGIR Conference on Information Retrieval*, (pp. 311-317).

Collection-Independent Document-Centric Impacts

Vo Ngoc Anh

Department of Computer Science and Software Engineering
The University of Melbourne
Victoria 3010 Australia

vo@cs.mu.oz.au

Alistair Moffat

Department of Computer Science and Software Engineering
The University of Melbourne
Victoria 3010 Australia

alistair@cs.mu.oz.au

Abstract *An information retrieval system employs a similarity heuristic to estimate the probability that documents and queries match each other. The heuristic is usually formulated in the context of a collection, so that the relationship between each document and the collection that contains it affects the scoring used to provide the ranked set of answers in response to a query. In this paper we continue our study of document-centric similarity measures, but seek to eliminate the reliance on collection statistics in setting the document-related components of the measure. There is a direct implementation benefit of being able to do this – it means that impact-sorted inverted indexes can be built with just a single parse of the source text.*

Keywords Information Retrieval.

1 Introduction

An information retrieval system employs a similarity heuristic to estimate the probability that documents and queries match each other. The answers presented to the user are the documents that have the highest probability of being related to the query, in the hope that they are relevant to the user's information need.

The similarity heuristic is usually formulated in the context of a collection. For example, many similarity computations include a factor derived from the inverse document frequency, $1/f_t$, where f_t is the number of documents in the collection that contain the term t. This means that the relationship between each document and the collection affects the scoring used to provide the ranked set of answers, and that similarity cannot be assessed between a single document and a query.

In this paper we continue our study of document-centric similarity measures [Anh and Moffat, 2002b], but seek to eliminate the reliance on collection statistics in setting the document-related components of the measure. Reducing the reliance on collection statistics brings a direct implementation benefit – it means that impact-sorted inverted indexes can be built with just a single parse of the source text.

Proceedings of the 9th Australasian Document Computing Symposium, Melbourne, Australia, December 13, 2004.

Previously, single-parse indexing mechanisms have been used to construct document-sorted indexes, but not the impact-sorted indexes that we have been using in our work. With the similarity formulations described in this paper, we are now able to achieve the same savings at index construction time, plus retain all of the other benefits associated with the use of impact-ordered indexes, including high ranking effectiveness, and fast processing of ranked queries.

A particular benefit of making all parts of the index independent of collection statistics is that retrieval on distributed collections can more readily be accommodated, since there is no need for any collation of global parameters for the super-collection. That is, the index for any composition of indexed collections can be built by simply concatenating the various partial indexes.

The rest of the paper is organized as follows. Section 2 reviews the local reordering technique. Section 3 then describes a number of heuristics to remove the IDF component and make the document impacts dependent solely on localized information. Section 4 presents experimental results showing that the proposed techniques work well in practice.

2 Similarity computation using impacts

The *local reordering* technique, introduced by Anh and Moffat [2002b], improves both the effectiveness and the efficiency of the ranking process. In this technique, an integer *impact* between 1 and k is associated with each term t in a document d, to indicate the "strength" or "importance" of t in d. The limit k is a small integer fixed in advance, and in our experiments is typically taken to be 10 (see [Anh and Moffat, 2002b]). The impact of t in d is denoted by $\omega_{d,t}$.

For a text collection $\mathbf{D}$ containing N documents and n distinct terms, a n-dimensional vector space is formed, with each dimension associated with one of the terms. Every document $d \in \mathbf{D}$ is represented by a *document impact vector*,

$$\Omega_d = \{\omega_{d,1}, \omega_{d,2}, \ldots, \omega_{d,n}\} ,$$

and every query q by a *query impact vector*,

$$\Omega'_q = \{\omega'_{q,1}, \omega'_{q,2}, \ldots, \omega'_{q,n}\} .$$

Document impacts and query impacts might be defined differently, which is why different notation has been used for them. If a term $t \in q$ does not have a corresponding dimension in the n-space, it is ignored.

Once the impact vectors for a document d and a query q have been formed, the similarity score $S(d, q)$ between d and q is defined as

$$\begin{aligned} S(d,q) &= \Omega_d \times \Omega'_q \\ &= \textstyle\sum_t \omega_{d,t} \cdot \omega'_{q,t} . \end{aligned}$$

In practical implementations an incomplete version of this computation is often performed, with only a partial evaluation of the inner product computed, using additional heuristics that are generically referred to as *pruning* techniques. Anh and Moffat [2002a] describe several pruning methods that can be used with impact-based similarity scores.

The local impacts introduced by Anh and Moffat are *document-centric*, since the impact $\omega_{d,t}$ of a term t in a document d is defined by comparing statistics of t with other terms that appear in the same document d. The process of assigning impacts to the terms of d consists of three phases, shown in Figure 1.

In the first *parsing* phase, documents are processed by identifying the various terms that appear in them, and recording information about them, primarily $f_{d,t}$, the number of times term t appears in document d.

Then, in the *sorting* phase, the list of distinct terms of d is rearranged into decreasing order of a *primary sort key*, with ties broken by a *secondary sort key*. In this paper the primary sort key value is always $f_{d,t}$, which represents the use of the well known TF factor. One possible secondary sort key value – as in Figure 1 – is the corresponding term IDF factor, calculated as $1/f_t$, where f_t is the collection frequency of t.

In the third *mapping* phase of assigning impacts, the ordered list of terms is partitioned used some pre-defined scheme, and divided into k consecutive non-overlapping segments numbered from k down to 1. The number of elements x_i in a segment i ($i = k \ldots 1$) is

$$x_i = (B-1) \cdot B^{k-i} ,$$

where

$$B = (n_d + 1)^{1/k} ,$$

and n_d is the number of elements in the list being partitioned – in other words, the number of distinct terms in document d. At the end of the mapping phase, each term t is assigned the integer impact i corresponding to its position in the ranked list, shown as step 3 in Figure 1. The result of this step is that a small number of terms are deemed to be of high impact in the document, and a much greater number of terms are deemed to be of low impact.

Once the impacts have been computed for each term in each document, the set of inverted lists required for the collection index can be formed. This is the fourth step shown in Figure 1. Each inverted list is ordered by decreasing impact, and because there are only k possible different impact scores for each term, the inverted list can be thought of as a sequence of k (or fewer) blocks, each of which contains document numbers d in which that term has the same impact. The result is an *impact-sorted index* that allows very fast query processing.

To process a query q, the first step is to parse the query and assign a query term impact value to each distinct term t of q. Anh [2004] described a number of schemes for determining query term impacts. Here we make use of a simpler scheme which also gives good retrieval effectiveness:

1. For each $t \in q$, the *weight* of t in q is defined as

   $$w'_{q,t} = (1 + \log_e f_{q,t}) \times (\log_e(1 + f^{\max}/f_t)) ,$$

   where $f^{\max}$ is the maximum value of f_t over the collection.

2. The set of term weights is transformed to a set of integer impacts $\omega'_{q,t}$, by linear scaling so that the maximum query term impact is exactly k.

It is this scheme that is applied for all the experiments in this paper. Worth noting is that this calculation does still include f_t in an IDF factor, but that it is not required until the index has been completed and queries are being processed.

After query term impacts have been decided, the inverted lists for the query terms are retrieved, and their blocks are processed an interleaved manner so that blocks with high product $\omega_{d,t} \cdot \omega'_{q,t}$ are processed first. The combination of pruning techniques `Continue`, `TermFine`, and `BlockFine` (see Anh and Moffat [2002a] for details) is applied to suppress the contributions from index blocks with a low product $\omega_{d,t} \cdot \omega'_{q,t}$, and allow the process to execute quickly.

In addition to these processes, special treatment is given to stop-words so that they are always assigned the lowest impact of 1. For this purpose, a list of 600 stop-words is employed, taken from the file `stoplist.orig` that is publicly available at the site `http://goanna.cs.rmit.edu.au/~jz/resources/stopping.zip`. The whole scheme of defining impacts in this way is, as in our previous work, denoted `(TF,IDF)`.

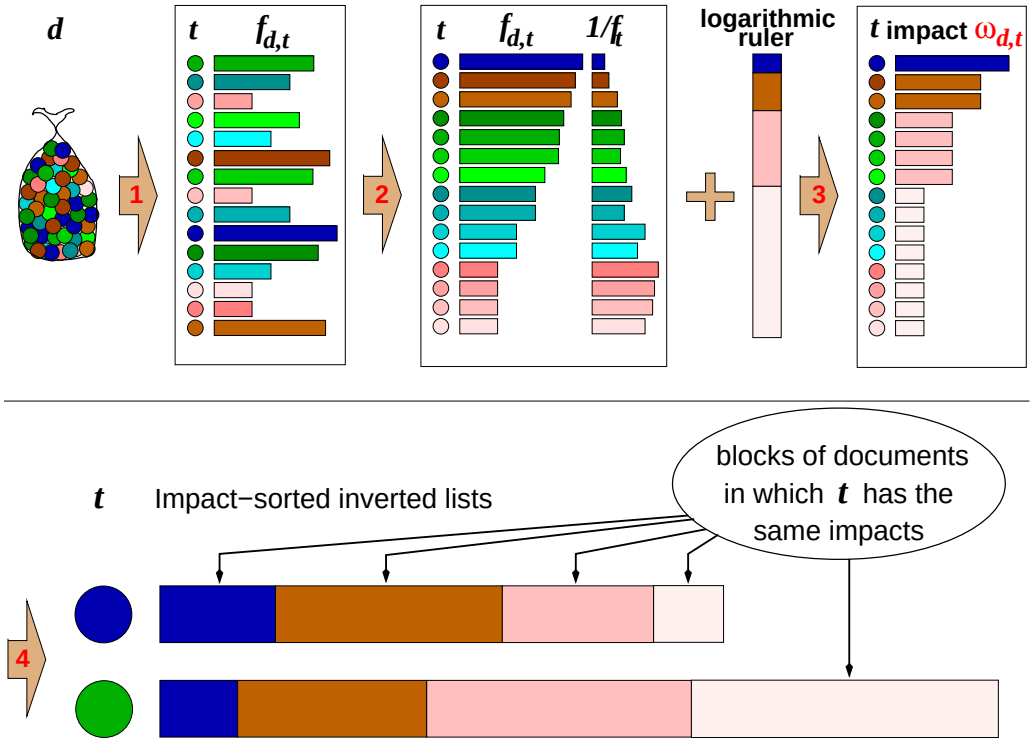

Figure 1: *Impact-sorted indexing.* Building a static-pruned impact-sorted inverted index with $k = 4$. The set of f_t values is used in step three as part of the sort key used to generate term ranks.

3 Non-IDF Impacts

One of the key factors contributing to the retrieval effectiveness of document-centric impacts is how well the sorting phase is able to arrange the terms in decreasing order of term importance. The `(TF,IDF)` technique reported above relies on the TF.IDF principle (see, for example, Salton [1989] and Witten et al. [1999]), and uses both the TF and the IDF factors as part of the sort key.

In this section we explore other heuristics for setting the document impacts. Our goal throughout is to avoid any reliance on collection-wide statistics. While this makes a partial violation of the TF.IDF principle inevitable, it is an attractive target for practical reasons, including ease of index construction.

Hence, we are interested in alternative quantities that are available while a single pass through the collection is being performed, and can stand in for IDF as an indication of a term's importance.

Independent Factors An obvious solution is to remove IDF from the computation, and replace it by a factor that is independent of not only the collection, but also the term and the document themselves. The simplest solution is sorting the term list on the decreasing order of the term frequencies alone.

The complete removal of the IDF component means that there is no secondary sorting key, which gives rise to a problem when assigning impacts to equal-frequency terms. In considering this situation, the following options are explored.

- `(TF)`: This is the simplest approach. Terms are sorted in decreasing order of their within-document frequencies. Then to deal with ties, two principles are applied. First, all terms that have the same frequency must have the same impact assigned. Second, if, according to the mapping scheme, an equal-frequency set of terms would be scattered across more than one impact segment, the average value (or the closest integer that is not less than the average) is chosen as the impact for all the terms in the set.

  This strategy means that the impact groups are not guaranteed to be as defined by the partition scheme of the mapping phase. Some groups may be larger than they should be, others smaller.

- `(TF,Random)`: This policy represents a simple and elegant way to deal with the tie problem. Distinct random numbers are generated for distinct terms and serve as the secondary key for the sorting process. In comparison with `(TF)`, the `(TF,Random)` policy does not guarantee that equal-frequency terms are assigned equal impacts. On the other hand, the number of elements in each impact group is kept precisely as was defined by the partition scheme of the mapping phase.

- (ITF): This is the dual of (TF), in that it sorts the terms in increasing (rather than decreasing) order of term frequency. It flagrantly violates the TF criterion of the TF.IDF principle, and there is no reason to believe that it will perform well. Nevertheless, inclusion of it in the experiments provides a useful litmus test – if the (ITF) method does not perform badly, then the whole notion of TF-based ordering is in doubt.

Term and Word Lengths Compared with using the IDF component as the secondary sort key, method (TF) may not provide enough discrimination to the terms, and while method (TF,Random) ensures the complete range is used, it is in a rather arbitrary way. A term attribute that can be a direct surrogate for the IDF factor might be a better option. Ideally, such an attribute would be highly correlated with IDF, while being collection-independent.

The term length (that is, number of characters in the term) is one possible candidate for this task. Terms that are used frequently (such as "sea" and "food") tend to be short, and rare terms (such as "microbiology" and "pedestrian") are normally long.

One potential issue is that in many document ranking systems, words in documents are stemmed before indexing (see, for example, Baeza-Yates and Ribeiro-Neto [1999]), and it is the stemmed words that are indexed. As a result of stemming, vocabulary terms are generally shorter than any of the words they represent.

Stemming complicates the choice of term or word length for the secondary sort key. In all the experiments conducted for this paper, a light stemming scheme has been applied. It is a variant of Porter's stemmer, but with the reduction rules condensed to cover only the regular cases of plural nouns; adverbs ending with "ly"; and verb forms ending with "s", "ed", and "ing".

Assuming some stemming policy is in place, the following options employing term lengths are considered:

- (TF,TL): The length of each stemmed term is used as the secondary sort key. The greater the length, the more important the associated term is presumed to be.

- (TF,WL): As in the case of (TF,TL), but the average unstemmed length over all appearances of words sharing the same stem is taken as the secondary sort key. Note that the average is taken locally for each document. Again, terms with longer average length are considered more important, but now words that in some variants are heavily pruned by the stemming are given extra weight.

- (TF,ITL): The inverse length of stemmed terms is used as secondary sort key. Poor performance from this option would support the idea of using term length as a surrogate for IDF.

Term Positions and Density It is likely that an ordering that better reflects our human perception of term importance can improve retrieval effectiveness, and human perception for terms in an isolated document cannot reflect any IDF component. Hence it is interesting to consider replacement of the IDF factor by a sort key that is perhaps completely uncorrelated.

One such factor is the position of the term in the document. Another factor is the degree of concentration of terms. The concentration of a certain term in a small area of text can heuristically show that this area is relevant to the term.

Based on these assumptions, three further options were explored, all based on term positions:

- (TF,ITFP): The assumption here is that important terms are likely to appear early in each document. So, the inverse position of the first appearance of terms (counting word appearances from the beginning of the document to the first appearance of this term) is used as the secondary sort key. There is one nice property of this policy – there are absolutely no ties amongst the sort key values.

- (TF,TLP): This policy is, to some extent, a dual of the previous one. Now the position of the last appearance of a term is taken as the secondary sort key. It reflects the intuition that, at least sometimes, important words reappear at the end of documents, for example, in conclusions.

- (TF,ITD): As a measure of term density, we use the inverse distance between the position of the two closest term appearances. If a term appears only once, the distance is set to the number of words in the document. If a term appears two or more times the distance is set to the number of words between its two closest appearances, with small distances deemed to be evidence of term importance. Ties are treated as in the case of the (TF) method.

Tags Many electronic documents are provided in a structured or semi-structured form incorporating SGML-like tags. Exploring this document structure might also lead to better term ordering.

A difficulty in using text structure is the great diversity of styles and markup used. A general solution for a heterogeneous collection or homogeneous collections might be hard to achieve, and careful study of a new homogeneous collection might produce better performance than can be attained by generic rules.

Figure 2 shows a typical document from the WSJ collection, which is very similar to other documents of other TREC data except for the Web data (see trec.nist.gov for descriptions of the TREC project and the TREC data).

For easy presentation, when a term appears under a tag, the tag is referred to as a "category" of the term. We can suppose that each appearance of any

```
<DOC>
     <DOCNO>
          WSJ870512-0150
     </DOCNO>
     < HL>
          Collins Foods International Sells ...
     </HL>
     <DD>
          05/12/87
     </DD>
     <SO>
          WALL STREET JOURNAL (J)
     </SO>
     <IN>
          CF TENDER OFFERS, MERGERS, ACQUISITIONS (TNM)
     </IN>
     <DATELINE>
          LOS ANGELES
     </DATELINE>
     <TEXT>
          Collins Foods International Inc. said it ...
     </TEXT>
</DOC>
```

Figure 2: *Sample* WSJ *document.* The document is an article in one of the *Wall Street Journal* editions in 1987. Most of the content part of the article has been omitted from the presentation.

term is associated with at least one category (category "DOC" in this case). In general, a term appearance is associated with a number of categories. For example, in our document model, several term appearances belong to both "DOC" and "TEXT" categories. The number of appearances of a term in a category (including the disjoint occurrences of the category) is called the term frequency in that category. The total number of word appearances in a category is called the cardinality of the category. The number of distinct categories associated with a term is called the term's diversity degree.

Various heuristics can be used with the category statistics. We explore the following simple strategies:

- `(DenseTag)`: The terms in a document are sorted in decreasing order of a complex key defined by term diversity degree and the term frequencies in the categories, where the categories are arranged in the decreasing order of their cardinality. Note that in this strategy, the primary sort key remains the within-document term frequency.

  The underlying assumptions of this strategy are: (a) the higher the diversity degree, the more valuable the term is – if it appears in the "HL" category and also in the "TEXT" category, it is more important than it would be if it appears only in one of them; and (b) the higher the cardinality, the more important the category is.

  The second assumption may not be correct for other collections with rich SGML markup, but reflects the level of markup in `WSJ`.

- `(RareTag)`: This policy is similar to `(DenseTag)`, except that the categories with low cardinality are considered as more important than the ones with higher cardinality. This policy is distinguished from all others explored in this paper in that the within-document term frequency stops being the primary sort key. Rather, it serves at the last element in the complex sort key.

- `(TF,RareTag)`: In this policy, the within-document term frequency is the primary sort key. The secondary sort key is the whole sort key which is defined for `(RareTag)`.

4 Experiments

To test the different sorting schemes, a series of experiments have been conducted. The desired outcome is an assessment of the extent to which any of the proposed heuristics are worthwhile in terms of maintaining the level of retrieval effectiveness achieved by the `(TF,IDF)` method. That is, we wish to establish whether any of the proposed methods is capable of producing retrieval effectiveness comparable with the level produced by the standard document-centric impacts which rely on collection-wide statistics.

Data The data collections and queries used for the experiments are derived from the TREC resources (see `trec.nist.gov`). The three text collections employed are (a) `WSJ` which is a set of 173,252 *Wall Street Journal* articles, supplied in Disk 1 and Disk 2 of the `TIPSTER` corpus; (b) `TREC12` which is all of the 741,856 documents on Disk 1 and Disk 2 of the `TIPSTER` corpus; and (c) `wt10g` which is a set of 1,692,096 web documents. Note that the three text collections represent three different types: homogeneous officially-printed, heterogeneous officially-printed, and web documents.

Each TREC collection is accompanied by a set of topics and relevance judgements. The topics are used to generate queries. In all of the experiments here, short queries derived from the "TITLE" field of the topics are used. A data set is then defined by the text collection name and the range of topic number. For example, the dataset `TREC12.051-200` reflects use of the collection `TREC12` and queries taken from the TREC topics numbered 051 to 200. More details of the text collections and topics can be found in the papers by Harman [1995] and Hawking [2001].

Four metrics are employed to compare the retrieval effectiveness of the different techniques: `Av.Prec`, which is the mean value (over the tested queries) of the average precision at 1,000 retrieved documents; `Prec.10`, which is the mean precision at 10 documents retrieved; `Recp.Rank`, which is the mean reciprocal rank; and `Recall`, which is the recall at 1,000 retrieved documents. More details of these effectiveness metrics, as well as their stability, can be found in Baeza-Yates and Ribeiro-Neto [1999] and Buckley and Voorhees [2000].

Retrieval Effectiveness In the first set of experiments, all of the methods were compared to the `(TF,IDF)` baseline on a query-by-query basis, over

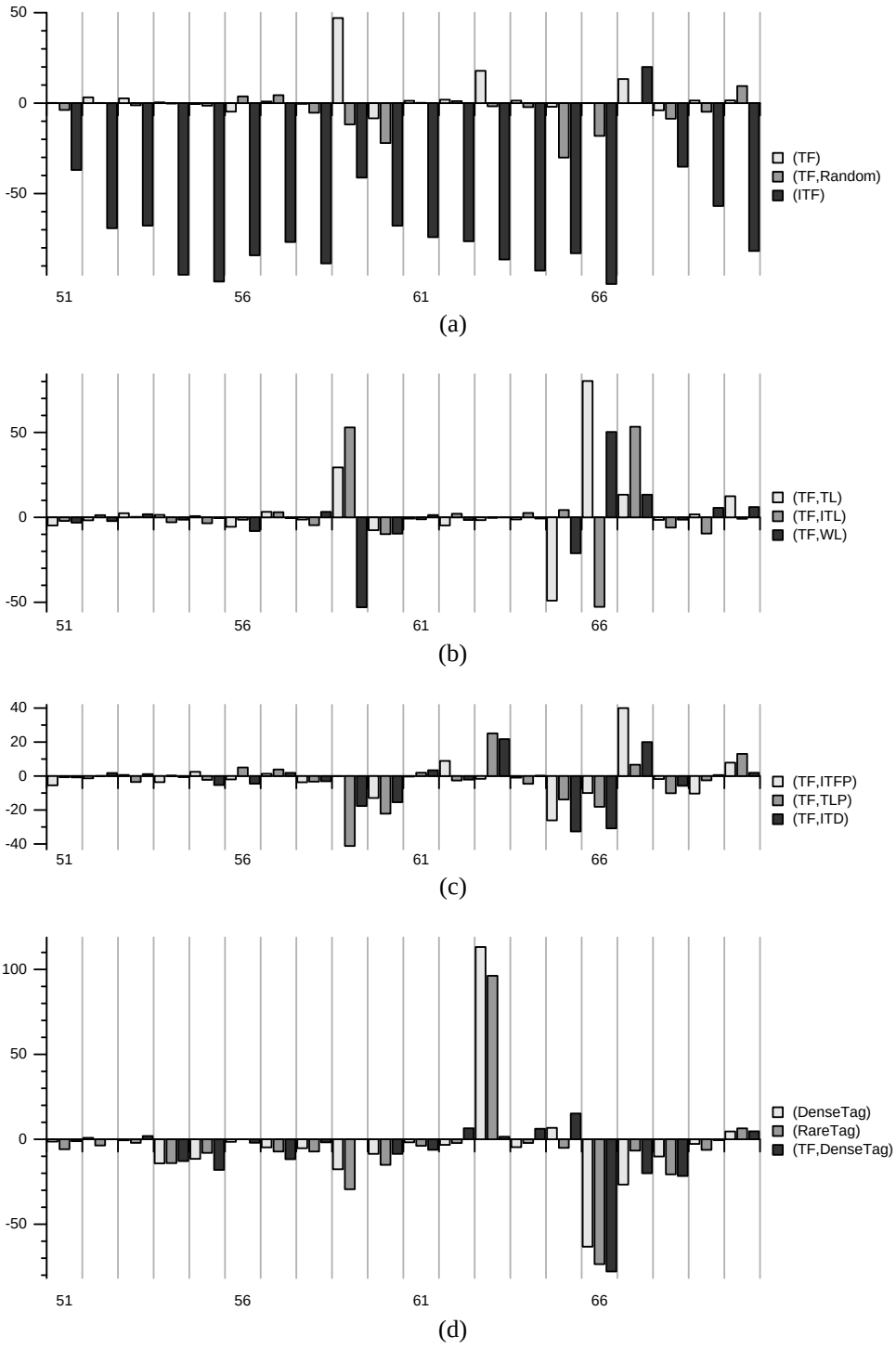

Figure 3: *Query by query relative performance.* Relative average precision for the data set `WSJ.051-070`, for variants of each of the sorting groups: (a) Independent Factors; (b) Term Lengths; (c) Term Positions; and (d) Tags. In each of the graphs, a bar for a method shows the difference between the score obtained by the method on that query and the score of the `(TF,IDF)` baseline method on that query, measured as percentage of the latter. For example, the third bar in graph (a), which has dark color, shows that for query `051`, the `(ITF)` method attains average precision 35% lower than that obtained by the baseline method.

a relatively small collection and a set of 20 queries – WSJ.051-070. The only effectiveness metric used in this preliminary testing was Av.Prec.

The outcome of the experiments is depicted in Figure 3, in which the methods are presented in their various families. The first graph in the figure shows that the method (TF) performs comparably to the baseline, and outperforms it by a considerable margin on query 059. Using the TF factor alone when determining impacts, and retaining ties rather than breaking them, is a good choice. The power of the TF factor is also demonstrated by the extremely poor performance of the (ITF) method, which reverses the (TF) sort ordering. The first graph also shows that the (TF,Random) method provides decreased effectiveness on a quarter of these twenty queries, and is somewhat inconsistent.

