ADCS 2008

Proceedings of the Thirteenth Australasian Document Computing Symposium

8 December 2008

Edited by
Rob McArthur, Paul Thomas, Andrew Turpin and Mingfang Wu

Proceedings of the Thirteenth Australasian Document Computing Symposium

Tasmanian ICT Centre, CSIRO, Hobart, Tasmania
8 December 2008

Published by
School of Computer Science and Information Technology,
RMIT University,
Melbourne VIC 3001, Australia.

Editors

Rob McArthur
Paul Thomas
Andrew Turpin
Mingfang Wu

ISBN: 13978 1 921426 21 6

http://es.csiro.au/adcs2008

Proceedings of the Thirteenth Australasian Document Computing Symposium
Hobart
8 December 2008

Chair's Preface

These proceedings contain the papers of the Thirteenth Australasian Document Computing Symposium hosted by the Tasmanian ICT Centre, CSIRO in Hobart, Tasmania.

The two keynote talks, seven long papers and three short papers reflect the breadth of interest of the Australian research community in the area of document computing. Continuing the tradition begun in 2007, ADCS is not only collocated with The Australasian Language Technology Workshop 2008, but we are sharing a paper session, keynote talk, and social functions with the Australian natural language research community.

The quality of submissions was very high this year. Of the 12 full papers submitted, 8 were accepted for presentation as full papers (67%), 1 was accepted as a short paper (8%), and 3 were rejected (25%). All of the 3 short papers submitted were accepted (100%).

All submissions received at least two anonymous reviews by experts in the area, while 60% of the papers received 3 reviews. Dual submissions were explicitly prohibited.

The members of the program committee and extra reviewers deserve special thanks for their professional reviews all received in the short time required for this ADCS conference.

We would also like to thank Funnelback, CSIRO, NICTA (Victoria) and RMIT School of Computer Science and IT for their generous sponsorship of the event.

The symposium includes many formal presentations, but perhaps its greatest benefit lies in the opportunity it provides for document computing practitioners and researchers to get together and informally share ideas and enthusiasm.

Symposium Co-Chairs

Rob McArthur	CSIRO ICT
Mingfang Wu	RMIT University

Program co-chairs

Paul Thomas	CSIRO ICT
Andrew Turpin	RMIT University

Program Committee

Peter Bailey	Microsoft Research
Peter Bruza	Queensland University of Technology
Bob Colomb	University of Technology Malaysia
Stijn Dekeyser	University of Southern Queensland
Shlomo Geva	Queensland University of Technology
David Hawking	Funnelback Pty Ltd
Judy Kay	University of Sydney
Robert McArthur	CSIRO ICT
Alistair Moffat	University of Melbourne
Gitesh K. Raikundalia	Victoria University
Falk Scholer	RMIT University
Milad Shokouhi	Microsoft Research, Cambridge
Amanda Spink	Queensland University of Technology
Jamie Thom	RMIT University
Andrew Trotman	University of Otago
Anh Vo	University of Melbourne
Ross Wilkinson	CSIRO
William Webber	University of Melbourne
Justin Zobel	NICTA

Additional Reviewer

Jing He	Victoria University

ADCS Advisory Committee

Peter Bruza	Queensland University of Technology
Judy Kay	The University of Sydney
David Hawking	Funnelback
Alistair Moffat	The University of Melbourne
Amanda Spink	Queensland University of Technology
Ross Wilkinson	CSIRO, Canberra
Justin Zobel	NICTA

Contents

Quo vadis information retrieval research

Kal Järvelin

The talk begins by an analysis of the tradition of IR research based on the TREC approach. The structure of IR experiments, so characteristic for IR research, is discussed in detail, with IR effectiveness as the dominating dependent variable. The strengths of the approach in research and in practice are acknowledged. The talk then moves on to pointing out mounting challenges to IR (evaluation) and IR system development:

- Sometimes there are no articulated information needs preceding information access. The searcher first needs to find a focus, then develop questions. Is it possible to support the searcher?
- Sometimes there are neither unique questions / queries nor unique right answers. The searchers are individual and inconsistent throughout. Is it possible to support the searcher?
- Practical IR is often a process, not a single shot at the database, while IR evaluation is by-and-large based on one-query sessions. IR processes have rarely been sufficiently described in recent times. Therefore they cannot be understood, properly supported, nor evaluated.
- Practical IR is rarely performed in vacuum at the center of the universe. Rather it is highly integrated with the other components of the searcher's information environment. This is unfortunately not reflected in IR evaluation.
- Relevance is dynamic and multidimensional while measured in the opposite way as stable, topical and binary. Simple measures like searchers' clicks are increasingly taken as indications of relevance while they are insufficient and do not reliably predict relevance.
- Searchers' task performance is independent of IR system effectiveness. People cope with clearly degraded retrieval systems as well as with better ones. Just their behavior is changed. Are we investing our research efforts optimally?

All this suggests that, in addition to search engine development, IR (experimentation) might be deserving of other serious foci. These studies might not almost exclusively, and certainly not primarily, focus on search engine effectiveness as the paramount dependent variable. The talk finishes by discussing a cognitive approach to IR as a way for developing material theories on information access with more varied designs of dependent and independent variables. It is the author's view that, by lifting one's eyes from the search engine effectiveness fixation, it is readily understood that 80% of the IR terrain is unmapped, even from the CS viewpoint. Or, if no alternative approaches matter, we might close up shop.

Kal Järvelin is an Academy Professor at the Academy of Finland, working at the Dept. of Information Studies, University of Tampere. He holds a PhD in Information Studies (1987) from the same university. Kal's research covers information seeking and retrieval, database management, and structured documents; and linguistic and conceptual methods in IR. He has authored over 200 scholarly publications and supervised fourteen doctoral dissertations. Kal has served the ACM SIGIR Conferences as a program committee member (1992-2005), Conference Chair (2002) and Program Co-Chair (2004, 2006). He is an Associate Editor of Information Processing and Management (USA).

Syntactic and semantic structure in web search queries

Rosie Jones

Traditionally, information retrieval examines the search query in isolation: a query is used to retrieve documents, and the relevance of the documents returned is evaluated in relation to that query. The query itself is assumed to consist of a bag of words, without any grammatical structure. However, queries can also be shown to exhibit grammatical structure, often consisting of telegraphic noun-phrases. In addition, users typically conduct web and other types of searches in sessions, issuing a query, examining results, and then re-issuing a modified query to improve the results. We describe the properties of real web search sessions, and show that users conduct searches for both broad and finer grained tasks, which can be both interleaved and nested. Reformulations reflect many relationships, including synonymy, hypernymy and hyponomy. We show that user search reformulations can be mined to identify related terms, and that we can identify the boundaries between tasks with greater accuracy than previous methods.

Rosie Jones is a Senior Research Scientist at Yahoo!. Her research interests include web search, geographic information retrieval, and natural language processing. She received her PhD from the School of Computer Science at Carnegie Mellon University under the supervision of Tom Mitchell. She is co-organizing the WSDM 2009 Workshop on Web Search Click Data (WSCD09). She served on the Senior PC for SIGIR in 2007 and 2008, and is a Senior Member of the ACM.

Term-Frequency Surrogates in Text Similarity Computations

Stefan Pohl *Alistair Moffat*

NICTA Victoria Research Laboratory,
Department of Computer Science and Software Engineering
The University of Melbourne
Victoria 3010, Australia

{*spohl,alistair*}*@csse.unimelb.edu.au*

Abstract *Inverted indexes on external storage perform best when accesses are ordered and data is read sequentially, so that seek times are minimized. As a consequence, the various items required to compute Boolean, ranked and phrase queries are often interleaved in the inverted lists. While suitable for query types in which all items are required, this arrangement has the drawback that other query types – notably pure ranked queries and conjunctive Boolean queries – do not require access to word position information, and that component of each posting must be bypassed when these queries are being handled. In this paper we show that the term frequency component of each posting can be completely replaced by a surrogate that allows skipping of positional information interleaved in inverted lists, and obtain significant speedups in ranked query execution without increasing the index size, and without harming retrieval effectiveness. We also explore two methods of reconstituting approximations to the original term frequencies that can be employed if use of the surrogates is deemed too risky. Our simple improvement can thus be used with all ranking functions that make use of term frequencies.*

Keywords Information retrieval, inverted index, skip pointer, proximity query, efficiency, effectiveness.

1 Introduction

Web search is an expensive operation, on which many millions of dollars are spent each year in terms of computing and energy costs. Small improvements in search efficiency can thus yield significant monetary savings, and are of considerable interest to the web search industry. Users of web search services are fickle, and if they do not get their results quickly, no matter if their query is common or rare, and easy or complex, they will switch their allegiance to a different product.

A significant fraction of the queries in a typical query log contain phrases. But even without explicitly specifying phrases in queries, documents are expected to be ranked higher if keywords occur in close proximity. To support these options, the inverted index has to store positional information for every word in every document. To minimize random access costs, it is common practise to compactly store them in the inverted lists along with the document numbers and document statistics in an interleaved fashion.

To rank documents in response to a query, similarity measures use heuristic computations based on a range of document statistics, including the overall document frequency f_t of each term t that appears in the collection, and the number of times $f_{d,t}$ that term t appears in document d. In this paper we show that the $f_{d,t}$ component in each posting in the inverted index can be replaced by a surrogate that allows skipping of the positional pointers in query modalities in which they are not required. This substitution significantly improves query execution speed, and compared to the alternative option of inserting an additional per-posting skip into each pointer, both reduces the cost of storing the index and also the cost of processing it. Because the surrogate is highly correlated with term frequency, it has only a marginal effect on retrieval quality, and in some of our trials, actually improved it. Nevertheless, we have also evaluated two approximation methods that allow reconstitution of a good approximation to the original term frequencies, to reduce the risk of effectiveness being eroded. Our simple improvement can thus be used with all ranking functions that make use of term frequencies.

Proceedings of the 13th Australasian Document Computing Symposium, Hobart, Australia, December 8, 2008.

2 Background

This section provides background information and introduces the relevant literature.

2.1 Inverted indexes

The inverted index is the most efficient access structure for fast document retrieval [22]. It consists of a vocabulary, storing the distinct terms t that occur in the document collection, including information such as f_t, the number of documents containing this term; and a set of inverted lists, one per term. Each term t in the vocabulary links to its inverted list, which stores a set of postings, each of which in turn reflects the occurrence of t in a document d. The term frequency $f_{d,t}$ –

the number of times term t occurs in document d – is usually also stored in each posting, which in simplest form has the structure

$$\langle d, f_{d,t} \rangle .$$

To evaluate a query, each term's entry is located in the vocabulary, which is held either completely in memory, or with the most frequently used portions in memory. The corresponding inverted lists are then retrieved, and depending on the query type, combined in some manner that yields a set of answers.

Different methods can be employed to evaluate inverted lists: *term-at-a-time*, which reduces the number of random accesses to disk [13]; or *document-* or *impact-at-a-time* [2, 17], which are appropriate if only the correct (or even an approximate) ranking of the top k documents is required. In the latter two approaches the inverted lists do not necessarily have to be completely processed, and the risk of additional disk seeks is mitigated by reduced processing costs. Document-level indexes are typically very compact, and when suitably compressed require just 5–10% of the space of the text they index [22].

If more complex queries are to be supported, and similarity measures employed that are based on the proximity of query terms, word positional information must also be stored in the index. A direct extension to the previous posting layout is to include the positions $p_1, \ldots, p_{f_{d,t}}$ of term t in document d in each posting:

$$\langle d, f_{d,t}, p_1, \ldots, p_{f_{d,t}} \rangle .$$

Having the positional information immediately available allows arbitrary position-based operators to be implemented, independently of the processing strategy, and can be also advantageous for snippet generation [18].

2.2 Compression

The space required by an index can be significantly reduced through compression. If document identifiers (respectively, term positions) are sorted in increasing order, the difference between consecutive entries can be stored instead. These are commonly referred to as d-gaps for documents, and p-gaps for positions. Because most d- and p-gaps are smaller than the original values, they can be stored in fewer bits, saving space. The drawback of the gap transformation is that the lists of document and position numbers need to be accessed sequentially, and it also makes it more difficult to bypass the positional component of each pointer if it is not required.

Different coding schemes for integer values have been proposed which are able to greatly reduce index size [22]. Reduction in the amount of data transferred from disk can also speed up query processing. The most effective integer coding schemes are bit-based, and unless care is taken with their implementation, add a non-trivial overhead to the computational cost of query processing. As a tradeoff between efficiency in space and time, the use of byte-aligned codes has also been suggested. Byte codes typically outperform bit-aligned compression schemes in terms of decoding speed, at the cost of a modest loss in compression effectiveness [15, 20]. In the simplest of byte code arrangements, one bit is spent to indicate whether or not the next byte is also part of the codeword for the current integer. The other seven bits in each byte store the actual integer value. More complex byte-aligned schemes have also been proposed [6, 7, 10] that reduce the amount of compression "leakage" that occurs compared to bit-aligned codes. Other fast-decoding methods (including word-aligned codes [3]) have also been devised, but typically require that the elements in the stream being compressed be drawn from the same distribution, which is not the situation when the components in each posting are interleaved.

Because not all query modes require access to the position lists, it is desirable to be able to bypass them in any given posting, and move immediately to the next posting. If positions are stored uncompressed, bypass is readily accomplished by stepping over the next $f_{d,t}$ stored values. But when the p-gaps are compressed using a variable length code, the natural ability to forwards seek over $f_{d,t}$ values is lost, a combination that means that if the positional components are not required during querying, each p-gap must be explicitly counted off. In the simplest byte code, this can be done by examining the top bit of each byte looking for stopper bytes, but even this degree of processing is a cost that is better avoided.

2.3 Index organization

The posting index layout indicated above is not the only way to store the inverted lists. An important design decision is the choice of where and how to store position information. There are three distinct alternatives.

Pointer interleaving This straightforward approach, suggested above, minimizes random accesses costs, as the lists for the terms in a query can be sequentially processed using the term-at-a-time strategy. However, in the case of pure ranked queries, in which the position information is not needed, the positions are of necessity still read from disk and accessed in memory, and have to be bypassed as every pointer is processed. On the other hand, the storage requirements are minimal, as there is no overhead incurred through storage of additional control information.

Term interleaving The positional information can also be placed into a separate part of each inverted list, or possibly even into a separate part of the inverted index. Extra space will then be required to allow coordination between the postings in the inverted lists, and their corresponding positions lists, and the space required by the overall index might approach that of

having separate document-level and word-level indexes side by side. With term interleaving, performance on ranked queries should be close to that obtainable using a strictly document-level index; but phrase queries may execute more slowly than is the case with pointer interleaving, because of the need to access additional disk locations to retrieve positional information.

Phrase indexes If phrase queries are the dominant operation to be supported, additional indexes for term-pairs can be built [21]. A document-level index combined with a word-pair index has good performance for both ranked and phrase queries, but is limited to these because no position information is explicitly stored. Another approach is to index only common phrases [9].

Augmentations There have been several approaches described for augmented index organizations, in which internal structures are added within each inverted list to accelerate either the search for pointers, or to bypass blocks of pointers.

In the context of document-level indexes, and conjunctive Boolean and pure ranked queries, Moffat and Zobel showed that their "skipping" approach allows significant time savings to be made [14]. The key part of their proposal was to enlarge the index, and to at regular intervals insert into each inverted list a forward pointer, expressed as a bit or byte offset. To allow decoding to resume after one or more blocks of postings have been bypassed, the document number at the destination of each forward pointer is stored as a gap relative to the document at the start of the skip, rather than relative to the immediately preceding posting. A range of similar techniques have been proposed for other situations, including when the entire index is being held in main memory [16].

An upper bound for performance improvements through skipping can be found if the skipped data is simply not stored. Of course, doing this in regards to whole postings results in loss of generality, and means that some queries might not be properly answerable; nevertheless, this is what is done by static pruning methods, and is approximated by approaches that separate the index into two disjoint parts and seek to resolve queries using only one part [8]. Similarly, skipping the position lists in a pointer interleaved index corresponds (at best) to the use of a document-level index; and the performance differences between document- and word-level indexes can be large (for example, see Hawking [12]). In particular, we use a document-level index as one of the baselines in our experiments with the pointer-skipping approach that is described in Section 3.

Horses for courses Systems employing term-at-a-time evaluation have the advantage that disk accesses are sequential, and the number of disk seeks is minimized. On the other hand, processing the inverted lists of all terms in parallel in pursuing the document-at-a-time approach simplifies the implementation of operators such as word adjacency and proximity. Term-at-a-time systems can also resolve mandatory phrases in queries in a pre-processing step, and then only score the documents that pass the phrase or proximity constraint. This approach avoids having to store position information temporarily in the term-at-a-time accumulators for later use. Either way, if the position information is not pointer-interleaved in the inverted lists, additional linkages into the index are necessary.

Our contribution Anh and Moffat discuss the relationship between index structure and query modes [5], and conclude that pointer-interleaved indexes are slower than term-interleaved ones. However, their experiments did not allow for the possibility that each set of word positions might be able to be rapidly bypassed, and it is that opportunity that we consider in this paper. But rather than simply add skipping information to the index to allow position list bypass, we *replace* the $f_{d,t}$ value (which is the length of the positions list, counted in pointers) by a surrogate, namely the length $b_{d,t}$ (counted in bytes) of the compressed positions list. When a byte code is used, the value of $b_{d,t}$ is never smaller than $f_{d,t}$ and is usually greater; nevertheless, our proposal is that $b_{d,t}$ then be used as a surrogate for $f_{d,t}$ in the ranking formulation. As we shall see, the result of this simple substitution is that index size is largely unaffected; retrieval effectiveness is largely unaffected; and processing speed for ranked queries (assuming a pointer interleaved index) is greatly improved.

2.4 Effectiveness-efficiency tradeoffs

Important factors determining the cost of a query evaluation are the size of document collections and the length and difficulty of queries. Consequently, improvements are promising to achieve if the number of (full or even partial) document evaluations can be reduced. A constant theme has thus been methods for trading (a hopeful small loss of) retrieval effectiveness for (a hopefully large gain in) retrieval efficiency.

Heuristic ranking algorithms can be categorized by their effect on the results, compared to full, exhaustive, evaluation. The least intrusive optimizations are those that guarantee to produce identical results, that is, the same set of k top scoring documents, along with the same final similarity scores. Documents not in the top k might be scored differently.

The next fidelity level is a requirement that the top k documents be correct, and that they be in the correct score ordering, but that the scores not be faithful compared with exhaustive evaluation. This level of approximation is completely acceptable if, for example, the users of a system see the answer rankings, but not the scores. In this case even top-ranked documents may not need to be fully evaluated.

The third category includes methods that guarantee that the right documents will appear in the top k, but not

that they appear in the right order. That is, the split between "shown to the user" and "not shown to the user" is correct, but the ranking might not be. Weakening the fidelity criterion allows additional short-circuiting of the similarity computation.

The last category consists of algorithms that produce rankings that are experimentally similar to those of the underlying exhaustive computation, but cannot be guaranteed in any way. This kind of approach is also of interest to system developers, since similarity functions are themselves heuristics, and there is no guarantee that slavishly executing any particular computation in full detail does in fact give the best possible retrieval effectiveness. A simpler and more succinct computation might do just as well in practice.

The restricted accumulator methods that can be used with term-at-a-time evaluation of ranked queries are examples of this final approach. The accumulators store intermediate results for each document, and represent partially evaluated similarity scores. Moffat and Zobel [14] proposed methods for limiting the number of active accumulators, saving on per-query memory costs, and also allowing the skipping pointers to gain additional traction. Many similar dynamic pruning approaches have been developed, including ones that make use of impact-sorted indexes in which processed and quantized $f_{d,t}$ are stored, rather than raw $f_{d,t}$ values. Impact-ordering also allows the effectiveness-efficiency tradeoff to be controlled on a per-query basis, and allows query evaluation to be terminated even before the end of any of the inverted lists has been reached [4]. Another recent proposal uses the similarity and dissimilarity between documents to build document clusters which can be represented in an index structure [1]. The parallel scoring of clusters and documents within those leads to speedups, but also requires new scoring functions.

3 Term-frequency surrogates

Skips that bypass groups of consecutive postings are useful in ranked querying optimizations that restrict the number of accumulators, or in conjunctive Boolean queries. The key operation in this case is to "forward search" for a specified document number, bypassing all pointers in which the document identifier is less. In this case, there is a balance to be struck between short and long skip groups – too short, and access gets slowed by the large number of skip pointers that themselves must be handled, as well as the index becoming enlarged; and too long, and it is likely that the pointer being sought appears in the very next block anyway. That is, any additional control information not only adds to the space requirement of the index, but also potentially introduces additional costs during query processing, if stored interleaved. Each augmentation value in a pointer-interleaved inverted list has to be read and at least inspected, even if it is not used to evaluate the query.

3.1 Skipping positions

In evaluating pure ranked queries, position information is not used, and it is natural to consider adding a skip to every pointer so that the positional components can be bypassed. But positional components are typically very fine-grained units, and adding another value to every posting is potentially expensive. Given that the typical posting in a text index contains around 5–10 values (a d-gap, an $f_{d,t}$ value, and then 3–8 positions on average), the overhead might be 10–20% in terms of space.

Hence, instead of adding a skip element to the posting and (hopefully) trading execution time for storage space, we fold the desired control information (the skip amount) into the information that must be read and decoded anyway (in particular, the $f_{d,t}$ value). With this small adjustment, skipping of positional lists suddenly becomes feasible. At risk, of course, is the fidelity of the similarity scoring process for ranked queries, since this is why $f_{d,t}$ values are included in the index.

The key observation that allows the substitution to take place is that these two values are reasonable well correlated – because of the monotonic relationship between integer values and the byte length of their byte-coded representations, lists that contain many p-gaps are almost always longer than lists that contain a smaller number of p-gaps. That is, we hypothesize that ranked querying retrieval effectiveness should not change dramatically if $b_{d,t}$, the byte length of the positions list in document d for term t, is used in place of $f_{d,t}$, the actual frequency of t in d. In particular, we note again that the similarity function is itself a heuristic, and should be resistant to small changes in document statistics.

3.2 Correlation

The skip pointer for a position list is just the number of bytes $b_{d,t}$ it takes to code the $f_{d,t}$-long position list of document d and term t, decremented by one if it is given that at least one position must appear in every posting. The compressed position list length in bytes using a byte code is at least as large as the term frequency, because at least one byte is necessary for every coded value. Perfect correlation would be achieved if each position gap could be coded in exactly one byte, which might in fact hold in domains with very short documents (containing $\leq$ 128 indexed words). Figure 1 illustrates the relationship between $f_{d,t}$ and $b_{d,t}$, and shows the least and greatest $f_{d,t}$ value associated with each $b_{d,t}$ value over the approximately six billion postings present in a full inverted index for the 426 GB TREC `gov2` collection. The strength of the relationship is clear.