The bottom three graphs tell a slightly different story. For each of the methods there is at least one query in which effectiveness decreases by more than 20% relative to the baseline; and another for which effectiveness increases by 20% or more. This inconsistent behavior suggests that reliance on any single process is risky. But, as a general observation, we note that the term length heuristics are better than the term position ones, and that the heuristics based on markup tags appear to be ineffective.

Adding more queries Table 1 shows retrieval effectiveness on the same WSJ collection, but with a full set of 150 queries incorporated in the average effectiveness values. Also included as an additional reference point in Table 1 is an implementation of the pivoted cosine method [Singhal et al., 1996]. In each cluster in the table, the best results in each of the three metrics are shown in bold.

Table 1 confirms that, in an average sense, the (TF) strategy – and also several of the other approaches – compares well to the previous (TF,IDF) mechanism. Moreover, the impact-sorted mechanism outperforms the pivoted cosine approach, confirming our claims in Anh and Moffat [2002b].

Other data sets The final set of experiments measured retrieval effectiveness for all three text collections, using in each case all of the available queries; that is, WSJ.051-200, TREC12.051-200, and wt10g.451-550. The methods taken to be compared include the (TF,IDF) starting point for this investigation; (TF), which performed best in the previous experiments; (TF,WL), which represents the group of methods based on the average matching word length; (TF,ITD), which is about the best of the term position methods; and finally (DenseTag), which is a reasonable choice from the tag based methods. As an additional baseline, Figure 4 also plots a standard pivoted cosine mechanism.

Figures 4(a) and 4(b) show that all of the impact-based methods give similar effectiveness on the properly-published WSJ and TREC12 collections, regardless of whether the collections are homogeneous or heterogeneous.

Method	Effectiveness Scores		
	Av.Prec.	*Prec.10*	*Recp.Rank*
Baselines			
(Pivot)	0.2384	0.4160	0.6428
(TF,IDF)	**0.2471**	**0.4440**	**0.6572**
Independent Factors			
(TF)	**0.2459**	**0.4440**	**0.6752**
(TF,Random)	0.2399	0.4360	0.6385
(ITF)	0.0655	0.1247	0.2583
Term and Word Length			
(TF,TL)	0.2419	**0.4433**	0.6479
(TF,WL)	**0.2425**	0.4413	**0.6693**
(TF,ITL)	0.2409	0.4347	0.6416
Term Position and Density			
(TF,ITFP)	0.2432	**0.4440**	**0.6669**
(TF,TLP)	0.2393	0.4300	0.6607
(TF,ITD)	**0.2436**	0.4413	0.6509
Tags			
(DenseTag)	**0.2410**	0.4400	0.7013
(RareTag)	0.2403	**0.4493**	**0.7052**
(TF,DenseTag)	0.2331	0.4427	0.6798

Table 1: *Relative performance of different impact schemes.* Document-centric impact variants compared to two baselines on the data set WSJ.051-200. The baselines are the well-known pivoted cosine vector space measure [Singhal et al., 1996], and the document-centric impact version (TF,IDF) [Anh and Moffat, 2002b]. The maximum score attained in each group and for each effectiveness metric is shown in bold.

Figure 4(c) shows more variability. The most significant observation is that method (DenseTag) gives inferior results, when compared to the other approaches. As described earlier, (DenseTag) was designed for the traditional TREC data, and not for web documents in which the use of SGML markup is more intense.

On the other hand, Figure 4(c) shows that, except (DenseTag), all other methods (including the (TF,IDF) baseline) perform similarly for the web data set wt10g.451-550, and dominate the pivoted cosine arrangement. The (TF) method is fractionally weaker than the other three if Recp.Rank is taken as the criterion. However, this metric is often unstable, and this slight weakness may not be significant.

As an overall assessment, Figure 4 confirms the suitability of the (TF) method as a replacement for the (TF,IDF) method we previously proposed. A number of other factors can also be taken as the secondary sort key without greatly affecting average effectiveness.

5 Conclusion

Retrieval approaches based on document-centric impacts are good candidates for use in large systems, because of their simple implementation, fast execution, and compact index requirements.

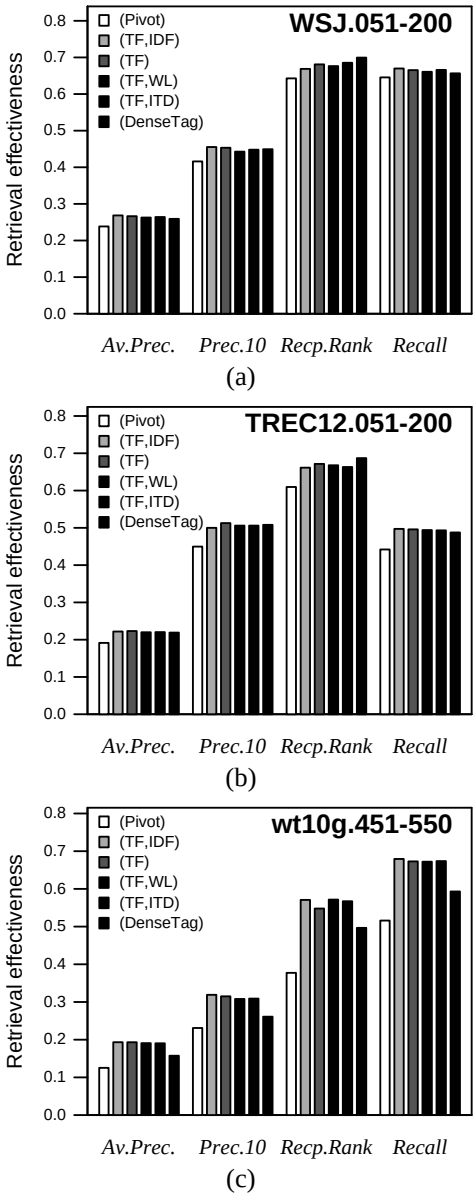

Figure 4: *Overall performance of selected document-centric impacts.* Retrieval effectiveness performance in terms of `Av.Prec`, `Prec.10`, `Recp.Rank`, and `Recall`, for (a) `WSJ.051-200`, (b) `TREC12.051-200`, and (c) `wt10g.451-500`.

This paper shows that a further desirable property can be added – reduced index construction times, achieved through the use of document impacts that are independent of the collection statistic f_t, thereby eliminating one of the two document parsing passes previously required when constructing indexes for the `(TF,IDF)` mechanism.

Collection statistics are, however, still required in the formulation of query impacts. To be truly collection independent, that reliance also needs to be circumvented. Whether it is possible to define the similarity score between a document and a query using nothing more than the document and the query themselves remains as an open – and very interesting – question.

Figure 3 also raises a further intriguing possibility – the variability of effectiveness shown in the lower three sections of the graph suggests that some form of combination of evidence may be able to outperform all of the methods we have used so far. We plan to explore that possibility as we further develop the document-centric approach.

Acknowledgements This work was supported by the Australian Research Council and by the Center for Perceptive and Intelligent Machines in Complex Environments.

References

V. Anh. *Impact-Based Document Retrieval*. PhD thesis, Department of Computer Science, The University of Melbourne, 2004.

V. Anh and A. Moffat. Impact transformation: effective and efficient web retrieval. In M. Beaulieu, R. Baeza-Yates, and S. Myaeng, editors, *Proc. 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–10, Tampere, Finland, August 2002a. ACM Press, New York.

V. Anh and A. Moffat. Vector space ranking: Can we keep it simple? In J. Kay and J. Thom, editors, *Proc. 2002 Australian Document Computing Symposium*, pages 7–12, Sydney, Australia, December 2002b.

R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, Addison Wesley, New York, 1999.

C. Buckley and E. Voorhees. Evaluating evaluation measure stability. In N. Belkin, P. Ingwersen, and M. Leong, editors, *Proc. 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 33–40, Athens, Greece, September 2000. ACM Press, New York.

D. Harman. Overview of the second Text REtrieval Conference (TREC-2). *Information Processing and Management*, 31(3):271–289, May 1995.

D. Hawking. Overview of the TREC-9 Web Track. In E. Voorhees and D. Harman, editors, *The Tenth Text REtrieval Conference (TREC 2001)*, pages 87–102, Gaithersburg, MD, November 2001. National Institute of Standards and Technology (Special Publication 500-250). URL `http://trec.nist.gov/pubs/trec10/t10_proceedings.html`.

G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison Wesley, Reading, Massachusetts, 1989.

A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In H. Frei, D. Harman, P. Schäuble, and R. Wilkinson, editors, *Proc. 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29. ACM Press, August 1996.

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

Novel Group Awareness Mechanisms for Real-time Collaborative Document Authoring

Gitesh K. Raikundalia *Hao Lan Zhang*

School of Computer Science and Mathematics
Victoria University
PO Box 14428
Melbourne City MC 8001 Australia

Gitesh.Raikundalia@vu.edu.au haolan@sci.vu.edu.au

Abstract *Group awareness has become important in improving the usability of real-time, distributed, collaborative writing systems. However, the current set of implemented awareness mechanisms is insufficient in providing extensive and comprehensive awareness in collaborative document authoring. Certainly, current mechanisms, such as telepointers and multi-user scrollbars, have contributed in providing awareness support in collaborative authoring. Yet, given the shortcomings of these mechanisms and the difficulty in providing rich interaction found in face-to-face collaboration, much more support needs to be provided for group awareness during authoring. This research extends the pool of all known awareness mechanisms (including those that have been discovered before but have yet to be implemented).*

This research discovered several awareness mechanisms not found and reported elsewhere, through conducting usability experiments with a real-time cooperative editor. This paper covers three of the mechanisms—Structure-based Multi-page View, Point Jumping Mechanism and User Info List—*discovered from the experiments. The paper also provides quantitative results supporting implementation of such mechanisms.*

Keywords Group awareness, awareness mechanisms, real-time collaborative document authoring.

1 Introduction

Real-time, distributed collaborative writing systems (RDCWS) allow distributed authors to work on documents at the same time. Examples of these systems include GROVE [1], SASSE [2] and ShrEdit [3]. In certain circumstances, RDCWS are very useful tools for a group that must carry out tasks on a document simultaneously. An example of the use of a RDCWS is in synchronous composition of essays. Collaborative essays may be used in teaching, such as in learning about negotiation of meaning [4].

However, in a workplace situation, a RDCWS may not necessarily be used to write an entire document in one sitting. Participants may use email or workflow to write parts of a document in an asynchronous manner, whilst writing other parts together synchronously. Participants may have an initial meeting to agree and work on the structure and content of the document together at the same time, leaving participants to finish the document separately at different times. On the other hand, medical researcher colleagues of this paper's first author work on a document at different times, only to come together *at the end of the process* to finalise the document. These medical researchers find greater efficiency in finalising the document together at the same time rather attempting to finalise it separately at different times.

Group awareness (GA) is an important feature enhancing the usability of RDCWS. GA provides users with sufficient knowledge about the status of a document itself and all activities other users perform upon the document. GA plays an essential and integral role in cooperative work by simplifying communication, supporting coordination [1], assisting "anticipation" [5] and supporting "convention" [6].

In face-to-face interaction, it is naturally easy for people to know who is present, what are others' responsibilities and what others are doing. When group members are geographically dispersed, supporting spontaneous interaction is more difficult. To enrich GA in real-time collaborative authoring, various awareness mechanisms such as telepointers [7], radar views [8] and multi-user scrollbars [2] have been used. Unfortunately, these mechanisms were implemented in editors without prior research on what awareness information users actually need or desire when writing collaboratively. Consequently, some awareness mechanisms are implemented in an ad-hoc manner. Although these approaches have found some relevant awareness mechanisms, some other possible

Proceedings of the 9th Australasian Document Computing Symposium, Melbourne, Australia, December 13, 2004.

mechanisms are easily overlooked or require excessive experimentation for their discovery. For instance, the emergence of a radar view reported in [8] is the result of a long process of improvement and enhancement, which could have been naturally avoided if the process producing the radar view initiated from end-users. Furthermore, previous research does not provide a potentially nearly full set of comprehensive awareness mechanisms; designers are often left without any clear sense of which awareness mechanisms should be implemented in RDCWS to support GA.

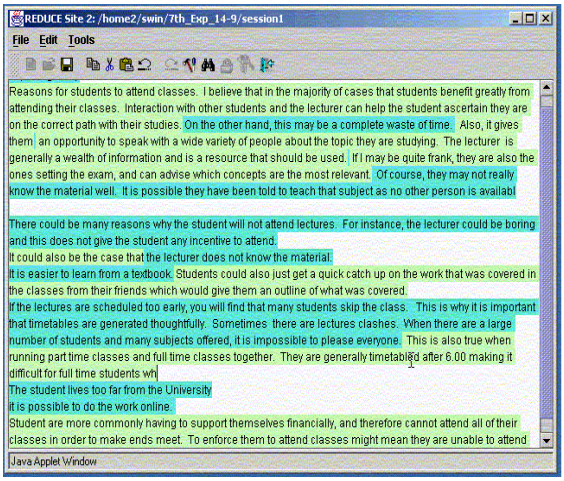

Figure 1: REDUCE collaborative editor

To produce a usable system, a designer must be directed by the principles of user-centred design [2]. Therefore, this research involves study of users' needs and identifies highly suitable candidates for mechanisms in providing GA. This involves conducting laboratory-based, usability experiments with REDUCE—Real-time Distributed Unconstrained Cooperative Editor [9]. REDUCE is a real-time and multi-user text editor allowing geographically distributed people to interact synchronously upon the same document without constraints (Figure 1). This research ensures that awareness mechanisms are developed in the light of users' needs.

New awareness mechanisms have been discovered by this research. These mechanisms have been proposed totally by end-users and have not yet been implemented. Hence, the mechanisms presented in this paper are mock-ups and still require implementation, evaluation and improvement before final implementation in any RDCWS. The main emphasis of this paper is to describe the following mechanisms found in this research:

- *Structure-based Multi-page Preview*
- *Point Jumping Mechanism*
- *User Info List*

This paper begins with coverage of major existing awareness mechanisms. It then describes the research methodology, the usability experiments conducted in this research, and finally, the mechanisms discovered.

2 Related work

The growing interest in GA results from the fact that awareness support has increasingly been identified as a crucial part of successful collaboration [3, 5]. Perceiving and understanding the activities and intentions of other members of a group is a basic requirement for human interaction. As in any groupware system, GA is a major concern in real-time collaborative authoring. Previous research has found a number of different awareness mechanisms supporting collaborative authoring.

Telepointers [7] are a mechanism allowing multiple cursors of users to be shown within the document. Telepointers usefully show all the sections of a document all users are working on in parallel. However, telepointers are only capable of providing other users' cursor positions when they are located in the same portion in the document. Consequently, if telepointers are removed from a local user's view, it is difficult for this user to gauge the location at which a remote user is working and how active is that user.

Radar views are miniaturisation techniques that provide an overall view of a document to show where all users are working on a document. Radar views have been proven to be useful in maintaining GA [8]. The major problem with a miniaturisation technique is that of limited scalability; a miniature view of a very large data space contains too little detail to be useful.

To overcome a radar view's limitations, especially to bridge the gap between local details and the global structure of a document, a fisheye view can be used [10]. A fisheye view is a distortion-oriented view that presents a single view displaying both local detail and global context on a continuous "surface". A fisheye view provides a seamless and smooth transition between local details and the global structure. When each user has a focal point, the location of other users and the details of their actions performed upon the workspace are provided. Remote users' focal points can be out of a local user's view when a document is too large to fit in the local user's view. The local user apparently loses track of the remote users' whereabouts. Also, when more than two enlarged areas overlap they hide one another, so part of the document appears lost.

Multi-user scrollbars allow a user to see the different parts of a document worked on by other users via scrolling within the document. In the literature, there are two different variations of multi-user scrollbars: version 1 in [2] and version 2 in [8]. In version 1, each remote scrollbar is located in a different vertical region; however, in version 2, all remote scrollbars are located in the same vertical region. The major problem of multi-user scrollbars occurs when a large number of users are working in

the workspace. In version 1, the display of a large number of remote users' scrollbars causes space constraints; it forces the area of the document portion viewed to be smaller. In the case of version 2, when views of more than two users intersect, it is hard to know exactly the location of remote users because many remote scrollbars overlap one another.

The Split Window View [11] is a mechanism that allows the user to view both working and viewing areas of other members of the group. In some cases, a user's working and viewing areas can be different as the user may be working on a particular part of the document, yet be looking somewhere else in the same document. Therefore, this mechanism allows a user to see both of these areas of all other users. When any of these other users' working areas are exactly the same as their viewing areas, this one area is only shown. This mechanism has not yet been implemented, but it is clear what are the major drawbacks are of this mechanism. There are space constraints in presenting possibly many working and viewing areas at the same time within the mechanism. The more areas that have to be displayed, the more screen space is required, which may mean making the areas smaller to fit them onto the screen. Also, when a large number of areas are shown, a low-fidelity display is used, meaning it is more difficult to read the text within an area.

The purpose of the Modification Director [11] is to show to a user that another user is modifying their work. The mechanism is helpful in conveying who the other user is that is altering their work and how they are altering it. The mechanism provides a document-related form of GA. It works based upon a flashing colour icon to indicate another user is modifying text, and clicking on the icon pops up a read-only window to show the modified text. An issue that has to be resolved in this mechanism's implementation is how to show multiple areas of a user's text being modified by the same user.

The Dynamic Task List [12] is a task-based technique for supporting document-related awareness. The mechanism provides a frequently-updated list of group members' tasks. A user is able to comment on other users' tasks and the author responsible for a task is informed of which other users are viewing their part of the document. This mechanism may prove to be more difficult to implement.

3 Research methodology

To produce a usable editor for supporting collaborative authoring, much research has exploited the user-centred approach in the study of how people write together [2]. Similarly, to provide usable *awareness mechanisms* for real-time collaborative editors, a designer must be directed by such principles. Therefore, this research conducted laboratory-based experiments with REDUCE. At present, REDUCE supports almost no GA features; hence, conducting experiments with REDUCE allows determination of awareness information users really need to perform a collaborative authoring task effectively and efficiently. The results of the experiments lead to the discovery of new awareness mechanisms that are potentially capable of supporting GA in real-time collaborative authoring. These mechanisms are discovered with a view to implementing them so that further usability experiments with REDUCE can be used to evaluate the mechanisms. That future set of experiments determines the effectiveness of the mechanisms through real-world use by end-users.

The usability experiments were carried out in the Swinburne Usability Laboratory of Swinburne University of Technology in Melbourne, Australia, in April 2004. The experiments involved twelve pairs of subjects working on three writing tasks, including *creative writing* (CW) (e.g., writing short essays from scratch), *document preparation* (DP) (e.g., writing a manual on REDUCE) and *brainstorming* (BS) (e.g., generating ideas). This research used these three categories for two main reasons. First, these

		Experimental sessions											
		E1	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**E8**	**E9**	**E10**	**E11**	**E12**
Verbalisation first	**CW**	T1, T2									T1, T2		
	DP							T5, T6					T5, T6
	BS									T3, T4		T3, T4	
Silence first	**CW**				T1, T2		T1, T2						
	DP			T5, T6					T5, T6				
	BS		T3, T4			T3, T4							

Table 1: Experimental sessions with REDUCE

categories represent a wide range of collaborative document authoring tasks. Second, the categories require different styles of collaboration. The types of awareness mechanisms that are needed in different contexts of collaborative authoring are found by using these varied tasks. The twelve pairs were allocated to perform the three tasks as such: 4 pairs worked on CW, 4 pairs worked on DP and 4 pairs worked on BS. The actual tasks used in experiments are shown in the Appendix. Table 1 shows the tasks used by each pair in the different sessions. For instance, in session E1, the first task given to the E1 pair is task T1 (see the "Experimental tasks" sub-section of the Appendix) of verbalisation, that is, communication via telephone. The second task given to this E1 pair is task T2 where there is silence during collaboration.

Subjects worked in pairs where each member of a pair was located in one of two separate subject rooms. A profile of the computer skills of subjects is shown in the Appendix. Subjects could not see each other from their rooms as is the case in distributed collaboration. A research assistant observed each pair from an observation room. Each pair participated in a two-and-a-half hour session that included:

- *Training in REDUCE* (30 minutes).
- *Experiment* (1 hour): Subjects worked in pairs to work on one task with verbal communication (verbalisation) for thirty minutes and on another task without verbal communication (silence) for thirty minutes. Conducting the experiments with and without support of verbal communication allowed identification of problems users had and the workarounds users resorted to when verbalisation was absent.
- *Questionnaire and interview* (1 hour): Subjects filled in a questionnaire, which included nineteen six-point scale (closed-ended) questions and thirteen open-ended questions. The closed-ended questions were questions asking subjects if they believed certain types of awareness were or were not important in collaborative authoring. These questions were analysed to provide results shown in histograms in this paper. The open-ended questions sought from subjects mechanisms they would appreciate being available for supporting GA. *It is from the open-ended questions that the proposed mechanisms of this paper were discovered*. The questionnaire is shown in the Appendix. Each subject filled in a questionnaire during an interview held by the research assistant where they could clarify the mechanisms they desired. Subjects drew onto the questionnaire their ideas of mechanisms they believed were useful to them in providing group awareness. Interviews were recorded onto audiotape for verification of subjects' responses during data analysis.

The next section describes three new awareness mechanisms identified in this research: *Structure-based Multi-page Preview*, *Point Jumping Mechanism* and *User Info List*.

4 Awareness mechanisms

4.1 Structure-based Multi-page View

A mechanism was suggested during interviews that could be used for supporting awareness of knowing the parts of the document on which other users are currently working (see Appendix). This mechanism, which the authors have named *Structure-based Multi-page View* (SMV), allows an overall view of the document. The SMVis shown in Figure 2.

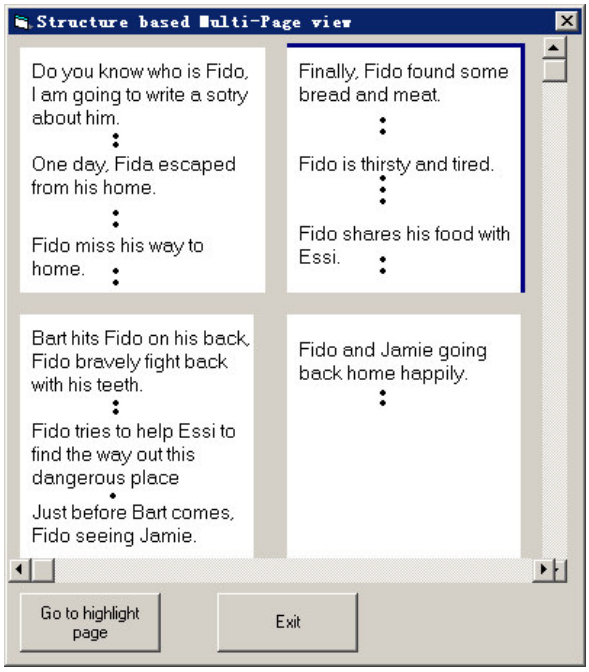

Figure 2: Viewing other's contributions with SMV

The mechanism shows how four users (there is one area in SMV for each user) are working on different pages of the same document. The user can click on an area to highlight that area (a border is shown around the area that has been selected). Then, the user can click on the "Go to highlighted page" button to be taken to that page of the document in the RDCWS. An amendment that the authors make to this mechanism, that was not thought of by the subject suggesting this mechanism, is to add details of which user is working on which page of the document. The users and their working areas are not indicated in the subject's suggested GUI in Figure 2.

The closed-ended question from the Questionnaire (see Appendix) that corresponds to the SMV is Knowing the parts of a document on which other users are currently working. Figure 3 shows the distribution of responses to this question from the subjects. 45% of subjects (almost half of them) found that a mechanism informing them of the parts of a

document on which others currently work to be a highly relevant mechanism. One third of subjects (33%) found such a mechanism to be "fairly important". Thus 78% of subjects believed that such a mechanism is useful in supporting them in collaborative authoring. The remaining one-fifth of subjects did not provide responses favourable for the existence of such a mechanism. These results indicate that designing and implementing a mechanism such as the SMV is worthwhile to support users. It must be noted that not all users are expected to use any single awareness mechanism during an authoring session. Each mechanism provides one specific type of awareness support, and is available to users if they wish to use it during an authoring session.

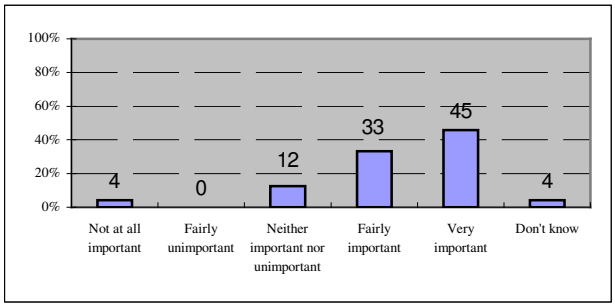

Figure 3: Knowing the parts of a document on which other users are currently working

The experimental subject's suggestion of this awareness mechanism has clearly been influenced by the Print Preview feature found commonly in software nowadays (for instance, in word processors, Web browsers, etc.). The subject that suggested this mechanism has thought that a familiar single-user GUI would be helpful in understanding who is working on which parts of the document. Suggesting a mechanism based on familiarity with another known GUI is an unsuprising result as users desire GUIs and functionality that make sense to them and are easy to learn and use. Indeed, the mechanisms of telepointers and multi-user scrollbars, which have been researched a number of years ago and are well-known mechanisms, have not been total innovations in that single-user pointers and software using scrollbars preceded the research on telepointers and multi-user scrollbars. In fact, familiarity with pointers and scrollbars has assisted the acceptance of these two awareness mechanisms and made them easier to learn.