3.3 Space overhead

The number of bytes needed to store an integer using the simple byte code is a monotonic function of its value, and because $b_{d,t} \geq f_{d,t}$, storing the surrogate instead of term frequency gives rise to a slight increase

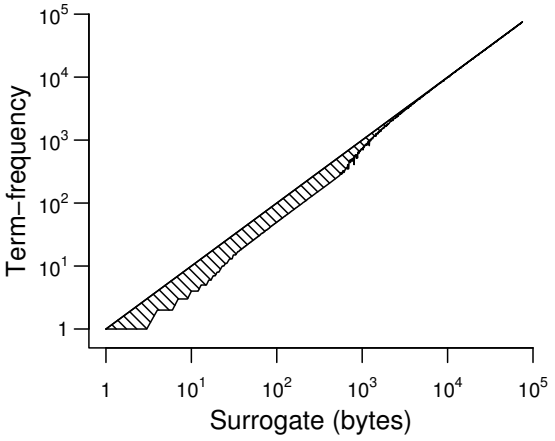


Figure 1: Term-frequency $f_{d,t}$ as a function of compressed byte length $b_{d,t}$, when positional p-gap lists are stored using the simple byte coder. The upper and lower lines record the extremes measured on the TREC gov2 collection. Except for relatively short lists, the two are very closely correlated.

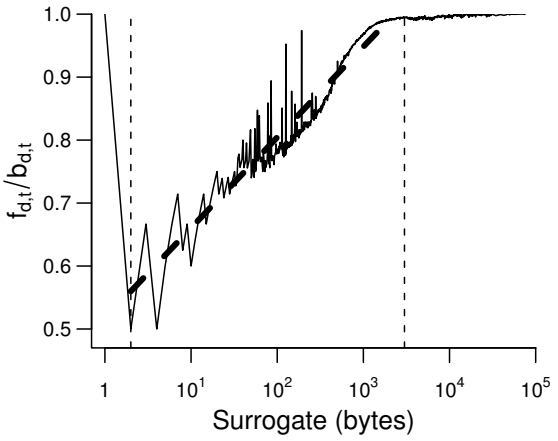


Figure 2: Average observed ratio $f_{d,t}/b_{d,t}$ as a function of the compressed byte length $b_{d,t}$, for the TREC gov2 collection.

System	Space	
	GB	%
Base	44.418	100.0
Full skips	50.174	113.0
Part skips, $f_{d,t} > 10$	44.673	100.6
Surrogate skips	44.421	100.0

Table 1: Space requirements of different word-level index arrangements for the gov2 collection. Row "Part skips, $f_{d,t} > 10$" provides for retention of the $f_{d,t}$ value included in rows "Base" and "Full skips", but adds skip information only when there are more than ten p-gaps in the position list. Row "Surrogate skips" replaces the $f_{d,t}$ values by $b_{d,t}$ values.

in index size. Table 1 shows space overheads for different alternative indexes relative to storing term frequency alone for the gov2 collection described in more detail in Section 4.2, including for a compromise arrangement that adds positional skip information only when there are more than ten p-gaps to be bypassed. Use of full positional skips adds 13% to the index, whereas storage of $b_{d,t}$ in place of $f_{d,t}$ translates into an increase in index size of an inconsequential 3 MB over a 44 GB index.

3.4 Reconstructing frequencies

Most similarity functions are based on $f_{d,t}$ in some way. To facilitate the incorporation of our surrogate with different similarity functions in a transparent manner, it is important to find ways to reconstitute original term frequencies. This way, the surrogate can be plugged into any search engine and is independent of the similarity metric being used.

One simple arrangement is for a lookup table to be employed, in which for each posting, a combination of context variable settings (such as F_t, the overall collection frequency of the term; f_t, the document frequency of the term; and $b_{d,t}$) is mapped to the average term frequency observed for that combination of conditioning values. This table can then be used to convert a given $b_{d,t}$ value, in a given context, back to an estimated $f_{d,t}$ that can be used in the similarity computation.

The two most interesting variables in search engines are the document-frequency f_t, indicating the number of documents in which a term occurs, and the correlated term frequency surrogate itself.

It turns out (based on experiments not reported here) that the term-document frequency f_t has a very low correlation with the ratio $f_{d,t}/b_{d,t}$, and the best single predictor of $f_{d,t}$ is $b_{d,t}$ alone, with no other context information. Calculating the average value of $f_{d,t}/b_{d,t}$ for each distinct $b_{d,t}$ value yields the relationship plotted in Figure 2. As expected, $b_{d,t} = 1$ uniquely indicates $f_{d,t} = 1$, and for large byte-lengths ($b_{d,t} > 3{,}000$), strongly indicates $f_{d,t} = b_{d,t}$. Between these two extremes, a small table serves to capture the observed average relationship.

An even more compact representation is to derive a formula to reconstitute term frequencies. As can be seen from the mid-section of Figure 2, a logarithmic function is a good fit for the indicated interval, and can be pre-calculated at indexing time. Then, during query evaluation, either the direct mapping or the formula is employed, depending on the value of $b_{d,t}$. The dashed line fitted to Figure 2 shows the relationship $f_{d,t} \approx b_{d,t}(0.5 + (1/7)\log_{10} b_{d,t})$.

Finally, note that – with a relatively small amount of computation – exact $f_{d,t}$ frequencies are still available if they are required. All that is necessary is that the next $b_{d,t}$ bytes be processed to count (in the case of the simple byte code and the (S, C)-byte code) the number of stopper bytes. This option could be employed conditionally when exact reproduction of a similarity formula is required. For example, the correct ordering of the top k ranked documents generated by the surrogate approach might then be fully scored, to place them into final presentation order.

4 Experimental results

We adapted version 0.9.3 of the freely available research search engine `zettair`.[1] To measure the impact of using $f_{d,t}$ surrogates, we performed experiments in terms of both efficiency and effectiveness.

4.1 Experimental arrangement

The TREC `gov2` collection is currently the biggest available research dataset, and was the main resource used in the efficiency experiments. It consists of more than 25 million documents and 426 GB of data, and represents a large portion of the available (crawlable) `.gov` part of the web, as of early 2004. A word-level byte coded index for `gov2` occupies 44 GB, and contains more than 6 billion postings.

We used the first 1,000 queries of the query log provided for the efficiency task of the TREC 2006 Terabyte Track. The queries are a mixture of different length ranked queries and do not contain phrases. This is not an issue, because the use of a surrogate does not affect query execution times for phrase queries. We performed neither stemming nor stopping of the index or queries. Timings are for production of a ranking of the top 20 result documents, without lookup of documents or generation of snippets, using a Linux server running Ubuntu 7.10 and kernel version 2.6.22, with dual quad-core Xeon E5345, 2.33 GHz, 64-bit processors and 4 GB of main-memory. The machine was otherwise under light load during experiments and memory was flushed between the runs for different systems and data sizes.

4.2 Efficiency

To measure the extent to which skipping position lists speeds up ranked query processing, we measured timings for the two limiting cases: when all of the inverted lists for each query have to be fetched from disk; and when all of the necessary data is readily available in main memory as a result of a recent execution of the same query. The first case is representative for search in large collections that do not fit into memory, while the latter measures the improvement for in-memory search. A real-world search engine under steady load will have performance somewhere between these extremes, because the index data for at least some of the terms in some of the queries is likely to be available in main memory via standard caching mechanisms.

To achieve these measurements, we executed the query stream with each query immediately re-evaluated after its first evaluation, and kept separate records of the two execution times. This way, the second query is likely to be executed without accesses to disk, because the index data required will probably be served from the disk-cache managed by the operating system. We measured timings for the same set of queries over a range of collection sizes, with random sampling from the whole of `gov2` used to form the sub-collections.

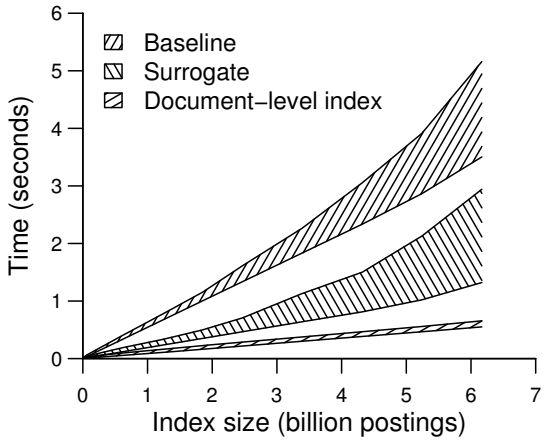


Figure 3: Average per query execution time of the first 1,000 efficiency queries of the TREC 2006 Terabyte Track. The upper and lower ranges for each system and index size show the cost of executing that query with and without disk accesses.

Figure 3 gives the results of these experiments, where the vertical axis denotes average query time (in seconds), and the horizontal axis shows the size of the index, measured in pointers. Three different systems were tested: the original `zettair` word-position index, with pointer-interleaved positional lists; a modified `zettair` in which $f_{d,t}$ is replaced in each pointer by $b_{d,t}$, and all (pointer interleaved) positional lists are bypassed during ranked query execution; and a document-level index in which no positional information is stored at all.

The word-level index using the surrogate is demonstrably faster than the original, and when disk access costs are also factored out, is close to the execution times of the much smaller document-level index. That is, the in-memory case appears to be CPU-bound and exhibits a linear growth in cost with increasing index size; on the other hand, the two upper limits show a distinctive curve as the index size grows beyond 40 GB, and it becomes harder for the operating system to exploit inter-query caching effects.

Other experiments not included here have shown that stopping the queries leads to much smaller execution times, but that the surrogate system keeps its advantage over the baseline; and that executing queries a third (or even fourth) time always leads to timings consistent with the second execution.

4.3 Effectiveness

To see how the use of surrogate $f_{d,t}$ values affected the quality of ranked retrieval, we compared four versions of `zettair` over three different sets of TREC topics and data: Topics 701–750 on the `gov2` collection; Topics 401–450 on the `wt2g` collection; and Topics 451–500 on `wt10g`. The title of the topic was always

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

System	TREC gov2		TREC wt2g		TREC wt10g	
	MAP	σ	MAP	σ	MAP	σ
Baseline	$0.237^{\dagger}$	0.184	0.294	0.219	0.203	0.220
Surrogate	0.250^{*}	0.182	0.294	0.214	0.199	0.215
Surrogate $b_{d,t}$	$0.243^{*\dagger}$	0.184	0.299	0.212	0.197	0.213
Surrogate Formula	$0.243^{*\dagger}$	0.184	0.298	0.214	0.198	0.214

Table 2: Effectiveness comparison using TREC topics 701–850 on the gov2 collection; topics 401–450 on the wt2g collection; and topics 451–500 on the wt10g collection. The MAP values were tested for significance at the 0.05 level, with * denoting significant relative to the Baseline, and † denoting significant relative to Surrogate.

taken as the query. We compared the performance of the unmodified baseline zettair system with our plain surrogate that uses $b_{d,t}$ instead of $f_{d,t}$; with a version that computed an approximate $f'_{d,t}$ using a lookup array indexed by $b_{d,t}$, as depicted by the plotted line in Figure 2; and with a version that computed an approximate $f''_{d,t}$ using a formula (but still implemented as a lookup table indexed by $b_{d,t}$), as depicted by the dashed fitted line in Figure 2.

Retrieval effectiveness was quantified using Mean Average Precision (MAP) and the standard TREC methodology, which measures MAP on the first 1,000 returned documents, and assumes that unjudged documents are irrelevant. We then performed pairwise t-tests at the 0.05 level to gauge significance. Within zettair we used the default Dirichlet-smoothed language modeling similarity function (with $\mu = 1{,}500$) that has been found to perform best across these datasets [11].

As can be seen from the results shown in Table 2, the influence of the surrogate is mixed, but always small. On the gov2 collection, use of surrogates gave rise to a significant *gain* in measured effectiveness, and the two reconstitution approaches then shifted effectiveness back towards that of the baseline zettair. On the other hand, on both wt2g and wt10g, no significant differences in MAP were recorded, perhaps partly because of the smaller number of topics in these two collections, but primarily because surrogates just didn't seem to make much of a difference.

Figure 4 shows, for the gov2 collection and Topics 701–850, the effect of replacing $f_{d,t}$ by $b_{d,t}$ on a topic by topic basis, and confirms that the use of surrogates perturbs almost all MAP scores up or down by a small amount, rather than create any large shifts, or move them all consistently in the same direction.

Rather than converting pairs of rankings to effectiveness scores and then comparing the scores, it is also possible to directly compare the relative fidelity of the two rankings, in terms of whether they retrieve the same documents in the same order, regardless of relevance. The *Rank-Biased Overlap* (RBO) computation of Webber et al. [19] provides a way of calculating the difference between two

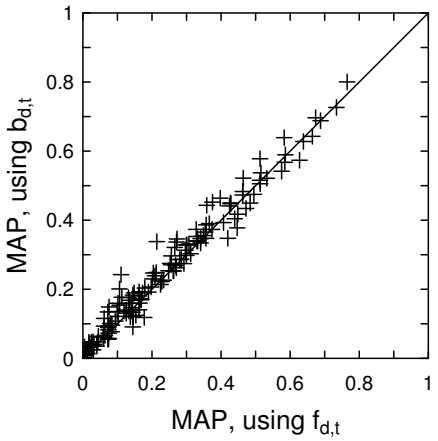


Figure 4: Surrogate versus Baseline, with MAP scores for Topics 701–850 using the $b_{d,t}$-based computation plotted as a function of the MAP scores attained using the original $f_{d,t}$-based computation.

System	RBO	
	$p = 0.9$	$p = 0.99$
Baseline	1.000	1.000
Surrogate	0.768	0.796
Surrogate $b_{d,t}$	0.782	0.839
Surrogate Formula	0.779	0.838

Table 3: Rank-biased overlap (RBO) for two different values of p using TREC Topics 701–850 on the gov2 collection.

indefinite rankings, with a bias towards early positions in the ranking, and the ability to deal with rankings over disjoint sets of objects. A parameter $0 \le p < 1$ controls the extent of the top-weightedness of the comparison, with p interpreted as being the *persistence* of a user who is side-by-side comparing overlap between the two rankings, and steps from rank r to rank $r + 1$ with probability p. A useful property of the RBO formulation is that the influence of the (unranked) tail sections of the lists is bounded, and the RBO score for any pair of rankings is a range that can be calculated without looking at all documents.

Table 3 lists RBO scores for the gov2 collection at two different values of p. Broadly speaking, $p = 0.9$

places the bulk of the emphasis on the top ten documents in the ranking, and an RBO of 0.8 suggests that around eight of the top-10 determined by the Baseline computation appear in the same top-10 positions as in the Surrogate one. Similarly, $p = 0.99$ spreads the emphasis to depth 100 and beyond, and a score of 0.8 suggests that around 80% of the elements are in, or not too far from, their original rank positions. Use of RBO in this experiment neatly indicates the quality of the reconstitution methods – both give RBO scores higher than the plain Surrogate one.

5 Conclusion

Interleaving position information in inverted indexes allows processing of queries with only one disk seek per term in term-at-a-time systems, and reduces the number of parallel pointers required in document- and score-at-a-time systems. We proposed a term frequency surrogate as a way of speeding up query processing in such indexes, without adding to the space required by the index. We have shown that the length of the compressed p-gaps is highly correlated with term frequency, and allows the use of $b_{d,t}$ instead of $f_{d,t}$ in similarity computations. We also used approximation functions of differing quality and efficiency to reconstitute the original term frequencies. The surrogate method is demonstrably faster than the baseline approach, and approaches the speed on a document-level index for ranked queries. Effectiveness is largely unchanged by the substitution.

Acknowledgements Justin Zobel took part in a number of fruitful discussions, and William Webber provided the RBO implementation. National ICT Australia (NICTA) is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

References

[1] I. S. Altingovde, E. Demir, F. Can and Ö. Ulusoy. Incremental cluster-based retrieval using compressed cluster-skipping inverted files. *ACM Trans. Information Systems*, Volume 26, Number 3, pages 1–36, 2008.

[2] V. N. Anh, O. de Kretser and A. Moffat. Vector-space ranking with effective early termination. In *Proc. 24th Ann. Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 35–42, New Orleans, Louisiana, United States, 2001. ACM.

[3] V. N. Anh and A. Moffat. Improved word-aligned binary compression for text indexing. *IEEE Trans. Knowledge and Data Engineering*, Volume 18, Number 6, pages 857–861, June 2006.

[4] V. N. Anh and A. Moffat. Pruned query evaluation using pre-computed impacts. In *Proc. 29th Ann. Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 372–379, Seattle, Washington, USA, 2006. ACM.

[5] V. N. Anh and A. Moffat. Structured index organizations for high-throughput text querying. In *Proc. String Processing and Information Retrieval Symposium*, pages 304–315, Glasgow, Scotland, October 2006. LNCS 4209, Springer.

[6] N. R. Brisaboa, A. Fariña, G. Navarro and M. F. Esteller. (S, C)-dense coding: An optimized compression code for natural language text databases. In *Proc. String Processing and Information Retrieval Symposium*, pages 122–136, Manaus, Brazil, October 2003. LNCS Volume 2857.

[7] N. R. Brisaboa, A. Fariña, G. Navarro and J. R. Paramá. Simple, fast, and efficient natural language adaptive compression. In *Proc. String Processing and Information Retrieval Symposium*, pages 230–241, Padova, Italy, October 2004. LNCS Volume 3246.

[8] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer and J. Y. Zien. Efficient query evaluation using a two-level retrieval process. In *Proc. 2003 ACM CIKM Int. Conf. Information and Knowledge Management*, pages 426–434, New Orleans, Louisiana, November 2005. ACM Press, New York.

[9] M. Chang and C. K. Poon. Efficient phrase querying with common phrase index. *Information Processing & Management*, Volume 44, Number 2, pages 756–769, 2008.

[10] J. S. Culpepper and A. Moffat. Enhanced byte codes with restricted prefix properties. In *Proc. String Processing and Information Retrieval Symposium*, pages 1–12, Buenos Aires, November 2005. LNCS Volume 3772.

[11] S. Garcia, N. Lester, F. Scholer and M. Shokouhi. RMIT University at TREC 2006: Terabyte track. In *Proc. 15th Text REtrieval Conference (TREC)*, Gaithersburg, MD, 2007. National Institute of Standards and Technology.

[12] D. Hawking. Efficiency/effectiveness trade-offs in query processing (from theory into practice workshop, 1998 SIGIR conf.). *SIGIR Forum*, Volume 32, Number 2, pages 16–22, 1998.

[13] M. Kaszkiel, J. Zobel and R. Sacks-Davis. Efficient passage ranking for document databases. *ACM Trans. Information Systems*, Volume 17, Number 4, pages 406–439, 1999.

[14] A. Moffat and J. Zobel. Self-indexing inverted files for fast text retrieval. *ACM Trans. Information Systems*, Volume 14, Number 4, pages 349–379, 1996.

[15] F. Scholer, H. E. Williams, J. Yiannis and J. Zobel. Compression of inverted indexes for fast query evaluation. In *Proc. 25th Ann. Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 222–229, Tampere, Finland, 2002. ACM.

[16] T. Strohman and W. B. Croft. Efficient document retrieval in main memory. In C. L. A. Clarke, N. Fuhr, N. Kando, W. Kraaij and A. P. de Vries (editors), *Proc. 30th Ann. Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 175–182, Amsterdam, The Netherlands, July 2007. ACM Press, New York.

[17] T. Strohman, H. Turtle and W. B. Croft. Optimization strategies for complex queries. In *Proc. 28th Ann. Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 219–225, Salvador, Brazil, 2005. ACM.

[18] A. Turpin, Y. Tsegay, D. Hawking and H. E. Williams. Fast generation of result snippets in web search. In *Proc. 30th Ann. Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 127–134, Amsterdam, The Netherlands, 2007. ACM.

[19] W. Webber, A. Moffat and J. Zobel. A similarity measure for indefinite rankings. Manuscript, November 2008.

[20] H. E. Williams and J. Zobel. Compressing integers for fast file access. *Computer Journal*, Volume 42, pages 193–201, 1999.

[21] H. E. Williams, J. Zobel and D. Bahle. Fast phrase querying with combined indexes. *ACM Trans. Information Systems*, Volume 22, Number 4, pages 573–594, 2004.

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

MetaView: Dynamic metadata based views of user files

James Bunton
School of IT
The University of Sydney
NSW 2120 Australia
jamesbunton@fastmail.fm

Judy Kay
School of IT
The University of Sydney
NSW 2120 Australia
judy@it.usyd.edu.au

Bob Kummerfeld
School of IT
The University of Sydney
NSW 2120 Australia
bob@it.usyd.edu.au

Abstract *Hierarchical file systems are the most common way of organising large collections of documents. However, there are several desirable features they do lack. These include: good support for placing files in multiple locations; dynamic views on the users' data; and explicit ordering of files. This paper introduces MetaView, a new approach to enhancing file systems so that they can present users with a fluid and dynamic view of their files based on metadata. MetaView allows users to describe how they wish to view their files by specifying an organisational structure based on a metadata path. Experiments indicate that this approach is viable for collections of up to several thousand files in size, enabling flexible organisation of substantial parts of a user's file system.*

Keywords Document Management, Metadata, Non-hierarchical file system.

1 Introduction

File system organisation is a very important area of computing. Millions of people use computers daily to store critical, often irreplaceable data. This includes text, spreadsheets, photos, videos, music and many other types of documents. Most of this data is stored as ordinary files in a traditional file system.

The file systems provided by the commonly used desktop operating systems [1] are implementations of the hierarchical model. From a conceptual perspective, this is essentially the same model used in the 1969 version of Unix [17]. In a hierarchical file system, users can create folders and subfolders to categorise their files, with an arbitrary level of nesting. This structure can be browsed to locate existing files and create new ones. Once a file is located, applications can read from and write to it.

There are some well known limitations with this system which MetaView aims to address. One of these is that operating systems have a deep model that means each file may only be saved in one location, even if it logically belongs under two folders for the purposes of categorisation that suit particular user tasks. Additionally the burden of organising files is placed on the user; switching to a different categorising scheme can require much tedious, manual effort to move files around.

Consider the example where a user has a collection of music which comes with extensive metadata. The user may wish to organise this music by genre, artist and then title. Many songs fall under multiple genres. However, there is no well supported, easy mechanism enabling the user to express this in existing hierarchical file systems. The operating system provides poor support for the case where a user organises files initially in one structure, say artist, but then later decides they would like to switch to viewing their music files by genre and then title. The larger the collection of files, the more onerous this task becomes.

While most operating systems support links as a method of extending the hierarchical model, these have several problems. Firstly, there are so many different kinds: Windows shortcuts, Unix symlinks and hardlinks as well as Mac OS aliases. Each of these has different semantics with positive and negative tradeoffs and none of them meets both the goals described above of allowing files to be organised automatically for users and seamlessly saving files allowing a file to exist in multiple locations.

MetaView's approach is to provide users with a mechanism to specify the structure of a "view" in which they want their files to appear. A user may tell the system to make use of metadata to display all of their music in a format such as: `pop/The Beatles/Yellow Submarine.mp3` where the genre is first, then the artist, followed by the title. The user could later choose any other organisation structure that suits their needs simply by creating or updating a "view". The same user may occasionally wish to browse by artist, then genre, so their music would appear in this format: `The Beatles/pop/Yellow Submarine.mp3`. This applies equally for any types of files, including collections of completely different file types, for which automatically collected or user-specified metadata is available.

Importantly, MetaView allows any existing application to use the standard open/close/read/write file API to access document, making use of metadata as in the examples above. This means the applications can op-

[1] Linux, Mac OS and Windows.

Proceedings of the 13th Australasian Document Computing Symposium, Hobart, Australia, 8 December 2008.

erate consistently with the user's mental model at the same time as allowing developers to use the existing well-known and understood tools for interacting with the file system.

The ultimate vision is that rather than specifying a location on disk when saving a file, users would have the option to tag the file with metadata, or simply save it. The system will collect the optional user-specified metadata, as well as automatically extracted metadata and use this to display the file in any appropriate views which the user defines. This relieves users of the burden of organising their files manually, but still allows for the familiar and powerful browsing interface. MetaView currently implements the "view" part of this interface; changes would be required to the operating system's save-dialogue API in order to support this alternate saving behaviour in graphical applications.

2 Related Work

There have been several studies of how users work with their personal data, such as [4, 18, 5, 16, 12]. These point to some of the limitations in current systems causing difficulties for users in organising documents within multiple locations in file hierarchies as well as the lack of support for the user to define an arbitrary, custom ordering of files within a directory. Reflecting the recognition of the limitation of the prevailing hierarchical file systems, both the research community and commercial organisations have explored alternative approaches designed to improve the situation. We now consider some key examples.

One class of systems has taken the approach of enhancing search functionality. These include SFS [11], BeFS [10], Connections [19], LISFS [15] and Spotlight [3]. All of these systems provide alternate ways of accessing the information stored in a hierarchical file system. However, even this facility does not address the basic goals of our work, to overcome the limitations of hierarchies. These search tools operate in the context of the existing hierarchical file systems and still leave the user with the burden of manually reorganising files if the user wants to organise documents into a different hierarchical view.

By contrast, some research systems have attempted a complete breakaway from the hierarchical model. Examples include LifeStreams [8], Presto [7], Placeless Documents [6], PosCFS [13] and LIFS [2].

Selected systems particularly relevant to MetaView will now be described in greater detail.

2.1 SFS

The Semantic File System [11] was the first implementation of a file system that was semantic in that it provided virtual directories for queries based on metadata extracted from files. For example, it made use of author, title or words contained in the file. The way that it worked was that a virtual directory was created on request, the name of the directory being the query. This virtual directory concept allowed for compatibility with all existing software.

A process called a transducer ran automatically in the background to keep the store of metadata up to date with the contents of the files. The virtual directories were provided using a custom NFS server.

The evaluation of this system was primarily focused on system performance. The amount of disk space required for the metadata store and its index was determined. The total time to index files as well as incremental updates to the index were also measured. The authors concluded that the performance and space requirements of realtime indexing were not overly taxing for the file server. The system was found to be useful for sharing files with other research groups, although there was little detail of this. It was also noted that in the case of file types for which no transducer had been written, these files were difficult to locate.

2.2 LISFS

The Logical Information Systems File System (LISFS) [15] is also an implementation of a semantic file system. Like SFS 2.1, queries are supported as virtual directories.

The system is implemented as a Linux VFS plugin. Transducers operate in the background to collect metadata and update indices.

To make a query, a directory path is constructed out of expressions about the collected metadata and can include logical constructs such as 'and', 'or', 'not'. Queries can be constructed incrementally. At each stage in the query the user may choose to look at the result set so far, or at a list of possible extensions to the query. This allows for a kind of browsing with flexible organisation of files, as the user proceeds.

Also supported is the ability to make a query within a file, for example queries within a BibTex database will return a portion of that file. The result set can be edited and changes propagate back to the original in the way that one would expect. This aspect of the system enables the user to think about abstract *documents* even when these are actually stored within a single file.

No formal evaluation of the system was mentioned in the paper, although several examples were given of the system in use. It operates on several file types, including mp3 collections, source code, BibTex files and email. LISFS is important in that it extends the file system metaphor to allow more powerful and flexible access to data in a manner compatible with existing software.

2.3 LiFS

The Linking File System (LiFS) [2] extends the hierarchical file model to include the concept of links between files. The authors point out that many applications have developed their own systems of storing and searching file metadata, including links. The main shortcoming of this approach is that any one computer system will

have several incompatible stores of metadata using a different interface.

The solution given by the authors is a file system that includes support for traditional file metadata as well as links between files. These links are intended to allow desktop search tools to detect importance, relevance and relations between documents in a similar manner to the way web search engines use hyperlinks in web pages.

New system calls were added to support creating, reading and modifying links between files. Unfortunately, this means that LiFS compromises compatibility with existing applications.

The file system is designed to operate on high speed, high capacity non-volatile memory. Several tasks were performed with LiFS on standard volatile RAM, compared with ext2 and XFS. LiFS outperformed these systems in metadata access, and was close in other areas such as creating/deleting files and read/write operations.

2.4 Connections

Connections [19] is a desktop search tool. It works similarly to content search tools like Spotlight [3], but augments the results with contextual information gathered from monitoring user activity.

The system traces all file activity by the user, building up a relation graph between files. Links are defined between files which were accessed at similar times where these links represent a weighted relationship. This graph is analysed, to discard irrelevant, and append relevant entries to the standard content search as well as to help rank these results.

The user evaluation performed by the authors showed that Connections improved both average recall and precision over a standard content search. Additionally the performance impact of collecting the trace data and analysing it to build the context graph was found to be minimal.

2.5 Presto

The Presto [7] system's primary method of locating documents is through "collections", where these are live queries over document metadata combined with an explicit document inclusion and exclusion list which the user may manipulate. Collections may also be nested.

One particularly novel feature is personal metadata. When attaching metadata to a shared file, a user may decide to keep that metadata private. This could be useful for marking files as "interesting", metadata that is not necessarily relevant to other users of that document.

Presto sourced documents from many locations including the local file system, network shares, web sites, email clients, etc. These data sources were implemented as plugins, each of which was responsible for allowing read and write support to its respective source data, as well as extracting metadata.

Two APIs were provided by Presto. A custom NFS server acted as a compatibility layer intended for existing applications. The primary API allowed full access to all of the functionality offered by the system.

A custom document browser was built in the primary API to allow users to create and manipulate collections and the files within them. When a user wishes to work with a document in a Presto-unaware application, that application is launched and given the path to a file on the NFS server.

There was no formal evaluation. However, the authors reported that their goal, creating a system where attribute-oriented access was the primary method of document interaction, was successful.

3 Approach and Overview

Many of these novel approaches to explore alternatives to the prevailing hierarchical file system structure use virtual directories, whose path name constitutes a form of search query string, as a user interface for finding files. By contrast, our approach in MetaView is to create views, which are populated with directories and files such that the path to a file describes its contents according to the metadata which the user is interested in. If we think of the full path name of a file in a conventional hierarchical file system as an ordered series of metadata tags, essentially MetaView makes it possible to generalise that to allow other orderings of the metadata to create different views.

To implement this, we considered several possibilities and implemented one based upon symbolic links on a regular file system. This means that MetaView retains full compatibility with all existing software and requires no kernel modules or modifications. This makes installation simple and means that users do not need to trust their collection of files to an unknown file system.

Like Presto, MetaView attempts to provide an alternative to the hierarchical file system for organising and retrieving files. Presto's "collections" are similar in purpose to a "view"; however, there are some important differences. Where a collection is a flat list of files which may contain other collections, a view consists of a possibly nested set of files and folders kept in a structure managed by MetaView.

The following figures are screenshots from the Mac OS X Finder. Each shows a different view of the same collection of files, as described below. The goal of MetaView is that users should be able to specify arbitrary views over their files as they wish, reflecting their immediate needs for different structuring of their files. For example, a user may decide to have several views of their music files. Let us suppose that the first is a flat list of files as shown in the small subset of a user's music files in Figure 1. This shows just the first few of a large number of music files. Each of these has metadata containing various attributes describing the file.

We now illustrate MetaView's power to enable the user to automatically reorganise this set of files for more

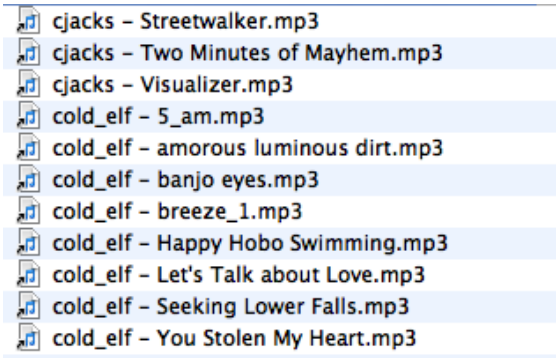


Figure 1: `Artist-Title.mp3`

convenient browsing. For example, suppose that over time, the user has amassed large numbers of these music files and at some stage, they decide to browse their music, thinking of it in terms of the genre first and within this, the artist. MetaView enables them to issue a command to define this new view as illustrated in Figure 2. Note that in this case, many files fit into multiple genres and the view takes care of this in the way that the user would reasonably expect, placing the music file into each genre folder it belongs in. This can be seen in Figure 3, the file 'Flat Ed - Growing Crows.mp3' appears under both the 'country' and 'indie' genres.

Figure 2: `Genre/Artist-Title.mp3` #1

Now we may suppose that at a later time, the user decides they want to be able to browse their music in terms of a different organisation, this time making the artist the first aspect and within that the title. This is illustrated in the example shown in Figure 4. Once again, some music files have multiple artists and the view will handle this correctly.

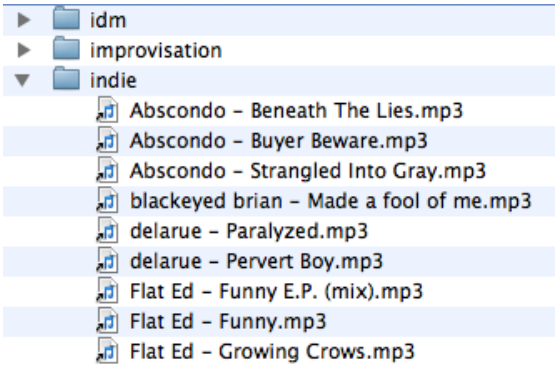


Figure 3: `Genre/Artist-Title.mp3` #2

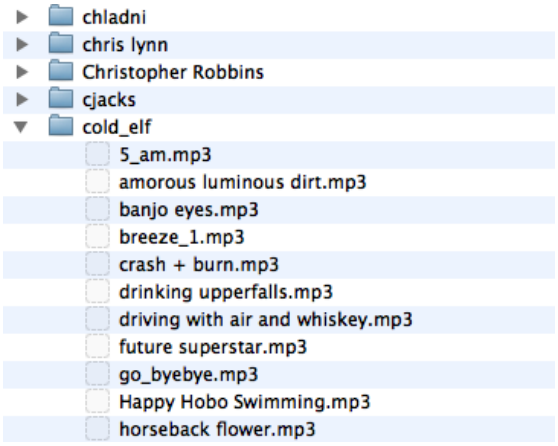


Figure 4: `Artist/Title.mp3`

The final example we will consider is of a combination of the previous two, browsing by genre, then artist with the filenames being the title of the song (Figure 5). This demonstrates the power of MetaView; users can easily construct these alternate views of their files according to their needs and wishes, enabling them to adapt their file structure to changes over time.

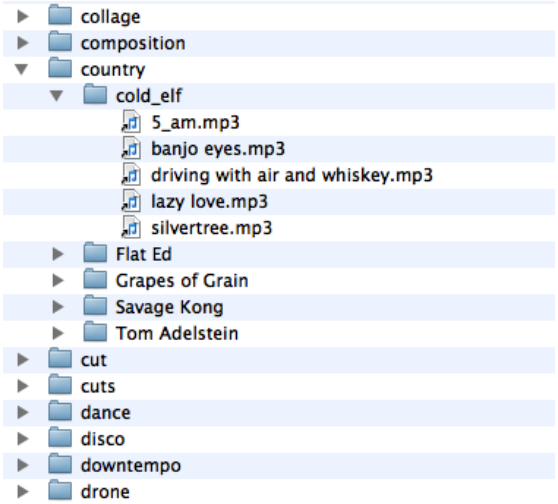


Figure 5: `Genre/Artist/Title.mp3`

4 Architecture

File metadata can be thought of as an extension to the hierarchical model. For our purposes, file metadata is both structured data extracted from a file as well as annotations or tags that a user may provide. For example, an email message may have the Subject, From, To, etc headers extracted as metadata and users may tag photos with the names of people in them.

MetaView has been implemented to run under Mac OS X 10.5, making heavy use of the metadata capabilities of the Spotlight API.

Apple's Spotlight was chosen for this project due to its wide support for existing file types and relative maturity compared to other systems. Spotlight has "importers" for many file types, these are called to examine a file by Spotlight and extract any metadata from it. This metadata is then stored and indexed in the Spotlight database. Apple provides good documentation on writing programs that make use of this database as well as writing new importers to support additional file types. MetaView can be used with any file type supported by Spotlight, these include: email, audio (MP3, AAC), office documents (Microsoft Office & OpenDocument), images (JPEG, PNG) and many others. There is also a commonly used technique that allows Spotlight to index individual documents even when they are all stored in a single file. In addition to the automatically extracted metadata, users may also annotate files with their own custom metadata such as tags for project names, or to mark a file as 'todo'.

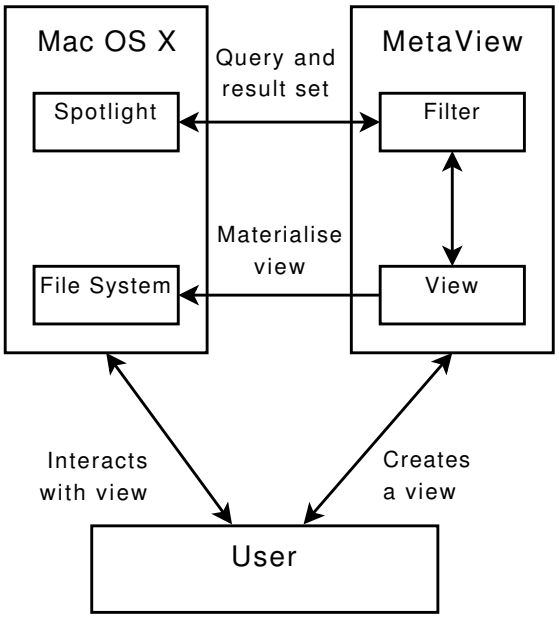


Figure 6: MetaView architecture

The system is implemented in two parts; the search filter and the metadata view (Figure 6). MetaView is written in Python and uses Apple's Spotlight API for searching and access to file metadata.

As input to MetaView, users provide an optional search query and a view specification which they wish the search results to be displayed in. The search query may be as simple as restricting the view to files in a particular directory, or it may be omitted entirely.

An example query for all MP3 files on the system would look like:

```
kMDItemContentType == 'public.mp3'
```

This query is expressed in the standard Spotlight query language. It could easily be constructed by users interacting with an application, such as the Mac OS X Finder.

The metadata view is the most important part of MetaView. In the case of this music example, metadata is extracted by Spotlight from the ID3 tag of the MP3 files. This is a standard tag format that is included with most MP3 music files. Users specify how they want to view their files in the form of a structured hierarchy with different metadata at each level. A view specification for browsing music by genre, artist and then title is expressed as:

```
$(genre)s/$(artist)s/$(title)s.mp3
```

View specifications are Python format strings that are evaluated with respect to each file to be placed in the view. This too could be made intuitive for users to formulate, with the support of a graphical interface.

MetaView creates a Spotlight search for each view that has a unique query. Spotlight then gives a list of query results which MetaView uses to populate a directory with links to the original files according to the view specification. In this way the user's existing workspace is not disrupted at all but they can still take advantage of the advanced functionality the MetaView offers.

It is important to note that the user's files remain on their existing filesystem and can still be accessed in the usual way if so desired. The view is implemented as symlinks to the original files in a directory managed by MetaView. As the user works on their system, creating, deleting and editing files, MetaView keeps the view up to date in real time.

Spotlight notifies MetaView whenever there is a change to any of the files that are being watched, whenever a new file becomes relevant and whenever an existing file is no longer relevant. This notification consists of a list of all the files which currently match the search results. MetaView takes this list and performs a `stat()` on each file to check its last modification time and sorts the Spotlight list into removed files and added files. A changed file is removed and then re-added.

These lists are then passed to the 'view' component which removes all links to the removed files and then inserts links to the added files into the appropriate places. By this process unchanged files are left in place rather than recreated each time Spotlight sends an updated result set.

The user may interact with the constructed view using any existing software. Views can be browsed from the command line or using the Finder. As symlinks are used, opening any link from any application causes that application to work on the original file.

In this way users can create views of their file collection and work with them using their existing software.

The concept of a view, providing flexible organisation of a users' file collections, not specific to any particular operating system. It is a generalisation of existing hierarchical file systems and is a generic concept that could be implemented under any operating system. However the prototype discussed is tied closely to Mac OS X. For a Linux version, the Spotlight metadata backend could be replaced with Strigi [1] or Tracker [9] while the symbolic link implementation of views would remain unchanged. To port MetaView to Microsoft Windows would require larger changes; both the view and the search filter components would need to be rewritten.

5 Scalability Evaluation

Since MetaView is designed to support new ways to organise collections of files, it is important to assess whether it can do this efficiently and to have an understanding of the scalability as the numbers of files grows. We need to assess the scalability of MetaView for its two main actions, constructing a new view and updating a view after relevant changes in the part of the filesystem managed by MetaView. A relevant change is a modification to a file's data or metadata such that the file needs to be added to the view, removed from it or moved within it. We now describe two forms of scalability analysis, the first analytical, based only on the operations required and the second empirical, based on results with actual sets of files.

5.1 Analytical

One of the key costs of handling changes to the managed file set is due to our use of Spotlight for the current implementation. The Spotlight API does not give a list of changes to search results. This imposes a small constant time cost for each file in the result list whenever a change is made. For each change to the view, there may be several calls to the more expensive `mkdir()`, `rmdir()`, `symlink()` or `unlink()` as well as the cost of extracting metadata attributes from the Spotlight API in order to effect the update. For each update posted by Spotlight, there is a cost $O(Sn + Um)$; where n is the number of files in the result set, m is the number of changed files, S is the cost of a `stat()` and U is the cost of updating the view for a single file.

If the Spotlight API were enhanced to give incremental updates to search results, rather than posting the entire result set each time, this performance could be improved. The complexity would drop to $O(Sm + Um)$, a dramatic improvement, especially for the important case where there is a large set of files and there is a change to a small number of them.

The cost of creating the view is the expected $O(Un)$.

The cost of using MetaView would remain the same for any file type supported by Spotlight. This is because every Mac OS X computer has already indexed every Spotlight-supported file on the system, and the cost of extracting metadata from the Spotlight database is independent of the original type of the file.

5.2 Empirical

For this test, we chose to use a set of files that could be readily assembled and where we could also gain metadata that would be useful for creating structures. We chose to use a collection of creative commons licensed music which could be used freely by others to repeat the experiment. The evaluation was conducted with a 10GiB collection of music from opsound.org [14] with 1836 MP3 files. For each of these files Spotlight allows access to metadata such as artist, title and usually multiple genres. The testing was performed on an Intel iMac running Mac OS X 10.5.5 with a Core 2 Duo 2.16 GHz processor and 1GiB of memory.

The cost of a single `stat()` system call was measured to be an average of approximately 0.0015 seconds over 1000 uncached files. This drops to about 0.00001 seconds for 1000 cached files. However in general, we would expect that the files that MetaView will be performing a `stat()` upon will not be cached, giving the slower time as the expected average. This corresponds to the S variable in the analytical analysis.

For the majority of cases in a large collection of files, updating the view will be a matter of a `symlink()`, `unlink()` or both for each file that needs to be updated. Over an average of 1000 files, the cost of a `symlink()` and an `unlink()` was measured to be 0.0003 seconds. Note that for each file that needs to be placed in several locations, this will require multiple system calls. So the actual time for the update will depend upon this.

MetaView took 20 seconds on average to construct a view in the format: `Genre/Artist/Title.mp3`. To update the view after changing one file took 3 seconds on average. Almost all of this time was spent in the `stat()` call, required to determine which file in the list from Spotlight was the changed one.

Additional tests were performed with different sized file collections, these can be seen in Figure 7 and 8. All tests were performed 3 times while the system was under no load and the result was averaged. In all cases there was less than one second difference between the trials.

The difference in time between the two views reflects the number of links that must be created in each case. In Figure 8 exactly one link is created for each

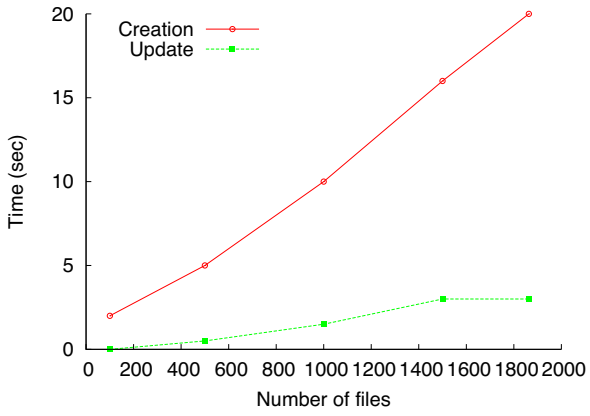


Figure 7: Results: `Genre/Artist/Title.mp3`

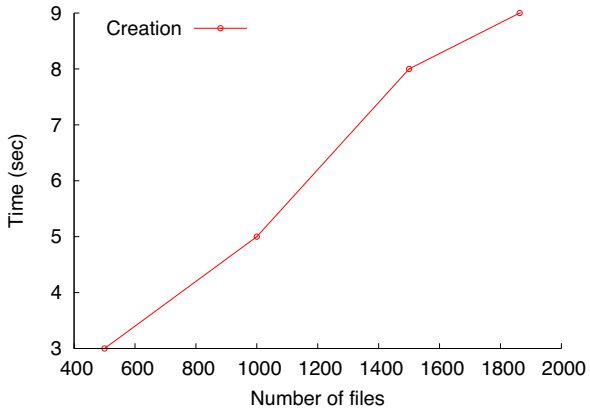


Figure 8: Results: `Artist - Title.mp3`

file, compared with Figure 7 where each file appears in several genres.