4.2 Point Jumping Mechanism

The Point Jumping Mechanism (PJ) is another awareness element enables a user know where other users are working in the document. Figure 4 shows where a given user is looking at, at the moment, in the document. PJ captures the same content as shown in REDUCE (including the different background colours for different users). Assume the user is Jennie. Jennie wishes to see where Kana is currently working in the document. PJ will allow Jennie to go instantaneously to the point in the RDCWS where Kana is working. Jennie will then highlight Kana in the list of users in the top right-hand corner of the PJ GUI. Jennie will click on the Jump to User" button.

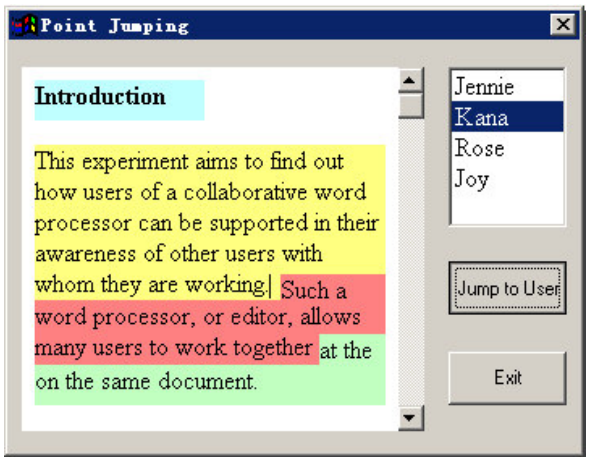

Figure 4: The PJ *before* Jennie jumps to Kana's working area

Figure 5 shows the result carrying out these steps. Jennie has been taken instantaneously to where Kana is working in REDUCE (note the vertical cursor at the bottom of Figure 5). It may be noted that the jump has been made within the awareness mechanism and not in REDUCE.

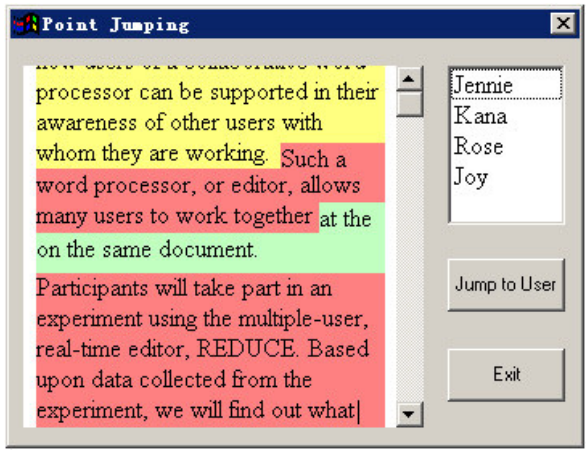

Figure 5: The PJ *after* Jennie jumps to Kana's working area

Since PJ also handles the awareness of where other users are working like the SMV, the support for PJ is also drawn from Figure 3. Thus, PJ is an alternative mechanism that also needs to be implemented and tested in real-world authoring to determine its effectiveness in supporting authoring.

4.3 User Info List

A mechanism was suggested to provide details about other users. This mechanism, the *User Info List* (UIL), is shown in Figure 6. The left-hand part of the GUI shows the contents of the RDCWS, whilst the right-hand part shows a panel with various details (User

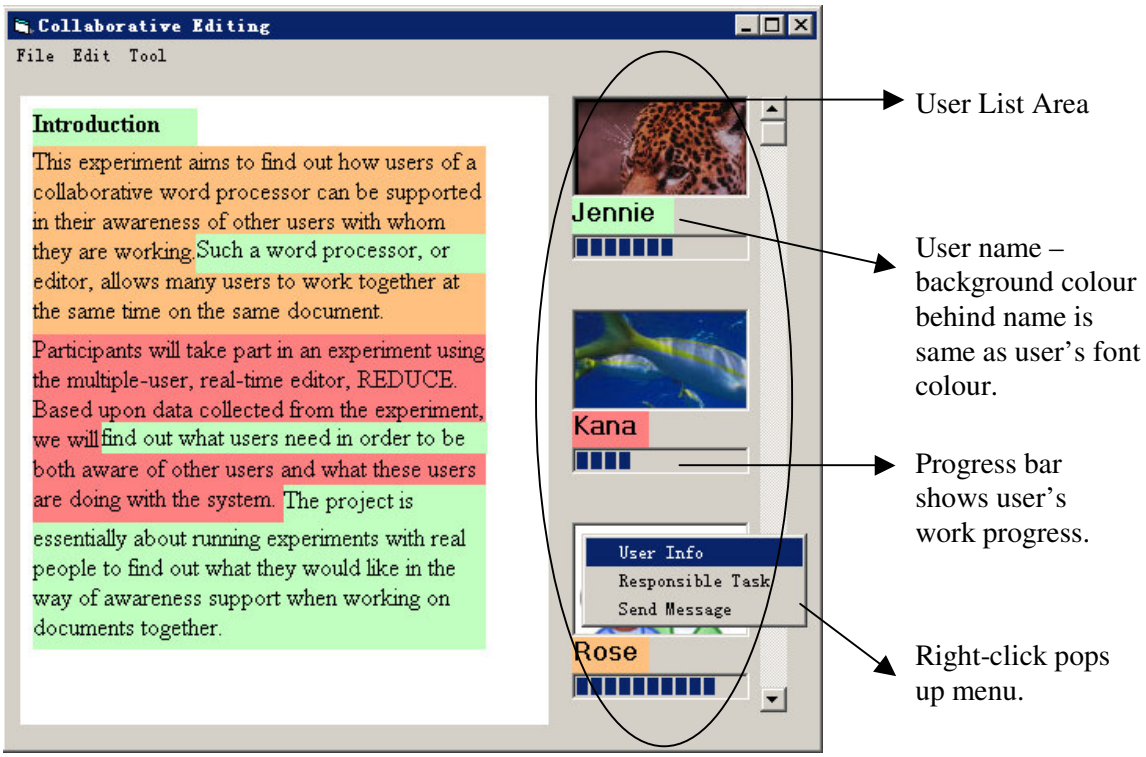

Figure 6: The User Info List and display of user information

List Area). The User List Area contains user's photographs. Below each area is the user's name on a background colour that is also their background colour in the RDCWS. Below the name is a progress bar that reflects how much of the assigned work of a user has been completed so far.

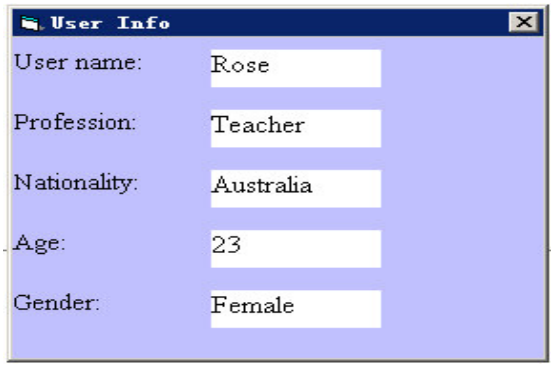

Figure 7: GUI providing awareness of who are the other users

Right-clicking on a user's photograph will pop up a menu as shown in Figure 6. This menu allows a user to either find out details about another user (selecting the "User Info" option will pop up the window shown in Figure 7), find information on the tasks the user is responsible for or to communicate a message to the user. The subject suggesting this UIL mechanism unfortunately did not provide an idea of what the GUIs looked like for determining others' tasks or sending messages.

This mechanism provides knowledge of how a user is progressing in their contribution to the document compared to the other users. A user can look at the progress bars of users and compare the completion of the individual contributions by comparing the number of blue notches in the progress bars. A closed-ended question asked of the subjects deals with this form of awareness. This is question number 12 in the Questionnaire, and its distribution of responses is shown in Figure 8. The distribution shows that 45% believe there is high or reasonable importance in having this knowledge. Thus in a collaborative authoring session, half of the subjects could be expected to use, to varying degrees, the UIL.

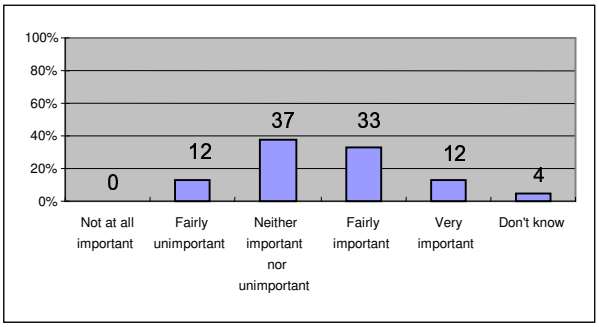

Figure 8: Knowing to what extent you have completed your work compared to the extent other users have completed their work

Another form of awareness is provided by this same mechanism. This is the awareness of Knowing the tasks for which other users are responsible (closed-ended question number 2 in the Questionnaire). In whatever user interface form this mechanism would take, it would be expected to be

used tremendously according to the results in Figure 9. Figure 9 shows that just over half the subjects wanted this form of awareness, with almost a third of them finding it reasonably important. In other words, being knowledgeable of what other users are going to contribute to the document and how they are going to do this, is overall important to subjects in being able to author a document together successfully. More than 80% of subjects find it of high or reasonable importance to be aware of others' responsibilities over a document.

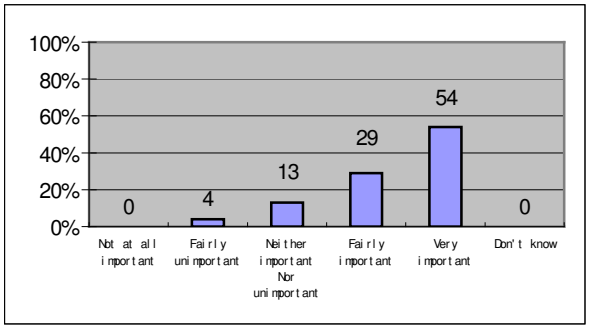

Figure 9: Knowing the tasks for which other users are responsible

5 Conclusion and future work

This paper has presented the three recently-discovered GA mechanisms of *Structure-based Multi-page Preview*, *Point Jumping Mechanism* and *User Info List* for collaborative document authoring. A series of experimental sessions in a usability laboratory—involving creative writing, (technical) document preparation and brainstorming tasks—was carried out with twelve pairs of subjects. Through usage of the REDUCE editor followed by interviews with experimental subjects, these mechanisms were discovered as proposals from these subjects. Mockups of these mechanisms and their functionality were explained in the paper. Analysis of closed-ended questions provided understanding of how many users did or did not favour different types of awareness support. These results supported the mechanisms presented in this paper.

However, the mechanisms need to be implemented and experiments with these implementations need to be carried out. Experiments are needed to determine in which cases the mechanisms are effective in assisting collaborative authoring. Experiments would involve subjects carrying out tasks with the mechanisms and being interviewed to determine how useful the mechanisms were in carrying out the tasks.

Acknowledgements This project has been funded by Victoria University Discovery Research Grant Scheme 2004. Many thanks to Assoc. Prof. Yun Yang and Minh Tran of Swinburne University of Technology and to John Craick, Manager of Swinburne Usability Laboratory.

References

[1] C. Ellis, S. Gibbs and G. Rein. Groupware: some issues and experiences. *Communications of the ACM,* Volume 34, Number 1, pages 39–58, 1991.

[2] R. Baecker, D. Nastos, I. Posner and K. Mawby. The user-centred iterative design of collaborative writing software. In *InterCHI'93*, pages 399–405 Amsterdam, 24 – 29 April 1993.

[3] P. Dourish and V. Bellotti. Awareness and coordination in shared workspaces. In *1992 ACM Conference on Computer Supported Cooperative Work*, pages 107–114, Toronto, Canada, November 1992.

[4] L. Irvin. Exercises: learning writing with computers through sharing texts, http://www.accd.edu/sac/english/lirvin/Exercises NCTE.htm, Accessed 3 June 2004.

[5] C. Gutwin and S. Greenberg. A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work*, Volume 11, Number 3-4, pages 411–446, 2002.

[6] J. Grudin. Groupware and social dynamics: eight challenges for developers. *Communications of the ACM*, Volume 37, Number 1, pages 92–105, 1994.

[7] S. Greenberg, C. Gutwin and M. Roseman. Semantic telepointers for groupware, In *Sixth Australian Conference on Computer–Human Interaction*, pages 54–61, Hamilton, NZ, November 1996.

[8] C. Gutwin, M. Roseman and S. Greenberg. A usability study of awareness widgets in a shared workspace groupware system. In *1996 ACM Computer-Supported Cooperative Work*, pages 258–267, Boston, USA, November 1996.

[9] Y. Yang, C. Sun, Y. Zhang and X. Jia. Real-time cooperative editing on the Internet. *IEEE* Internet *Computing*, Volume 4, Number 1, pages 18–25, 2000.

[10] S. Greenberg, C. Gutwin and A. Cockburn. Awareness through fisheye views in relaxed-WYSIWIS groupware. In *Graphics Interface 1996*, pages 28–38, Toronto, Canada, 1996.

[11] Tran, M., Raikundalia, G. and Yang, Y. Split Window View and Modification Director: innovative awareness mechanisms in real-time collaborative writing, In *Human Factors 2002*, Melbourne, 25 - 27 November, 2002.

[12] Tran, M., Raikundalia, G. and Yang, Y. Usability experiments for determining document-related awareness in real-time cooperative editing, In *ADCS '01*, pages 95-98, Coffs Harbour, 2001.

Appendix

Experimental tasks

Creative Writing
T1: "Fido is a dog living in Melbourne and owned by a boy, Jamie. Write a fictional story about the adventures of Fido."
T2: "Write a fictional story about the various events that occur in a sports team playing in a particular match. For instance, a soccer team or a cricket team or a basketball team, etc. playing a particular match."

Brainstorming
T3: "Stress affects people in modern life. There are clearly many different ways of escaping the stress and difficulties of modern life. Write down and explain various ways of reducing stress."
T4: "Write down different problems and difficulties that you feel occur when being taught in an educational setting (e.g., university lecture, workshop carried out in a company, etc.)"

Document Preparation
T5: "Write a research paper on an agreed topic with the other participant."
T6: "Write a manual or guide about REDUCE. This manual/guide must instruct and teach the reader how to use REDUCE."

Questionnaire

Six-point scale closed-ended questions

1. Knowing who is in the workspace
2. Knowing the tasks for which other users are responsible
3. Knowing how much time has elapsed since other users have used REDUCE
4. Knowing where other users are physically located
5. Knowing how long other users have been in the workspace
6. Being able to view the list of past actions carried out by a specific user
7. Knowing the parts of a document on which other users are currently working
8. Knowing the parts of a document at which other users are currently looking
9. Knowing what actions other users are going to take in the future
10. Knowing what actions other users are currently taking
11. Seeing the position of other users' cursors on the screen
12. Knowing to what extent you have completed your work compared to the extent others have completed their work
13. Knowing to what extent a portion of a document has been completed
14. Knowing if other users can know what you have been doing
15. Being able to comment on what other users have done
16. Knowing if other users are satisfied with what you have done
17. Having voice communication
18. Having video communication
19. In the case of nonverbal communication, having a communication tool that supports communication between users.

Open-ended questions

How would you expect REDUCE to show you who is in the workspace?
How would you expect REDUCE to show you which tasks other users are responsible for?
How would you expect REDUCE to show you how long other users have been in the workspace?
How would you expect REDUCE to show you the list of past actions carried out by a specific user?
How would you expect REDUCE to show you which parts of a document other users are currently working on?
How would you expect REDUCE to show you what actions other users are currently taking?
How would you expect REDUCE to show you what actions other users are going to take in the future?
How would you expect REDUCE to show you where other users are physically located?
How would you expect REDUCE to show you which parts of a document other users are currently looking at?
How would you expect REDUCE to show you to what extent a portion of a document has been completed?
How would you expect REDUCE to show you an overall view of the document?
How would you expect REDUCE to show you to what extent you have completed your work compared to the extent other users have completed their work?
What communication tools do you think can be used to support communication between users?

User profile

User Level	Number of subjects
Strong computer knowledge (Expert level)	4
Good computer knowledge (Advanced level)	3
Average computer knowledge (Average level)	9
Modest computer knowledge (Less-than-average level)	6
Non-computer user	2

Mechanism discovery

SMV

Twelve subjects suggested interest in a mechanism that provides an overall view of the document. A representative response by a subject, written in their own copy of the questionnaire, was:

"I want to see many pages in one screen."

This subject also drew a diagram of the mechanism. The authors' mockup of this subject's diagram is shown in Figure 2 of this paper.

PJ

Four subjects suggested this mechanism in different ways. One subject drew a mechanism that resembles the mockups shown in Figure 4 and Figure 5.

Is CORI Effective for Collection Selection? An Exploration of Parameters, Queries, and Data

Daryl D'Souza Justin Zobel James A. Thom
School of Computer Science & Information Technology
RMIT University, Melbourne 3001, Australia
{djds,jz,jat}@cs.rmit.edu.au

Abstract *In distributed information retrieval, a wide range of techniques have been proposed for choosing collections to interrogate. Many of these collection-selection techniques are based on ranking the lexicons; of these, arguably the best known is the CORI collection ranking metric, which includes several parameters that, in principle, should be tuned for different data sets. However, parameters chosen in early work on CORI have been used without alteration in almost all subsequent work, despite drastic differences in the data collections. We have explored the behaviour of CORI for a range of data sets and parameter values. It appears that parameters cannot reliably be chosen for CORI: not only do the optimal choices vary between data sets, but they also vary between query types and, indeed, vary wildly within query sets. Coupled with the observation that even CORI with optimal parameters is usually less effective than other methods, we conclude that the use of CORI as a benchmark collection selection method is inappropriate.*

Keywords Lexicon indexing, distributed retrieval, information retrieval.

1 Introduction

In distributed information retrieval, the search process involves passing the query to each of a set of search servers, then collating their responses. Each such server indexes a collection of documents. The cost of search can be reduced by only passing the query to a limited number of servers, giving rise to the *collection selection* problem: identification of those collections most likely to contain answers.

A method of collection selection that has been widely described in the research literature is to use information about each collection's lexicon (Callan, Lu & Croft 1995, Craswell, Bailey & Hawking 2000, French, Powell, Callan, Viles, Emmitt, Prey & Mou 1999, Gravano & Garcia-Molína 1995, Hawking & Thistlewaite 1999, Meng, Yu & Liu 2002, Yuwono & Lee 1997, Zobel 1997). In a common approach, the central service maintains a copy of the complete lexicon of each collection, which should be much smaller than indexes for the collections, or the text of the collections themselves. These lexicons can then be cheaply compared to the query to establish which are the most promising.

Proceedings of the 9th Australasian Document Computing Symposium, Melbourne, Australia, December 13, 2004.

A range of query-to-lexicon similarity measures have been proposed. Of these, arguably the best known is CORI, first described by Callan et al. (1995), and used in their subsequent work (Callan 2000, French et al. 1999, Larkey, Connell & Callan 2000). It has been used in many papers as a standard (Abbaci, Savoy & Beigbeder 2002, Callan & M.Connell 2001, Callan 2000, Lu & McKinley 1999, Powell, French, Callan, Connell & Viles 2000, Rasolofo, Abbaci & Savoy 2001, Si & Callan 2002), and even some recent papers report experiments showing it to be an effective collection-selection metric (Conrad, Guo, Jackson & Meziou 2002, French et al. 1999, Larkey et al. 2000, Si & Callan 2003). While this verdict is not universal, this continued investigation of CORI demonstrates that many researchers view it as a key approach.

In recent work, we were surprised by the relatively poor behaviour of CORI on some datasets, in the worst cases achieving only around half the effectiveness of other methods (D'Souza, Thom & Zobel 2004). We investigated the results to identify the issues—was our implementation at fault, for example? What we discovered was rather more serious. CORI is derived from a theoretical argument, giving a formulation in which key parameters are undetermined. In an early CORI paper, values for these parameters were chosen by exploration on a particular collection (Callan et al. 1995). In all subsequent papers the same parameter values have been used, with no reported attempt to reinvestigate them.

We searched for the best values for the CORI parameters over several sets of collections and queries, and found that in many cases the standard parameters give significantly inferior results than those observed with other parameter choices. For comparison, we show results on the same collections achieved by methods discussed by Zobel (1997); in 13 of 21 cases, the simple (and unparameterised) Inner product is superior to standard CORI, while in 15 of 21 cases the CORI is worse than Highsim, the other method tested. In 9 of the 21 cases, standard CORI is worse than the baseline of sized-based ranking (SBR) of simply selecting the largest collections.

Not only does the optimal choice of CORI parameter values vary dramatically from test set to test set, but it varies dramatically between query sets on the same data and between queries of the same type on the same data. Moreover, other issues are suggested by our investigation. While CORI has performed well in some other experiments, it appears that is because the test data con-

sisted of a small number of similarly-sized collections of unrelated documents—an environment that does not allow much discrimination between methods. Taking these problems together, it is far from clear that CORI is a wise choice of collection-selection metric.

2 CORI

In the collection-selection problem, it is assumed that the set C of collections has N members, and collection $c \in C$ contains f_c documents. Many scoring mechanisms used for selecting collections are rather like conventional similarity measurement, in that each collection is treated as a bag of words, just as a document is a bag of words in document retrieval. Therefore similar statistics are used, in particular $f_{c,t}$, the number of documents containing term t in collection c, and f_t, the number of collections containing t. Collections that score the highest are assumed to be the most likely to contain documents that are relevant to the query.

CORI is based on Bayesian inference networks. In CORI the similarities between a user query and a set of known document collections is computed, in order to rank the collections. The query is then submitted to each selected (highly ranked) collection to retrieve its set of top ranked documents; these separate document sets are then merged.

The similarity of a query to a given collection is the sum of the belief probabilities of the query terms appearing in the collection. The CORI similarity between a query q and collection c can be computed as

$$\mathit{CORI}(q,c) = \frac{\sum_{t \in q \& c} (d_b + (1-d_b) \cdot T_{c,t} \cdot I_{c,t})}{|q|}$$

where d_b, the minimum belief component, is set to 0.4, $T_{c,t}$ is the weight of the term in the collection, $I_{c,t}$ is the inverse collection frequency, and $|q|$ is the number of distinct terms in the query. The value $|q|$ can be ignored as it is constant for a given query. The inverse collection frequency $I_{c,t}$ can be computed as

$$I_{c,t} = \frac{\log((N+0.5)/f_t)}{\log(N+1.0)} \tag{1}$$

In an early version of CORI (Callan et al. 1995), the weight $T_{c,t}$ is computed as

$$T_{c,t} = d_t + (1-d_t) \cdot \frac{\log(f_{c,t}+0.5)}{\log(max_c+1.0)}$$

where d_t is the minimum term frequency component, set to 0.4 in earlier experiments (Allan, Ballesteros, Callan, Croft & Lu 1995, Callan et al. 1995), and max_c is the number of documents containing the most frequent term in collection c.

A suggested improvement to $T_{c,t}$ is to scale $f_{c,t}$ by adding a constant K (Callan et al. 1995). When ranking collections, it was argued, it is better to make K sensitive to the number of documents (as opposed to percentage of documents) on a topic. Furthermore, it was proposed that K should be large, because the $f_{c,t}$ values will generally be large. The computation of $T_{c,t}$ is modified to

$$T_{c,t} = d_t + (1-d_t) \cdot \frac{f_{c,t}}{f_{c,t}+K} \tag{2}$$

where K is computed as

$$K = k \cdot \big((1-b) + b \cdot (F_c/\bar{F}_c)\big) \tag{3}$$

where $\bar{F}_c = \sum_{c \in C} F_c/N$, and k and b are parameters. The parameter k controls the magnitude of K, while varying b from 0 to 1 increases the sensitivity of K to the size of the collection. The value F_c is the "number of word (term) occurrences" in c (French et al. 1999).

In the initial description of CORI, experiments were used to identify suitable k and b values (Allan et al. 1995, Callan et al. 1995). The first TREC CD (disk volume 1) was broken into seven collections, varying in size from a few million to a few tens of millions of words, and was tested with queries 51–100. The space of k and b values was searched using the relevance judgements for this data, giving the values $k = 200$ and $b = 0.75$. This yields

$$T_{c,t} = d_t + (1-d_t) \cdot \frac{f_{c,t}}{f_{c,t}+50+150 \cdot F_c/\bar{F}_c}$$

$$\mathit{CORI}(q,c) = \frac{\sum_{t \in q \& c}(d_b + (1-d_b) \cdot T_{c,t} \cdot I_{c,t})}{|q|} \tag{4}$$

In some experiments (Callan 2000, French et al. 1999, Larkey et al. 2000), d_t is dropped; that is, the previously used default (Callan et al. 1995) of $d_t = 0.4$ is replaced by $d_t = 0$.

In the experiments reported in this paper, we explore the choice of values for the parameters d_t, k, and b. We use CORI as in Equation 4, and $I_{c,t}$, $T_{c,t}$, and K as in Equations 1, 2 and 3 respectively. In almost every paper that uses CORI, the values used for the key parameters are $k = 200$ and $b = 0.75$. The only exceptions of which we are aware are the work of Conrad et al. (2002), where it is reported (without discussion of how the parameters were explored) that $k = 300$ and $b = 0.6$ are superior; and of Lu & McKinley (1999), where a small number of combinations are explored in the context of replication.

As a thought experiment, it is interesting to examine the expected behaviour of CORI in different environments, using default parameters. Values of N, f_c, and F_c are shown for our test data (discussed in the next section) in Table 1. Consider now the collection DATELINE-M. The ratio $F_c/\bar{F}_c$ varies from 0.00039 to 119.62. For a rare term with $f_{c,t} = 1$, possible $T_{c,t}$ values range from 0.020 down to 0.000056. For $f_{c,t} = 100$, the value of $T_{c,t}$ for the largest value of F_c rises to 0.0056. Even if $f_{c,t} = 1000$, the value rises to only 0.053. Thus a large collection can only be selected if it contains a large number of documents with one of the query terms—it is unlikely that a collection with a small number of relevant documents would be highly ranked. Conversely, a small collection with a couple of

Table 1: *Data set summaries, showing the number of collections N and the distribution of f_c and F_c values.*

	N	f_c			F_c		
		min	avg	max	min	avg	max
ZDISK2	43	1,642	5,377	7,888	1,317,038	1,716,025	1,948,747
ZDISK3	91	14	3,696	22,853	996	1,002,430	19,494,646
ORIGINAL17	17	6,711	63,422	226,087	2,898,248	15,783,397	29,996,344
SYM236	236	1	2,928	8,302	500	943,716	2,653,311
UDC236	236	2,891	2,928	3,356	588,842	943,716	8,863,449
BYLINE-M	2,239	1	39	6,440	21	13,173	1,848,436
BYLINE-C	2,239	1	39	6,440	18	13,173	2,139,015
BYLINE-R	2,239	1	39	6,440	71	13,173	2,141,808
DATELINE-M	530	1	289	30,507	29	75,083	8,981,080
DATELINE-C	530	1	289	30,507	61	75,083	7,819,445
DATELINE-R	530	1	289	30,507	23	75,083	7,948,978

occurrences of any query term is automatically highly ranked. Experimentally, we found that CORI rarely ranks large collections highly, even though they are often the best source of relevant documents.

In a recent paper, Si & Callan (2003) explore the limitations of CORI when collection size varies, and found the same defect. They propose modification to CORI based on estimated database size to compensate for this effect but they do not address the difficult issue of parameter choice. We plan to test this compensation in future work, but it is not clear that the positive results would be observed in the collections we use, where the variation in size and number of collections is in some cases much greater.

3 Test data

We use a range of test sets in our experiments. The first and second sets are contents of TREC disks 2 and 3. They are denoted ZDISK2 and ZDISK3, divided into 43 and 91 collections, respectively (Zobel 1997). In the former of these, each of the 43 collections is of similar size. In the latter, the divisions between the 91 collections were chosen at random, and the sizes vary dramatically.

The third data set used is ORIGINAL17, the contents of TREC disks 1 to 3 divided into seventeen collections as in the original work of Callan, Lu, and Croft (they used seven of these collections to determine the default CORI parameters). Collection sizes vary from 6,711 documents to 226,087 documents. As can be seen, this set is very different to the others.

The fourth data set, denoted SYM236 was developed by French, Powell, Viles, Emmitt & Prey (1998) to explore selection in larger databases (with at least 100 collections), and was a partitioning of TREC data based on source, year and month boundaries. The fifth data set, denoted UDC236 is reorientation of this same set of documents partitioned on the basis of approximately equi-sized collections (Powell et al. 2000).

The last six were derived from the Associated Press data on TREC CDs 1 and 2 (Harman 1995), and have been used by us in other work to explore the impact of different ways of classifying documents (D'Souza et al. 2004). The first of these is BYLINE-M, where the data was divided into 2239 collections according to the <BYLINE> field. Documents without a byline were omitted. Collection sizes were highly skew—most had only one or two documents, 150 or so had between 100 and 700 documents, and one had 6440 documents. We hypothesised that such a breakdown might reflect how documents were created and stored in a workplace such as a news provider, and thus provides a realistic real-world test of distributed document retrieval.

The second of these sets was BYLINE-C; this was a chronological breakdown of exactly the same set of documents into 2239 collections of exactly the same distribution of sizes. The third of these sets was BYLINE-R; a random breakdown of the same set of documents into 2239 collections of exactly the same distribution of sizes.

The fourth of these AP-sourced sets was DATELINE-M, where documents were classified by the <DATELINE> field. This yielded 153,020 documents in 530 collections; the sizes were again skew. The second last of these sets was DATELINE-C, a chronological breakdown of the DATELINE documents into 530 collections with the same size distribution. The last of these sets was DATELINE-R, a random breakdown of the DATELINE documents the same size distribution.

The TREC queries include a short form, or heading, and a longer exposition. This allows each query set to be used twice, in SHORT or LONG form. For ZDISK3, only short queries are available.

4 Experiments

A standard way to measure the performance of collection-selection metrics is to count the number of relevant documents in the highly-ranked collections. For each query, the number of documents that have been judged relevant is known, thus each collection makes a known contribution to the recall for that query. For example, suppose that for a given query there are 200 known relevant documents, and the three collections ranked highest have 16, 2, and 10 relevant documents respectively. Then the recall

Table 2: *Best CORI parameter combinations and recall@10 figures for each test set and query set, compared with baselines SBR, RBR and with standard CORI,* Highsim*, and Inner product.*

		Best (k, b, d_t)	Baselines SBR, RBR	Best CORI	Std. CORI	High-Sim	Inner product
ZDISK2	LONG	200.00, 0.8, 0.4	37.1, 77.1	57.0	**56.4**	54.2	54.9
ZDISK2	SHORT	21.54, 0.4, 0.8	37.1, 77.1	55.1	51.6	**54.0**	53.1
ZDISK3	SHORT	215.44, 0.0, 0.6	36.2, 78.8	51.6	39.6	35.0	**51.1**
ORIGINAL17	LONG	215.44, 1.0, 0.4	78.2, 99.3	93.5	**91.8**	83.6	82.9
ORIGINAL17	SHORT	464.16, 0.75, 0.6	78.2, 99.3	92.5	**91.8**	86.2	85.4
SYM236	LONG	1000.00, 0.6, 0.8	11.0, 47.1	25.7	**25.3**	23.6	22.7
SYM236	SHORT	464.16, 0.2, 0.6	11.0, 47.1	23.5	**22.2**	**22.2**	21.5
UDC236	LONG	464.16, 0.8, 0.2	1.8, 36.4	12.6	11.7	**15.1**	9.5
UDC236	SHORT	100.00, 0.4, 0.0	1.8, 36.4	13.4	12.6	**14.4**	11.3
BYLINE-M	LONG	2.15, 0.6, 0.2	11.2, 75.8	43.1	39.2	**39.8**	35.6
BYLINE-M	SHORT	21.54, 0.2, 0.6	11.2, 75.8	37.3	27.9	**39.8**	35.4
BYLINE-C	LONG	4.46, 0.2, 0.8	13.6, 52.9	18.0	7.9	**18.1**	16.7
BYLINE-C	SHORT	10.00, 0.0, 0.0	13.6, 52.9	17.6	4.5	**20.3**	17.6
BYLINE-R	LONG	10.00, 0.2, 0.4	12.8, 49.0	16.9	6.5	**16.5**	15.0
BYLINE-R	SHORT	1.00, 0.0, 0.2	12.8, 49.0	16.5	2.9	**19.0**	16.1
DATELINE-M	LONG	21.54, 0.4, 0.8	59.8, 89.7	70.6	60.0	**67.3**	66.4
DATELINE-M	SHORT	21.54, 0.0, 0.8	59.8, 89.7	69.0	34.1	**69.8**	68.0
DATELINE-C	LONG	21.54, 0.4, 0.8	48.5, 69.8	50.2	40.3	**49.8**	49.1
DATELINE-C	SHORT	21.54, 0.0, 0.8	48.5, 69.8	49.6	20.5	**49.7**	49.4
DATELINE-R	LONG	21.54, 0.4, 0.8	49.1, 65.6	49.6	46.8	**49.3**	49.2
DATELINE-R	SHORT	100.00, 0.0, 0.8	49.1, 65.6	49.3	24.8	**49.2**	49.0

available if the query is only passed to the top-ranked collection—that is, the recall@1—is 8%. The recall@3 is $(16 + 2 + 10)/200 = 14\%$.

Two benchmarks can be used to bracket possible performance. One is the "perfect" (or RBR, relevance-based ranking) score, if the collections are sorted by decreasing numbers of relevant documents. It is not possible to exceed this figure. The other is the "fixed" (or SBR, sized-based ranking) score, if the collections are sorted by decreasing size, on the simple heuristic that large collections should contain more relevant documents. Selection metrics that do no better than the fixed ordering are uninteresting. Plotting recall against the number of collections, it can be seen that a typical value such as recall@10 is a good indicator of overall performance.

In our experiments, we explored ranges of values for each of k, b, and d_t. The k values are large but not unbounded; we let k range through the geometric series given by progressively dividing 1000 by the value $\sqrt[3]{10} = 2.1544$, terminating at 0.21544, giving 13 values in total. In addition, we tested $k = 200$. We let b take the values 0.00, 0.20, 0.40, 0.60, 0.75, 0.80, and 1.00. We let d_t take the values 0.0, 0.2, 0.4, 0.6, 0.8, and 1.0. A value of $b = 1.0$ means that CORI simply counts the terms in common between query and collection; a value of $d_t = 1$ means CORI just sums up the $I_{c,t}$ terms. With inclusion of the values $k = 200$ and $b = 0.75$, the standard CORI ($d_t = 0.0$) is one of the values tested. In total $13 \times 7 \times 6 = 546$ combinations were explored for each set of queries and test set. We additionally tested varying d_b, but not as exhaustively.

In our first experiment, we set $b = 0.75$ and $d_t = 0.0$, and varied k. Results are shown in Figure 1. In this figure, the values shown are recall at 10 documents retrieved for each value of k and each collection and query set, where the recall values have been rescaled so that 0 is the recall for the "fixed" baseline and 100 is the recall for the "perfect" baseline. (The values of these baselines are shown in Table 2.)

As can be seen, the peak k figure varies wildly. Considering the SHORT queries shown in the left-hand-side graphs, the best value varies from 0.22 to 464.42, depending on the collection, and incorrect choice of k can seriously degrade precision: tuning on one data set does not give good results on the other data sets. This effect is even more pronounced for the LONG queries shown in the right-hand-side graphs, where the best k values range from 0.22 to 1000.00: the best k value for BYLINE-M reduces the score for ZDISK2 from 49 to 20.

More disturbingly, the best k value also varies between query sets on the same data. In the worst instance, DATELINE-M, the best value is 10.00 for SHORT but is 1000.00 for LONG. These queries were derived from the same topics. The parameters commonly used for CORI, which have been justified by performance on only one set of collections, are in some cases an extremely poor choice. Indeed, it is difficult to identify a good choice of parameters for a given set of documents and a query type. We counted the number of queries for which each of the tested k values is the best choice. For example, of the 100 SHORT queries on ZDISK2, there were eight for which $k = 0.22$ was best and 23 for which $k = 1000.00$ was best. Overall, the best choice of k for this data was 21.54 (see Table 2), despite

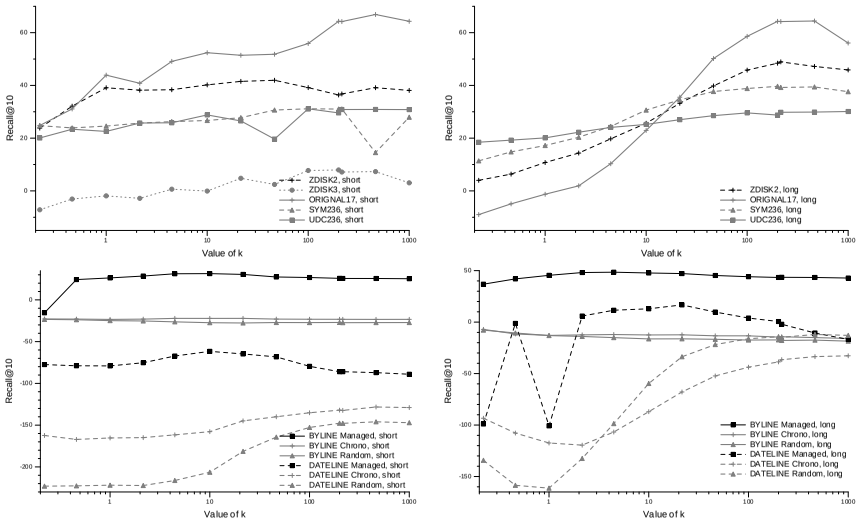

Figure 1: *Effect on recall@10 of varying k for fixed $b = 0.75$ and $d_t = 0.0$, on all collections. Left graphs: for* SHORT *queries. Right graphs: for* LONG *queries. In each case, the recall@10 has been rescaled so that the "fixed" SBR result is 0 and the "perfect" RBR result is 100.*

the fact that this was not the best choice for the great majority of queries.

The effect of varying b is similar to the effect of varying k, but not as extreme. Holding k constant at values such as 10 and 200, and varying b, we observed variations of up to about 15% on the rescaled recall values. These are not vast changes, but neither are they insignificant.

To identify the degree to which CORI performance varies for alternative choices of parameters, as discussed above we completely explored the $13 \times 7 \times 6$ space of parameter values to find the best combination in each case.

Results are in Table 2, which shows, for each test set and query set, the most successful combination of CORI parameters, the recall@10 achieved for this combination, and the recall@10 for the standard combination. Results for two other methods, Highsim and Inner product, are also shown; the figure in bold in each line is the best of standard CORI, Highsim, and Inner product.

We found, not surprisingly, that the poorest CORI results were observed with $d_t = 1.0$. However, for all other values of d_t, the value chosen had no discernible effect on performance. Similarly, we tested d_b, and found that it had little effect. For this reason we do not report changes in performance as a function of these two parameters; note that in many of the previous papers on CORI the parameter d_b is fixed at 0.4.

These results show the best b value varying from 0.0 to 1.0, and the best k value varying from 1.00 to 1000.00. The largest set of collections, the BYLINE data, showed the greatest improvement by tuning CORI and the greatest deviation from the original parameters. There is no consistent optimal combination of parameters for best CORI; indeed, no best CORI values coincide with the choice of $k = 200$ and $b = 0.75$ for standard CORI.

Even the best CORI results are often below the results observed with other lexicon-based methods, such as the Highsim method of Zobel (1997) or even the simple Inner product. Table 2 shows that these other ranking formulations often outperform CORI. Highsim is superior to CORI in 8 of the 21 cases, and superior to best CORI in 8 of the 21 cases; Inner product, although generally poorer than Highsim, is superior to CORI in 13 out of 21 cases. Only on the smallest set of collections, ORIGINAL17, is CORI the best method.

5 Conclusions

CORI has been used in a range of experiments in recent work, in some cases in comparisons with other collection-selection algorithms. In most of these experiments, the CORI formulation has used fixed values for parameters k (set to 200) and b (set to 0.75). These values were based on a set of seven collections extracted from TREC disk 1.

Experiments that used these recommended k and b values did so within a variety of experimental settings. We explored several variations with the aim of establishing optimal choices of k and b for variations in several factors. Our analysis of CORI, within this framework, showed that the greatest CORI effectiveness was for parameter values that did not coincide with

the usual choice. The experiments show that the CORI parameters k and b are highly sensitive to the variations in data sets, and that best k values are widely distributed even for a single data set and type of query. There is no obvious mechanism for setting the CORI parameters, and the use of standard CORI as a benchmark collection-selection method is not justified.

Acknowledgements

This work used computing facilities supported by the Australian Research Council.

References

Abbaci, F., Savoy, J. & Beigbeder, M. (2002), A methodology for collection selection in heterogeneous contexts, *in* 'Proc. IEEE Int. Conf. on Information Technology: Coding and Computing (ITCC'02)', Las Vegas, USA, pp. 529–535.

Allan, J., Ballesteros, L., Callan, J., Croft, W. B. & Lu, Z. (1995), Recent experiments with INQUERY, *in* D. Harman, ed., 'Proc. Fourth Text Retrieval Conf. (TREC-4)', National Institute of Standards and Technology Special Publication 500-236, Gaithersburg, Maryland, pp. 49–57.

Callan, J. & M.Connell (2001), 'Query-based sampling of text databases', *ACM Transactions on Information Systems* **19**(2), 97–130.

Callan, J. P. (2000), Distributed information retrieval, *in* W. B. Croft, ed., 'Advances in Information Retrieval: Recent Research from the Center for Intelligent Information Retrieval', Kluwer Academic Publishers, pp. 127–150.

Callan, J. P., Lu, Z. & Croft, W. B. (1995), Searching distributed collections with inference networks, *in* E. A. Fox, P. Ingwersen & R. Fidel, eds, 'Proc. ACM SIGIR Int. Conf. on Research and Development in Information Retrieval', ACM, pp. 21–28.

Conrad, J., Guo, X., Jackson, P. & Meziou, M. (2002), Database selection using actual physical and acquired logical collection resources in a massive domain-specific operational environment, *in* 'Proc. 28th Int. Conf. on Very Large Data Bases (VLDB'02)', Hong Kong, China.

Craswell, N., Bailey, P. & Hawking, D. (2000), Server selection on the world wide web, *in* 'Proc. Fifth ACM Conf. on Digital Libraries', pp. 37–46.

D'Souza, D., Thom, J. A. & Zobel, J. (2004), 'Collection selection for managed distributed document databases', *Information Processing and Management* **40**(3), 527–546.

French, J. C., Powell, A. L., Callan, J., Viles, C. L., Emmitt, T., Prey, K. J. & Mou, Y. (1999), Comparing the performance of database selection algorithms, *in* M. Hearst, F. Gey & R. Tong, eds, 'Proc. ACM SIGIR Int. Conf. on Research and Development in Information Retrieval', ACM, pp. 238–245.

French, J. C., Powell, A. L., Viles, C. L., Emmitt, T. & Prey, K. J. (1998), Evaluating database selection techniques: A testbed and experiment, *in* W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson & J. Zobel, eds, 'Proc. ACM SIGIR Int. Conf. on Research and Development in Information Retrieval', ACM, pp. 121–129.

Gravano, L. & Garcia-Molína, H. (1995), Generalising GlOSS to vector-space databases and broker hierarchies, *in* 'Proc. Int. Conf. on Very Large Data Bases, September 11-15, Zurich, Switzerland', pp. 78–89.

Harman, D. (1995), 'Overview of the second text retrieval conference (TREC-2)', *Information Processing and Management* **31**(3), 271–289.

Hawking, D. & Thistlewaite, P. (1999), 'Methods for information server selection', *ACM Transactions on Information Systems* **17**(1), 41–76.

Larkey, L., Connell, M. & Callan, J. (2000), Collection selection and results merging with topically organized U.S. patents and TREC data, *in* 'Proc. Ninth ACM Int. Conf. on Information and Knowledge Management (CIKM'00)', pp. 282–289.

Lu, Z. & McKinley, K. (1999), Partial replica selection based on relevance for information retrieval, *in* 'Proc. ACM SIGIR Int. Conf. on Research and Development in Information Retrieval', ACM, Berkley, CA USA, pp. 97–104.

Meng, W., Yu, C. & Liu, K.-L. (2002), 'Building efficient and effective metasearch engines', *ACM Computing Surveys* **34**(1), 48–89.

Powell, A. L., French, J. C., Callan, J., Connell, M. & Viles, C. L. (2000), The impact of database selection on distributed searching, *in* 'Proc. ACM SIGIR Int. Conf. on Research and Development in Information Retrieval', ACM, pp. 232–239.

Rasolofo, Y., Abbaci, F. & Savoy, J. (2001), Approaches to collection selection and results merging for distributed information retrieval, *in* 'Proc. Tenth ACM Int. Conf. on Information and Knowledge Management, 2001 (CIKM'01)', Atlanta, USA, pp. 191–198.

Si, L. & Callan, J. (2002), Using sampled data and regression to merge search engine results, *in* 'Proc. ACM SIGIR Int. Conf. on Research and Development in Information Retrieval', ACM, Tempere, Finland, pp. 19–26.

Si, L. & Callan, J. (2003), Relevant document distribution estimation method for resource selection, *in* 'Proc. ACM SIGIR Int. Conf. on Research and Development in Information Retrieval', Toronto, Canada, pp. 298–305.

Yuwono, B. & Lee, D. L. (1997), Server ranking for distributed text retrieval systems on the internet, *in* 'Proc. 5th Int. Conf. on Database Systems for Advanced Applications (DASFAA'97)', Melbourne, Victoria, Australia, pp. 41–49.

Zobel, J. (1997), Collection selection via lexicon inspection, *in* 'Proc. 2nd Australian Document Computing Symposium (ADCS'97)', Melbourne, Victoria, Australia, pp. 74–80.

Co-training on Textual Documents with a Single Natural Feature Set

Jason Chan
School of Information Technologies
The University of Sydney
NSW 2006 Australia
jchan3@it.usyd.edu.au

Irena Koprinska
School of Information Technologies
The University of Sydney
NSW 2006 Australia
irena@it.usyd.edu.au

Josiah Poon
School of Information Technologies
The University of Sydney
NSW 2006 Australia
josiah@it.usyd.edu

Abstract *Co-training is a semi-supervised technique that allows classifiers to learn with fewer labelled documents by taking advantage of the more abundant unclassified documents. However, conventional co-training requires the dataset to be described by two disjoint and natural feature sets that are redundantly sufficient. In many practical situations datasets have a single set of features and it is not obvious how to split it into two. This paper investigates the performance of co-training with only one natural feature set in two applications: Web page classification and email filtering.*

Keywords Text categorization, Web page classification, spam filtering, co-training

1 Introduction

As the number of on-line documents increases, finding information that is really relevant to the users' needs becomes very important. This is one aspect of the information overload problem faced by a growing number of people. Research in Text Categorization [10] and Machine Learning has shown that it is possible to build effective classifiers for intelligent information retrieval given a sufficiently large set of labelled examples. However, obtaining labelled documents requires a great deal of human effort and is also a time consuming and tedious process.

Blum and Mitchell [2] introduced a new technique to overcome this problem. This method, called *co-training*, was shown to be capable of converting unlabelled Web documents into labelled Web documents by initially starting off with only a small pool of classified examples. The authors stated two main requirements on the dataset to be satisfied in order for co-training to be beneficial. Firstly, the dataset must be described by two disjoint sets of features that were sufficiently strong. That is, using only either one of the sets of attributes, a classifier can be built with reasonably high accuracy. For example, in their experiment that dealt with the problem of classifying Web pages, the two sets of features used to describe a page were the words in the body of the page and the words in hyperlinks of other documents referring to that particular page. Secondly, the two feature sets should be conditionally independent given the class.

In the great majority of practical situations, there do not exist two natural sets of features that can describe the dataset. In other cases, the data collected may only belong to one of the possible natural feature sets. In this paper, we investigate the applicability of co-training to such datasets. We compare co-training of a single natural feature set and co-training with two natural feature sets. By analysing the results, we address the question of when co-training with a random split of features is likely to be useful. The experiments are based on two applications: Web page classification and spam filtering. This paper extends our work presented in [4] by including addition experimental data and theoretical insights.

This paper is organised as follows. The next section provides important background information on co-training. In section 3, previous work on co-training is covered. Section 4 describes the experimental set-up and summarises the experimental objectives, while section 5 presents the experimental results. Section 6 gives a detailed discussion. The final conclusions and potential future work are given in section 7.

2 The Co-training Algorithm

In a given application, we may have a set of redundant features that can be used to classify the instances. That is, it is possible to split up all the features into two sets so that we can build two independent classifiers that can still label the instances correctly. These sets of features are said to be *redundantly sufficient*. As an example, emails can be classified accurately with just the header information (sender, subject, etc.) or just the content in the body of the message.

These two classifiers are trained with a small set of labelled instances so that we have two weak classifiers. They are then employed in a loop to classify all the unlabelled examples. Each classifier selects the most confidently predicted examples and adds these into the training set. Both classifiers then re-learn on the enlarged training set so that they take into account the newly added (and previously unlabelled) data. The loop is then repeated for a

Proceedings of the 9th Australasian Document Computing Symposium, Melbourne, Australia, December 13, 2004.

number of iterations to maximize performance on a separate validation set.

The co-training algorithm is summarized below:

Obtain a small set **L** of labelled examples
Obtain a large set **U** of unlabelled examples
Obtain two sets $\mathbf{F_1}$ and $\mathbf{F_2}$ of features that are redundantly sufficient
while U is not empty do:
    Learn classifier $\mathbf{C_1}$ from **L** based on $\mathbf{F_1}$
    Learn classifier $\mathbf{C_2}$ from **L** based on $\mathbf{F_2}$
    for each classifier $\mathbf{C_i}$ do:
        $\mathbf{C_i}$ labels examples from **U** based on $\mathbf{F_i}$
        $\mathbf{C_i}$ chooses the most confidently predicted examples **E** from **U**
        E is removed from **U** and added (with their given labels) to **L**
    end-for
end-while

The intuition for why this algorithm should work is as follows. One classifier, with its set of features, can confidently predict the class of an unlabelled example, because it is similar to the training instances. However, it may only be similar to the training instances for this classifier's set of features; the other classifier may not have been so sure about this instance's classification. Because of the confidence with which the first classifier predicts this example's class, it will be labelled accordingly and placed into the training set. Hence, the second classifier will be able to learn from this instance and adjust better in future.