The empirical results match up with analytical analysis showing that MetaView scales linearly with the number of files it manages.

6 Conclusions and Future Work

A major performance cost is processing the complete set of results that Spotlight posts whenever a change is detected in even a single file. MetaView could use the FSEvents API to watch for changes to files on the system, in this way it would know which files have been changed in a Spotlight result set without resorting to a `stat()` of each file. This approach adds the cost of processing each file change event on the system, which may even be more work. A better solution would be possible if Spotlight optionally provided updates to existing result sets rather than reposting the complete list of files each time.

Presto provides users with the ability to add and remove files from a collection and have these actions stay persistent. MetaView nearly gets this ability for free by using the regular file system. If a file is added to a view or removed from a view by another program, MetaView will simply ignore it. However additional support for this would be useful. For example, the action of placing a file into a directory which contains files tagged as "todo" could tag that file. This means it would appear in other views that show files tagged as "todo".

There are several promising directions for future work. One of these would provide support for "bundles". These are groups of files which should be treated as one logical unit. Particularly on Mac OS X, many applications create directories full of files that appear to the user as one bundle. MetaView should also treat this as one when populating views and not delve inside the logical unit.

A complementary direction would support users in creating views of abstract "documents" which are stored within a single file. One important example of this is the standard mbox file which contains a collection of mail items, each of which the user may think of as a separate document. Another class of example is a file containing a collection of consistent elements, such as bibtex entries. Since the user may wish to organise these virtual documents into different structures, it would be valuable for MetaView to be able to operate at this level. The simplest way to do this is to make these abstract documents appear to Spotlight as individual files. Many Mac OS X programs already use this technique, including Apple's Address Book for contacts as well as Safari for bookmarks and history. Stand-in files are created for each abstract document with just enough information to allow the owner application to find the actual content.

The system could be further integrated with applications by giving users the ability to apply custom metadata to files when saving them. A save dialogue that allowed users to tag files, displaying a list of commonly used tags, might be a good way to achieve this.

MetaView represents an exploration of a new mechanism for supporting flexible organisation of personal information, in terms of arbitrary sets of hierarchical organisations of documents. With MetaView, users can flexibly create views of their file system where these views structure the documents in the ways that suit the user's current needs, even if this was not anticipated when the files were first saved and organised. These views are automatically updated as the contents of the file system changes, keeping the user's workspace up to date and organised with minimal effort on their part. Importantly, the user's existing file system organisation is left untouched by MetaView, allowing a user a safe entry point, so they can use MetaView without altering their existing work practices and without the risk of being unable to use the multitude of existing operating system services.

MetaView provides a flexible and adaptable new means to organise and access files. While it is consistent with the mental model of traditional hierarchical file systems that most users are familiar with today, it enables the user to extend that same mental model to arbitrary new organisation possibilities, while preserving compatibility with existing software and work practices.

References

[1] Strigi - the fastest and smallest desktop searching program. http://strigi.sourceforge.net 2008.

[2] Sasha Ames, Nikhil Bobb, Kevin M. Greenan, Owen S. Hofmann, Mark W. Storer, Carlos Maltzahn, Ethan L. Miller and Scott A. Brandt. Lifs: An attribute-rich file system for storage class memories. In *Proceedings of the 23rd IEEE / 14th NASA Goddard Conference on Mass Storage Systems and Technologies*, 2006.

[3] AppleComputer. Working with spotlight. http://developer.apple.com/macosx/spotlight.html 2006.

[4] Deborah Barreau and Bonnie A. Nardi. Finding and reminding: file organization from the desktop. *SIGCHI Bull.*, Volume 27, Number 3, pages 39–43, 1995.

[5] Richard Boardman, Robert Spence and M. Angela Sasse. Too many hierarchies? the daily struggle for control of the workspace. In *Proceedings of HCI International 2003*, pages 616–620, New Jersey, USA, 2003. Lawrence Erlbaum Associates.

[6] Paul Dourish, W. Keith Edwards, Anthony LaMarca, John Lamping, Karin Petersen, Michael Salisbury, Douglas B. Terry and James Thornton. Extending document management systems with user-specific active properties. *ACM Transactions Information Systems*, Volume 18, Number 2, pages 140–170, 2000.

[7] Paul Dourish, W. Keith Edwards, Anthony LaMarca and Michael Salisbury. Presto: an experimental architecture for fluid interactive document spaces. *ACM Transactions Computer-Human Interactaction*, Volume 6, Number 2, pages 133–161, 1999.

[8] Scott Fertig, Eric Freeman and David Gelernter. Lifestreams: an alternative to the desktop metaphor. In *CHI '96: Conference companion on Human factors in computing systems*, pages 410–411, New York, NY, USA, 1996. ACM.

[9] Gnome Foundation. Tracker - a personal search tool and storage system. http://live.gnome.org/Tracker/WhatIsTracker 2008.

[10] Dominic Giampaolo. *Practical File System Design with the Be File System*. Morgan Kaufmann Publishers, Inc., 1999.

[11] David K. Gifford, Pierre Jouvelot, Mark A. Sheldon and Jr. James W. O'Toole. Semantic file systems. In *SOSP '91: Proceedings of the thirteenth ACM symposium on Operating systems principles*, pages 16–25, New York, NY, USA, 1991. ACM.

[12] William Jones, Ammy Jiranida Phuwanartnurak, Rajdeep Gill and Harry Bruce. Don't take my folders away!: organizing personal information to get things done. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1505–1508, New York, NY, USA, 2005. ACM.

[13] W. Lee, S. Kim, J. Shin and C. Park. Poscfs: An advanced file management technique for the wearable computing environment. *Lecture Notes in Computer Science*, Volume 4096, pages 966, 2006.

[14] OpSound.org. http://opsound.org 2008.

[15] Yoann Padioleau, Benjamin Sigonneau and Olivier Ridoux. Lisfs: a logical information system as a file system. In *ICSE '06: Proceeding of the 28th international conference on Software engineering*, pages 803–806, New York, NY, USA, 2006. ACM.

[16] Pamela Ravasio, Sissel Guttormsen Schär and Helmut Krueger. In pursuit of desktop evolution: User problems and practices with modern desktop systems. *ACM Transactions Computer-Human Interaction*, Volume 11, Number 2, pages 156–180, 2004.

[17] Dennis M. Ritchie and Ken Thompson. The unix time-sharing system. *Commun. ACM*, Volume 17, Number 7, pages 365–375, 1974.

[18] C.A.N. Soules and G.R. Ganger. Why can't i find my files? new methods for automating attribute assignment. *Proceedings of the 9th conference on Hot Topics in Operating Systems*, Volume 9, pages 20–20, 2003.

[19] Craig A. N. Soules and Gregory R. Ganger. Connections: using context to enhance file search. *SIGOPS Oper. Syst. Rev.*, Volume 39, Number 5, pages 119–132, 2005.

On the relevance of documents for semantic representation

Laurianne Sitbon
National ICT Australia
Queensland University of Technology
Brisbane, Australia
laurianne.sitbon@nicta.com.au

Peter Bruza
Queensland University of Technology
Brisbane, Australia
p.bruza@qut.edu.au

Abstract *The subject of this paper is the quality of semantic vector representation with random projection under various conditions. The main effect we are watching is the size of the context in which words are observed. We are also interested in the stability of such representations since they rely on random initialisation. In particular we investigate the possibility of stabilising terms representations through documents representations. The quality of semantic representation was tested by means of synonym finding task using the TOEFL test on the TASA corpus. It was found that small context windows produces the best semantic vectors with 59.4 % of the questions correctly answered. Processing the projection between terms and documents representations several times was found not to improve the stability of the representation. It was also found not to improve the average quality of representations.*

Keywords Natural Language Techniques and Documents, Semantic spaces, Random projection.

1 Introduction

In computational linguistics, information retrieval and applied cognition, words are often represented as vectors in a high dimensional space computed from a corpus of text. In a variety of studies from cognitive science there have been encouraging results using such representations to replicate human word association norms, for example, semantic association (see, for example, [11], [10], [14]). Therefore, there is some evidence such vector representations do capture semantics of words in a way which accords with those we carry around "in our heads". We will call such representations "semantic vectors". The aim of this paper is to evaluate the effect of granularity on the quality of semantic vectors. Both documents (low granularity) and windows (high granularity) will be used to compute semantic vectors.

Dimensionally reducing the term-documents matrix has often been shown to improve the quality of semantic vectors, for example, latent semantic analysis. However, singular value decomposition, the means for dimension reduction is computationally expensive. Random Indexing [12] offers a computationally inexpensive alternative to dimension reduction [15]. However, the semantic vectors computed by RP are not stable due to the final semantic vector representations depending on initial random seeds. We aim at investigating if the use of an iterative repetition of the projection process between terms and documents representation will lead to more stable representations. We will also investigate if it is more efficient.

The structure of this paper is as follows. In the next section the semantic vector model of random projection will be described. Thereafter the TOEFL test will be used as a means of evaluating semantic vector representations in relation to the questions just raised.

2 Semantic space models

The idea behind semantic spaces is that the meaning of a word is carried by the words that co-occurs with it, and that two words are semantically related if they tend to co-occur with the same words. Co-occurrence is defined with respect to a context, for example, a window of fixed length, or even a document. Co-occurring words can be stored into matrices where the rows can represent the terms and the columns can represent contexts. Each row corresponds to a vector representation of a word. The strength of semantic association between words can be computed by using cosine - the smaller the angle between words representations, the more semantically related they are assumed to be.

Proceedings of the 13th Australasian Document Computing Symposium, Hobart, Australia, 8 December 2008.

2.1 Random Projection

Random Projection (RP) is based on the fact that a term-document matrix computed from a corpus is sparse. The sparsity is large enough that the vector representations can be projected onto a basis comprising a smaller number of randomly allocated vectors. Due to sparseness condition, the basis of random vectors has, in general, a high probability of being orthonomal [2]. The algorithm proceeds in 4 steps after the creation of a document- (or term-) term matrix : (1) create an empty matrix where rows are documents and the columns new random vectors of dimension t, (2) randomly insert in each document vector $t/6$ of positive seeds and $t/6$ of negative seeds, (3) generate a matrix where the rows are terms and the columns new dimensions by adding the corresponding random vector to a term each time it appears in a document, (4) generate the new matrix of documents in new dimensions by adding the corresponding term vector each time a document contains a term.

This can be seen mathematically as the new representation M^{random} of an initial term-document matrix M spanning N terms in d documents and then reduced to t dimensions through a random matrix as in Equation 1.

$$M^{random}_{t\times N} = Random_{k\times d} M_{d\times N} \quad (1)$$

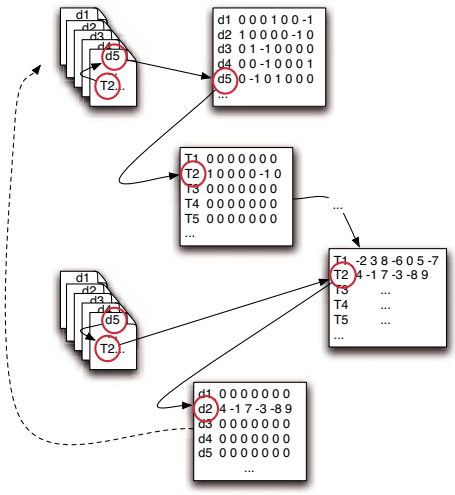


Figure 1: Process of random projection to compute term and documents matrices.

The process is illustrated on Figure 1 where it is suggested that the last two steps could be repeated, using the previously computed matrix for documents instead of the initial random one.

The number of positive and negative random seeds initially followed a Gaussian distribution but it has been shown [1] that a probabilistic distribution with 1/6 is equivalent. This method can be applied to retrieve documents and is referred to as Random Indexing [12]. The initial representation can also be based on contexts [8].

3 Experiments

3.1 Experimental setup

As a means to compare the semantic vector models above, the TOEFL synonym task on the the TASA corpus was used. The basic hypothesis is the higher the TOEFL score, the better the quality of the underlying semantic vector. This choice follows many similar evaluations in the literature and allows our results to be placed in the perspective of other published results. The TOEFL synonym test comprises 80 questions. Each question is multiple choice made of a question word and four potential answers. A question is "incomplete" if the question term is unknown to the model in question, for example, because the question words were not present in the model. In the main experiment both the number of correct answers and the number of answerable questions will be reported. In the best results section the scores will be calculated according to the measure introduced in [9] where non-answerable questions will be scored 0.25 each thereby simulating guessing. The TASA[1] contains 44,486 documents of "General Reading up to 1st year college" . It is assumed American students can learn relevant vocabulary and language usage from these readings. These documents contain 148,221 different non-stop terms for a total of 8,605,497 words. We have performed the experiments using a java implementation of Random Projection provided by the semantic vectors package[2][15]. Both corpus and questions were stemmed with a Porter Stemmer implementation[3] and the corpus is indexed with Lucene[4] to generate the initial matrix. Both term-document and term-context matrices were investigated. The minimum frequency of terms in the initial representation is set to 2 and the values of the initial seeds are either -1 or +1. Over the 80 questions of the TOEFL test, two are incomplete within all models constructed using Random Projection with stemming and 6 are incomplete without stemming. As mentioned previously, the semantic vectors produced by Random projections are somewhat unstable due to the use of

[1]We are grateful to Tom Landauer for providing the TASA corpus
[2]http://code.google.com/p/semanticvectors/
[3]http://tartarus.org/ martin/PorterStemmer/
[4]http://apache.lucene.org

random seeds during initialization. Therefore, the experiments are reported on the basis of 5 runs.

3.2 The effect of dimension reduction

The effect of varying dimension size is evaluated on word-document matrices constructed from the corpus. The notion of projection aims at some generalisation over the initial content of documents. Stemming is a first dimension reduction in that sense since it projects words onto word stems. In the context of random projection the number of random vectors used as a basis for representing the terms is another means of dimension reduction.The average results of testing RP with various dimensions are reported on Figure 2.The average results for non-stemmed data represented on the dashed line are well under the accuracy of stemmed data. Interestingly the highest average value of 37.4

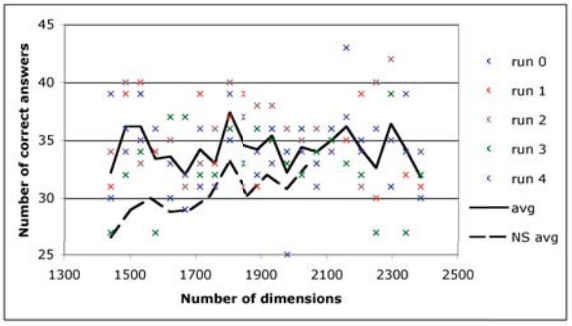


Figure 2: Accuracy of Random Projection for various numbers of dimensions.

correct answers out of 78 (33.2 without stemming) is obtained with 1800 dimensions which is the number of dimensions recommended in [2]. The five individual results for each run on stemmed data consistently exhibit a quite large variation suggesting the underlying vector representations are not that stable.

3.3 Stabilising representations with cycles

The idea suggested in Figure 1 of repeating the last two steps could lead to more stable representations. We have experimented with this idea by using different numbers of cycles (iterations) for the best set of parameters according to previous experiment : stemmed documents projected on 1800 random dimensions. The iterative reallocation of values on random vectors from terms matrix to document matrix and vice-versa doesn't improve neither the quality of representations according to figure 3 nor the stability between two representations.

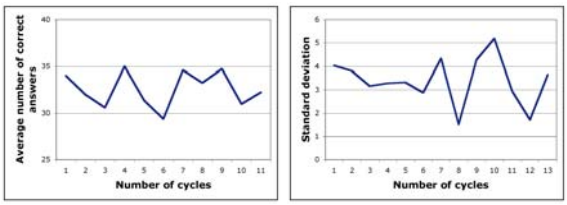


Figure 3: Average (left) and standard deviation (right) of the number of correct answers on 5 RP models for various numbers of cycles.

3.4 Reducing the granularity : context windows

As a mean of evaluating the semantic impact of full documents on semantic representation we have also built models based on an initial term-term matrix computed with a sliding window. This leads to the optimal size of the context in which words are considered to be co-ocurring. Figure 4 shows the average results for various context window sizes with random vectors. The random vector have been computed with 1800 dimensions since this size lead to the best results in previous experiments. The window sizes refer to the total number of words taken into account including the target word. The results show

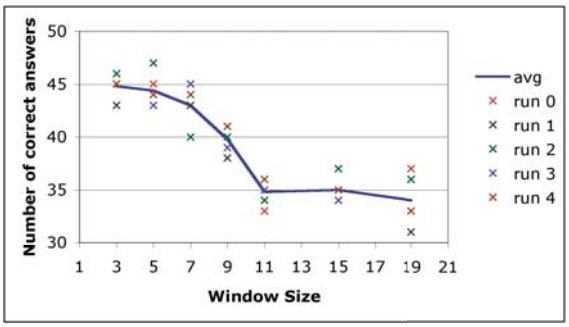


Figure 4: Accuracy of Random Projection using a word-word matrix with different context window sizes.

that the smallest context window (3 words) provides the most accurate results on the TOEFL test with 45 correct answers out of 78 in average. This implies that the model constructed based on the co-ocurrencies of the words only with the previous and the next word (these not being stop-words) performs the best for the synonym test. With a context window size of up to 9, the results are higher (40 correct answers out of 78) than when using whole a document as context. It is however important to note that context based models are more computationally expensive.

4 Conclusion

The best average obtained with a minimal window of size 3 words leads to a score of 59.4% of accuracy according to TOEFL evaluation. Comparison with previous published work should be viewed in light of doubt regarding the size of the underlying corpus. In this paper, the corpus comprises 44,486 documents whereas in other studies reported in the literature, the size is either 37,600 or 30,473 articles. We are unable to explain this discrepancy. Several results have been reported on the use of LSA. [9] had 64.5% of correct answers, [6] report results of 55.31% correctly answered questions for LSA and [4] found 63.6 % of correct responses using the cosine similarity and 61.5% using an inner product instead. Random Indexing [7] using word contexts gave 35-44% with unnormalised 1800 dimensional vectors and 48-51% with normalised vectors .

The models developed for information retrieval purposes tend to show that representing semantic spaces at the document level might benefit from a context window representation of words. One of the reason of the failure of document sized contexts could be the variety of topics present in single documents resulting in noisy representations. A intermediate solution still to be tested is to create topically coherent sub-documents using a linear segmentation algorithm.

In the future, it will also be worth investigating how stable semantic vectors are with slight corpus changes, or on larger corpora. Potential other tasks for examining semantic vectors are replications of free association [13] and priming [5].

Acknowledgements NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

[1] D. Achlioptas. Database-friendly random projections. In *Proc. of the Symposium on Principles of Database Systems*, pages 274–281, 2001.

[2] E. Bingham and H. Mannila. Random projection in dimensionality reduction : applications to image and text data. In *Proc. of the 7th KDDM*, pages 245–250, New York, NY, USA, 2001.

[3] D. M. Blei, A. Y. Ng and M. I. Jordan. Latent dirichlet allocation. *Journal of machine learning research*, Volume 3, pages 993–1022, 2003.

[4] T. L. Griffiths and M. Steyvers. Topics in semantic representation. *Psychological review*, Volume 114, Number 2, pages 211–244, 2007.

[5] M. N. Jones, W. Kintsch and D. J. K. Mewhort. High-dimensional semantic space accounts of priming. *Journal of memory and language*, Volume 55, pages 534–552, 2006.

[6] M. N. Jones and D. J. K. Mewhort. Representing word meaning and order information in a composite holographic lexicon. *Psychological review*, Volume 114, Number 1, pages 1–37, 2007.

[7] P. Kanerva, J. Kristoferson and A. Holst. Random indexing of text samples for latent semantic analysis. In Erlbaum (editor), *Proc. of the 22nd annual conference of the cognitive science society*, New Jersey, USA, 2000.

[8] J. Karlgren and M. Sahlgren. *Foundations of real-world intelligence*, Chapter From Words to Understanding, pages 294–308. Uesaka, Y., Kanerva, P. & Asoh, H., 2001.

[9] T. Landauer and S. T. Dumais. A solution to Plato's problem : the latent semantic analysis theory of acquisitionm induction and representation of knowledge. *Psychological review*, Volume 104, Number 2, pages 211–240, 1997.

[10] W. Lowe. Towards a theory of semantic space. In J. D. Moore and K. Stenning (editors), *Proc. of the 23rd Annual Conference of the Cognitive Science Society*, pages 576–581. Lawrence Erlbaum Associates, 2001.

[11] K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behaviour research methods, instruments and computers*, Volume 28, Number 2, pages 203–208, 1996.

[12] M. Sahlgren. An introduction to random indexing. In *Proc. of Methods and Applications of Semantic Indexing Workshop*, Copenhagen, Denmark, 2005.

[13] L. Sitbon, P. Bellot and P. Blache. Evaluation of lexical resources and semantic networks on a corpus of mental associations. In *Proc. of the 6th LREC*, Marrakech, Morocco, 2008.

[14] D. Widdows. *Geometry and Meaning*. CSLI Publications, 2004.

[15] D. Widdows and K. Ferraro. Semantic vectors : a scalable open source package and online technology management application. In *Proc. of the 6th LREC*, Marrakech, Morocco, 2008.

Exploring the benefit of contextual information for boosting TREC Genomic IR performance

Bader Aljaber , Nicola Stokes‡ , James Bailey* ‡‡ , Yi Li* ‡‡*

* Dept of Computer Science and Software Engineering, The University of Melbourne, Australia
‡School of Computer Science and Informatics, University College Dublin, Ireland
‡‡NICTA Victoria Research Laboratory, Australia

{*baljaber, jbailey, yli8*}*@csse.unimelb.edu.au, nicola.stokes@ucd.ie*

Abstract *Query Expansion is a widely used technique that augments a query with synonymous and related terms in order to address a common issue in ad hoc retrieval: the vocabulary mismatch problem, where relevant documents contain query terms that are semantically similar, but lexically distinct. Standard query expansion techniques include pseudo relevance feedback and ontology-based expansion. In this paper, we explore the use of contextual information as a means of expanding the context surrounding the unit of retrieval, rather than the query, which in this case is a document passage. The ad hoc retrieval task that we focus on in this paper was investigated at the TREC 2006 Genomic tracks, where systems were required to retrieve relevant answer passages. The most commonly reported indexing strategy was passage indexing. Although this simplifies post-retrieval processing, retrieval performance can be hurt as valuable contextual information in the containing document is lost. The focus of this paper is to investigate various contextual evidence of similarity outside of the passage such as: query/full-text similarity, query/citation sentence similarity, query/title similarity, query/abstract similarity. These similarity scores are then used to boost the rank of passages that exhibit high contextual evidence of query similarity. Our experimental results suggest that document context provides the strongest evidence of contextual information for this task.*

Keywords Passage Retrieval, Contextual Document Expansion and Ranking Strategies.