For example, suppose we have two email classifiers with one classifier using the headers of emails and the other using the words in the body of the message. The first classifier has been trained to categorize any email with the word "assignment" to be placed in the folder for teaching. If another email comes along with "assignment" in its subject header, the first classifier will be very confident that this message should be put in the folder for teaching. This will allow the second classifier to learn more about those words in the body of a message that can be used to determine that an email belongs in the folder for teaching. The second classifier learns because the words used in the body of the first email will not necessarily be the same as the words used in the second email.

3 Previous Work

Blum and Mitchell [2] performed the first experiments on co-training. The task was to identify the home Web pages of academic courses from a large collection of Web pages collected from several Computer Science departments. Their co-training implementation used the following two natural feature sets: the words present in the Web page (page-based classifier) and the words used in another page's link that pointed to the page (hyperlink-based classifier). The results showed that the error rate of the combined classifier was reduced from 11% to 5%. It was also proved that if the feature sets used by the classifiers are *conditionally independent* given the class, and the target classification function can be approximated, then any initial weak classifier can be improved to arbitrarily high accuracy using co-training. More recent research shows that this condition can be relaxed to a certain extent [9]. For example, it was proven that for two classifiers with weak dependence, the rate of disagreement between them provides an upper-bound limit on their error rate [1].

Kiritchenko and Matwin [5] applied co-training to the domain of email classification. They found that the performance of co-training is sensitive to the learning algorithm used. In particular, co-training with Naïve Bayes (NB) worsens performance, while Support Vector Machines (SVM) improves it. The authors explained this with the inability of NB to deal with large sparse datasets. This explanation was confirmed by significantly better results after feature selection.

Nigam and Ghani [9] investigated the sensitivity of co-training to the assumptions of conditional independence and redundant sufficiency. In their first experiment, co-training was applied to the Web pages database from [2]. The results showed that co-training using NB was not better than Expectation Maximization (EM) even when there is a natural split of features. Both EM and co-training with NB improved the performance of the initial classifier by approximately 10%. The second experiment was performed on a dataset that had been created in a semi-artificial manner so that the two feature sets are truly conditionally independent. In addition, the condition of redundantly sufficient features was met, since the NB trained on each of the data sets separately was able to obtain a small error. It was found that co-training with NB well outperformed EM, and even outperformed NB trained with all instances labelled. Their third experiment involved performing co-training on a dataset whereby a natural split of feature sets is not used. The two feature sets were chosen by randomly assigning all the features of the dataset into two different groups. This was tried for two datasets: one with a clear redundancy of features, and one with an unknown level of redundancy and non-evident natural split in features. The results indicated that the presence of redundancy in the feature sets gave the co-training algorithm a bigger advantage over EM.

Together with theoretical insights, the results of these experiments led the researchers to conclude that co-training has a considerable dependence on the assumptions of conditional independence and redundant sufficiency. However, even when either or both of the assumptions are violated, the performance of co-training can still be quite useful in improving a classifier's performance. In particular, in many practical settings, co-training is likely to be beneficial.

4 Experimental Setup

4.1 Objective

In the large majority of cases, datasets consist of only a single set of features with no obvious or natural way to divide them into two separate sets. Hence, the question of whether co-training can be useful with only a single natural feature set is of great practical importance. This paper investigates the performance of co-training with only one natural feature set in comparison to the use of two natural feature sets. The main question that we address is: *How useful is co-training with a single natural feature set*?

To tackle this question, we performed two sets of experiments in the two domains of Web page classification and spam filtering. Table 1 summarizes the four different experiments that we conducted, and the names that we have assigned to them.

Experiment	Web Page Classification	Spam Filtering
Supervised Learning	*WebSL*	*SpamSL*
Co-Training	*WebCT*	*SpamCT*

Table 1. Experiment names

The Supervised Learning Experiment deals with supervised learning. Here, we gain some insights into the different classifiers and feature sets, and in particular, determine how redundantly sufficient the different feature sets are. Good classification performance on a given feature subset implies that it is redundantly sufficient.

In the Co-Training Experiment, we perform co-training and compare the performance between using the natural split against using a random split of all the features. We also compare these performances against traditional supervised learning on the initial labelled set to see what improvement the co-training process gives a learner.

4.2 Pre-processing and Feature Selection

In both the Web page classification and spam filtering experiments, the documents were initially pre-processed by applying a stop-list[1], with the Web pages being further processed by removing certain html tags.

The standard bag-of-words representation was used and feature selection was performed with Information Gain [12]. Feature selection is a common method to reduce running time by using only the most important attributes, and has been shown previously to improve performance, such as in [12]. Upon inspection of the word lists, it was decided that the top 100 words was a suitable cut-off for experiments in both domains, resulting in a dimensionality reduction of about 98%. Hence, for each feature set, each document was represented using the term frequencies of the selected 100 features. Note that feature selection was applied individually to each feature subset.

4.3 Web Page Classification

Four topics with approximately 90 Web pages on each topic were retrieved and rated by four users. The Web pages thus formed 16 datasets. The phrases "nuclear fusion", "circulatory system", "food pyramid" and "greenhouse effect ozone layer" were used as queries in the Google[2] search engine to obtain Web documents on four topics.

For each topic, the users were given an objective for which they had to attempt to achieve with the given Web pages. A rating of either "good" or "bad" was assigned, depending on how useful a given user found that particular page for the task assigned to that topic. Each user's rating for the different pages was saved separately. Table 2 summarizes the feature sets used in the Web page classification experiments.

Feature set	Description
Headers	all words that appear in either titles, headings or hyperlinks
Words	all words that appear in the Web page without counting occurrences in the titles, headings, or hyperlinks
Half1	a random selection of half of the feature set *Words*
Half2	the other half of the words not found in *Half1*

Table 2. Feature sets used in Web page classification experiments

The two feature sets *Half1* and *Half2* are created to test the hypothesis that it is possible to randomly split the feature set into two smaller feature sets to obtain useful results in co-training.

The *Words* and *Headers* feature sets in the domain of Web page classification will hereon be referred to as the *natural feature sets*, while the other feature sets that contain a random selection of words will be referred to as the *random selection feature sets*. Our experiment with supervised learning applied to Web page classification is given the name *WebSL*, while co-training with Web page classification is called *WebCT*.

[1] http://alt-usage-english.org/excerpts/fxcommon.html

[2] http://www.google.com

4.4 Spam Filtering

To test the domain of email classification, we used the LingSpam[3] corpus. This dataset consists of 2883 emails of which 479 are spam and 2404 are genuine emails. Each email is broken up into two sections: the text found in the subject header of the email and the words found in the main body of the message. A distinction was made between the words that appeared in the subject header and those that appeared in the body. A summary of the feature sets used in the experiments is given in Table 3.

Feature set	Description
Body	all words that appear in the body of an email
Subject	all words that appear in subject of an email
Half1	a random selection of half of the feature set consisting of the combination of *Subject* and *Body*
Half2	the other half of the features not found in *Half1*

Table 3. Feature sets used in email filtering experiments

In similar style to Web page classification, the *Body* and *Subject* feature sets in the domain of spam filtering will hereon be referred to as the *natural feature sets*, while the other feature sets that contain a random selection of words will be referred to as the *random selection feature sets*. The experiment with supervised learning applied to spam filtering is given the name *SpamSL*, while the co-training experiment in spam filtering is called *SpamCT*.

4.5 Classifiers

Four types of classifiers were tested: Decision Tree (DT), Random Forest (RF) [3], Naïve Bayes (NB) and Support Vector Machine (SVM). In previous work on co-training [5], NB has often been used as a benchmark. The SVM was used in text categorization and email classification [5] with great success. Implementations of these classifiers were obtained from WEKA [11].

4.6 Evaluation

In this paper, f-measure will refer to macro-averaged f1-measure, which is given by the formula: f1 = 2pr / (p+r) where *p* is the macro-averaged precision and *r* is the macro-averaged recall. The macro-averaged precision is the average of the precision of each of the two classes; similarly for recall.

[3] http://www.mlnet.org/cgi-bin/mlnetois.pl/?File=dataset-details.html&Id=963839410Ling-Spam

5 Experimental Results

5.1 Experiment *WebSL*

In order to focus on general trends, our results were averaged over all users and topics. We analysed the results for each user and topic and found them to be consistent with our general findings. Table 4 summarizes the classification performance using 10-fold cross validation. Note that in each case, the classifiers have access to the top 100 features in the respective feature sets since feature selection was performed on each feature set individually.

Classifier	Words	Headers	Random halves	All features
DT	74.0	65.6	73.2	73.4
RF	79.1	75.9	78.6	78.5
NB	80.5	75.1	80.5	80.5
SVM	81.6	75.7	81.2	81.6

Table 4. F-measures (%) using different feature sets in *WebSL*

5.2 Experiment *WebCT*

Table 5 summarises the results of the co-training experiments over all users and topics, after labelling all initially unlabelled instances. Shown are the maximum classification performance (max) that is achieved and the performance improvement (increase) achieved over the base classifier trained with only the initial labelled set of 10%.

Classifier	Natural Feature Split		Random Feature Split	
	max	increase	max	increase
DT	67.5	0.0	67.9	0.5
RF	72.3	3.3	72.0	3.6
NB	73.6	3.0	71.9	1.1
SVM	40.2	0.5	75.0	1.9

Table 5. F-measures (%) with natural vs random split in *WebCT*

Another 10% of the instances were set aside for the test set, and the remaining data was used as initially unlabelled data. Stratification was performed to ensure an even distribution of instances from the two classes. Ten different combinations of labelled, unlabelled and test sets were used, and the results were averaged over these 10 runs. This somewhat resembles 10-fold cross validation.

The proportion of positive instances to negative instances varied between each user and topic. However, in most cases, this ratio was approximately 2 negative instances to 1 positive instance. For the results shown here, the number of positive and negative instances to be transferred from the unlabelled set to the labelled set was set to 1 and 2 respectively.

5.3 Experiment *SpamSL*

Table 6 summarizes the results. 10-fold cross validation was performed, with 2595 instances used in the training set and 288 instances in the test set. The results of RF are not reported here because we experienced memory constraints on our system.

Classifier	Body	Subject	Random halves	All features
DT	92.7	45.5	89.2	92.7
NB	86.9	45.5	85.0	86.9
SVM	88.9	63.0	87.3	88.9

Table 6. F-measures (%) using various feature sets in *SpamSL*

5.4 Experiment *SpamCT*

Table 7 summarises the results. Again, the best classification performance (max) that is achieved and the performance improvement (increase) achieved over the base classifier is given. Figure 1 illustrates the difference in performance over the number of co-training iterations completed for NB.

Classifier	Natural Feature Split		Random Feature Split	
	max	increase	max	increase
DT	45.5	0.0	45.5	0.0
NB	84.9	8.2	86.8	10.1
SVM	78.0	0.0	79.5	1.5

Table 7. F-measures (%) with natural vs random split in *SpamCT*

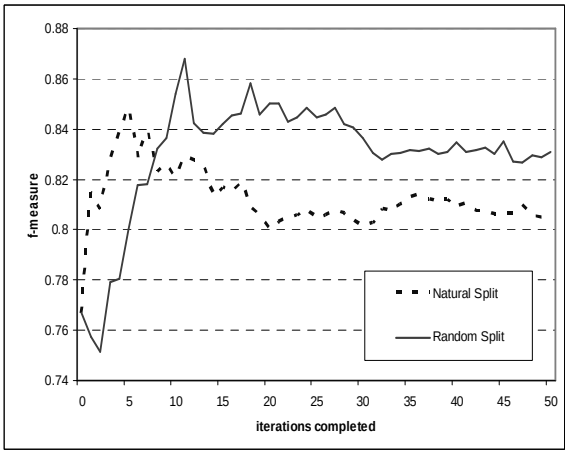

Figure 1. Performance of NB in *SpamCT*

We started off with a labelled set of 1 spam and 1 genuine email. 10% of the dataset were retained for testing, while the rest were used as initially unlabelled data. The ratio of spam to genuine emails in LingSpam is 1:5. Following this distribution, 1 newly-labelled spam and 5 newly-labelled genuine emails were transferred from the unlabelled set to the labelled set on each iteration of co-training. We repeated each trial 5 times, obtaining a different sample of labelled, unlabelled and test set each time. The above results are averaged over these 5 trials.

As shown in Figure 1, co-training with a random split of features improves the performance of a classifier in comparison to only using the initial labelled set. We found this to be the case in most settings.

6 Discussion

6.1 Supervised Learning Experiments

Table 4 and 6 show that the best performing feature sets (for all classifiers) were *Words* in Web page classification and *Body* in spam email filtering, respectively, and the worst were *Headers* and *Subject*. The performance of *Words* is better than *Headers* by 3-9%. (WebST) and *Body* outperforms *Subject* by 25-37% (EmailST). In fact, the performance of *Words* and *Body* alone is the same as using all features. That is, combining *Headers* with Words and *Subject* with *Body* does improve performance.

However, it is very interesting to notice that the performance does not suffer any significant degradation when a random half of the features was selected (compare *Random halves* with *All features*). For example, using half of the words randomly selected from all the available words only decreases the classification performance by no more than 4% in both applications. Hence, the supervised learning results indicate that there is redundant sufficiency in the feature sets from which the random halves were drawn. This suggests that the two domains have satisfied the necessary requirements for co-training.

For Web page classification, we even went one step further by performing a more aggressive sampling and randomly selecting only one-fifth of the feature set. The results (not shown here) indicated that the performance was reduced by a maximum of only 3.3% and 4.2% in comparison to using half and all of the features, respectively. This indicated a significant redundancy among the words in the body of the Web documents.

An interesting question is why the *Headers* and *Subject* feature sets do not perform well. The words found in the *Header* and *Subject* feature sets tend to be more meaningful than other words found in the main body of the document, since they summarise the main topics and their words are usually more selectively chosen than the words found in the main body. The most likely reason for the poor performance of the *Header* and *Subject* features is the significantly lower number of word tokens present in them in comparison to the entire email message or Web page, which makes them poor indicators of the document class. In [2], Blum and Mitchell also reported that their hyperlink-based classifier has inferior performance due to similar reasoning. Another reason is the increased sensitivity to noise when using a fewer word tokens. This means that non-discriminating words are more likely to be treated as significant for the classification when the total number of word tokens is small.

6.2 Co-training Experiments

The *increase* columns in Table 5 (WebCT) and Table 7 (SpamCT) show that co-training does not degrade the performance of the initial classifier trained with the small number of positive and negative examples. This holds for both natural-split and random-split co-training.

The same tables also show that co-training with a random split of the features produces results that are comparable with using the natural feature sets. In many cases, the f-measure is higher for the random feature split in comparison to the natural feature split. Why is co-training with a random split of the features so comparable, and in some cases, even better than using the natural feature sets?

There are two reasons for this. Firstly, one of the classifiers used in co-training with the natural feature sets is even weaker than either of the two classifiers using randomly generated feature sets. As shown in the supervised learning experiments, the *Header* and *Subject* feature set are much weaker than any of the random feature sets that were produced. As a result, the classifier using such a feature set is incorrectly labelling many instances in comparison with classifiers built using the random selection feature sets, hence transferring many incorrectly labelled instances into the labelled set.

Secondly, the supervised learning experiments also found that there is redundant sufficiency in the *All* feature set. That is, using a random selection of half of the features from all the features resulted in classifiers that only perform slightly worse than a classifier using all the attributes available. As a result, when performing co-training, both classifiers using their respective half of the features are able to improve the training set because they label unlabelled instances with a sufficiently high classification accuracy.

These two reasons combined suggest that co-training with a random split of a redundantly sufficient feature set can be just as competitive as and even better than co-training with two natural feature sets. This is especially the case when there exists a considerable difference between the classification performances of the two classifiers using the natural feature sets separately. As shown in the supervised learning experiment, a natural feature set consisting of fewer words, such as using the hyperlinks of a Web page, or using the subject header of an email, may produce significantly poorer results. In this event, co-training without the use of this lower quality feature set is likely to be more beneficial.

6.3 Comparison between the Classifiers

From Table 5 and 7, it is very clear that the DT classifier performs much worse than any of the other classifiers. Only in the *SpamSL* experiment does the DT do well. This classifier's poor performance is expected, since the branching of a DT is not very effective when there exists a large number of weak features in the dataset.

In the supervised learning experiments, there was little observed difference in performance between the NB and SVM classifiers (see Table 4 and 6). Previous research [5] showed that the SVM classifier was superior to NB in the application of email classification. Our experiment does not obtain this result because we perform drastic feature selection and keep only the best 100 features in each feature set. Hence, the SVM, which performs in high-dimensional feature spaces, does not get to illustrate its advantage over NB in such a setting.

It should be noted that while some classifiers perform best in the supervised classification experiments, others perform better in the co-training experiments, both in terms of highest f-measure obtained and greater improvement. This is the case because the supervised classification experiments are performed using 10-fold cross validation of the entire dataset, which means that the training set is 90% of the dataset. But with co-training, the base classifier starts off with only a few labelled instances, so a classifier's performance before co-training begins will not be similar to the 10-fold cross validated supervised classification results.

Hence, improvement with co-training is obtained if the base classifier is sufficiently strong to make good classifications for the unlabelled data. In particular, the classifier needs to be accurate with those instances in which it has high confidence in, since these are the instances that are selected and transferred from the unlabelled set to the labelled set during co-training. At the same time, we require the classifier to be weak enough so that extra labelled data will help it learn something new.

From Table 7, it can be seen that for spam filtering, the SVM classifier did not improve much with co-training in comparison to the NB classifier. In the case of Web page classification, it is not so clear from Table 5 which classifier improved more. These results occur because in the spam filtering case, the initial labelled set is just 2 instances, compared with 8 in the Web page classification experiment. Hence, in the spam filtering case, a very poor base classifier is built with the SVM, whereas NB can better classify the most confidently labelled instances. In the Web page classification experiment, the SVM base classifier has more training data.

Table 5 shows that in Web page classification, the classifier that improved most was the random forest. The hypothesized reason for this is that the RF classifier is more accurate in selecting the most confident prediction. This is because the RF classifier uses bagging of individual trees [3], which is more noise resilient than using a single classifier. The instance that is considered to be the most confident to label by the RF classifier is supported by a large number of decision trees that are individually confident with the classification of the same instance.

Thus, the performance of co-training depends on the choice of classifier. The encouraging results discussed in sections 6.1 and 6.2 are shown to be valid provided that a suitable classifier is used for co-training.

7 Conclusion

Document filtering is becoming increasingly important as excessive quantities of data become available in the Information Age. Classifiers can be built to filter out unwanted information but typically require many labelled examples. Co-training has been shown to be a beneficial tool in improving the performance of a classifier that is given only a small training set. However, conventional co-training requires the documents to be able to be described by two natural sets of features, which is not always possible.

The primary objective of the supervised learning experiments was to determine whether the two corpora were redundantly sufficient. It was found that classifiers could be built using a random selection of only half of all available features and still obtain very good classification performance. This implies that the datasets are redundantly sufficient.

In the co-training experiments, the first natural feature set contained the words used in the main body of the Web pages or emails messages, while the second feature set consisted of the words occurring in the subject header for emails and header information for Web pages. It was found that co-training using a random split of all the features was just as competitive as, and often outperformed, co-training with the natural feature sets. Also, classification performance generally improved over the initial classifier trained on the small initial labelled set with random-split co-training.

An important element that is needed in a feature set for co-training with a random split to work well is a dataset with high redundancy. When this condition is met, a random split of the feature set will produce two subsets, each of which can still be used on its own by a classifier to achieve a sufficiently high classification performance.

Also, co-training with a random split of a single natural feature set should be more preferable than co-training with two natural feature sets if one of the natural feature sets is considerably weaker than the other. This is particularly the case with Web pages and emails, where feature sets other than the words in the main body will typically be weak because of their small size. In such cases, co-training with a random split of all the features should produce comparable and possibly better results.

Further research is needed in other domains to determine just how successful this approach will be in comparison to co-training with natural feature sets currently used. Another possible path for future work is to compare the performance of co-training on two sets of features that are randomly split from a single natural feature set, with self-training [9] on this single natural feature set. A deeper investigation into the relation between choice of classifier and performance of co-training would also be beneficial.

Finally, rather than using a random split with reasonable results, developing a view-factorization algorithm capable of obtaining the optimal (or a near-optimal) split of the features presents a future challenge of considerable importance. There has already been a view-validation algorithm proposed [7] that can predict whether a given pair of views for a task will allow for a multi-view algorithm to outperform its single-view counterpart. This is the first step towards implementing a view-factorization algorithm. Also, a greedy heuristic method that gradually builds two sets of classifiers, adding strong features to each classifier one at a time, has been introduced in [6]. However, this method has been used in applications [8] where using the natural feature split has been reported to perform better.

8 References

[1] S. Abney, Bootstrapping, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguisitics*, 2002.

[2] A. Blum, T. Mitchell, Combining Labeled and Unlabeled Data with Co-Training, *Proceedings of the Workshop on Computational Learning Theory*, 1998.

[3] L. Breiman, Random Forests, *Technical Report, Department of Statistics, University of California, Berkeley*, 1999.

[4] J. Chan, I. Koprinska, J. Poon, Co-training with a Single Natural Feature Set Applied to Email Classification, *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI'04)*, 2004.

[5] S. Kiritchenko, S. Matwin, Email Classification with Co-Training, *Proceedings of CASCON*, 2001.

[6] C. Muller, S. Rapp, M. Strube, Applying Co-training to Reference Resolution, *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, 2002.

[7] I. Muslea, S. Minton, C. A. Knoblock, Adaptive View Validation: A First Step towards Automatic View Detection, *Proceedings of International Conference on Machine Learning*, 2002.

[8] V. Ng, C. Cardie, Weakly Supervised Natural Language Learning without Redundant Views, *Proceedings of HLT-NAACL*, 2003.

[9] K. Nigam, R. Ghani, Analyzing the Effectiveness and Applicability of Co-Training, *Proceedings of the 9th International Conference on Information and Knowledge Management*, 2000.

[10] F. Sebastiani, Machine Learning in Automated Text categorization, *ACM Comuting. Surveys*, 2002.

[11] I. H. Witten, E. Frank, *Data Mining: Practical machine learning tools with Java implementations.* Morgan Kaufmann, 1999.

[12] Y. Yang, J. O. Pedersen, A Comparative Study on Feature Selection in Text Categorization, *Proceedings of the 14th International Conference on Machine Learning*, 1997.

A Testbed for Indonesian Text Retrieval

Jelita Asian Hugh E. Williams S.M.M. Tahaghoghi
School of Computer Science and Information Technology
RMIT University, GPO Box 2476V, Melbourne 3001, Australia.
{jelita,hugh,saied}@cs.rmit.edu.au

Abstract *Indonesia is the fourth most populous country and a close neighbour of Australia. However, despite media and intelligence interest in Indonesia, little work has been done on evaluating Information Retrieval techniques for Indonesian, and no standard testbed exists for such a purpose. An effective testbed should include a collection of documents, realistic queries, and relevance judgements. The TREC and TDT testbeds have provided such an environment for the evaluation of English, Mandarin, and Arabic text retrieval techniques. The NTCIR testbed provides a similar environment for Chinese, Korean, Japanese, and English. This paper describes an Indonesian TREC-like testbed we have constructed and made available for the evaluation of ad hoc retrieval techniques. To illustrate how the test collection is used, we briefly report the effect of stemming for Indonesian text retrieval, showing — similarly to English — that it has little effect on accuracy.*

Keywords Indonesian, queries, collection, relevance judgements, stemming

1 Introduction

The Text REtrieval Conference, TREC, began in 1992 in response to the need for a common testbed for evaluating Information Retrieval techniques, and the need to meet and discuss those techniques [3]. TREC is divided into *tracks*, and each track into *tasks*. A track investigates a retrieval paradigm, and each task a subelement of that paradigm. For example, the 2003 Web track investigates retrieval techniques for the Web, with two tasks that investigate topic and home (or named-page) finding. The original TREC track was *ad hoc*, proposed to investigate ad hoc searches for new topics in archived data.

The ad hoc paradigm is that used by most users of web search engines. A typical ad hoc query is a phrase or set of keywords that describe an information need, and the correct or *relevant* responses are those documents that meet that information need. The ad hoc paradigm is no longer explicitly investigated as a TREC track, but it is still widely investigated as a task in many tracks such as the genomics, terabyte, and robust tracks. Indeed, it is well-known that the ad hoc queries from earlier TREC conferences are the standard testbeds used by IR researchers developing new techniques.

Indonesia has the world's fourth largest population, with over 230 million people. However, very little research in IR has investigated techniques for Indonesian. It is important for Australian organisations to be able to retrieve, process, and monitor Indonesian for media, political, and intelligence applications. One possible reason that research in Indonesian IR has not flourished as well as other major languages is the absence of a publicly available Indonesian testbed. The Indonesian document collections that do exist [2, 5, 6] either do not have topics and relevance judgements, or are not published.

In this paper, we describe a TREC-like testbed for the evaluation of ad hoc Indonesian text retrieval. We have defined 20 ad hoc queries for which known answers exist in a collection of 3,000 newswire documents. For each query, we have exhaustively examined the documents and identified those that meet the information need as the set of relevant answers. To illustrate the usefulness of the testbed, we report experiments with a stemming scheme we have recently developed [1]. Our results show that stemming has little benefit for the accuracy of Indonesian ad hoc text retrieval.

2 Testbed Construction

A testbed for evaluating ad hoc retrieval requires three parts: a document collection, a list of query topics, and a set of *relevance judgements* [8]. This section explains how we constructed each component. Additional information and the testbed itself are available online[1].

2.1 Collection

A collection for evaluating ad hoc retrieval must be a static set of documents. Motivated by TREC, we obtained a collection of newswire articles by crawling daily news from the popular online Indonesian newspaper Kompas[2]. We retrieved 3,000 articles between January and June 2002 inclusive, leading to a collection of around 0.7 megabytes in size with 38,601 distinct words. Similar to TREC, we followed

[1] http://www.cs.rmit.edu.au/~jelita/corpus.html
[2] http://www.kompas.com/

Proceedings of the 9th Australasian Document Computing Symposium, Melbourne, Australia, December 13, 2004.

```
<DOC>
<DOCNO>news10513-html</DOCNO>
Mayjen Syafrie Samsuddin akan Jadi
Kapuspen TNI JAKARTA (Media):  Mantan
Pangdam Jaya Mayjen Syafrie Samsuddin
akan menjadi Kapuspen TNI
menggantikan Marsekal Muda Graito
Husodo.Menurut informasi yang
diperoleh Antara Jakarta Kamis, Syafrie
Samsuddin menjadi Kapuspen TNI dan serah
terima jabatan akan dilakukan pada akhir
Februari 2002.  Namun kebenaran
informasi tersebut hingga kini belum
dapat dikonfirmasikan ke Kapuspen TNI.
( M-1 )
</DOC>
```

Figure 1: *An example Kompas newswire document from our test collection, marked up in a TREC-like format.*

```
<top>
<num> Number:  14
<title> nilai tukar rupiah terhadap
dolar AS
<desc> Description:  Dokumen harus
menyebutkan nilai tukar rupiah terhadap
dolar AS.
<narr> Narrative:  Asalkan dokumen ada
menyebutkan nilai tukar rupiah terhadap
dollar tanpa indikasi menguat atau
melemah sudah dianggap relevan.
Prediksi nilai tukar dianggap
tidak relevan.
</top>
```

Figure 2: *An example topic. Its English translation — not included in the testbed — is shown in Figure 3.*

the principle of keeping the data as close to original as possible, and did not correct any faults such as spelling mistakes or incomplete sentences [8].

The collection of documents are stored in a single file, marked-up using standard TREC tags. The tags <DOC> and </DOC> mark the beginning and end of a document respectively, and each document has a document identifier delimited by the <DOCNO> and </DOCNO> tags. An example document is shown in Figure 1.

2.2 Topic Construction

The next step in building a testbed is to define a set of queries or *topics* that represent user information needs. (The words topic and query are used interchangeably in this paper.) There are different formats of TREC topics from different years of the workshops [7], with recent examples containing fewer fields. We followed the final ad hoc track format from TREC-8 [9].

The ad hoc topics from TREC-8 have three major fields: title, description, and narrative. The title (encapsulated in a <title> element) is a short title that summarises the information need. The description (<desc>) is a longer, one-sentence description of the topic. The narrative (<narr>) gives more detailed explanation that aims to completely describe which documents are relevant to the query. The topics also have the additional <top> and </top> tags to delineate each query in a file and a <num> element that is used to denote the query identifier.

The Kompas newswire is different in topicality and timespan to the newswire collections used at TREC. Therefore, we defined our own topics following the TREC approach. At TREC, candidate topics are brought by participants and the NIST TREC team decide the final topics by approximating the number of relevant documents per topic [7]. In our case, we began by reading all the 3,000 documents to see what topics were available. With only one native speaker involved in the project, we then limited ourselves to defining twenty topics represented in the collection. These topics have two major types: *general* and *specific*. A general topic is one for which many documents meet the information need. For example, many documents answer the query "Who is Megawati Soekarnoputri's husband?" (topic 9). A specific topic has a small set of document answers and assessors need to read carefully to obtain the answer. An example specific query is "What are the symptoms and causes of asthma?" (topic 10).

An example of a formatted Indonesian topic is shown in Figure 2 and its English translation is shown in Figure 3. To follow the exact format of TREC, the English translations do not form part of the distributed testbed, but are available upon request.

2.3 Relevance Judgements

The final step in constructing a testbed is to make *relevance judgements*, that is, to define which documents are relevant to the information needs expressed by each query. The relevance judgements are then used as the benchmarks to decide whether the documents deemed to be relevant by retrieval systems are indeed relevant according to humans; we report such experiments as an example later.

In TREC, relevance judgements are normally performed though *pooling* [7]. Pooling works as follows: first, for each query, the top 100 documents returned from each *run* (experiment) performed by each participant are returned to NIST; second, a pool is created by grouping together all answers from each run and eliminating duplicates; last, human assessors read each document in the pool and assess its relevance to the topic. The drawback of pooling is that documents that are not retrieved by any system are considered not relevant. Relevance at TREC is binary: a document is either relevant or not, and there is no degree of relevance.

```
<top>
<num> Number:  14
<title> The exchange rate between
rupiah and US dollar
<desc> Description:  Document shall
mention the exchange rate of Indonesian
rupiah against US dollar.
<narr> Narrative:  The document is
relevant as long as it mentions the
exchange rate of rupiah against USA
dollar, even without indication whether
rupiah strengthened or weakened.
Exchange rate prediction is not
relevant.
</top>
```

Figure 3: *English translation of the Indonesian topic shown in Figure 2.*

The number of documents and queries in our testbed is limited, and we did not use pooling. Instead, each of the 3,000 documents was read and judged manually to see whether it was relevant to any of the twenty queries, resulting in an exhaustive tabulation of $20 \times 3,000 = 60,000$ relevance assessments. The relevance judgements are formatted in TREC-like format[3] as follows:

```
14 0  NEWS12738-HTML 0
14 0  NEWS12739-HTML 1
```

The first column indicates the topic number. The second column indicates feedback iteration and this is ignored. The third column is the document identifier appearing between the <DOCNO> and </DOCNO> tags in the documents. The last column indicates whether the document is relevant to the topic. In this example, document NEWS12738-HTML is not relevant to topic 14 and NEWS12739-HTML is relevant. Relevance judgements are ordered by topic, and by document identifier within each topic.

While this testbed is not comparable in size to the 475 megabytes of the TREC Disk 5 Los Angeles Times collection, it is a useful resource that can be extended with collaborative input from other researchers.

3 An Example Evaluation

This section shows an example application of our testbed to Indonesian IR research: evaluation of whether stemming — the removal of suffixes, infixes, and prefixes to derive the morphological root — aids retrieval effectiveness. To experiment with our testbed, we used the zettair search engine[4], which has native support for TREC collections, topics, and relevance assessment. We used a modified Indonesian stemmer that we have recently described [1] to preprocess the collection and topics, and compared this to searching with the unmodified data. A detailed discussion of our stemming approach appears elsewhere [1].

Measure	Without Stemming	With Stemming
Average Precision	0.4394	0.4801
Precision at 10	0.3750	0.3550
R-Precision	0.4210	0.4534

Table 1: *Performance before and after stemming.*

For both the stemmed and unstemmed data, we searched with the titles from the 20 queries on our collection of 3,000 documents, returning 100 answers per query. (The normal TREC practice is to return 1,000 answers but with only 3,000 in the collection we recommend a setting of 100.) We found — in unreported experiments — that title-only search is most effective of all possible combinations of fields. These answers were then evaluated against the relevance judgements using the trec_eval program[5], a standard package used by the TREC workshops to evaluate ad hoc tasks [9]. The software reports different interpretations of *recall* and *precision*: recall is the fraction of relevant answers retrieved, and precision is the fraction of answers that are relevant; recall requires that exhaustive relevance judgements be available, and can therefore be calculated using our testbed.

Table 1 shows the results of our experiment. The first row shows average precision, with stemming improving retrieval perfomance by around 4%. Average precision is calculated by determining the precision after each relevant document is retrieved, summing those precision values, and dividing by the number of relevant documents found; the overall average values for each query are then averaged over all queries. The second row shows average precision after processing 10 documents, averaged over all queries, and shows a drop in performance by 2% for stemming. The final row shows the precision for each query where the number of answers processed equals the number of relevant answers for that query, averaged over all queries. The R-precision results favour stemming by around 3%.

We used the Wilcoxon signed ranked test to examine whether the differences in performance are significant. They are not, at both the 95% and 99% confidence intervals. These results are perhaps surprising; Indonesian words have many more variants than those in English, and we expected that the removal of prefixes, infixes, and suffixes should improve retrieval performance. However, these results are consistent with those observed in English text retrieval [4].

To investigate further, Figure 4 shows the per-query performance. For each topic, three bars are shown: to the left, the total number of relevant documents; in the middle, the number of relevant documents found

[3] http://trec.nist.gov/data/qrels_eng/index.html

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

[5] The trec_eval program can be obtained from ftp://ftp.cs.cornell.edu/pub/smart

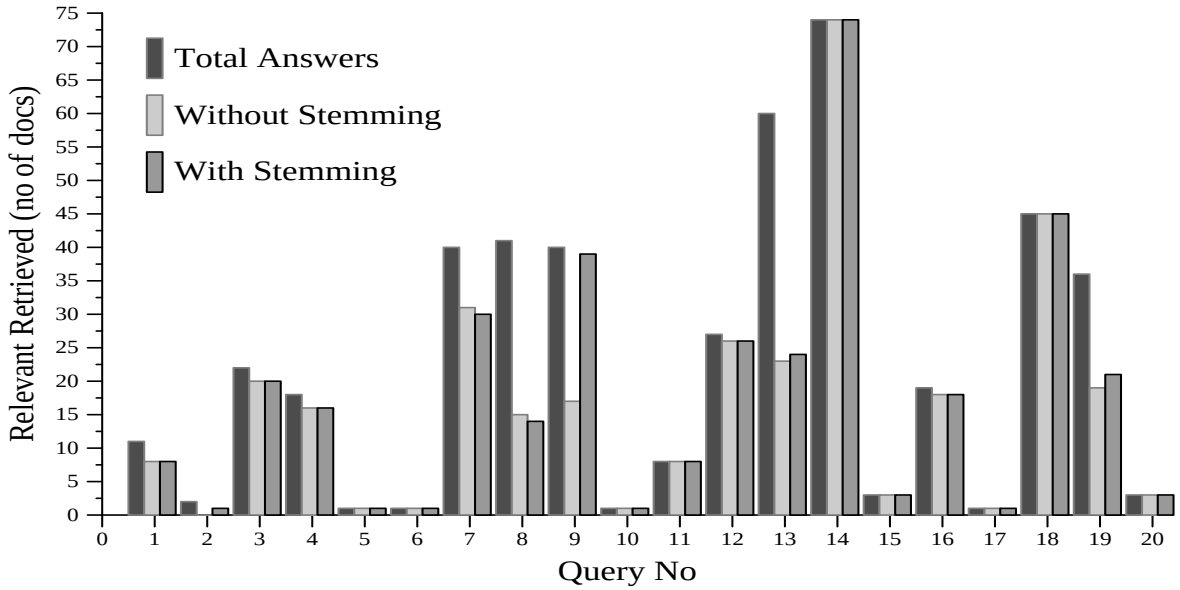

Figure 4: *Topic-by-topic performance with and without stemming. For each topic, the left column shows the number of relevant documents. The middle column shows the number retrieved without stemming, and right column the number retrieved with stemming. The queries used are only the titles.*

without stemming; and, on the right, the number of relevant documents found with stemming. The results show that — with the exception of topics 2, 9, and 19 — there is little difference between performance with and without stemming.

We suspect that this is because some relevant documents answer the query implicitly and do not contain the query terms. For instance the query for "nama bos Manchester United"(the name of the boss of Manchester United) does not retrieve one document that discusses "the manager of MU". A human assessor understands that "manager" is a synonym of "boss" and "MU" is the acronym of "Manchester United"; automated retrieval systems generally use words directly from the query, and stemming is ineffective here.

4 Conclusion

In this paper, we have described the first testbed for Indonesian Information Retrieval. It includes 3,000 documents from newswire texts, 20 topics, and exhaustive relevance judgements. The testbed is stored in the TREC format, and can be used in TREC-like ad hoc evaluations with standard TREC retrieval and evaluation tools.

We have presented a brief experiment with the collection. The experiment shows that an accurate stemmer does not significantly aid retrieval performance on ad hoc queries, despite our expectations to the contrary. We intend to investigate stemming further as a result of this first evaluation. We also plan further fundamental work on techniques for Indonesian Information Retrieval using our testbed.

References

[1] J. Asian, H.E. Williams and S.M.M Tahaghoghi. Stemming Indonesian. In V. Estivill-Castro (editor), *Proceedings of the Twenty-Eighth Australasian Computer Science Conference (ACSC2005)*, 2005. To Appear.

[2] I. Fahmi, March 2004. Personal Communication.

[3] D.K. Harman. Overview of the First TREC conference (TREC-1). In *Proceedings of the First Text REtrieval Conference (TREC-1)*, pages 1–20. NIST Special Publication 500-207, 1992.

[4] D. A. Hull. Stemming Algorithms: A Case Study for Detailed Evaluation. *Journal of the American Society of Information Science*, Volume 47, Number 1, pages 70–84, 1996.

[5] F. Tala. A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia. Master's thesis, University of Amsterdam, July 2003.

[6] V. B. Vega. Information Retrieval for the Indonesian Language. Master's thesis, National University of Singapore, July 2001.

[7] E. M. Voorhees and D.K. Harman. Overview of the Sixth TREC conference (TREC-6). In E.M. Voorhees and D.K. Harman (editors), *Proceedings of the 6th Text REtrieval Conference (TREC-6)*, pages 1–24. NIST Special Publication 500-240, 1997.

[8] E. M. Voorhees and D.K. Harman. Overview of the Eighth TREC conference (TREC-8). In E.M. Voorhees and D.K. Harman (editors), *Proceedings of the 8th Text REtrieval Conference (TREC-8)*, pages 1–24. NIST Special Publication 500-246, 1999.

[9] E. M. Voorhees and D.K. Harman. Overview of the Ninth TREC conference (TREC-9). In E.M. Voorhees and D.K. Harman (editors), *Proceedings of the 9th Text REtrieval Conference (TREC-6)*, pages 1–14. NIST Special Publication 500-249, 2000.

Phrases and Feature Selection in E-Mail Classification

Elisabeth Crawford

Computer Science Department
Carnegie Mellon University
PA 15213 USA

ehc@cs.cmu.edu

Irena Koprinska and Jon Patrick

School of Information Technologies
University of Sydney
NSW 2006 Australia

{*irena,jonpat*}*@it.usyd.edu.au*

Abstract *In this paper we study the effectiveness of using a phrase-based representation in e-mail classification, and the affect this approach has on a number of machine learning algorithms. We also evaluate various feature selection methods and reduction levels for the bag-of-words representation on several learning algorithms and corpora. The results show that the phrase-based representation and feature selection methods can be used to increase the performance of e-mail classifiers.*

Keywords E-Mail Classification, Text Categorization, Feature Selection

1 Introduction

E-mail management is a significant and growing problem for individuals and organisations. E-mail users commonly try to manage the large amount of e-mail they receive by sorting it into folders. Most e-mail managers allow the user to hand construct rules to automatically assign e-mails to folders. However, this feature is rarely used [4]. A system that can automatically *learn* how to classify e-mail messages is desirable.

Several systems for automatic e-mail classification have been developed. Cohen [1] used RIPPER to induce keyword-spotting rules. Bayesian approaches have been used e.g. [11] as well as Nearest-neighbour techniques [13]. Previous studies have been limited because they have failed to explore the use of feature selection and phrase representations. In this paper we show how these techniques can be used to increase classification accuracy.

2 Phrase Representation

Bag of Words (BOW) is the approach most commonly used to represent documents in Text Categorization (TC). In BOW, each document is represented by a vector that contains an importance weighting for every word in the corpus. Phrase-based approaches use whole groups of words as features in the bag and as such preserve more of the document's semantics.

Proceedings of the 9th Australasian Document Computing Symposium, Melbourne, Australia, December 13, 2004.

In order to use a phrase-based document representation we need a method for choosing the phrases. Syntactic and semantically derived phrases have been used without any significant improvement over the BOW approach e.g. [8, 5, 12]. Statistically selected phrases have proved more successful. Mladenic and Grobelnik [9] found that for WWW documents, n-grams (of length 3-4) selected using odds ratio improved the performance of Naive Bayes. Similar results were reported by Furnkranz [6] on Reuters data for the RIPPER classifier using frequency based selection of 2 and 3-grams.

We construct the phrase-based representation used in this paper by first stemming and removing stopwords. We then generate all 1 and 2-grams and place them in a bag of features weighted by their normalized tf-idf. Phrases are then selected statistically using one of the methods described in the next section.

3 Feature Selection

Feature selection is often an essential step in TC as text collections can have more than 100,000 unique terms (words or phrases). Removing less informative and noisy terms reduces the computational cost and often improves classifier generalization. Feature selection works by ranking all the terms and then selecting some percentage. A variety of ranking criteria have been used in TC with varying degrees of success. Some of the more successful approaches are variants of χ^2 [17], information gain [17] and odds ratio [10].

We have chosen to experiment with χ^2, Information Gain (IG) and Document Frequency (DF). We included DF becuase it is computationally efficient and Yang and Pedersen [17] showed that except for aggressive levels of feature selection (bigger than 90 percent), it performed similarly to the first two.

4 Experimental Setup

We use a corpus consisting of e-mail messages for 4 different users (first 4 users in [2]). Each of the users has a different set of criteria for classifiying their e-mails. For instance, users 1 and 3 categorize e-mail mostly on the basis of topic and sender, while user 2 categorizes their mail based on when it needs to be acted upon. User 4 is different again, classifying according to the actions performed (e.g Delete, ReplyAndKeep) as well as topic and sender. The corpora contain between

430 and 972 email messages classified into between 7 and 39 folders. The e-mails in each corpus were ordered by date with the first two thirds becoming the training set and the final third the test set. Note that the strongly temporal nature of e-mail makes cross validation an unsuitable option. For each corpus 5, 10 and 30 percent of the features in the BOW representation were calculated. These numbers were then used to define the number of features selected both for the BOW and phrase representation, i.e. the *same* number of features was used for both.

We have taken the approach of building for each category a binary classifier. This allows e-mails to be placed in more than one category as necessary. We look at six learning algorithms: Support Vector Machines (SVMs), K-Nearest Neighbour (KNN), Decision Trees (DTs), Perceptron, Widrow-Hoff (WH), and Naive Bayes (NB). The following options were used: KNN with cosine similarity as distance metric and distance weighted voting, for k equal to 1, 5 and 30; SMO algorithm for SVMs with both linear and quadratic kernels, C4.5 for DTs and the standard algorithms for WH, NB and Perceptron. We implemented WH and used the WEKA implementations [14] of the other algorithms. Micro-averaged F1 is used to measure the average perfromance of the ML algorithms over multiple categories.

5 Results and Discussion

5.1 Effect of ML aglorithm

Figures 1 to 4 show, for each user, the performance of both the BOW and 2-gram phrase approach for the different learners. For each learner we chose the best result achieved on any of the feature selection settings.

The results demonstrate a strong difference in the difficulty of automatically categorizing different users' e-mails. Consistent with [2] our results show that the coarser grained e-mail sets of users 1 and 3, where many e-mails were classified according to topic, were easier to classify than the finer grained user 2 and 4 corpora. These corpora were harder to classify due to there being less training data per category and the action based classifaction policies of the users.

For the phrase-based representation, SVMs produced the highest classification performance for users 1, 2 and 3, and DTs for user 4; WH and KNN also performed very well. For the BOW representation, KNN produced the highest results for users 2 and 4, WH for user 1 and 3. Overall, SVMs performed the best and NB the worst, which is consistent with Yang and Liu's [16] comparison on Reuters data using BOW. The NB classifier had very high recall, but low preicision. This problem could perhaps be lessened by careful thresholding. WH also performed very well (phrases: 2nd for user 3, 3rd for user 1; BOW: 1st for users 1 and 3). We note that while KNN was the top learner on Reuters and Ohsumed [15], it was less successful on e-mail data. Compared to these corpora, e-mail contains a great deal of noise because of different writing styles and most probably inconsistencies in classifications.

5.2 BOW vs Phrases

Figure 1 shows that on the user 1 corpus, the phrase based representation led to performance increases for all learners except WH and kNN5 with highest improvement for SVM1 and SVM2. On the user 2 data (Figure 2), the phrases worked better on five learners, almost equal on two and worse for NB and KNN1. The biggest improvement was achieved for SVM2 and DTs. Similar results were obtained for user 4 (Figure 4). The classification performance of BOW and phrases is closest on the user 3 corpora: phrases outperformed BOW on four learners, achieved similar performance on another four and were only slightly worse on NB.

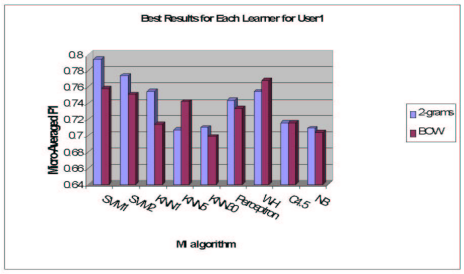

Figure 1: Best results for user 1

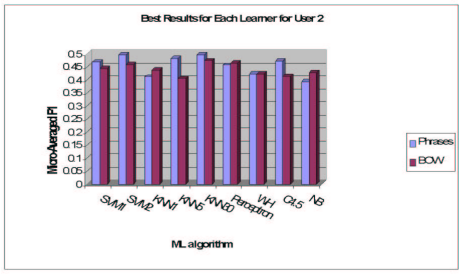

Figure 2: Best results for user 2

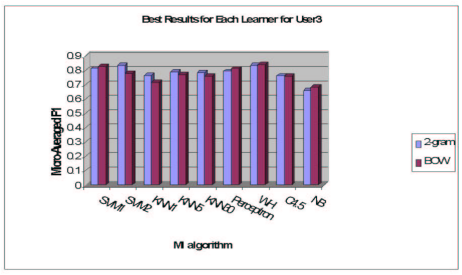

Figure 3: Best results for user 3

The results clearly show that the proposed phrase based representation is useful for e-mail classification — the representation has improved the performance of a variety of ML algorithms over varied corpora. In the next two sections we examine how feature selection can be used to improve e-mail classification using a BOW representation.

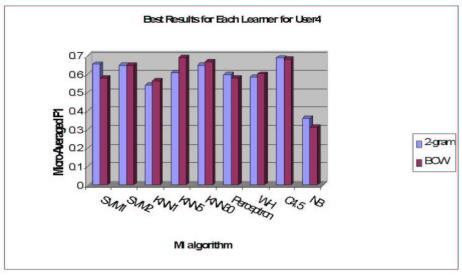

Figure 4: Best results for user 4

5.3 Feature Selection and ML algorithms

The effect of the feature selection algorithms and reduction level has been investigated for BOW on the standard TC corpora. For example, Joachims [7] found feature selection improved the perfromance of KNN and C4.5, but not linear and quadratic SVMs or NB on Reuters and Ohsumed. Yang and Pedersen [17] found that DF, IG and χ^2 have similar characteristics (prefer common terms) and similar performance on Reuters data using KNN and LLSF classifiers. DF was found to perform comparably with IG and χ^2 with up to 90% term removal. To see how these results translate to e-mail data, we first look at the effect of the feature selection algorithm and then the effect of feature selection level.

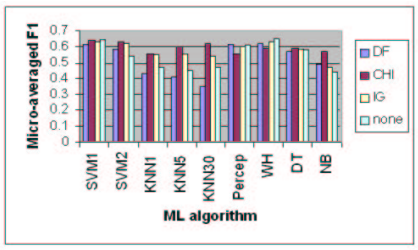

Figure 5: Comparison of feature selection methods across learners for BOW

To investigate the effect of the feature selection methods on each learner, we averaged the performance results across the four corpora and the feature selection levels. Figure 5 shows feature selection was useful for all learners except SVM1 and WH. These results support the theoretical and empirical evidence that SVMs learn independently of the dimensionality of the feature space [7]. Our results show that SVM1 is a very stable classifier regardless of the different feature selection mechanisms and levels. However, for user 4, χ^2 level 5 significantly increases the performance of SVM1 over the full data set. Thus, the corpora has an effect on the usefulness of the feature selection even for stable classifiers.

The results for the quadratic kernel SVM (SVM2), however, are not consistent with [7]. As can be seen, SVM2 benefits from feature selection and χ^2 was found to be the best selector. χ^2 was also found to be the best feature selection method for KNN (k=1, 5 and 30), DT (C4.5) and NB while DF was the best for Perceptron. Unlike Yang and Pedersen, we see that for KNN DF is not comparable to IG and χ^2 even for relatively low feature reduction; in fact it was even worse than using the full feature set. The biggest improvement due to the use of feature selector is for KNN30 and NB, where χ^2 improves the performance by more than 11%. The combination of ML algorithm and feature selection mechanism is clearly important. Different feature selection mechanisms produce different changes in performance across a variety of ML algorithms.