1 Introduction

Query expansion is a technique used in Information Retrieval (IR) to address the synonymy problem. More specifically, a relevant document, which contains semantically related words that are lexically dissimilar to the query, will appear less related than it actually is. This is also referred to as the*vocabulary mismatch problem* [3]. This is a very common problem, which affects IR effectiveness more than the problem of query term ambiguity [5]. An alternative to query expansion is document expansion - the process of adding related terms to the document's representation. In this paper, we explore the use of document expansion in a passage retrieval task. The TREC 2006 and 2007 (Text REtrieval Conference) Genomic track task requires the retrieval of extracted answer passages, in response to natural language questions. The most commonly used indexing strategy used by track participants for this task was passage indexing. Although this simplifies post-retrieval processing, retrieval performance can be hurt as valuable contextual information in the containing document is lost by this indexing strategy. Thus, expansion techniques are needed. Work in [7] investigated the impact of various *query* expansion term types on passage retrieval effectiveness in this Genomic IR task. The results showed that a significant improvement can be gained when ontologically related words (synonyms, hypernyms, hyponyms) are used in query expansion.

In this paper, we extend the work presented by Stokes et al. [7] by exploring different types of *contextual information* as a means of expanding the context surrounding the unit of retrieval (a passage), rather than the query. This is a type of document expansion. So, we investigate the use of various sources of contextual evidence of similarity outside of the passage such as: query/full-text similarity, query/citation sentence similarity, query/title similarity and query/abstract similarity. These similarity scores are then used to boost the rank of passages that exhibit high contextual evidence of query similarity. Our results indicate that document context is the strongest source of contextual evidence for this task.

Related Work. Similarly to query expansion, document expansion can be used to overcome the problem of synonymy. Document expansion techniques, enrich documents off-line with related terms during indexing. This type of expansion can reduce the overheads of query expansion at query time.

Billerbeck and Zobel [1] proposed two new corpus-based methods for document expansion. In the first method, each document is treated as a query and augmented by related terms. In the second method, each term in the corpus is treated as a

Proceedings of the 13th Australasian Document Computing Symposium, Hobart, Australia, 8 December 2008.

query and augmented by related terms and used to rank documents accordingly. Overall, Billerbeck and Zobel's experiments showed that compared with query expansion, document expansion methods achieved relatively poor improvements. That might be because the specific topic of the original documents is significantly changed when related terms are added.

2 TREC Evaluation Data and Metrics

In this section, we will describe the data collection, retrieval task and the evaluation metrics we use. All our experiments were conducted on the document collection used in *2006 and 2007 Genomic TREC task*[1]. The TREC collection consists of *162,259* full-text journal articles from 49 journals which are electronically published via the Highwire Press site. Besides that, 28 topics expressed as natural language questions are also provided. Participants of the task were required to implement a retrieval system and submit the first 1,000 ranked passages returned by their systems for each of the topics (Hersh et al. [4]). Passages in this task can be defined as text sequences that must occur within paragraph boundaries (delimited by HTML tags). For evaluation, human judges evaluate the relevance of passages retrieved. More precisely, passage boundaries were defined, and each relevant answer was assigned a set of topic tags (called *aspects*) from a control vocabulary of MeSH terms. MeSH stands for ***Me***dical ***S***ubject ***H***eadings[2].

To evaluate the system effectiveness, we use *M*ean *A*verage *P*recision (**MAP**), which is considered one of the most common IR evaluation metrics. In document retrieval systems, the document MAP score is calculated as follows: for a given query, the average of all the precision values at each recall point in document ranked list is first calculated. Then, the mean of all the query average precision scores is determined. The TREC Genomics Track also defines a variant of this MAP score. The Passage MAP is similar to the document MAP. However, since passage retrieval is a question answering task, a special metric which factors in the length of the passages retrieved is introduced. So, the passage MAP is calculated as the fraction of characters in the system passage overlapping with the gold standard answer, divided by the total number of characters in every passage retrieved up to that point in the ranked list. Consequently, extra characters retrieved will (negatively) affect the final MAP score.

Stokes et al. [7] defined another version of the MAP, called the *paragraph MAP* score. The paragraph MAP calculates the fraction of paragraphs retrieved that contain a correct passage, divided by the total number of paragraphs retrieved. As before, the average of these scores at each recall point is the final score for that topic. In this metric, extra characters retrieved cannot affect the final MAP score and the system will get "full-marks" if it returns the paragraph that the gold standard passage occurs in.

In our experiments, Mean Average Precision (MAP) is used to evaluate system performance at three different levels of information granularity: Passages, Documents and Paragraphs.

3 System Description

In this paper, we augment an IR query expansion system first proposed in [7]. The authors introduced a novel concept normalization ranking metric, which maximizes the impact of query expansion in the genomic domain. More specifically the system ensures that documents containing multiple unique concepts are ranked higher than those which make reference to the same concept multiple times; and expansion terms (synonyms and related terms) for the same concept are not given undue influence by the ranking metric.

Briefly, we will describe the genomic retrieval system presented in [7] with emphasis on the part that we will extend; followed by an explanation of our extension to the system. The architecture of that genomic retrieval system is shown in Figure 1. The data collection is first prepared and separately indexed on paragraph and other contextual information representations; a query is then taken and expanded with synonyms found in other external resources such as MeSH terms which stand for ***Me***dical ***S***ubject ***H***eadings. Based on that, two sets of ranked outputs are retrieved. First, a set of candidate paragraphs is retrieved based on paragraph indexing and then ranked. Second, a set of documents is retrieved based on the indexing of other contextual information such as the document contexts (that is the original document representation) and then ranked. The candidate paragraphs are reduced (that is what we will call later as a *passage reduction*) in order to extract relevant answer passages to the query (in our case, queries are natural language questions). These extracted passages are then ranked and presented to the user. The overall process is referred to as a *PASSAGE* run.

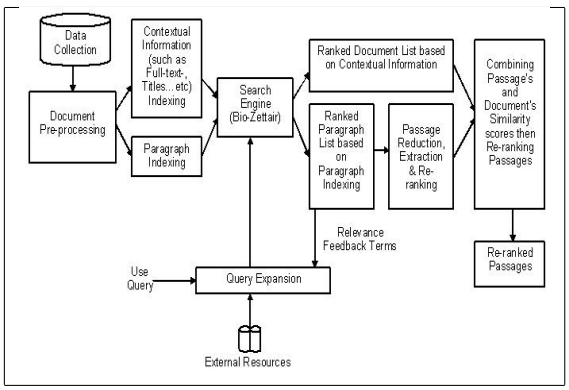


Figure 1: The architecture of the passage retrieval system.

[1] http://ir.ohsu.edu/genomics/2006protocol.html

[2] MeSH terms are managed and created by the United States National Library of Medicine (NLM), http://www.nlm.nih.gov

Stokes et al. [7] found that query expansion with synonyms from domain-specific terminology resources achieves a significant performance over a baseline system. Further, improvements in Passage MAP score were achieved when the passage reduction process was performed on retrieved paragraphs. However, although Passage MAP increased significantly (from 0.108 to 0.127), Document level MAP drop significantly (from 0.534 to 0.507).

To address this problem, Stokes et al. [7] proposed a new passage re-ranking method which considers both the relevance of the passage to the query, and incorporates the relevance score of the document containing that passage. In other words, the similarity scores of the retrieved passages are linearly combined with the similarity scores of their containing documents. This method boosts Passage and Document MAP scores and can be summarized as follows:

1. First perform query expansion, query the index, and then use passage extraction and re-ranking to find the top 1000 passages for each query. A query is defined as a set of concept and non- concept terms or phrases. For example, the query "What is the role PRNP in Mad Cow Disease?", has two concept terms 'PRNP' and 'Mad Cow Disease' and one non-concept term 'role', the rest are stopwords. By splitting the query in this manner, we can ensure that the occurrence of less informative, non-concept terms do not have an inflated weight of importance in the similarity calculation.

2. The top 1000 passages are then divided into different concept level groups. That is, we group documents in the ranked list based on the number of query concept terms they contain, where documents with all query concepts reside at the top of the ranked list.

3. Within each group, passages are re-ranked by combining their similarity scores with their containing document's similarity score.

So, for a passage i which has a similarity score *Pi* and whose containing document has a similarity score *Di*, the final combination score *Si* is calculated as:

$$Si = Pi \times \frac{Di}{Dmax} \times Pmax \qquad (1)$$

where *Pmax* and *Dmax* are the maximum similarity scores of all the 1000 passages and their containing documents.

Our Contribution: Our extension to the work in [7] is based on the investigation of additional types of contextual information for re-ranking the retrieved passages, in order to attain better IR performance for this task. In other words, as well as using the document context's similarity scores in re-ranking the retrieved passages, we also examine and evaluate the effect of using similarity scores based on different types of contextual information, which are outlined in the next section.

4 Experimental Results

It has been shown in [7] that when the document context's similarity scores are used to re-rank retrieved passages, MAP score increases for both Passage level MAPs (that is, from 0.108 to 0.137) and document level MAPs (that is, from 0.534 to 0.543) are observed. Hence, as already mentioned, use this result to motivate our investigation of exploring the benefit of using other contextual information for boosting TREC Genomic IR performance such as Citation Contexts, Titles, Abstracts and MeSH terms.

A citation context is essentially the text surrounding the reference markers (e.g the 'cite' command in LaTeX) used to refer to other scientific works. These citation contexts are essentially descriptive fragments and are likely to contain synonymous or related terms to the document being cited. Consequently, they can be used as an alternative representation of the contents of a document. Citation contexts have been used by a number of techniques in information retrieval [6].

From the entire collection, we were able to extract the citation contexts for *3475* documents. More specifically, we have omitted documents which have been rarely cited by other documents in our collection, as no meaningful citation representations can be used for these documents. Also, we have used a fixed citation window size of 50 terms before and after the citation marker as suggested by Bradshaw [2].

Experimental results shown in this paper present the MAP scores of the system at the Passage level, Document level and Paragraph level (that are paslev, doclev and parlev respectively) for the following system runs:

- Baseline: the best expansion run presented in [7].
- PASSAGE: the baseline system when paragraphs are reduced to answer passages (called passage reduction in the previous section).
- PASSAGE + Doc: the PASSAGE run where the retrieved passages are re-ranked using the *Document* context's similarity scores.
- PASSAGE + Cit: the PASSAGE run where retrieved passages were re-ranked using the *Citation* context's similarity scores.
- PASSAGE + Title: the PASSAGE run where the retrieved passages were re-ranked using the *Title* context's similarity scores.
- PASSAGE + Abstract: the PASSAGE run where the retrieved passages were re-ranked using the *Abstract* context's similarity scores
- PASSAGE + MeSH: the PASSAGE run where the retrieved passages were re-ranked using the MeSH context's similarity scores.

Looking at Table 1, we can see that at the passage (paslev) and paragraph (parlev) MAP scores, show performance improvements over the baseline run. Not only that, but at the passage level MeSH contexts (PASSAGES +MeSH) can marginally outperforms document contexts (PASSAGES + Doc). While, at the document evaluation level (doclev), no representation can obtain better than the PASSAGE + Doc run score.

	paslev MAP		doclev MAP		parlev MAP	
Baseline	0.108		0.534		0.356	
PASSAGES	0.127	17.84%	0.507	-5.05%	0.362	1.78%
PASSAGES + Doc	**0.137**	27.13%	**0.543**	1.67%	**0.384**	8.01%
PASSAGES + Cit	0.119	10.33%	0.500	-6.42%	0.353	-0.83%
PASSAGES + Title	0.126	16.94%	0.508	-4.88%	0.363	2.15%
PASSAGES + Abstract	0.123	14.42%	0.525	-1.71%	0.376	5.75%
PASSAGES + MeSH	**0.138**	28.21%	0.519	-2.74%	0.365	2.71%

Table 1: Table showing the effectiveness of re-ranking passage retrieval results (that is PASSAGES) with other contextual evidence of query/passage similarity.

Table 2 presents a second set of context experiments, where in this case every containing document of a passage is now represented by a combined representation, which combines either its title, abstract or MeSH terms with its citation contexts (if any), that is *PASSAGE + (Title+Cit)*, *PASSAGE + (Abstract+Cit)* and *PASSAGE + (MeSH+Cit)*). For the addition of citation context terms, despite the fact that some minor increases at particular MAP levels can be seen, overall the results are inconsistent. This may be explained by the fact that in many cases we do not have sufficient citation sentences to make up a citation representation for a combining document. A final combination run is also included in this table which combines all contextual information (PASSAGES + (Abs+Title+MeSH+Cit)). However, the PASSAGE+DOC run still performs this. Using a paired Wilcoxon signed-rank test, this PASSAGE+DOC run was found to be statistically significant better (at all MAP levels) when compared with the baseline and PASSAGE runs at the 0.05 confidence interval.

	paslev MAP		doclev MAP		parlev MAP	
Baseline	0.108		0.534		0.356	
PASSAGES	**0.127**	17.84%	0.507	-5.05%	**0.362**	1.78%
PASSAGES + Doc	**0.137**	27.13%	**0.543**	1.67%	**0.384**	8.01%
PASSAGES + (Title+Cit)	**0.124**	15.17%	0.508	-4.88%	**0.363**	2.22%
PASSAGES + (Abstract+Cit)	**0.123**	14.32%	0.526	-1.58%	**0.376**	5.77%
PASSAGES + (MeSH+Cit)	**0.135**	25.43%	0.518	-2.90%	**0.362**	1.86%
PASSAGES + (Abs+Title+MeSH+Cit)	**0.127**	17.87%	0.528	-1.16%	**0.380**	6.79%

Table 2: Table showing additional combinations of context information, which are used to re-rank passages returned by the PASSAGE run.

The results in Table 3 show MAP scores for the top performing systems on the TREC 2006 Genomic Track tasks. TREC_MEDIAN refers to the median values of each MAP score for the official TREC results. Okapi BM25 is the baseline used in the task. Some of the other top performing runs have a detailed description given in [7]. We can see that the majority of MAP scores achieved by our context re-ranking runs outperform the scores of these system, with the exception of UIC_SIGIR and UIC_SIGIR for document level MAP score.

Discussion. In summary, a number of conclusions can be drawn from our experiments:

Run	paslev MAP	doclev MAP	parlev MAP
TREC_MEDIAN	0.037	0.308	0.124
UIC_GenRun3	0.123	0.532	0.342
THU2	0.099	0.434	0.265
NLMinter	0.084	0.473	0.272
UIC_SIGIR	NA	0.539	NA
Okapi	0.048	0.336	0.137

Table 3: Table showing performance of the top performing TREC systems on the Genomics Track.

- The use of the document context brings the best IR performance at passage, document and paragraph level MAPs
- The use of other contextual information, especially abstracts and MeSH terms, can also boost IR performance compared with baseline system, particularly for passage and paragraph level MAPs
- In the absence of the availability of the document context (since the source of many documents is not easily/freely available), the use of other, publically available information contexts (such as MeSH terms, Abstracts and Titles), is a useful way to improve IR performance.
- Use of citation contexts appears promising for improving IR performance. However, it is difficult to obtain citation contexts for most documents. In our collection, only 2.14% of documents could be sufficiently described using citation contexts.

5 Conclusions

A successful implementation of the retrieval ranking method using different types of contextual information can deliver further improvements. In particular, for this passage retrieval task we found that MeSH terms and Document similarity contexts, further boost the performance of an already competent query expansion information retrieval system.

References

[1] B. Billerbeck and J. Zobel. Document expansion versus query expansion for ad-hoc retrieval. In *the 10th ADCS*, pages 34–41, 2005.

[2] S. Bradshaw. Reference directed indexing: Redeeming relevance for subject search in citation indexes. In *the 7th ECDL*, pages 499–510, 2003.

[3] G. Furnas, T. Landauer, L. Gomez and S. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, Volume 30, Number 11, pages 964–971, 1987.

[4] W. Hersh, A. Cohen, P. Roberts and H. Rekapalli. Trec 2006 genomics track overview. In *The 15th TREC*, November 2006.

[5] R. Krovetz and W. Croft. Lexical ambiguity and information retrieval. *ACM Trans. Inf. Syst.*, Volume 10, Number 2, pages 115–141, 1992.

[6] A. Ritchie, S. Teufel and S. Robertson. Using terms from citations for ir: Some first results. In *the 30th ECIR*, pages 211–221, 2008.

[7] N. Stokes, Y. Li, L. Cavedon and J. Zobel. Exploring criteria for successful query expansion in the genomic domain. *Information Retrieval*, 2008.

WebKnox: Web Knowledge Extraction

David Urbansky
School of Computer Science and IT
RMIT University
Victoria 3001 Australia
davidurbansky@googlemail.com

James A. Thom
School of Computer Science and IT
RMIT University
Victoria 3001 Australia
james.thom@rmit.edu.au

Marius Feldmann
Department of Computer Science
University of Technology Dresden
Germany
feldmann@rn.inf.tu-dresden.de

Abstract *The paper describes and evaluates a system for extracting knowledge from the web that uses a domain independent fact extraction approach and a self supervised learning algorithm. Using a trust algorithm, the precision of the system is improved to over 70% compared with a baseline of 52%.*

Keywords Information Extraction, Web Mining

1 Introduction

Given the vast quantity of repeated information available on the web, it has become possible to more reliably extract factual knowledge about many different entities. Therefore it is useful to have a automatic approach that finds pages containing facts and extracts the best answers. This paper describes and evaluates a system WebKnox (Web Knowledge eXtraction) for extracting knowledge from the web.

WebKnox's input consists of the following parts:

1. The **concepts**, e.g. `Car` or `Country`.
2. The **attributes** for each concept. The attributes determine which facts are searched for each entity in the concept. E.g. for the `Country` concept attributes could be `population` and `capital`.
3. The **entities** for each concept. The entities and attributes together build the templates that are filled in the fact extraction process. E.g. for the `Country` concept, `Australia` and `Germany` are valid entities.

The following are the main contributions in this paper. We present a domain independent fact extraction approach that retrieves web pages with factual information, analyzes those semi-structured pages, and extracts facts from generic structures and formats that commonly occur on websites. We introduce a self supervised learning algorithm that automatically estimates the precision of the different structures used to extract the facts. We show how the extraction precision for numeric values can be increased by cross validating them with numeric values from other entities of the same concept. We demonstrate how a trust value can be assigned to the extracted facts which is used to rank the extractions and help the end user determine which facts can be trusted.

Proceedings of the 13th Australasian Document Computing Symposium, Hobart, Australia, 8 December 2008.

2 Background

2.1 Web Information Extraction

This section gives background information about information extraction in general and the extraction tasks for the web in particular.

In contrast to information retrieval (IR), where the task is to find relevant information for a given query and to rank the results, information extraction (IE) is the process of extracting information to a given target structure such as a template or an ontology. The IE tasks are defined by an input (e.g. an HTML page) and an output (e.g. a populated database) [2].

There are several main tasks in information extraction.

1. *Named Entity Recognition* (NER) is the task that identifies entities in a given text. It is the easiest task, however it is more recognition than extraction because no new entities are extracted.
2. *Coreference Resolution* (CO) identifies identity relationships between entities in texts.
3. *Entity Extraction* (EE) is the task of discovering new instances of a concept.
4. *Fact Value Extraction* (FVE) is the task of finding values for given attributes for a given entity. e.g.

the entity "Australia" and the attribute "population" are given and the value for the attribute is searched.

5. *Fact Extraction* (FE) is a similar task to FVE but no attributes are given for the extraction process.

2.1.1 Web Information Sources

The choice of the information extraction technique depends on the format of the source. The world wide web consists of documents that belong to one of the three main types of sources: unstructured, semi-structured and structured.

In web information extraction, semi-structured sources are mainly HTML files. That is because they contain lots of unstructured data as texts but use tags to structure that data for rendering purposes.

Internal Representation Before examining the main techniques that access and extract from these web sources it is important to understand that the content can be represented in two main forms.

1. A *hierarchy of nodes* represents the source as a tree of nodes (such as element or text nodes). This representation is usually instantiated using the *Document Object Model* (DOM).

2. A *tokenized string* represents the source as a parsed string of words, numbers etc. (so called tokens). This representation is often used for free text as there is no other structure that can be used to represent the information.

2.1.2 Information Extraction Techniques

Natural Language Processing (NLP) can be employed for web information extraction using a set of techniques that make the natural language more machine readable. Those techniques are for example *tokenization*, *sentence splitting*, *orthomatching* (coreference resolution) and *regular expressions*.

Wrapper Induction "A program that makes an existing website look like a database is called a wrapper." [3]. Wrappers perform pattern matching to find the information of interest. The main goal in learning a wrapper is to find a general description of what the information that is supposed to be extracted looks like. Writing a wrapper by hand is labor intense and over the time more automation has been introduced. Chang et al. [2] classify these wrapper techniques in four categories with increasing automation:

1. *manually-constructed wrappers*. The user writes a wrapper for every web site he wants to extract information from which means that he has to have a profound understanding of programming languages. That requirement makes it however impractical for a broad domain approach.

2. *supervised wrapper construction*. For supervised learning a wrapper, a human labels information on a set of HTML pages he wants to have extracted. These labels are taken as positive examples for the learning algorithm. Non-labeled data serves as negative examples. The user does not need any programming knowledge but needs only to be able to mark up the content or use a graphical user interface to do that.

3. *semi-supervised wrapper construction*. Semi-supervised systems require the user not to give a whole exact labeled set of web pages but rather guess extraction patterns on given examples. The user then has to decide which pattern is the correct one, thus the extraction process becomes supervised.

4. *unsupervised wrapper construction*. The unsupervised approach requires no user interaction. While former approaches needed a user to specify the data of interest, the extraction target in supervised extraction are data rich regions of the website [2].

Wrapper induction techniques are primarily used for structured or semi-structured sources as many techniques rely on the DOM tree.

2.1.3 Correctness of Extracted Information

Traditional IE focuses on extracting as much information as possible from a small corpus whereas web information extraction systems often rely on the redundancy of web content [6]. That means that the focus of the extraction techniques should be set on the precision as the recall automatically comes with many mentions of the entity or fact that is extracted. One major problem with web information extraction systems is that the quality of the extractions can vary, i.e. extracted entities may not really belong to the concept they were assigned to or facts are wrong.

Simple Scoring For the fact extraction task a simple scoring, based on the number and quality of sources can be used to decide which fact extraction is correct and which is not [7]. The effectiveness of simple scoring relies however on the assumption that correct facts are extracted more often than incorrect facts which also depends on the extraction technique and the type of the fact. Rare facts for example might be extracted correctly but do not score very high.

Pointwise Mutual Information Etzioni et al. [4] use patterns as discriminators to ensure the correctness of an extracted fact or entity. That means they use these discriminators as queries for web search engines and calculate pointwise mutual information (PMI) between the extraction and the discriminator with the hit counts. If E is an extracted entity and D is the discriminator phrase, the PMI can be calculated as in the Equation 1.

$$PMI = \frac{Hits(E+D)}{Hits(E)} \quad (1)$$

2.2 State-of-the-art Systems

KnowItAll [4] is a domain independent, unsupervised system that automatically extracts entities and facts from the web. KnowItAll is redundancy-based which means it relies on the assumption that a fact or entity occurs many times on the web. The system's strength is finding new entities for a given class (EE task). To do that, it uses a set of domain independent patterns and queries a search engine with that pattern.

The input for KnowItAll is a set of concepts, attributes and relations. The extractor module queries search engines with extraction patterns and performs a shallow syntactic analysis. A discriminator is an extraction pattern with alternative text. The assessor module queries search engines with discriminators to validate a particular extraction and ensure the precision of the system. For that purpose, KnowItAll uses PMI.

KnowItAll is specialized in extracting entities and has its limitations in extracting facts. It can extract entity relations found in free text but much information, especially numbers (e.g. the population of a country) is given in table structures that are not evaluated by KnowItAll. Also the PMI score for validating the extractions would most likely not work with numeric extractions.

Textrunner [1] goes one step further beyond the capabilities of KnowItAll, as it does not require any user input which is more scalable and easier to apply for new domains. Its only input is the corpus of web pages, and information is extracted in a single pass. This happens in three steps for every sentence read: (1) The noun phrases of the sentence are tagged, (2) nouns that are not too far away from each other are put into a candidate tuple set and (3) the tuples are analyzed and classified as true or false.

GRAZER [7] is a system that corroborates and learns new facts. The input for GRAZER are seed facts (attribute-value pairs) for given entities. Entities and seed facts are automatically generated using specialized wrappers. For the given entities, relevant pages are obtained. Relevant pages are those that have a mention of the entity. On these pages the seed facts are corroborated and new facts are extracted. The system searches for mentions of the seed facts on the relevant pages and adds the source if the fact was found on the page. The corroboration happens in free text and in structured HTML as all tags are removed and only the area around the attribute name is searched for the mention of the value.

Although GRAZER does not need an ontology about the knowledge domain as an input it relies on a set of seeds for entities and facts. These seeds are obtained in an non generic way by inputting the data by hand, which is labor intense or by scraping sources with specialized wrappers. The same facts are extracted several times and are treated as new facts when they have a different attribute which is just a synonym, e.g. "Birthday:17.01.1962" is another fact than "Date of Birth:17.01.1962".

3 Design

This section introduces the design of our system for fact extraction from the web. First, the knowledge to be extracted is encoded in an ontology, then entities are given, and then the fact extraction process automatically finds the values for the specified attributes.

3.1 Knowledge Representation

Before the extraction process can start WebKnox needs to know what concepts, attributes and entities exist. This knowledge is called *prior knowledge*. The prior knowledge for WebKnox is modeled in an ontology using OWL. Therefore, all concepts and attributes are defined in the *knowledge ontology* and the entities and facts that are extracted are stored in another separate *data ontology*.

The purpose for the knowledge ontology is (1) to define the knowledge represented in the data ontology and (2) to serve as an input for the extraction process. In the knowledge ontology every attribute gets an *OWL data type property* assigned to it. This determines which type of value the attribute will have and can be used for (1) other programs reading the ontology, trying to parse the data and (2) for the extraction process to know which values on a source are candidates for the attribute. A datatype property can have any XSD datatype[1] but WebKnox only uses the following:

1. **String**: A string is a sequence of characters, WebKnox will however only consider proper nouns as fact candidates for a string attribute, i.e. only a sequence of words starting with a capitalized character or a number are considered to be possible answers.

2. **Boolean**: Attributes with a boolean value can either have true or false as a value. WebKnox searches boolean values only in tables and looks for "yes" and "no" occurrences.

3. **Decimal, Double, Float, Integer, Int, Long**: These are numeric attributes which are all handled equally by WebKnox. Every numeric attribute is handled as a double and only occurrences of numbers are extracted as fact candidates for the attribute.

4. **Date**: An XSD date is a string given in a standardized UTC format: YYYY-MM-DD. WebKnox will look for several representation of dates on web sources and tries to transform these back to the UTC format.

[1] `http://www.w3.org/TR/xmlschema-2/#d0e11239`

5. **AnyType**: Attributes with values that do not match any previously mentioned data type can have the AnyType property. WebKnox takes all characters around or after the attribute on the web source into account when determining the fact candidates. Thus AnyType can be used for strings that are not proper nouns.

3.2 Fact Extraction

The first process is the retrieving of the source pages which gets an entity and its attributes as input. The extraction process then extracts the values for the entity's attributes from the websites retrieved. The extracted facts are normalized and eventually the trust in the extractions is calculated.

3.2.1 Retrieving Fact Pages

Retrieving relevant pages that host the searched facts is a crucial process that has to be tightly coupled with the extraction process. As input data the source retrieval process gets the names of the entities and attributes that are being searched for. The process then queries a search engine and outputs the retrieved pages together with information about which attributes are expected on the page. This output is fed into the extraction process. The focus lies on retrieving semi-structured HTML pages as they are easy to access via generic search engines as Google[2].

To retrieve pages that have the searched facts present, WebKnox uses two kinds of generic queries. The first kind is called *multi-attribute query*, it tries to find pages relevant to the entity and extract all searched facts from the retrieved pages (e.g. the query "Australia"). The second kind is the *single-attribute query* and is focused on each single attribute, i.e. it queries the search engine with attribute specific terms (e.g. the query "Australia population").

The retrieved websites are then passed to the fact extraction process. The fact extraction process also gets information about the type of the query so that it only looks for a single attribute on single attribute pages and tries to find all attributes on general fact pages retrieved by multi-attribute queries.

3.2.2 Exploiting Structure and Format of Web Pages

The quality of extracted facts can be increased by using different extraction structures for different types of fact appearances. As covered in the background a common approach for fact extraction is to use the complete website content and simply remove all HTML tags (as done by the GRAZER system [7]). That however also removes all advantages that come with the semi-structured type of HTML documents. WebKnox differs from current approaches as it takes the *extraction structures*, i.e. the different generic formats and structures into account that are used to represent facts on web pages.

Definition 1 (Extraction Structure)**.** An **extraction structure** is the pattern or format the extracted fact is represented on a source.

These extraction structures are *phrases*, *tables*, *colon patterns* and *free text*.

Phrases are natural language representations of facts for a specific entity. For example the phrase "The capital of Australia is Canberra" is used on a website. The phrase covers the fact `capital:Canberra` for the entity `Australia`. Ideally the searched value for the attribute appears right after the "is" in the phrase. WebKnox uses only two phrases: `the ATTRIBUTE of ENTITY is` and `ENTITY's ATTRIBUTE is`. These phrases are also used by the source retrieval process to discover pages that state these phrases.

Tables are important HTML structures on the web that are used to represent many facts, which led to numerous wrapping techniques. Keeping the HTML structure allows to traverse in the DOM Tree of the website and find corresponding attribute-value pairs in tables. Figure 1 shows an example[3] of a rendered HTML table in a) and the DOM representation of that part in b). That is a very easy example of a table but also a very common one. By identifying the td-element with the attribute, the sibling td-element with the value can be found and only the text inside that element is extracted.

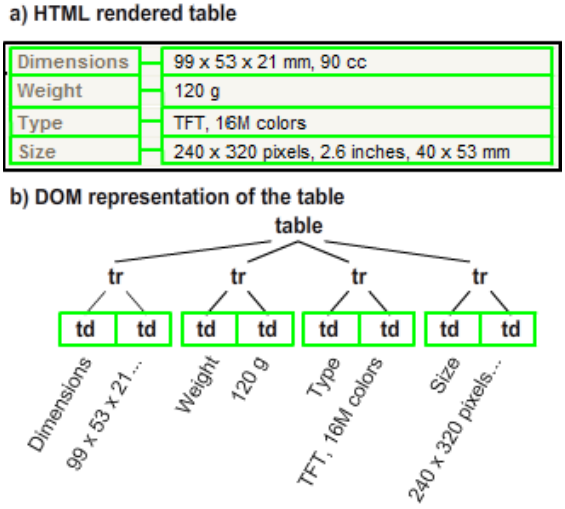


Figure 1: A table for mobile phone specifications

Colon pattern is the text that is right after a colon (":"). Often facts are given in an unstructured way (no tags) but with the format `ATTRIBUTE:VALUE` so that only the text after the colon needs to be extracted. Figure 2 shows an example[4] of this representation, where

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

[3] Table from `http://gsmarena.com/nokia_n95-1716.php`

[4] Data from `http://engadget.com/2008/08/30/msis-wind-u90-to-boast-8-9-inch-display/`

a) depicts the HTML rendered version while b) shows the text as it is seen when the separating tags are removed (replaced with whitespace). If one would try to extract the processor attribute (`PROC`), expecting a numeric value and not noticing the format, the 2008 would be extracted as it is closer to the processor attribute than the correct value `1.6GHz` after the colon. The colon pattern can therefore help increasing the fact extracting precision in a very simple manner.

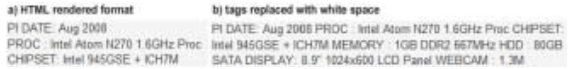


Figure 2: An example for fact representation in a colon pattern, a) shows the presentation in rendered HTML, whereas b) shows the data when tags are removed (replaced with white space)

Free text is the absence of structure (tags) and additional format (phrase or colon pattern). Facts can also appear in long paragraphs of text but as no further information about the structure and format is given, all text around the attribute has to be considered as a valid answer for the attribute's value. It is assumed that always the next matching value closest to the attribute is extracted. WebKnox takes the sentence in which the attribute appears as the boundary. This way incorrect information further away is not extracted as well. Using information found in free text increases the recall and must be considered, especially for rare facts that do not appear in tables or other structures and formats.

Some extraction structures are more reliable than others. It is also necessary to take all possible extraction types as it increases the recall and some facts can only be found looking in a certain structure. The trust in the fact values extracted by a structure must therefore take the employed extraction structure into account. The next section describes how the trust in extracted facts is calculated.

3.2.3 Calculating the Trust in Extractions

Once values for attributes have been extracted, they need to be ranked in order to determine the value that is most likely to be the correct one for the attribute. It is now necessary to find the correct ones by assigning *trust* to each extraction.

Definition 2 (Trust)**.** The **trust** is a non negative number. The higher the number the more reliable the extracted value.

The following equations assign trust values and aim to improve the ranking of the extracted values, i.e. to put the correct ones on top.

The easiest way to rank the extracted values it by just counting the number of extractions. The more often a value has been extracted, the higher the trust value. Equation 2 shows how the trust value is calculated in that case, with N being the number of extractions for the given value, and x being a tuple consisting of *concept*, *entity*, *attribute* and *value*, $x =< x_{concept}, x_{entity}, x_{attribute}, x_{value} >$. This way of assigning a trust value is called "Quantity Trust" from now on.

$$\text{QuantityTrust}(x) = N \tag{2}$$

The Quantity Trust does not make use of additional information like **where** (the source) and **how** (extraction technique/structure) the fact was extracted. This information must be considered when determining the trust for an extraction.

Determining the Source Trust Some pages that are retrieved for the extraction process mention the attribute and its value several times. For example, suppose a page that is retrieved, when searching for the entity `Nokia N95` and the attribute `talk time`, mentions the attribute several times, two times with the correct value of 6.5 hours but three times with different values that do not relate to the entity but to other mobile phones. The source trust can therefore be reduced whenever there is more than one value for the searched attribute as shown in Equation 3, where D is the number of different values found for the given attribute and source. The source trust can have values between 0 and 1 with one being highest trust and zero being no trust.

$$\text{SourceApplicability}(attribute,source) = \frac{1}{D} \tag{3}$$

Determining the Extraction Structure Trust Extraction structures have different precisions that must be taken into consideration when calculating the trust for a fact value. The values determined in the test set are not representative for all possible concepts and domains. Since WebKnox aims to be domain independent, the precisions determined for the test set cannot be taken as references. WebKnox uses self supervised machine learning to automatically estimate the trust for the four extraction structures used. The trust value for the extraction structures is an estimated precision, i.e. it is a number between 0 and 1 with one being highest trust (all extractions were correct) and zero being no trust (all extractions were incorrect).

For all extraction structures e, information about the number of extractions N(e), and the number of correct extractions C(e) is kept. The ExtractionStructureTrust is then calculated as the ratio of correct extractions to total extractions (Equation 4):

$$\text{ExtractionStructureTrust}(e) = \frac{C(e)}{N(e)} \tag{4}$$

Initially all extraction structures are initialized with a trust value of 0.5. The three steps are then as follows:

1. The input for the first step is the extraction result with an assigned trust. In the first step the highest trusted fact is searched throughout all concepts

and attributes. It is then assumed that this fact is really a correct one, since it has a high trust. All extraction structures used to extract that very fact value get credit for a correct extraction, i.e. , $C'(e) = C(e) + 1$ and $N'(e) = N(e) + 1$. Extraction structures that led to wrong fact values for that attribute, get credit for a wrong extraction, i.e. $N'(e) = N(e) + 1$. In the next iteration that highly trusted fact is not considered anymore when looking for the highest trust.

2. In the second step, the trust for the extraction structures is updated based on the number of correct and total extractions that have been revised in the former step, i.e. the extraction structure trust is recalculated using Equation 4.

3. In the third step, the trust for all extracted values is recalculated by using the updated trust for the extraction structures. After this step, the ranking of the extracted values for each attribute might change. The newly ranked list is then again input for the first step to repeat the process. The iteration can be stopped when the trust for the different extraction structures converges. In case the trust does never converge, the iteration will only stop after all highest trusted facts have been evaluated in step one.

Combining Source and Extraction Structure Trust Taking both, the source trust and the trust in the extraction structure, into consideration, the trust for an extracted value can be calculated as shown in Equation 5. S is the set of sources the given fact has been extracted from, ExtractionStructureTrust(e) is the trust of the extraction structure e used and SourceApplicability(s) is the trust for the source s. The trust will therefore be high, when the value has been extracted in many trustworthy sources using numerous highly trusted extraction structures. This trust formula shall be called "Combined Trust".

$$\text{CombinedTrust}(x) = \sum_{s \in S}(\sum_{e \in E} \text{ExtractionStructureTrust}(e) * \text{SourceApplicability}(x_{attribute}, s)) \quad (5)$$

Normalization Facts can be represented in different formats which still represent the same thing. For example dates can be written in many ways, such as `January, 17th 1962` or `17/01/1962`. Also many numeric facts have units. Not taking the unit into account leads to the extraction of two different facts where actually only one is mentioned, e.g. `2 inch` and `5.08 cm` is the same fact. Normalization helps to find facts from different formats and to cluster them.

Validating Numeric Fact Values across Entities Another problem with extracted facts is that some attributes do not have a single absolutely correct value. The `population` attribute for example is not mentioned correctly on any website on the entire web as it changes almost every second. Instead there are values that are *almost* the same and can be considered correct. Fact values for attributes with fuzzy values tend to not corroborate well. For example, the following fact values might have been extracted for the `population` attribute for `Australia`:

```
300 (3 times)
21000000 (1 time)
21340000 (1 time)
22578420 (1 time)
20452340 (1 time)
```

The problem here is that the exact same number for the population is not mentioned on more than one source. The incorrect extraction `300` however is extracted several times and therefore gains higher trust.

To solve that problem, two assumptions are made:

1. The order of magnitude (OOM) for numeric facts is often the same for entities within the same concept; there are exceptions such as the population of countries.

2. There are well-known entities where the information about the numeric attribute can be extracted with relatively high trust because they appear on very many pages.

Both assumptions were supported in our test set for most of the fact values. A bigger test set with more entities (and more well-known ones) would most likely further support that assumption.

To make advantage of the fact that the OOM is often the same, WebKnox uses a validation process across all entities for a given attribute. This process is called "Cross Validation" and is part of the second step in the self supervised learning loop. It works as follows:

1. For all numeric attributes, an OOM distribution is constructed.

2. If the highest trusted value from the first step of the learning loop is a numeric value, the number is considered to be correct and the OOM of that number is given credit in the attribute's OOM distribution.

3. In the next iteration the trust for the fact values for the same attribute will be calculated as shown in Equation 7. The *CrossValidationFactor* for a numeric fact value is one plus the *support* of the OOM, which is a number between 0 and 1 with 1 being 100% support (all other entities of the concept had values with exactly the same OOM for that attribute) and zero being 0% support (no other entity of the same concept had the same OOM for that attribute).

$$\text{CrossValidationFactor}(x) = 1 + \text{support}(\lfloor log_{10}(x_{value}) \rfloor, x_{concept}) \quad (6)$$

$$\text{CrossValidationTrust}(x) = \text{CombinedTrust}(x)) * \text{CrossValidationFactor}(x) \quad (7)$$

4 Evaluation

The Test Set contains six different concepts (five entities each) and five data types. In total there are 255 facts to extract.

The entities for each concept were chosen manually by applying following criteria to gain a more representative sample for each concept: `Notebooks`, `Mobile Phones` and `Cars` were chosen from different manufacturers; small and large `Countries` were chosen; `Movies` were chosen based on popularity; and only well known `Actors` were chosen.

For the evaluation, 2420 HTML pages were retrieved using the REST web service from Yahoo![5]. Each entity was searched for using two multi-attribute queries and each attribute of an entity resulted in three single-attribute queries. For each query, only the top eight retrieved URLs were used for the fact extraction process. Not all queries led to eight answers from the search engine, in that case all answers were taken.

The standard measures for comparing extraction systems are *precision* and *recall*. In web information extraction, the precision measures the ratio of correctly extracted facts or entities to the total extractions, and the recall measure determines the ratio of the performed extractions and the extractions expected. Additionally, the measure *found* is used several times. *Found* is the ratio of extracted facts (correct or not) to expected facts. A found value of one means, that for every attribute at least one value has been extracted. If not otherwise stated, the measures always relate to the complete test set of the WebKnox system.

Baseline Basically two different approaches are used today: (1) wrapper induction and extracting from tables and (2) treating the website as a long (tokenized) string by removing the tags. As the developed approach extracts information not only from tables it is appropriate to compare it to the latter technique.

The baseline extraction works similar to the technique from the GRAZER system [7]. All tags are removed from the website, all occurrences of the attribute are evaluated and the corresponding fact values are expected before or after the attribute. Only the first 150 characters before and after the attribute are searched for the matching value to delimit noise. The trust is calculated only by counting the number of extractions (Quantity Trust).

[5] `http://developer.yahoo.com/search/`

Evaluation of Source Retrieval WebKnox uses a set of generic queries to retrieve websites from a search engine that are likely to have a mention of the facts searched for. When retrieving the top eight results, 95% of the facts were found. All further evaluations for the fact extraction rely on the test set that was gained by taking the top eight results from Yahoo! for all queries.

Evaluation of Extraction Structure Trust Learning Figure 3 shows the learned trust values for the four extraction structures after every iteration of the learning loop. The dashed lines visualize the manually determined precision values in the test set for each extraction structure and the solid lines are the automatically calculated trust values from the learning loop. All trust values were initialized with 0.5. The graphic shows that the *free text* (red) and *table* (green) extraction structure do not change very much after the first forty iterations. The *phrase* and the *colon pattern* extraction structure however, seem to drop an raise quickly even after 40 iterations. This behavior is due to the occurrences of these structures. The "found" value for these two structures was lower than for *free text* and *table*, which means that the extraction structure occurs more rarely and therefore it takes longer to gain a stable trust value. The loop did not stop before all iterations have been performed since the trust values did not converge so far. 172 iterations was the maximum for the test set with 255 facts since not all facts have been found. After 172 iterations, three of the four extraction structures got an automatically assigned trust value that is in a 4% margin to the "correct" precision value for the extraction structure in the test set. The highest discrepancy can be seen with *phrase* that is 5.2% away from the correct trust value. This again can be explained by the low occurrence number, only every fourth fact can be found by the *phrase* extraction structure. The black line in Figure 3 depicts the overall precision of WebKnox. It is shown that the precision does in fact increase as the extraction structures get trust values closer to their real precision. Through the learning loop, an overall precision gain of 7.4% is gained, recall is also affected positively with an increase of about 7%.

Cross Validation When comparing the extraction precision for the numeric data type we find that (both with and without crossvalidation) WebKnox found 143 of the 145 numeric facts in the test set. The learning loop was performed 172 times in both cases, until no more iteration was possible. Without cross validation 63.19% of the extracted numeric values were correct. Using cross validation showed a gain in precision of almost 7% to 70.13%. The difference between the baseline and WebKnox for numeric fact values now increases to over 25% in precision.

Overall Fact Extraction Performance Figure 4 shows the comparison between the baseline and the fact extraction process of WebKnox for all concepts of the test set. The evaluated fact extraction process used

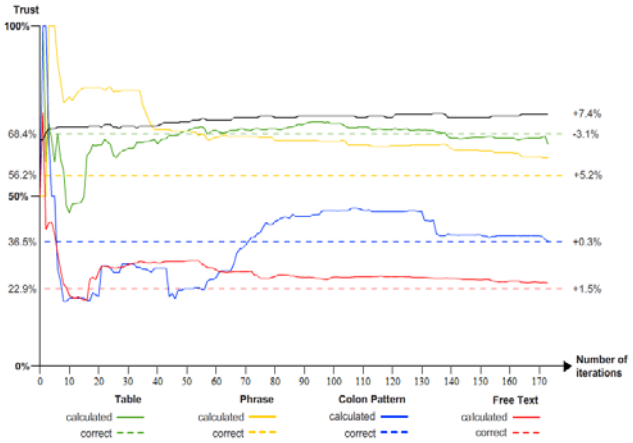


Figure 3: Evaluation of the self supervised learning loop for the extraction structure trust.

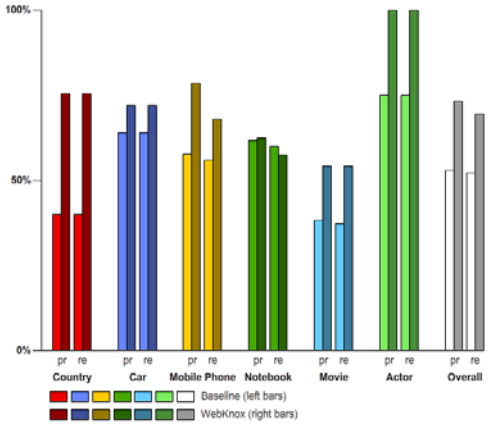


Figure 4: Evaluation of the WebKnox system against the baseline across six concepts.

Equation 7 (with cross validation) and stopped after 172 iterations to weight the extraction structures and apply cross validation for numeric facts. The measures in the figure are *pr* for precision and *re* for recall. There are two bars for each measure and concept, where the left is the one for the baseline and the darker right one is the measured value for WebKnox. In five of the six concepts, WebKnox reaches a higher precision and recall than the baseline. For the car and notebook concept it does not perform considerably better than the baseline. That is because the normalization step sometimes fails to normalize the numbers correctly. In the car and notebook domain most of the facts are numeric facts and several times there is no unit given with the fact. Overall, the system achieves precision and recall over 70% compared with the baseline of just approximately 52%.