Figure 6 shows the effect of the feature selection level on each learner. We have averaged the effects of the different feature selection algorithms and corpora over the three levels. Overall, level 5 (i.e. 95% term reduction) was the best for KNN, DT and NB. The quadratic kernel SVM (SVM2) performed best with level 30 feature selection while Perceptron, WH and SVM1 performed best without feature selection. NB was the most stable in terms of feature selector and level it always prefers DF and high reduction (level 5, 10).

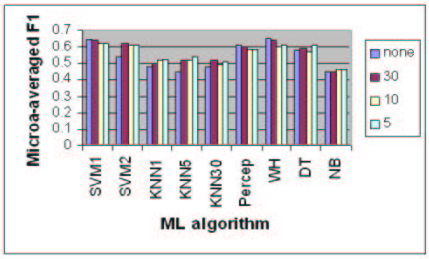

Figure 6: Comparison of feature selection levels across learners for BOW

For each corpora we averaged, the performance of DF, χ^2 and IG averaged over all ML algorithms (figures not shown). We found DF performs slightly worse than IG and χ^2 and there is no evidence that its performance drops at 90% term reduction as Yang and Pedersen found on Reuters. The KNN results (i.e. without the averaging across ML algorithms) also do not show this. On the other hand, like Yang and Pedersen we did find that IG and χ^2 are highly correlated suggesting that this patter might be general as opposed to corpus dependent.

5.4 Corpora and Feature Selection

To study the effect of the *feature selection methods* on each e-mail corpora, we have averaged performance results for all learners and feature selection levels (Figure 7). Users 1 and 3 (who classify based on topic and subject) benefit most from feature selection. Overall, IG is the best feature selector for all corpora and χ^2 obtained almost the same performance as IG on two corpora. For one of the difficult corpora (user 2), the performance of all feature selectors is comparable and not significantly better than not using feature selection. The reason for this is that user 2 classifies e-mails mainly based on the action performed on them, thus, since the correct class cannot always be predicted based on the text, feature selection less useful. Feature selection was

beneficial for the other difficult corpora - user 4. This is because user 4 classifies e-mails according to actions to a lesser extend than user 2 and the vocabulary for this corpus is much larger than any of the others. A large vocabulary implies there may be more noise in the training data makeing feature selection more useful.

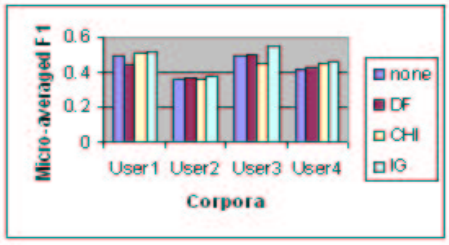

Figure 7: Comparison of feature selection methods across corpora for BOW

Figure 8 shows the effect of the *feature selection levels* on each corpus. We have averaged performance results for all learners and feature selection methods. Aggressive feature selection was found to work best for user 1, 3 and 4. For the more difficult corpora users 2 and 4, the performance of all reduction levels is similar due to the reasons discussed above, overall the moderate level 30 works best.

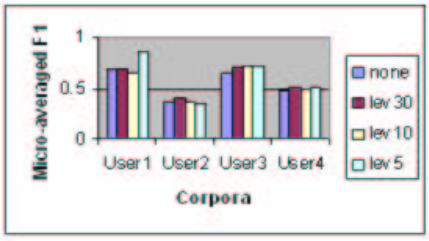

Figure 8: Comparison of feature selection levels across corpora for BOW

6 Conclusions and Future Directions

In summary:

- There can be a strong difference in the difficulty of automatically categorizing different users e-mail;
- The *combination* of ML algorithm and feature selection mechanism effects performance. Thus, it is important for feature selection mechanism in TC to be evaluated across a variety of ML algorithms, which often has not been the case in previous work;
- The phrase-based representation improves performance;
- Overall, SVMs (both linear and quadratic kernel), WH and KNN were found to be the best classifiers for both phrases and BOW;
- Feature selection is useful and overall improved the performance of all ML algorithms except for linear kernel SVM and WH;
- χ^2 was found to be the best feature selector for four out of nine learners;
- Aggressive feature selection (90-95% reduction) worked quite well on all corpora.

When analysing our experiments we noted how performance differed according to the *combination* of ML algorithm, feature selection mechanism and level, text representation and corpora. In the future we would like to explore an approach similar to that described in [3] to choosing a combination of feature selection mechanism and level, text representation and ML algorithm that is best suited to the particular user's e-mail.

References

[1] W. Cohen. Learning rules that classify e-mail. In *AAAI Spring Symposium on Machine Learning in Information Access, pp. 18-25*, 1996.

[2] E. Crawford, J. Kay and E. McCreath. Iems - the intelligent email sorter. In *19th Int. Conf. on Machine Learning*, 2002.

[3] E. Crawford, I. Koprinska and J. Patrick. A multi-learner approach to e-mail classification. In *ADCS*, 2002.

[4] N. Ducheneaut and V. Bellotti. E-mail as habitat: an exploration of embedded personal information management. *Interactions v.8, n.5, pp.30-38*, 2001.

[5] S. Dumais, J. Platt, D. Heckerman and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proc. of CIKM-98*, 1998.

[6] J. Furnkranz. A study using n-gram features for text categorization, 1998.

[7] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *10th ECML*, 1998.

[8] D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proc. of SIGIR-92*, pages 37–50, 1992.

[9] D. Mladenic and M. Grobelnik. Word sequences as features in text learning. In *Proc. of the 17th Electrotechnical and Computer Science Conference*, 1998.

[10] Dunja Mladenic. Feature subset selection in text-learning. In *ECML*, pages 95–100, 1998.

[11] P. Pantel and D. Lin. Spamcop: A spam classification & organization program. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[12] S. Scott and S. Matwin. Feature engineering for text classification. In *Proc. 16th ICML*, 1999.

[13] R. Segal and M. Kephart. Mailcat: An intelligent assistant for organizing e-mail. In *3d Int. Conf. on Autonomous Agents, pp.276-282*, 1999.

[14] I. Witten and E. Frank. *Data Mining - Practical Machine Learning Tools and Techniques with Java Implm.* Morgan Kaufmann, 2000.

[15] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval v.1, n.1/2, pp.69-90*, 1999.

[16] Y. Yang and X. Liu. A re-examination of text categorization methods. In *22nd ACM International Conference on Research and Development in Information Retrieval*, pages 42–49, 1999.

[17] Y. Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *ICML*, pages 412–420, 1997.

Towards a new approach to tightly coupled document collaboration

Speculative Short Paper

Stijn Dekeyser

Mathematics and Computing
University of Southern Queensland
Queensland 4350 Australia

dekeyser@usq.edu.au

Abstract *Currently document collaboration typically proceeds using tools such as* CVS *or vendor-specific Computer Supported Collaborative Work (*CSCW*) and Electronic Meeting (*EM*) messaging systems. Both regulate essentially asynchronous* loosely coupled *collaboration. The prime disadvantages of these technologies are that often documents are checked out or distributed in their entirety and that human interaction is needed in case of unresolvable conflicts. On the side of* tightly coupled *distributed collaborative work, emerging XML databases are employing database-type concurrency control techniques, but unfortunately tend to lock entire documents preventing simultaneous updates. XML-enabled relational databases have the same intrinsic problems, leading to the question if another way is possible.*

In this speculative short paper we describe a novel approach toward tightly coupled document collaboration, involving database-style synchronous client-server collaboration tailored to semi-structured documents. It is partly based on previous theoretic results which introduced path locks to control concurrency on semi-structured data. We also describe how clients may use a future communication protocol based on the path locks.

Keywords Document and XML Databases, Document Management, Document Collaboration.

1 Introduction

Collaboration on digital documents is not only a hot topic, it's a very important one as well. The economic benefits of software allowing easy collaboration on any kind of digital information are enormous. Currently there is a wide range of techniques available, both for general and specific purposes. Often these techniques are classified using the two dimensions of *time* and *location*. Two main classes are physically distributed *asynchronous* collaborative work, and physically distributed *synchronous* collaborative work (we do not discuss collaborative work in one location). We list some of them here, and give a very brief indication of some of their individual drawbacks. Then we proceed with outlining a new approach to document collaboration.

Proceedings of the 9th Australasian Document Computing Symposium, Melbourne, Australia, December 13, 2004.

Asynchronous, loosely coupled

CVS and related technologies. **Drawbacks:** (1) The update method is based on lines in text files, rather than on a conceptual representation of the semantics of a document. (2) In case of multiple updates on a given line, the system requires human intervention to solve the conflict.

Tracking The technique of marking-up sections of a document with change information is typically used by word processing software and other office tools. **Drawbacks:** (1) Multiple instances of a document exist among authors, making file management by hand unavoidable. (2) human intervention is needed to solve conflicts. (3) Documents must be distributed via email, disk, FTP or other means.

CSCW and EMS (Messaging) A large body of commercial systems manage group work from a technical, managerial, and social perspective [10, 16, 1]. The level of automatic synchronization differs from product to product, but often the entire workspace is distributed at each participant's site, while each copy is kept up-to-date by interchanging appropriate control messages [12]. **Drawbacks:** Apart from those of the previous two technologies, a distributed groupware system can suffer concurrency control problems due to events arriving out of order. Other problems are outlined in [12].

Synchronous, tightly coupled

XML-enabled DB Documents can be stored in relational tables, and users can update them using transactions which are based on classic concurrency control mechanisms. **Drawbacks:** it has been shown [4, 9] that relational databases, also those that are XML-enabled, are far from adequate for generic document collaboration

purposes. This is mainly due to locking mechanisms being table-based while several parts of a document (or its entirety) may be stored in the same table.

Native XML DB A document may be converted to XML and stored in emerging native XML databases [4]. **Drawbacks:** as with relational databases, locking mechanisms are currently primarily document-based, meaning that concurrent updates on a single document are not allowed.

Looking only at synchronous collaboration systems, we notice that the main drawbacks of current methods are related to the unsuitability of the concurrency control technique that is being used. What is needed is a new approach that controls concurrency at a logical level within semi-structured documents (e.g. XML [17]), rather than on a document-per-document basis. We will present such an approach in Section 3.

2 Usage Scenarios

In this section we briefly describe some application areas where a new type of collaboration would be beneficial.

2.1 Writing Documents

Typically, researchers using LaTeX collaborate on papers with co-authors using CVS or sending fragments to a designated editor. Authors using other word processors for more general purposes typically exchange entire documents, with parts marked up to represent changes made by different persons. This method is even less satisfactory than using files in CVS repositories.

An alternative approach where the document is stored on a server and clients interact with it synchronously allows for a far better collaboration experience. If a semi-structured locking mechanism is used by the server, logical sections of a document, however small or large, can be locked for write-access. This ensures that the semantics of the document is used in regulating concurrent access, and that human intervention is not required to solve conflicts. Such an approach would also make sense for web content management systems.

2.2 Drawing Plans

Architects, engineers, or artists working in a team on a vector graphic currently have even fewer options than authors of text-based documents [5, 6, 8], unless the graphic can be saved in a text format and kept in a CVS repository as described above. Certain products have facilities akin to those of word processors, where parts are marked up to indicate editing by different persons. However, someone must still manually manage a master document and decide which updates are retained. Other products use collaboration mechanisms based on messaging. Here, too, multiple instances of a single drawing exist, and users must intervene in the management and resolution of conflicting updates. Certainly, in some cases these properties are beneficial, but more often they are not wanted.

Here, too, a different approach would likely be better suited. Consider that the graphic is stored as SVG [11] on a document collaboration server that uses the document's semantics to allow users concurrent access to logical parts of the document while scheduling potentially conflicting updates. Collaborators would see the entire graphic evolve as changes are made to it. Individual authors could be given read- and update restrictions on any logical part, akin to database privileges on tables or views. Only one consistent, authoritative instance — managed by the document collaboration server — exists (but versioning is possible as in Section 2.3) at any given time, although individuals may be allowed to make copies of any version branch.

Some of these features are offered by DR. DWG [6], but this is not a generic tool for use outside the particular application. Furthermore, it involves storing the entire drawing at each user, letting one user at a time make changes which are sent to the other collaborators in real time. Similar comments hold for [13].

2.3 Programming

As a final use case, consider how collaborative programming could be vastly improved if program source code is treated logically rather than as a sequence of lines. As in Section 2.1, the document collaboration server regulates concurrent updates from different clients, disallowing conflicts. The important advantage of using CVS, not just controlling updates but also managing versions, can be effectively simulated in this approach as well. For this, consider that the source code is represented as an XML tree [3]. Special tags at any location in the tree can indicate different versions of code. A smart editor (e.g. Epic [2] already does this) may manipulate these elements in diverse manners, making version management realistic and flexible.

3 Path Lock Concurrency Control

In previous work [9] we have presented a novel concurrency control technique for use in schedulers for semi-structured databases. It uses a new kind of locks which are closely related to XPath queries to indicate precisely which logical parts of a document have been read or written by individual clients.

Clients *query* a document by sending XPath-like statements to the document server. Appropriate path (read) locks are set in the document. The document server's scheduler checks for conflicts with other clients' locks and sends back a node-set if there are no conflicts.

Clients may also *update* parts that they have queried within the context of a transaction, resulting in addi-

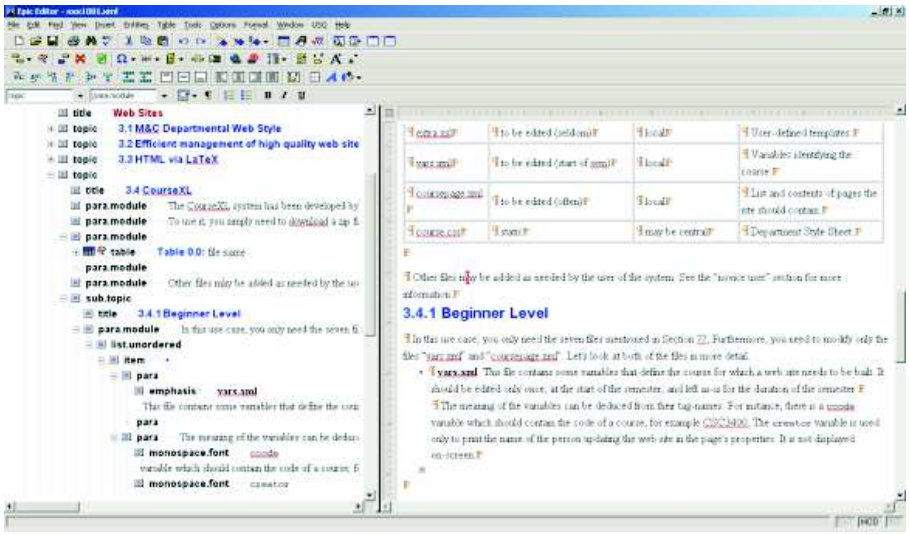

Figure 1: Epic Client View

tional path (write) locks. Path lock conflict rules are described which enable checking for conflicts. The conflict rules exist in two versions, *Path Lock Propagation* and *Path Lock Satisfiability*, representing a trade-off between time and space complexity in conflict checking algorithms using the two schemes.

The paper also introduces two schedulers, a *Conflict* Scheduler and a *Commit* Scheduler. Both schedulers guarantee serializability of schedules consisting of query and update statement belonging to various transactions. Furthermore, their use of Path Locks and corresponding conflict rules allow for a much higher level of concurrency compared to traditional relational and hierarchical methods [9].

3.1 Transaction Operations

Making the above more concrete, we briefly list the operations that clients may use within transactions. Some details, notably constraints on their use, are omitted to increase readability.

A(n, a) The *addition* operation creates a new node n' with tag-name a and an edge (n, n') in the document tree.

D(n) The *deletion* operation removes the node n and the edge incident to n in the document tree.

Q(n, p) The *query* operation returns the set of nodes selected by the path expression p started from the context node n.

C$()$ The *commit* operation signals the end of a transaction.

Note that the above update operations are sufficient to support an XML update language such as XUpdate [14], and that they can update both content and structure.

4 Communication Protocol

As yet unexplored in the context of this paper is a communication protocol (DCP — *document collaboration protocol*) for use by document collaboration servers and client document editors. Clearly, commands sent by the client include the operations listed in Section 3.1. Additionally, a *rollback* and *abort* operation will be needed, as is a *handshake* operation that identifies client and transaction to the scheduler.

Replies from the server include sets of nodes as a result for queries and updates, and notification of conflicts, and failures of both operations and transactions.

As a protocol for communication between a database server and client, it is apparent that state must be preserved during the lifetime of a transaction. And since this technology must be usable over the Internet, integration of the protocol with IP is essential.

5 Client Issues

Perhaps the most exciting challenge to realize this new approach to document collaboration lies with redesigning graphical clients. A general-purpose XML authoring client such as Epic offers a flat and a hierarchical view of the document being edited (see Figure 1).

Clients implementing the DCP protocol described in Section 4 must show the contents of parts of the document as being uncertain. When a client (say, `A`) first starts talking to the server, the full document is received, yielding a flat and hierarchical view as in Figure 1. However, to allow others to edit the document, client `A` must *commit* to release its read locks, which at that point cover the entire document. An end-user working with `A` can subsequently traverse the *assumed* document (possibly requesting refreshes whenever needed), while querying a specific part of the document immediately prior to making a change.

Clearly, the exact query posed before the update will determine the level of concurrent authoring allowed by other clients. Thus, client `A` must allow for an intelligent way to describe the section that will be modified. Many different queries are possible; for example, a positional query (`/section[3]`) may be preferred over a content-specific query (`/section[title='ab']`), with both yielding different path locks and hence resulting in different concurrent updates being allowed or refused.

The above can be summarized as the following formal research problem: given a set of path locks L and a write operation o, give all query expressions q such that the locks required by o and q do not conflict with L.

In addition to intelligently querying the part of the document that the end-user is editing, the client must be flexible in displaying concurrent changes (the conflict rules make sure that no problems can arise, however). For instance, if `A` is working on a specific section while `B` is moving that section to a different location, then upon `B`'s commit, `A` should refresh its view to show the new location of the section being edited. These and other issues with clients need to be researched for both general-purpose and specific-purpose (e.g. drawing) editors.

6 Realization

To realize this type of document collaboration, at least five areas need further research and/or development:

- The Path Locks technique referred to in Section 3 needs to be expanded to allow for more expressive update commands and to support queries expressed in the full surface-syntax of XPath [7]. This research is proceeding smoothly.
- The communication protocol described in Section 4 needs to be defined exactly by network protocol experts. Integration with the Internet Protocol is essential.
- A general-purpose document collaboration server needs to be extended with an implementation of the Path Locks concurrency control theory and the communications protocol. An open source native XML database server such as eXist [15] would be a good target.
- As an alternative to a general-purpose document collaboration server, existing applications such as word processors, vector graphics authoring systems, spreadsheet programs and others could to be extended to include a light-weight server allowing others to collaborate using the communication protocols in their clients.
- Clients need to be re-thought to allow parts of the document they are editing to go out of date, and read-lock only those parts that are actively seen by the end-user, and write-lock the even smaller fragment that is being updated at any given time. Research is necessary to investigate how this can be done generally, while allowing a maximal level of concurrent updates.

References

[1] *Proceeding on the ACM 2002 Conference on Computer Supported Cooperative Work*, New Orleans, Louisiana, USA, November 2002. ACM.

[2] Arbortext. Epic editor overview. Arbortext.com Web Article, 2004. `http://www.arbortext.com/html/epic_editor_overview.html`.

[3] G. Badros. JavaML: A markup language for Java source code. In *Proceedings of the Ninth International Conference on the World Wide Web*, May 2000.

[4] R. Bourret. XML and databases. Website, 2004. `www.rpbourret.com/xml/XMLAndDatabases.htm`.

[5] B. Burchard. Sharing your drawings: An introduction to collaboration. Autodesk.com Web Article, 2004.

[6] Dr DWG Knowledge Center. The CAD View Collaborator. White Paper, 2004. `http://www.drdwg.com/techcenter/cadviewcollaborator.html`.

[7] J. Clark and S. DeRose. XML Path Language (XPath). Recommendation, World Wide Web Consortium (W3C), 1999. `http://www.w3.org/TR/xpath`.

[8] P. Coffee. Collaboration's new age. eWeek.com Web Article, February 2004. `http://www.eweek.com/article2/0%2C1759%2C1539706%2C00.asp`.

[9] S. Dekeyser, J. Hidders and J. Paredaens. A transaction model for XML databases. *World Wide Web Journal*, 2004.

[10] D. Eseryel, R. Ganesan and G. S. Edmonds. Review of computer-supported collaborative work systems. *Educational Technology & Society*, Volume 5, Number 2, 2002.

[11] J. Ferraiolo, J. Fujisawa and D. Jackson. Scalable vector graphics (SVG) 1.1. Recommendation, World Wide Web Consortium (W3C), January 2003. `http://www.w3.org/TR/SVG/`.

[12] S. Greenberg and D. Marwood. Real time groupware as a distributed system: concurrency control and its effect on the interface. In *Proceedings of the ACM conference on CSCW*, pages 207–217. ACM Press, 1994.

[13] C. Ignat and M. Norrie. Grouping/ungrouping in graphical collaborative editing systems. In *ECSCW*, 2003.

[14] L. Martin. XUpdate – XML Update Language. Draft requirements, XML:DB, November 2000.

[15] W. Meier. eXist: An open source native XML database. In *Web, Web-Services, and Database Systems. NODe 2002 Web- and Database-Related Workshops*. Springer LNCS Series 2593, 2002.

[16] J. F. Nunamaker, Alan R. Dennis, Joseph S. Valacich, Douglas Vogel and Joey F. George. Electronic meeting systems. *Commun. ACM*, Volume 34, Number 7, pages 40–61, 1991.

[17] F. Yergeau, T. Bray, J. Paoli, C. M. Sperberg-McQueen and E. Maler. Extensible markup language (XML) 1.0. Recommendation, World Wide Web Consortium (W3C), February 2004. `http://www.w3.org/XML/`.

GOOD Publishing System: Generic Online/Offline Delivery

Short Paper

Jacek Radajewski

Distance and e-Learning Centre
University of Southern Queensland
Queensland 4350 Australia

jacek@usq.edu.au

Sally MacFarlane

Distance and e-Learning Centre
University of Southern Queensland
Queensland 4350 Australia

macfarla@usq.edu.au

Stijn Dekeyser

Mathematics and Computing
University of Southern Queensland
Queensland 4350 Australia

dekeyser@usq.edu.au

Abstract GOOD *is a tailor-made, fully integrated publishing system that creates output documents for multiple media types used in both online and offline teaching modes at the University of Southern Queensland. It is used in the Distance and e-Learning Centre of USQ to create course material for thousands of on-campus, online and external students. Among the end products generated from a single XML input document containing study material for a specific course are study books, introductory books, and web sites in a variety of formats. Future end products currently being investigated include voice rendering. The* GOOD *system is entirely based on open standards such as* XML, XSLT, DOM, *and* XSL:FO *and implemented with* JAVA/J2EE *technology. Among its features is a smart editing client to allow technically non-proficient staff to edit their own course material.*

Keywords Document Management and Publishing, XML authoring.

1 Introduction

USQ is among the leading Australian universities catering to international and off-campus students. The Distance and e-Learning Centre (DeC) provides services to lecturers in publishing educational material both in print form and for web based systems.

Aside from using the GOOD system presented in this paper, course material at USQ is typically written using commercial word processing or publishing software, or LaTeX in the university's mathematics and computing department. Academics working with DeC staff collaborate on a wide range of file formats. The main drawback of this approach is that the aim of "write once, publish in any format" is difficult and expensive to reach. To this end, the DeC in 2000 started work on a generic system to author course material in XML and deliver output in a wide range of formats for both online and offline delivery.

Proceedings of the 9th Australasian Document Computing Symposium, Melbourne, Australia, December 13, 2004.

Related Work

To our knowledge, a complete, fully integrated publishing system specifically for authoring and delivery of course material based on XML technology did not yet exist when work commenced on the one described here. The well-known DocBook [15] system is too complex for use by technically non-proficient academics. Furthermore, DocBook was designed for authoring IT-specific documentation, making its DTD unsuitable for course material.

Additionally, commercial and other XML authoring clients currently assume some knowledge of XML and tend to offer only DTD-based checking of structure. The absence of a more intuitive interface to write course material in XML format is a significant limiting factor for the adoption of such generic editors.

As well as not being an XML-based system (thus making conversion from it to any other format non-trivial in general), LaTeX often is too complex for non-computing oriented staff.

2 GOOD Document Type

To specify the document type for GOOD, the DTD technology was chosen instead of the more expressive XML Schema [9] (XSD) because at that time XSD's specification was not yet a W3C recommendation. Furthermore, types are not so important in this very document-oriented application. Even so, adoption of XSD is planned for the future (see Section 6).

The GOOD DTD [11] consists of four main sections. These are meta-data, introductory material, study modules and selected readings. Cross-referencing is permitted between all sections. We briefly describe each

of these sections. The *meta-data* section holds information about the course, for example course code, year and semester offered. The *introductory* section includes information about assessment, general study resources and faculty policies, as well as the course introduction. The *study modules* section consists essentially of any number of modules, which can be grouped into parts. Modules include the basic content of the course, as well as learning objectives, recommended references and activities. Modules can be further divided into topics, sub-topics and ideas. Arbitrarily deep nesting of topics is not allowed for typesetting and pedagogical reasons. Other elements which are optional in this section include 'executive overview', 'index', 'glossary' and 'appendix'. Finally, the *selected readings* section contains any number of selected readings which may be grouped by module, or be independent of modules. Each reading may include a scanned reading or a reference to a URL.

Elements of varying degrees of granularity, from paragraph up to module, can be marked for conditional delivery. For example, some content may only be intended for print delivery and is required to be excluded from the web delivery. This is implemented as a group of optional attributes on specific elements. Finally, the DTD provides support for images, standard LaTeX equations, and various types of media, such as Macromedia Flash.

2.1 Meta DTD

The GOOD DTD is contained within a "MetaDTD" XML file [11]. Naturally this file is governed by a DTD itself. The MetaDTD document contains 'help' information for each element in the actual GOOD DTD, as well as the DTD definition for each element. This enables developers to easily update the help information as elements are added or altered. From the MetaDTD, using Java, DOM [10] and LaTeX, we render HTML help, PDF documentation, and the GOOD DTD itself. Use of our MetaDTD tool will also facilitate the eventual migration to XML Schema.

2.2 DTD updates

The GOOD DTD is constantly evolving, with attributes and elements being added, deleted and updated. This means that occasionally the DTD is changed in such a way that existing documents become invalid with respect to the new version of the DTD. Each such DTD change is released with an accompanying updater, essentially a SAX [5] parser which is run over existing documents to make any necessary changes. This prevents users from being exposed to validation errors caused by DTD changes.

3 Client: XML Authoring

As mentioned in the introduction, authoring documents conforming to a specific DTD using current editors typically is not very intuitive. For this reason, the GOOD system comprises an editor (Arbortext's Epic [2]) that has been extended with specific GUIs to make editing of course material by end-users much easier. Even so, the GOOD system's architecture is neutral to whatever XML editor is used. Thus, sophisticated end-users may use their preferred tools.

Epic displays the traditional two views of the XML document: a tree-based map view, and a styled-up view using FOSI [6]. The FOSI stylesheets enable automatic numbering of elements such as modules, topics and readings; display of grouped data in html tables; and display of information from cross-referenced elements. Thus, the use of Epic allows authors to get a swift, if inaccurate, feel for how the final product will look.

In the next three sections, we describe two additional features of the extended Epic editor. The first is specific to Epic, while the second and third comprise our own extensions.

3.1 Change Tracking

It is important to the workflow of GOOD documents that users can see the changes made to the document by themselves and other users. Epic implements change tracking using elements (such as add and delete) within a namespace. These elements which are specific to change-tracking remain in the document until the change is either accepted or rejected by a user through Epic.

Epic automatically handles the validation of documents containing change-tracking with respect to the DTD. Outside Epic, however, documents containing change-tracking cannot be validated with respect to the DTD. All the XML parsers used as part of the rendering process are for this reason non-validating, leaving the validation to the editor. As part of future work, we will investigate the development of an XML Schema that would accept both forms of the document.

3.2 Custom-built Java Swing GUIs

There are also a number of custom-built GUIs, written in Swing [8], which are used to simplify certain operations for the user.

The GOOD DTD supports some twenty reference types, including book, database, journal and email. The user enters information through a reference GUI which clearly indicates mandatory fields and other requirements, based on the reference type. These GUIs are based on a common framework so that it is relatively simple to change the requirements for existing reference types or add new ones. Fields or groups of fields that are commonly used across various reference types, for example URL or group of authors, are represented by Java classes.

The GOOD DTD specifies a number of cross-reference types, each referencing different elements, including textbook, reading, resource and module, as well as a generic cross-reference that can reference any type of element. The cross-referencing GUI

displays a tree of elements, filtered depending on the cross-reference type, from which the user selects a cross-reference. This enables users to easily select an element to cross-reference, with validation performed by the application.

There is also a GUI to enable the user to perform *check in/out* operations, without knowing anything about the version control system in use. Access to documents is restricted by access control lists. Also, the user is able to view the HTML help for any element in the GOOD DTD. This help is produced from the MetaDTD presented in Section 2.1. The LaTeX *viewer* GUI enables the user to preview LaTeX equations. Finally, the *render* GUI allows the user to specify exactly what they wish to produce, for example web, print, draft, cd label or print cover pages.

3.3 Document Checker

Some constraints on document contents cannot be specified by DTD or schema. These constraints include restrictions on empty elements, duplicate references and invalid URLs. The *document checker* GUI performs this validation, reporting errors and warnings back to the user, with a cross-reference to the location of each error so that the user can easily rectify the problem.

4 Rendering Servers

The server side of the GOOD system is mainly concerned with managing course documents and rendering them to any of the supported output formats. The rendering phase is somewhat different for print products than for web publishing. However, both have a number of steps in common, as described next.

4.1 Common Rendering Steps

The rendering process essentially consists of a number of sequential DOM and SAX parses and transformations of the document, after which the resultant XML is transformed using XSLT to the desired output type. As mentioned previously, all of these XML parsers are non-validating.

The *SAX Document Builder* performs initial processing of the document. This includes normalizing filenames and converting images to different formats depending on the target. For PostScript output, high quality EPS images can be used, while for PDF or web output these images are converted to a lower quality image format. The SAX Document Builder also dereferences non-ASCII character entities.

Next, the *Delivery Filter* filters out content which is marked not to be delivered to the render target. This is done before pre-processing so that excluded sections do not affect the numbering or format of the final output.

To enable users to view the effect of their changes on the rendered output, GOOD has two options for rendering documents containing change tracking. Users can render the document "without" change tracking, that is render the document as though all changes had been accepted. There is also an option to render "with" change tracking, that is render the document with added content highlighted in green and deleted content in red. The change tracking filter makes the necessary changes to the document.

The *Pre-processor* uses both XSL and Java to number certain elements, collate reference lists, normalise image height and width attributes and populate element titles.

4.2 Print Rendering Steps

After pre-processing, the print render process is simple. To generate the XSL:FO [1] document, the appropriate XSLT stylesheet is chosen, depending on the document type being rendered (study book, introductory book, selected readings or solutions manual). XEP [16] is then used to render either PDF or PostScript from the FO document. The table of contents, including bookmarks, is done by the XSL. Cross-references are rendered as internal links.

4.3 Web Rendering Steps

Unlike print output, web output consists of any number of separate pages, with cross-references rendered as links within and between these pages. A SAX parser processes the cross-references, modifying the links so that they will work once the content is broken up into separate pages. The content is then broken up into chunks, each chunk representing a discrete section of the course, for example module, reading or course overview. Each chunk is then passed to an XHTML XSL transformer to generate a separate HTML page. The same chunk is passed to an XSL:FO transformer to produce a PDF version of the same page.

The navigation for the site, including a JavaScript menu and site map, are also generated using XSL. Images, CSS and other content are then packaged with the generated XHTML files to make a complete web site. For output of hypertext files to a hybrid cd, an ISO-9660 CD image file is created.

5 Implementation Issues

We now turn to a few implementation-related issues in the GOOD system. Since this is a short technology paper, this section is necessarily brief. More information about the system's design and implementation can be found in [13].

5.1 Scalability

At present, the GOOD system is used primarily by DeC staff and a small number of academics from different faculties. As the number of users of the GOOD system grows, scalability is increasingly important. A number of users may be rendering concurrently, or simultaneously require access to the document repository. The following two sections describe how the GOOD system is designed to deal with such issues.

5.1.1 JMS render queue

To enable multiple users to render concurrently, the render process is implemented as an asynchronous process. The client sends a render request which is queued on a JMS [14] queue of render jobs. There are a number of distributed rendering servers, each with a Message Driven Bean [14] monitoring this queue. The Message Driven Bean picks up the job and initiates the render. This solution is scalable since while each rendering server executes only one render at a time, any number of rendering servers can connect to the same queue.

The rendering server sends progress messages to the client, at various stages of the render process. When the render completes, the output file is automatically opened on the user's machine.

5.1.2 XEP memory use

XEP uses a significant amount of memory, particularly for rendering images. Initially some documents were causing out of memory errors on the rendering servers. There is now a separate queue and rendering server for large render jobs.

5.2 Version control

GOOD currently uses CVS for version control. CVS commands are run from shell scripts called from Java. There are some transaction issues with this, since it can be difficult to tell whether the shell script has failed.

6 Future Work

While GOOD is already in operation, it has a sizable wish list attached. We classify each of these future features based on a time-frame for their implementation.

Short Term

- *Role-based access* to course documents (Read-only, Write, Team Leader).
- *Document preferences* allowing a user to set up preferences relating to document structure, so that based on these preferences we can create a course document skeleton. Also for example, when they add a module, the editor will be able to set attribute values and create required elements based on these preferences.
- *Access to* GOOD from outside USQ.

Medium Term

- *Migration to XML Schema* [9] instead of DTD.
- *Voice rendering* using VoiceXML [12] to support visually impaired students.
- *Closer integration* with other university systems such as calendar and course specifications.
- *Cross-platform support*.

Long Term

- *Migration to XML Databases* [3] instead of CVS.
- *Concurrent Authoring* either by use of CVS or more elaborate mechanisms [3, 7].
- *Import/Export* to and from LaTeX and to and from the emerging Open Document Standard [4].

References

[1] S. Adler, A. Berglund, J. Caruso et al. Extensible stylesheet language (XSL) version 1.0. Recommendation, World Wide Web Consortium (W3C), October 2001. http://www.w3.org/TR/xsl.

[2] Arbortext. Epic editor overview. Arbortext.com Web Article, 2004. http://www.arbortext.com/html/epic_editor_overview.html.

[3] R. Bourret. XML and databases. Website, 2004. www.rpbourret.com/xml/XMLAndDatabases.htm.

[4] M. Brauer, G. Edwards, D. Vogelheim et al. Open office specification 1.0. Committee draft 1, Oasis, March 2004. http://www.oasis-open.org/committees/office/.

[5] D. Brownell. *SAX 2*. O'Reilly, January 2002. ISBN: 0-596-00237-8.

[6] G. Charlebois. Standard generalized markup language (SGML): Overview and new developments. *Network Notes*, Volume 3, 1994. http://www.collectionscanada.ca/9/1/p1-202-e.html.

[7] S. Dekeyser, J. Hidders and J. Paredaens. A transaction model for XML databases. *World Wide Web Journal*, 2004.

[8] R. Eckstein, M. Loy, D. Wood et al. *Java Swing*. O'Reilly, second edition, 2002. ISBN: 0-596-00408-7.

[9] D. Fallside. XML Schema. Recommendation, World Wide Web Consortium (W3C), May 2001. http://www.w3.org/XML/Schema.

[10] A. Le Hors, P. Le Hegaret, L. Wood et al. Document object model (DOM) level 2. Recommendation, World Wide Web Consortium (W3C), November 2000. http://www.w3.org/DOM/DOMTR.

[11] S. MacFarlane, J. Radajewski et al. GOOD and MetaDTD DTDs. Technical report, Distance and e-Learning Centre, USQ, 2004. http://www.sci.usq.edu.au/staff/dekeyser/Good/index.php.

[12] S. McGlashan, D. Burnett, J. Carter et al. Voice extensible markup language (VoiceXML) version 2.0. Recommendation, W3C, March 2004. http://www.w3.org/TR/voicexml20/.

[13] J. Radajewski et al. The GOOD System. Technical report, Distance and e-Learning Centre, USQ, 2004.

[14] B. Shannon. Java 2 platform, enterprise edition (J2EE). Specification, v1.4, Sun Microsystems, 2003. http://java.sun.com/j2ee/j2ee-1_4-fr-spec.pdf.

[15] N. Walsh and L. Muellner. *DocBook: The Definitive Guide*. O'Reilly, October 1999. ISBN: 1-56592-580-7.

[16] xAttic. XEP XSL Rendering Engine. xAttic.com Web Article, 2003.

NLPX - An XML-IR System with a Natural Language Interface

Alan Woodley

Centre for Information Technology Innovation
Faculty of Information Technology
Queensland University of Technology
Queensland 4001 Australia

ap.woodley@student.qut.edu.au

Shlomo Geva

Centre for Information Technology Innovation
Faculty of Information Technology
Queensland University of Technology
Queensland 4001 Australia

s.geva@qut.edu.au

Abstract *Traditional information retrieval (IR) systems respond to user queries with ranked lists of relevant documents. The separation of content and structure in XML documents allows individual XML elements to be selected in isolation. Thus, users expect XML-IR systems to return highly relevant results that are more precise than entire documents. This paper presents such a system. The system accepts queries in both natural language (English) and formal XPath-like format (NEXI) and matches to a set of relevant and appropriately-sized elements using an effective ranking scheme.*

Keywords Information Retrieval, Natural Language Queries

1.0 Introduction

The widespread use of Extensible Markup Language (XML) documents in digital libraries has lead to development of information retrieval (IR) methods specifically designed for XML collections. Most traditional IR systems are limited to whole document retrieval; however, since XML documents separate content and structure, XML-IR systems are able to retrieve the relevant portions of documents. Users interacting with XML-IR system could potentially receive highly relevant and highly precise material. However, it also means that XML-IR systems are more complex than their traditional counterparts.

We describe a system that attempts to solve some of the challenging problems of XML-IR. In what follows, we first describe how queries are interpreted by the system. Two query formats are examined: natural language, and NEXI queries, an XPath variant where users express their information need in a formal language. We then very briefly describe the internal storage structure of the XML collection and the ranking scheme that is used to order results. Finally we present some performance results from the INEX 2004 Workshop.

Proceedings of the 9th Australian Document Computing Symposium, Melbourne, Australia, December 13, 2004.

2.0 Query Interpretation

The system presented here was designed to participate in the 2004 Initiative for the Evaluation of XML Retrieval (INEX) Workshop [2]. The INEX Workshop is similar to the TREC workshop. It is an annual event that provides a world-class benchmark for the evaluation of XML systems. INEX provides a test collection of 12,000 IEEE journal articles, a set of queries and a set of evaluation metrics. Two types of queries are used in INEX: CO and CAS. Content Only (CO) queries ignore document structure and only contain content requirements. Contrastly, Content and Structure (CAS) queries explicitly express both content and structural requirements. Both CO and CAS queries are expected to return appropriately sized elements – not just whole documents. Figures 1 and 2 are examples of both query types.

```
<inex_topic topic_id="XX" query_type="CO">
<title>
  "multi layer perceptron" "radial basis
functions" comparison
</title>
<description>
  The relationship and comparisons between
radial basis functions and multi layer
perceptrons
</description>
</inex_topic>
```

Figure 1 A CO Query

Both the description and title tags express users' information needs. The description expresses users' need in a natural language (e.g. English). The title expresses users' information need in either a list of keywords/phrases (CO) or as a formal XPath-like language (CAS) called Narrowed Extended XPath I (NEXI) [5].

```
<inex_topic topic_id="XX"
query_type="CAS">
<title>
  //article[about(.,information
retrieval)]//sec[about(.,compression)]
</title>
<description>
  Find sections about compression in
articles about information retrieval.
</description>
</inex_topic>
```

Figure 2 A CAS Query

NEXI's syntax is **//A[about(//B,C)]** where **A** is the context path, **B** is the relative path and **C** is the content requirement. Each **'about'** clause represents an individual information request. So the query **//A[about(//B,C)]//X[about(//Y,Z)]** contains two requests: **//A[about(//B,C)]** and **//A//X[about(//Y,Z)]**. However, in NEXI only elements matching the leaf (i.e. rightmost) **'about'** clause are returned to the user, and the others are used to support the return elements in ranking.

In 2004 INEX introduced its natural language track. At the INEX 2003 Workshop more than two-thirds of the proposed queries had major semantic or syntactic errors [4] that required 12 rounds of corrections. Since experts in the field of structured information retrieval are unable to easily use formal query languages, one cannot expect an inexperienced user to do so. However, most users are able to intuitively express their information need in a natural language. There already exists an extensive body of research into natural language processing in the specific area of Information Retrieval, largely thanks to The Text Retrieval Conference (TREC) and the Special Interest Group for Information Retrieval (ACM-SIGIR). However, work on an XML-IR interface is still largely un-documented and many problems remain unsolved.

2.1 Natural Language Query (NLQ) to NEXI Translator

Our system was originally developed for participation in the Ad-hoc track using NEXI. We adapted it to handle natural language queries by converting NLQs to NEXI.

Step 1 Lexical and Semantic Tagging

Suppose that the description tags in Figure 1 and 2 are input into the system as natural language queries (NLQ). Translating the NLQs into NEXI format takes several steps. First each word is tagged as either as a special connotation or by its part of speech. Special connotations are words of implied semantic significance within the system. Our system uses three types of special connotations: structural words that indicate the structural requirement of the user (e.g. article, section, paragraph, etc.), boundary words that separate the user's structural and content requirements (e.g. about, containing) and instruction words that indicate if we have a return or support request. All other words are tagged by their part of speech. Any part-of-speech tagger could perform this task; however, our system uses the Brill Tagger [1]. Figure 3 is an example of the NLQ after tagging.

NLQ 1: The/DT relationship/NN and/CC comparisons/NNS between/IN radial/JJ basis/NN functions/NNS and/CC multi/NNS layer/NN perceptions/NN
NLQ 2: Find/XIN sections/XST about/XBD compression/NN in/IN articles/XST about/XBD information/NN retrieval/NN

Figure 3 A Tagged CO and CAS Natural Language Query

Step 2 Template Matching

The translator's second task is to derive information requests from the tagged NLQ by matching the tagged NLQ to a predefined set of grammar templates. The grammar templates were developed by inspection of previous years' INEX queries. NLQs have a narrow context and require the understanding of only a subset of natural language. A system that interprets NLQs requires fewer rules than a system that attempts to understand natural language in general. Inspection of previous INEX queries reveals that most queries correspond to a small set of patterns. By extracting these patterns we were able to formulate grammar templates that matched a majority of queries. Figure 4 shows some of the grammar templates.

Query: Request+
Request : CO_Request | CAS_Request
CO_Request: NounPhrase+
CAS_Request: SupportRequest | ReturnRequest
SupportRequest: Structure [Bound] NounPhrase+
ReturnRequest: Instruction Structure [Bound] NounPhrase+

Figure 4 Grammar Templates

Each grammar template corresponds to an individual information request. Each

information request has three attributes: **Content,** a list of terms or phrases expressing users content requirements, **Structure,** a logical XPath expression that describes the structural constraints of the request. And **Instruction,** "R" if we have a return request or "S" if we have a support request. Figure 5 is an example of the information requests derived from the templates.

NLQ 1:
Structure: /*
Content: relationship, comparisons, radial basis functions, multi layer perceptions
Instruction: R

NLQ 2:

	Request 1	**Request 2**
Structural:	/article/sec	/articlec
Content:	compression	information retrieval
Instruction:	R	S

Figure 5 Derived Information Requests

Step 3 NEXI Query Production

The final step in the translator is to merge the information request into a single NEXI query. Return requests are output in the form **A[about(.,C)]** where A is the request structural attribute and C is the request content attribute. To add support requests, we must first locate the longest matching string in the return request and then add the support request in the form **D[about(E,F)]** where D is the longest matching string, E is the remainder of the support request structural attribute and F, is the support requests content attribute.

Figure 6 is how the NEXI queries would appear after the information requests for each NLQ have been merged.

NLQ 1:
//*[about(**.,**relationship, comparisons, radial basis functions, multi layer perceptions)]
NLQ 2:
//article[about(.,information retrieval)]//sec[about(.,compression)]

Figure 6 NLQ-to-NEXI Queries

2.2 Processing NEXI Queries

Once NEXI queries are input into the system they are converted into an intermediate language called the RS query language. The RS query language converts NEXI queries to a set of information requests. The format of RS queries is

Request: Instruction 'l' Retrieve_Filter 'l' Search_Filter 'l' Content.

The Instruction and Content attributes are the same as they were in the previous section; however, the Structural attribute has been divided into a Retrieve and Search Filter. While both are logical XPath expressions the **Retrieve Filter** describes which elements should be retrieved by the system, while, the **Search Filter** describes which elements should be searched by the system. Figure 7 is an example of the queries introduced earlier converted to RS queries.

RS Query 1:
R|//*|//*| relationship, comparisons, radial basis functions, multi layer perceptions
RS Query 2:
R|//article//sec|//article//sec|compression
S|//article|//article| information retrieval

Figure 7 An Example of an RS Query

3.0 System Structure

We index the XML collection using an inverted list. Given a query term we can derive the filename, physical XPath and the ordinal position within the XPath that it occurred in. From there we construct a partial XML tree containing every relevant leaf element for each document that contains a query term. Further information on our structure can be found in [3].

4.0 Ranking Scheme

Elements are ranked according to their relevance. Data in an XML tree is mostly stored in leaf elements. So first we calculate the score of relevant leaf elements, then, we propagate their scores to their ancestor branch elements.

The relevance score of leaf elements is computed from term frequencies within the leaf elements normalised by their global collection frequency. The scoring scheme rewards elements with more query terms. However, it penalises elements with frequently occurring query terms, and rewards elements that contain more distinct query terms.

The relevance score of a non-leaf is the sum of the children scores. However leaf element scores are moderated by a slight decay

factor as they propagate up the tree. Branch elements with multiple relevant children are likely to be ranked higher then their descendents – as they are more comprehensive - while branch elements with a single relevant child will be ranked lower than the child element as they are less specific.

5.0 Results

The system was entered into both the Ad-hoc and NLP tracks at INEX2004. In the Ad-hoc track the system ranked 1st from 52 submitted runs in the VCAS task, and 6th from 70 submitted runs in the CO task. In the NLP track the system was ranked 1st in the VCAS task and 2nd in the CO task. While the NLP track was limited to 9 participants initially, of which only 4 made official submissions, the most encouraging outcome was that the NLP system outperformed several Ad-Hoc systems. In fact, if the NLP submission was entered in the Ad-hoc track it would have ranked 12th from 52 in VCAS and 13th from 70 in CO. This seems to suggest that in structured IR, natural language queries have the potential to be a viable alternative, albeit not as precise, to a formal query language such as NEXI (an XPath derivative).

The Recall/Precision Curves for the Ad-hoc track, along with the R/P curve for our NLP runs are presented in Figures 8 and 9. The top bold curve is the Ad-hoc curve, the lower is the NLP curve, and the background curves are of all the official Ad-hoc runs at INEX 2004.

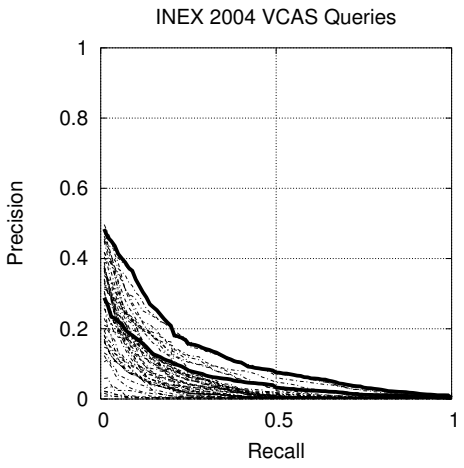

Figure 8 The INEX 2004 VCAS R/P Curve

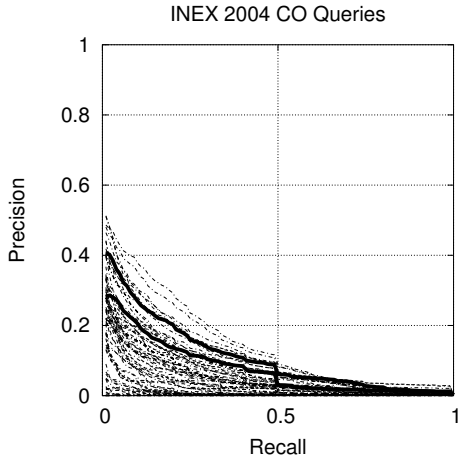

Figure 9 The 2004 INEX CO R/P Curve.

6.0 Conclusion and Future Outlook

This paper presents an XML-IR system responds to user queries with relevant and appropriately sized results. Our ranking scheme is comparable with the best INEX alternatives. The NLP interface requires further development; however, initial results are promising. The system provides a working example of the potential of XML-IR systems.

References

[1] E. Brill. A Simple Rule-Based Part of Speech Tagger. In *Proceedings of the Third Conference on Applied Computational Linguistics* (ACL), Trento, Italy. 1992.

[2] N. Fuhr and S. Malik. Overview of the Initiative for the Evaluation of XML Retrieval (INEX) 2003. In *INEX 2003 Workshop Proceedings*, Schloss Dagstuhl, Germany, December 15-17, 2003, pages 1-11. 2004.

[3] S. Geva and M. Spork. XPath Inverted File for Information Retrieval, In *INEX 2003 Workshop Proceedings*, Schloss Dagstuhl, Germany, December 15-17,2003, pages 110-117. 2004.

[4] Trotman, A. and O'Keefe, "The Simplest Query Language That Could Possibly Work", In *INEX 2003 Workshop Proceedings*, Schloss Dagstuhl, Germany, December 15-17,2003, pages 167-174, 2004.

[5] A. Trotman and B. Sigurbjörnsson, *Narrowed Extended XPath I (NEXI)*, http://www.cs.otago.ac.nz/postgrads/andrew/2004-4.pdf, 2004.