5 Conclusion and Further Work

We showed that we can increase the fact extraction performance by searching facts in different formats and structures of HTML documents. Furthermore, we introduced an algorithm that can learn a trust value for those structures in a self supervised manner. We are able to assign a trust value to the extracted facts based on a source trust and the trust for the extraction structures. Further work needs to be done especially:

1. Determining how well the trust value indicates for the end user the reliability of the automatic extraction.
2. Finding further criteria to calculate the source trust more accurately for the extraction process.
3. Investigating in further domain independent formats and structures that are used to represent facts on websites.
4. Automate identification and extraction of entities (extending the work of Vercoustre et al. [5] on entity ranking from Wikipedia).

References

[1] Michele Banko, Micheal J. Cafarella, Stephen Soderland, Matt Broadhead and Oren Etzioni. Open Information Extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2670–2676, 2007.

[2] Chia-Hui Chang, Mohammed Kayed, Mohed R. Girgis and Khaled F. Shaalan. A Survey of Web Information Extraction Systems. *IEEE Transactions on Knowledge and Data Engineering*, Volume 18, Number 10, pages 1411–1428, 2006.

[3] William W. Cohen, Matthew Hurst and Lee S. Jensen. A flexible learning system for wrapping tables and lists in HTML documents. In *Proceedings of the 11th International Conference on World Wide Web*, pages 232–241. ACM, 2002.

[4] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld and Alexander Yates. Web-scale information extraction in knowitall: (preliminary results). In *WWW '04: Proceedings of the 13th International Conference on World Wide Web*, pages 100–110. ACM, 2004.

[5] Anne-Marie Vercoustre, James A. Thom and Jovan Pehcevski. Entity ranking in Wikipedia. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 1101–1106. ACM, 2008.

[6] Alexander Yates. *Information Extraction from the Web: Techniques and Applications*. Ph.D. thesis, University of Washington, Computer Science and Engineering, 2007.

[7] Shubin Zhao and Jonathan Betz. Corroborate and Learn Facts from the Web. In *KDD '07: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge discovery and data mining*, pages 995–1003. ACM, 2007.

Anonymous folksonomies for small enterprise webs: a case study

Tom Rowlands
CSIRO ICT Centre and ANU DCS
ACT 2601 Australia
tom.rowlands@ieee.org

David Hawking
Funnelback
ACT 2601 Australia
david.hawking@acm.org

Ramesh Sankaranarayana
Dept. of Computer Science
Australian National University
ACT 2601 Australia
ramesh@cs.anu.edu.au

Abstract *Tags and emergent folksonomies are a potentially rich new source of document annotations, offering query independent and dependent evidence for exploitation by information retrieval systems. Previous research has shown that tags may facilitate improved web search in an environment where each tagging action generates a (user, tag, resource) triple.*

For websites operated by a public institution, operational or privacy concerns may prevent the recording of data capable of identifying individuals. This leads to a simpler anonymous tagging system but is likely to reduce user motivation for tagging, since the user cannot access their own set of tags. It also means that votes for tags are not counted, and a potentially useful joining attribute is not available.

Using webpage, metadata, query, click, anchortext and tag data provided by a public museum, we demonstrate that, despite these limitations, tag data collected by an anonymous tagging system has the potential to improve retrieval effectiveness.

Keywords Information Storage and Retrieval

1 Introduction

'Tagging' a resource is the action of tying a typically short, and often white-space free, string, the 'tag', to a resource. The resource may be a web page, document, picture, person, or a reference. Tags are used on many social networking websites such as `flickr.com`, `delicious.com`[1] and `citeulike.org` and in some blogs, allowing users to read blog posts of a particular tag. There is no restriction on the text a tag may contain; no controlled vocabulary or, necessarily, a particular meaning attached to any particular tag.

[1] Formerly known as `del.icio.us`

Proceedings of the 13th Australasian Document Computing Symposium, Hobart, Australia, 8 December 2008.

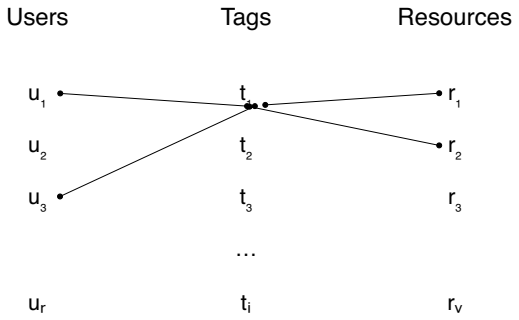


Figure 1: The relationship between users, the tags they use and the resources that are tagged, expressed as a graph. It is possible for the one resource to be tagged multiple times with the same tag, but only by different users.

On services permitting multiple taggers, such as `delicious.com`, users can typically see the tags others have applied, and over time a 'taxonomy of the folk' develops. Importantly, users can re-apply the same tag to objects already exhibiting a tag, thereby reinforcing the tag.

Bao et al. [1] have shown that this type of tagging can be used to improve retrieval effectiveness in web search. It is a more open question as to whether folksonomy tagging can, in practice, deliver retrieval benefits at an enterprise or website level. Please note that, although most folksonomy tagging systems are web-based, tags could potentially be applied to non-web data.

1.1 Anonymous tagging

Anonymous tagging systems are different to those described above in that user information such as user-id, IP address or geo-location is not recorded. All tags are public. Consequently, one of the potential incentives for tagging, organisation and ease of reference for the individual [6], is removed. While an underlying database perhaps permits a resource to be tagged more than once with the same tag, this doesn't tend to happen in practice.

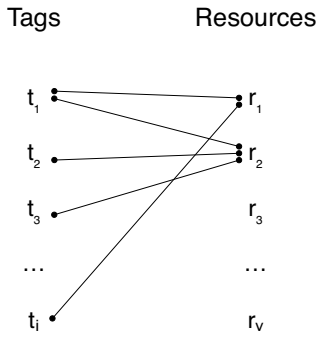


Figure 2: The relationship in an anonymous tagging environment is such that tags are applied to resources, independent of the users. This arrives at a simpler graph than in Figure 1, in which each object can only be tagged with each tag once.

Anonymous tagging offers some advantages to an organisation operating a website. There is no need to track individual users or securely store their associated logins and passwords. There is less chance that a user will feel ownership over their data, thereby reducing the risk the service provider will need to disseminate data should the service be ceased. Notwithstanding the loss of incentive mentioned in the abstract, casual visitors to the site may be more likely to tag resources, since they do not have to log in to do so.

To the best of our knowledge, anonymous tagging systems have not previously been studied from an information retrieval perspective.

1.2 Aims and scope of the present work

We present a case study of a data collection comprising document content, metadata, anchortext, user queries and clicks, as well as folksonomy tags from the anonymous tagging system of a public institutional website.

We characterise the collection and compare the distribution of number of items tagged per unique tag with that observed in a user-based tagging system.

Our principal aim is to test the hypothesis that tags collected in an anonymous tagging environment are capable of boosting retrieval effectiveness within an institutional website.

Accordingly, we investigate the following questions:

- What proportion of resources are tagged?
- How quickly is the untagged proportion of the collection likely to diminish at currently observed tagging rates?
- To what extent do queries match tags?
- Do tags permit retrieval of documents not retrieved on the basis of text created by author and/or publisher, i.e. content and official metadata.

2 Related work

2.1 Annotations

Significant retrieval effectiveness and efficiency gains have been demonstrated by the use of annotation data.

Craswell et al. [3] demonstrate superior site finding performance with an anchortext surrogate index 'an order of magnitude' smaller than content. Eiron et al. [5] study the utility of anchortext for information retrieval based on the idea that queries, anchortext and titles are created by a similar thought process.

Xue et al. [12] use surrogates and compensate for a relatively small quantity of click data by using co-visitation.

Dmitriev et al. [4] invite users to add 'explicit' annotations to an intranet. They argue that such annotations are expensive to produce as users have to be asked to produce them and often do not find the time. They also examine 'implicit' annotations, such as queries associated with documents through clicks.

2.2 Tags and folksonomies

Mathes [10] proposes some attractions of user generated textual metadata, arguing the relative simplicity of tagging systems, their 'low cognitive cost', rapid feedback and development of communities, all as attractions to potential taggers. Golder and Huberman [6] discuss how folksonomies are distinct from taxonomies in their being non-hierarchical and inclusive. They analyse data from `delicious.com` and demonstrate some interesting effects by comparing users. They also report that tags are not always used as a description of the document content and so document and tag vocabularies can be expected to be different.

Bao et al. [1] define *SocialSimRank*, estimating the similarity between queries and web pages based on the tag graph. They also define *SocialPageRank*, estimating the query independent value of a web page based on the tag graph. They use both in a whole of Web search task, combining with other forms of evidence using a support vector machine, and show good results.

The challenge of tag segmentation due to lack of explicit boundaries (e.g. 'informationretrieval') in tags, discussed in [1], is not a problem in our case.

Halpin et al. [7] suggest a generative model for tagging that arrives at a power law (see [2]) based on preferential attachment. The central idea is that users are more likely to tag a resource with a tag that has already been used for that resource. They demonstrate tags on `delicious.com` following a power law pattern.

3 Data

The primary data used in this paper has been gathered from an online museum catalogue. Pictures and descriptions of the museum's artifacts are published, along with various metadata, as HTML pages and are accessible via a standard web interface. We collected the document data using a commercial web crawler. The museum provided tag and click data as a database dump. The site is particularly interesting because it offers non-trivial quantities of content, anchortext, click associated query and tag evidence.

The tagging data covers only those items within the museum's 'online collection', which comprises 132327 of the 135216 documents on the museum's website. We consider only the online collection. The majority of documents describe a single museum exhibit and include Title, Description (usually identical to title), Keywords, and content.

The click data covers the period 14 July 2008 to 14 August 2008. The content of sixty documents for which we have tag data were not yet downloaded by the time our crawl was terminated. The crawl was tempered to reduce load on the museum's servers. It started on 20 August 2008 and lasted just under five days.

There are 7221 distinct tags with 11 509 applications of those tags. Each application included the date on which the tag was applied to the document. There were no cases of the same tag being applied more than once to the same document.

A conventional query log was not available. We have no information about queries which were submitted but which did not lead to any click. Instead, we have a click log which shows 10 747 distinct user-composed queries[2] associated with a total of 364 310 clicks. It is known that the site search facility makes use of the tags.

The average length of the tags, queries and anchortext is 1.5, 1.4 and 8.1 words respectively. The distribution of lengths is shown in Figure 6. Anchortext tended to repeat the title, often truncated, of the target page.

For the study, all tags and queries have been case folded and leading and trailing whitespace has been removed. In our data, tags are associated with document identifiers rather than URIs. Sometimes multiple URIs share the same docid e.g. `http://museum.com/getdoc?docid=1&image=1` and `http://museum.com/getdoc?docid=1&image=2`. In such cases, the tag or click has been considered to 'apply' to both URIs.

4 Experiments

4.1 Experiment 1: Characteristics of anonymous tags

In this section, we investigate the frequency of applications of tags to particular resources. It has been previously established that, in non-anonymous tagging systems, sufficiently popular tagged resources and entire folksonomies from the one system yield power law-like distributions. An example of such a distribution is shown in Figure 4.[3] The anonymous tagging system that is the focus of this paper is shown in Figure 3. This shows graphically that, like queries and non-anonymous tagging systems, anonymous tagging systems yield a few tags occurring a relatively large number of times, with many tags occurring very rarely.

[2]We eliminated a large number of records in which the query was generated by a user clicking on a navigational link within the site.

[3]This example data is taken from the `citeulike.org` facility.

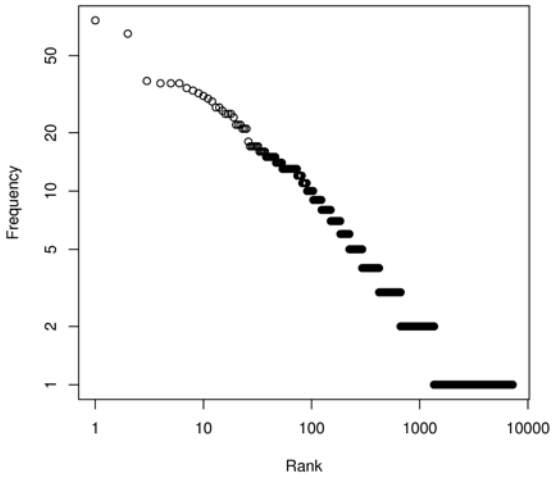


Figure 3: A log-log graph of the number of times each tag has been used across the corpus, ordered by that number. For example, the most frequent tag, ranked 1, has been applied to 76 objects. As the tagging system is anonymous, each tag can only be applied to each object once. The distribution is similar to that in the more traditional case, shown in Figure 4, despite the lack of reinforcement.

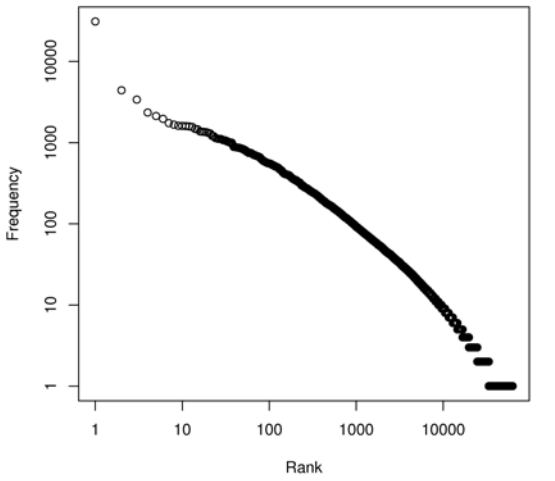


Figure 4: A log-log graph, similar to Figure 3, but from a non-anonymous tagging system where tags can be reinforced by other users. This data is from `citeulike.org`.

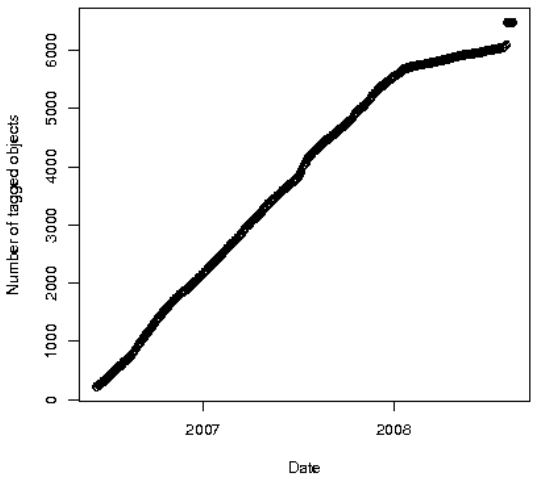


Figure 5: The number of tagged objects over time. The sudden jump in the number of tags at the far right reflects an import of tags made available to users through the Flickr site. Flickr is very popular. Note that this is the number of tagged objects, not the number of tags; if an object is tagged multiple times it is only counted once.

4.2 Experiment 2: How prevalent are tags?

We counted the number of documents (museum objects) to which tags had been applied and observed how that increased with time.

4.2.1 Results

The number of tagged objects plotted against time is shown in Figure 5. The number of tagged objects is far fewer than the number of separate articles in the museum, and reflects only around five per cent of the collection. The overall average rate of tagging over two years is three and a half thousand objects tagged per year.

It is not clear why there is a 'knee' in the plot around the beginning of 2008. The museum suggests this may be due to a declining number of people interested in tagging. Note the sudden jump in the second half of 2008 due to the import of tags from Flickr.

4.2.2 Discussion

If, optimistically, the average rate of tagging were to be maintained, it would be approximately another thirty five years before all the items in the *present* collection received at least one tag.

A substantial increase in the number of tagged objects is provided by making the objects available on the external Flickr site.[4] All of the tags added by users of Flickr were to objects that were previously untagged, and in some cases there were multiple tags added to the same object.

[4]See `http://www.flickr.com/commons/`.

4.3 Experiment 3: Do tags match queries?

To assist in retrieval, tags must match queries actually received by the information retrieval system. We consider three different matching models:

- Exact match: the query and tag strings are identical.
- AND match: all the words in the query are present in the tag.
- OR match: at least one query word is present in the tag.

Note that each of these models is applied to tags individually, not to the collection of tags applied to an object. Obviously, the second and third models are the same when the query consists of only one word.

For contrast, a similar calculation is conducted for anchortext. 'Canned' or automatically produced links containing query text or tags have been filtered out to avoid inflating the results in this case. An example of such links is the automatically generated list of 'recently applied tags'.

4.3.1 Results

Table 1 shows the percentage of query instances that might be answerable by tags and anchortext using three different matching schemes.

Overall, 88% of the query instances share at least one term with a tag and, as a consequence, *might* be at least partially answerable by that tag. This percentage drops to 69% for AND match and 61% for Exact match.

Table 2 looks at queries and annotations the other way around—what percentage of annotations are useful in answering at least one query? The percentages are reasonably high for tags. Overall, 81% of tags are potentially useful (OR match) in answering at least one query and 57% of tags exactly match a query.

A high proportion (88%) of anchortext annotations achieve an AND match with at least one query, but there are no exact matches.

4.3.2 Discussion

No anchortext annotations exactly match any queries, even though there is a very high degree of AND match. The lengths of strings used as anchortext (usually the title or an abridged title of the target document) are quite different to those of tags and queries; this can be seen in Figure 6. An exact match function would not be appropriate for use with anchortext on this site.

4.4 Experiment 4: Do tags contribute useful additional terms?

Here we are interested in queries answerable by tags but not by document content, metadata or anchortext.

Table 1: Percentage of the query workload, of various lengths, matching at least one annotation. For example, twenty seven per cent of query instances of length two AND match a tag.

Match type	Exact					AND					OR				
Query length	1	2	3	≥ 4	all	1	2	3	≥ 4	all	1	2	3	≥ 4	all
Tags	75	21	24	2	61	85	27	27	3	69	85	96	99	100	88
Anchortext	0	0	0	0	0	97	93	90	87	96	97	100	100	100	98

Table 2: Percentage of annotations, of various lengths, matching at least one query instance. For example, twenty three per cent of tags of length three match a query exactly.

Match type	Exact					AND					OR				
Annotation length	1	2	3	≥ 4	all	1	2	3	≥ 4	all	1	2	3	≥ 4	all
Tags	68	39	23	11	57	68	81	86	93	73	75	94	98	99	81
Anchortext	0	0	0	0	0	68	66	98	92	88	72	83	100	99	96

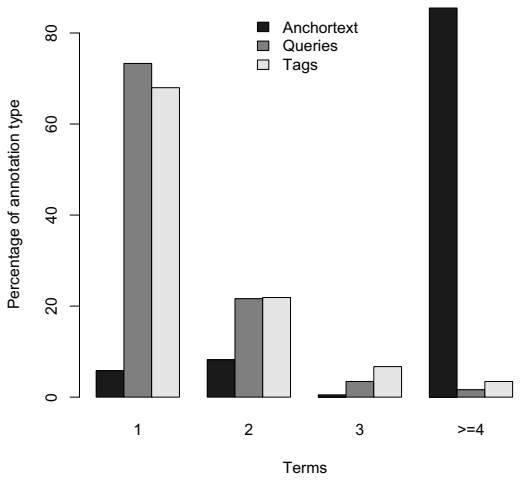


Figure 6: The percentages of different lengths of annotation data. There is a large proportion of anchortext that is four or more terms long, while tags and query instances both tend to be short.

4.4.1 Results

For 48% of distinct queries, a tag matching the query (at least one intersecting term) reveals at least one new document with which the query did not share a term. This represents 54% of the workload. Document content included all text and metadata.

The percentages of query instances matching tags but not the associated document is shown in Table 3.

5 Discussion

We have seen that in this instance, even after two years offering a tagging interface a relatively small number of objects have been tagged. This is a disappointment to the museum which they partially addressed by posting items on Flickr and collecting tags.

Many of the reasons for tagging outlined by [9] and [6] do not apply in the anonymous tagging environment. This may partly explain why objects displayed in Flickr are tagged at a much higher rate than on the museum site.

We note that it may be possible to derive further tags implicitly from website referrer logs. Another possibility might to be to provide tagging incentive by instituting a tagging game (see e.g. [11]).

Not every object must be tagged for the tags to contribute useful evidence. For example, an anonymous tagging system on an intranet may assist staff in finding key pages more quickly even with only a small subset of important pages tagged.

Fifteen per cent fewer queries were potentially answerable by tags than by anchortext, but 'potentially answerable' is an extremely optimistic metric. Examining from the 'other direction', however, it is most often very long anchortext that matches queries.

The rough power law distribution shown by [7] is seen in the folksonomy distribution shown in Figure 3. Halpin et al. suggested in their generative model of tagging systems that the likelihood of a tag being applied to an object was influenced by the tags already applied to that object. In the case of anonymous tagging systems, the distribution of tag application is 1—a straight line—and yet the distribution across the corpus is still a, roughly, power law distribution with a steep decline for the most frequent tags.

Any system permitting users on the Web at large to add or remove information at will opens itself to the issue of spam [8]. The tags on the site investigated here seemed to be relatively free of spam. By restricting the resources to those available on the site it is made less attractive to spammers attempting to manipulate rankings in commercial search engines. Private systems allowing links beyond the site itself are also possibly immune for the same reason; their impact on commercial search will be small or nil.

Table 3: Percentage of query workload where a query matches a tag but does not match the content of a tagged document (including Title, Keyword and Description metadata) or anchortext pointing to the tagged document. For example, 49% of queries containing two terms matched a tag that was applied to at least one document containing neither of the query's terms.

Match type	Exact					AND					OR				
Query length	1	2	3	≥ 4	all	1	2	3	≥ 4	all	1	2	3	≥ 4	all
Content	28	1	0	0	21	57	1	0	0	42	57	49	37	20	54
Anchortext	36	1	0	0	27	77	2	0	0	58	79	94	97	99	83

6 Conclusions

Anonymous tagging systems, like that deployed at the museum which is the object of the present study, do not provide the same incentive to tag as do user-centric tagging systems on the Web. When objects from the museum are displayed in Flickr, they are tagged at a much higher rate than on the museum's own site. Anonymous tags provide only a binary signal as to the importance of a resource with respect to a tag. There is no voting aspect; either a tag is applied or it is not.

Despite these differences, anonymous tag data from the museum shows a similar distribution of tags to that described by [7].

Although the sparsity of tag data and its slow rate of accumulation mean that a retrieval system for the museum could not be based on tags alone, we found that a relatively high proportion (54%, assuming OR-match) of the query load for which answers could be identified using the tags that were not identified by text or metadata generated by the author or publisher. This suggests that future research on combining anonymous tags with other evidence in a retrieval system would be worthwhile.

Acknowledgements The authors would like to thank the museum who kindly granted access to the data, without which the experiments could not have been run.

References

[1] Shenghua Bao, Gui-Rong Xue, Xiaoyuan Wu, Yong Yu, Ben Fei and Zhong Su. Optimizing web search using social annotations. In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider and Prashant J. Shenoy (editors), *WWW*, pages 501–510. ACM, 2007.

[2] A. Clauset, C.R. Shalizi and MEJ Newman. Power-law distributions in empirical data. *Arxiv preprint arXiv:0706.1062*, 2007.

[3] Nick Craswell, David Hawking and Stephen Robertson. Effective site finding using link anchor information. In *Proceedings of ACM SIGIR 2001*, pages 250–257, 2001.

[4] Pavel A. Dmitriev, Nadav Eiron, Marcus Fontoura and Eugene Shekita. Using annotations in enterprise search. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 811–817, New York, NY, USA, 2006. ACM Press.

[5] Nadav Eiron and Kevin S. McCurley. Analysis of anchor text for web search. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 459–460, New York, NY, USA, 2003. ACM.

[6] Scott A. Golder and Bernardo A. Huberman. Usage patterns of collaborative tagging systems. *J. Inf. Sci.*, Volume 32, Number 2, pages 198–208, 2006.

[7] Harry Halpin, Valentin Robu and Hana Shepherd. The complex dynamics of collaborative tagging. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 211–220, New York, NY, USA, 2007. ACM.

[8] Monika R. Henzinger, Rajeev Motwani and Craig Silverstein. Challenges in web search engines. *SIGIR Forum*, Volume 36, Number 2, pages 11–22, 2002.

[9] Cameron Marlow, Mor Naaman, Danah Boyd and Marc Davis. Ht06, tagging paper, taxonomy, flickr, academic article, to read. In *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*, pages 31–40, New York, NY, USA, 2006. ACM.

[10] Adam Mathes. Folksonomies-Cooperative Classification and Communication Through Shared Metadata, December 2004.

[11] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326. ACM Press New York, NY, USA, 2004.

[12] Gui-Rong Xue, Hua-Jun Zeng, Zheng Chen, Yong Yu, Wei-Ying Ma, WenSi Xi and WeiGuo Fan. Optimizing web search using web click-through data. In *Proc. ACM CIKM '04*, pages 118–126, 2004.

The Effect of Using Pitch and Duration for Symbolic Music Retrieval

Iman S. H. Suyoto and Alexandra L. Uitdenbogerd
School of Computer Science and Information Technology
RMIT University
Vic. 3001 Australia
{*Iman.Suyoto,Alexandra.Uitdenbogerd*}*@rmit.edu.au*

Abstract *Quite reasonable retrieval effectiveness is achieved for retrieving polyphonic (multiple notes at once) music that is symbolically encoded via melody queries, using relatively simple pattern matching techniques based on pitch sequences. Earlier work showed that adding duration information was not particularly helpful for improving retrieval effectiveness. In this paper we demonstrate that defining the duration information as the time interval between consecutive notes does lead to more effective retrieval when combined with pitch-based pattern matching in our collection of over 14 000 MIDI files.*

Keywords Music information retrieval, Information retrieval, Multimedia resource discovery, Pattern matching

1 Introduction

The field of music information retrieval has as its aim the development of technology to enable users to find music that they are searching for. There are many ways that users may wish to search for music, such as locating information about a song for which a small fragment is remembered, finding music that is of a similar style to an example, or simply searching for music that the user might like. The reason for searching could be simply to satisfy the user's curiosity, check for copyright infringement, or to purchase new music.

One of the main problems studied in the field of music information retrieval is that of retrieving music given a query that is a melody fragment, such as a few notes of the sung component of a verse of a song. The problem's complexity varies depending on the format of the query and the music collection, with the simplest being search of a symbolically encoded collection of melodies using a symbolically encoded melody. In this paper, we use symbolic melody queries and a polyphonic (multiple notes at once) collection of music. Most of our early work [29, 34, 35] was restricted to search using a representation of both queries and music from the collection as sequences of pitches. Rhythm information was ignored. This approach was shown to be competitive with more complex techniques in recent evaluation exchanges [24, 31, 32] where the collection was symbolically encoded. However, improvement may be possible with the introduction of rhythm information, potentially allowing matching techniques to yield greater effectiveness for sung queries that are likely to be less precise than those issued via a musical keyboard or text-based encoding. In our experiments we explore two different methods of encoding rhythm: encoding the duration of each note in a melody and *inter-onset intervals* (IOI) — the time interval between successive notes. We found that improvement in retrieval effectiveness is possible using an IOI representation of rhythm.

Proceedings of the 13th Australasian Document Computing Symposium, Hobart, Australia, 8 December 2008.

2 Related Work

Much of previous research has shown that the pitch feature is sufficient to support effective content-based retrieval of music. The usage of both pitch and rhythm has also been examined in past work by, for example, McNab et al. [15], Chen and Chen [1], Lemström et al. [12], Dannenberg et al. [2], Ferraro and Hanna [5], Hanna et al. [8], Typke et al. [27], and Lemström et al. [13]. Other than in our previous work [23], the relative value of these features for matching on large polyphonic collections has not been measured. In addition, the benefit of string-matching approaches in this scenario have not been thoroughly investigated yet. We discuss each of these papers below.

McNab et al. [15] investigated what combination of pitch and duration features has the best discriminatory power to distinguish one musical piece from others. Their collection consisted of 9 600 folksong melodies. They examined both exact matching and approximate matching (using dynamic programming as given in Mongeau and Sankoff [18]). To represent the pitch component of notes, they used pitch interval, which is the difference in pitch between two adjacent notes, and pitch contour, which is the movement direction from a previous note to a current note, described further in Uitdenbogerd and Yap [33]. They found that for highly effective exact matching with rhythm, five notes are sufficient. Without rhythm, about seven notes are

required. For approximate matching (with rhythm), the number of required notes increases to twelve.

A technique for retrieval by rhythm was proposed in Chen and Chen [1]. Every piece was represented by a rhythm string, representing solely the rhythmic patterns in that piece. In particular, a piece was divided into measures, and the note durations in every measure in a piece were captured as a unit. Pitches were ignored. Every measure was stored as a node in a tree-based index structure. Their paper emphasises the efficiency of their approach, but fails to present how effective it is. Their test collection only consisted of 102 folk songs (the format of which is unspecified). The relatively small size of the collection and the lack of effectiveness benchmark make the merit of this approach questionable.

Lemström et al. [12] introduced a technique that represents a note as a combination of its pitch interval (with respect to the note preceeding itself) and duration of a monophonic music sequence, called relative interval slope. A sequence consists of n notes, each of which is a pair of its pitch and its duration. The interval slope sequence consists of n symbols, each is the signed difference between the pitch of the current note and that of the previous note, over the duration of the previous note. The first symbol is a special case; it is the pitch of the first note over the the duration of the last note. If every symbol in the interval slope sequence is denoted by $a_i; 1 \leq i \leq n$, the relative interval sequence consists of n symbols, each is a_i for $1 \leq i \leq 2$ or $\frac{a_i}{a_{i-1}}$ for $i > 2$. They conducted their experiment on a collection of 6 070 monophonic MIDI tracks. Only exact matches were considered. It is not clear how many queries were used. It is mentioned that the experiment run consisted of 18 000 searches, but the number of unique queries is not mentioned. For queries with pattern length of 13 up to 20, no false positive was generated.

In Dannenberg et al. [2], rhythmic information was used for query-by-humming retrieval, with an answer collection of MIDI files. Three melody encoding approaches were evaluated. In the first approach, a note is represented using its pitch interval and inter-onset interval ratio. An inter-onset interval ratio is encoded as a quantised value of five possible values as devised in Pardo and Birmingham [19], which a pitch interval is encoded as a quantised value of 25 possible values. These make this encoding tempo-invariant and transposition-invariant. Edit distance was used as the similarity measure. In the second approach, based on Mazzoni and Dannenberg [14], a piece was divided into frames of equal time length, from each of which the fundamental frequency is estimated. In this case, note boundaries were ignored. The obtained melody was then transposed 24 times, half a semitone each time. Dynamic time warping was used for matching. In the third approach, based on Meek and Birmingham [16], a note was represented using its pitch class and inter-onset interval, quantised based on a log scale. Matching was performed using a hidden Markov model. Two experiments were conducted. The first experiment involved 160 queries (80 for training and 80 for testing) and a collection of 10 000 synthetically generated pieces with a mean length of 40 notes as noise and 10 folk songs as targets. How the 10 000 pieces were generated is not described. As the result of this experiment, the third approach caused 73.75% of the test queries to obtain the target answer in the first rank position, but the results for the other two approaches were not reported. The second experiment used two query sets. The first query set consists of 131 queries, whereas the second one consists of 165 queries. The first query set was run against a collection of 258 Beatles pieces, and the second query set was run against a collection of 868 popular songs. The third approach was superior for the first query set, yielding a mean reciprocal rank value of 27.0% (compared to 21.0% for the second approach and 13.4% for the first approach). For the second query set, the second approach was superior, yielding a mean reciprocal rank value of 32.9% (compared to 31.0% for the third approach and 28.2% for the first approach).

Ferraro and Hanna [5] and Hanna et al. [8] explored the use of duration information for monophonic music matching. They examined using duration differences between two notes. It is not clearly specified which two notes are meant. Combination of similarity evidence is used to combine the pitch similarity score (s_{pitch}) with the duration similarity score (s_{duration}) using the formula given in Mongeau and Sankoff [18]: $s_{\text{total}} = s_{\text{pitch}} + k s_{\text{duration}}$ where k is a weighting parameter. They claim that at $k = 0.20$, using duration information improves retrieval effectiveness over the use of pitch only.[1] The statistical significance of their result is not reported. Ferraro and Hanna [5] and Hanna et al. [8] claim to obtain significantly different results from using duration and disagree with our conclusion [23] that says otherwise. However, they were using monophonic music, whereas our experiments used polyphonic music. On the improvement significance aspect, we admitted that there was a slight improvement when duration information was used, albeit not statistically significant. On the other hand, they have shown no proof of statistical significance of their claim. Moreover, they did not contrast the input sizes used in both papers. Their work used the testbed of MIREX 2005, which had a collection of 558 MIDI pieces with only 11 queries. This is clearly much smaller than ours (more than 10,000 pieces in the collection and 24 queries) and an indication that the complexity of the problem they were discussing was much smaller.

All work mentioned above involved the use of duration on monophonic collections. There has been research that attempts to use duration-based information on polyphonic music, such as Typke et al. [27] and

[1] The k value is reported in Hanna et al. [8] but not in Ferraro and Hanna [5].

Lemström et al. [13]. Typke et al. [27] described several retrieval tasks in MIREX 2006.[2] Two of them involved polyphonic music:

1. Symbolic melodic similarity using 1 000 polyphonic karaoke files with five queries (referred as the karaoke task herethereafter).

2. Symbolic melodic similarity using 10 000 MIDI files downloaded from the Web, most of which are polyphonic, with six queries (referred as the mixed polyphonic task herethereafter).

In their approach, a melody extraction routine was applied to obtain monophonic representations of the polyphonic pieces. A skyline algorithm[3] was used. Which specific skyline algorithm was not specified. These monophonic representations are divided into overlapping segments with different lengths. They used lengths of 5 to 16, except for the second task, where they used 5 to 7. The segments were then indexed using vantage indexing [36] using the Proportional Transportation Distance [28] as the distance measure. A note was represented as a two-dimensional point [28], with pitch and onset time as the dimensions. The duration of the note was used as the weight of the point. For the two tasks, their method achieved a MAP value of 0.875 and 0.903 respectively.

In Lemström et al. [13], a geometric sweepline algorithm called P3 was used. Every piece was represented by its piano roll [20] representation. The features used were pitch and the start and end times (which can be used to derive durations) of notes. To determine the similarity between a query and an answer, the maximum overlap was determined over keys to ensure transposition invariance. Although this caters for difference in keys, it will likely fail if the tempi of the query and the answer are different. To address this, they proposed SCALEDP3, which extends P3 by scaling the query tempo by a scaling factor. However, it performed poorly on the MIREX 2006 symbolic polyphonic retrieval tasks.

3 Feature Extraction

Our approach assumes that we are working with polyphonic symbolic music. The string representations mentioned in this paper imply that a sequence is one-dimensional, since we cannot have any overlap in a string. However, in polyphonic music, notes can overlap, and as such, it is two-dimensional. Previously, Uitdenbogerd and Zobel [29] showed that reducing the two-dimensional space into one dimension by extracting a representative note for a particular time point can support effective retrieval. The output from feeding polyphonic music into this process is therefore a monophonic melody, representing the polyphonic music. The ALL-MONO algorithm has been shown to be a highly effective melody extraction algorithm. If there is a note m of length l_m sounding at time t_m and another note n sounding at t_n so that $l_m + t_m > t_n$, then l_m will become $l'_m \leftarrow t_n - t_m$. In other words, note overlaps are removed. The ALL-MONO algorithm is outlined in Algorithm 1.

Algorithm 1 ALL-MONO melody extraction algorithm. A note is expressed as a tuple $n = \langle p, d, o \rangle$ where p is the pitch, d is the duration, and o is the onset time. The base index is 0. P is the sequence of the representative bass part. "π_x" is the relational operator for projecting the x attribute.

Require: array of notes **N**
  Sort **N** by ascending onset time as the first sort key and descending pitch as the second sort key.
  {Start taking the highest note at any onset time.}
  for $i = 0 \ldots |\mathbf{N}| - 2$ **do**
    if $(\pi_o n_i \neq \pi_o n_{i+1})$ **then**
      Append $\pi_p n_i$ to P.
    end if
    if $(\pi_o n_i + \pi_d n_i > \pi_o n_{i+1})$ **then**
      $d' \leftarrow \pi_o n_{i+1} - \pi_o n_i$
      $n_i \leftarrow \langle \pi_p n_i, d', \pi_o n_i \rangle$
    end if
  end for
  Append $\pi_p n_{|\mathbf{N}|-1}$ to P.
  {End.}
  return P

4 Matching Technique

To support approximate matching, we convert the melody into standardisations. The pitch standardisation used for the experiments described in this paper is the directed modulo-12 approach [23, 26, 30], described in Section 4.1. As our experiments also make use of the duration feature in notes, we also need to encode the durations into a searchable representation. For this purpose, we use the extended contour standardisation, to be described in Section 4.2.

4.1 Pitch Directed Modulo-12 Standardisation

In the directed modulo-12 standardisation, a note is represented as a value r which is the interval between a note and its previous note scaled to a maximum of one octave [21, 30]:

$$r \equiv d(1 + ((I - 1) \bmod 12)) \qquad (1)$$

where I is the interval between a note and its previous note (absolute value) and d is 1 if the previous note is lower than the current note, -1 if higher, and 0 if otherwise. For example, the melody shown in Fig. 1 is encoded as "`7 4 1 -5 -5 2 3 -2 -1 -2`".[4]

[2] See `http://www.music-ir.org/mirex2006`.

[3] A skyline algorithm takes from a set of overlapping items the one with the extreme value of a certain feature of set of features. The ALL-MONO algorithm (Algorithm 1) is an example skyline algorithm.

[4] A figure is treated as a symbol. Hence, it is a 10-symbol string.

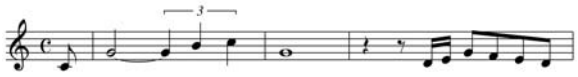


Figure 1: "Melbourne Still Shines" by ade ishs.

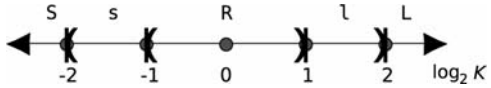


Figure 2: Duration extended contour quantisation. $K = \lambda_C/\lambda_P$ where λ_C and λ_P are respectively the current and previous note durations. The current note is represented as "R" if $|\log_2 K| < 1$; "l" if $1 \leq \log_2 K < 2$; "L" if $\log_2 K \geq 2$; "s" if $-2 < \log_2 K \leq -1$; and "S" if $\log_2 K \leq 2$.

4.2 Duration Extended Contour Standardisation

The extended contour standardisation is partly inspired by a pitch standardisation called the pitch extended contour standardisation [30], which encodes a note as a movement direction of the previous note pitch to its pitch. There are five distinct symbols, each representing a set of pitch intervals: "S" if the current note is the same pitch as the previous note, "u" if the current note pitch is a little higher than the previous note pitch, "U" if the current note pitch is much higher than the previous note pitch, "d" if the current note pitch is a little lower than the previous note pitch, and "D" if the current note pitch is much lower than the previous note pitch.

Just as in pitch contour-based standardisations, the extended contour standardisations also employ five distinct symbols to represent a note. In the case of duration, we use "S", "s", "R", "l", and "L" for "much shorter", "a little shorter", "same", "a little longer", and "much longer" respectively. Interestingly, Moles [17] describes an approach for encoding duration quantisation. The quantisation we use in our experiments is based on the encoding given in that literature. Let λ_C be the current note, λ_P be the previous one, and $K = \lambda_C/\lambda_P$. A note is represented based on the ranges of $\log_2 K$ as illustrated in Figure 2. For example, the melody shown in Figure 1 is represented as "L S R L S R l R R R".

4.3 Alignment

Kageyama et al. [11] suggested the use of note durations as penalty scores for insertion and deletion operations in calculating weighted edit distances. How the scores are calculated is not formally defined however. In this work, we also use a dynamic programming technique, that is, the local alignment algorithm [7], It is useful to find the substring with the highest similarity within a string. Query tunes are usually represented by short strings while answer tunes are usually represented by long strings, so the alignment is more suitable than global alignment [29].

For a query-answer pair, two scores are produced: one pitch similarity score, and one duration similarity score. These scores are to be fused using a similarity evidence combination technique described in the following section.

4.4 Combining Pitch and Duration Similarity Scores

We experiment with a vector model to combine similarity evidence from both pitch and duration matching. The pitches and durations are represented using the respective standardisations. For the purpose of fusing the pitch and duration similarity scores, they are modelled as vectors perpendicular to each other, making the resultant similarity vector become the overall similarity. The following formula is based on one in our previous work [22], where we represent pitch and duration as perpendicular unit vectors. To allow better fine-tuning, we now also assign weights for both pitch and duration components:

$$\vec{\Sigma} \equiv w_\pi \varsigma_\pi \hat{\pi} + w_\delta \varsigma_\delta \hat{\delta} \tag{2}$$

where $\vec{\Sigma}$ is the resultant similarity vector, ς_π is the pitch similarity, ς_δ is the duration similarity, w_π and w_δ are both weight constants, and $\hat{\pi}$ and $\hat{\delta}$ are respectively pitch and duration unit vectors. Ranking is then based on the magnitude of the resultant similarity vector, $|\vec{\Sigma}| = \sqrt{w_\pi^2 \varsigma_\pi^2 + w_\delta^2 \varsigma_\delta^2}$.

5 Experimental Setup

As the aim of our experiment is to identify whether note duration information is useful for melody retrieval, we use a collection of polyphonic MIDI files and a set of queries manually constructed by human subjects. The collection contains 14 193 MIDI files, which form a superset of the collection used in experiments by Uitdenbogerd and Zobel [29, 34] and Uitdenbogerd et al. [35]. A total of 24 queries were constructed by a musician after listening to a set of polyphonic pieces. The relevance judgement set was generated by human users. They were presented with top answers from several matching techniques and asked to give a binary relevance judgement. More detail can be found in Uitdenbogerd et al. [35].

As the baseline of our experiment, for pitch matching, we used $M(x,x) = 1$ for a match, $M(x,y)|_{x \neq y} = -1$ for a mismatch, and $I = -2$ for an insertion/deletion (see Section 4.3) as used elsewhere [23, 29, 34]. For duration matching, we used 21 scoring matrices as in Suyoto and Uitdenbogerd [23]. The scoring matrices were obtained by varying the variables $a, b, c, \ldots, i$ shown in Figure 3, as detailed in Table 1. The matrix means if there is a match "S"-"S", $M(\text{"S"},\text{"S"}) = c$; a mismatch "S"-"s", $M(\text{"S"},\text{"s"}) = d$; etc. At any time, $a \geq b \geq c \geq d \geq e \geq f \geq g \geq h \geq i$. The values of these variables correspond to the rewards/penalties based on the likelihood that there is an actual match when the symbols do not actually match. In other

	S	s	R	l	L
S	c	d	f	h	i
s	d	b	e	g	h
R	f	e	a	e	f
l	i	g	e	b	d
L	h	i	f	d	c

Figure 3: Scoring matrix for duration extended contour standardisation. "S", "s", "R", "l", and "L" respectively indicate a "much shorter", an "a little shorter", a "same", an "a little longer", and a "much longer".

SS	a	b	c	d	e	f	g	h	i
1	1	1	1	1	−1	−1	−1	−1	−1
2	2	1	1	1	−1	−1	−1	−1	−1
3	3	1	1	1	−1	−1	−1	−1	−1
4	3	2	1	1	−1	−1	−1	−1	−1
5	3	3	1	1	−1	−1	−1	−1	−1
6	3	3	2	1	−1	−1	−1	−1	−1
7	3	3	3	1	−1	−1	−1	−1	−1
8	3	3	3	3	−1	−1	−1	−1	−1
9	3	3	3	2	−1	−1	−1	−1	−1
10	3	3	3	1	−1	−1	−1	−1	−1
11	3	3	3	0	−1	−1	−1	−1	−1
12	3	3	3	−1	−1	−1	−1	−1	−1
13	3	2	1	0	−2	−3	−3	−3	−3
14	3	2	1	0	−3	−3	−3	−3	−3
15	3	2	1	0	−1	−2	−3	−3	−3
17	3	2	1	0	−1	−1	−3	−3	−3
17	3	2	1	0	−1	−1	−2	−3	−3
18	3	2	1	0	−1	−1	−1	−3	−3
19	3	2	1	0	−1	−1	−1	−2	−3
20	3	2	1	0	−1	−1	−1	−1	−3
21	3	2	1	0	−1	−1	−1	−1	−2
22	3	2	1	0	−1	−1	−1	−1	−1

Table 1: Scoring schemes (SS) for duration extended contour standardisation. For all scoring schemes, $a \geq b \geq c \geq d \geq e \geq f \geq g \geq h \geq i$.

words, if a symbol is replaced by a substitute, the matrix values represent how much it will change the rhythmic pattern of the melody. If an "R" matches an "R" (thus the score a is rewarded), it is very likely that the two notes represented by the symbols have the same relative duration or inter-onset interval. By an extreme contrast, the likelihood that two notes, each represented by "S" and "L" (thus the score i is given), have the same relative duration or inter-onset interval is small.

6 Results

In our experiment, queries were matched against all tunes in our collection 23 times, once for pitch matching using the directed modulo-12 standardisation and 22 times for duration matching using the 22 scoring schemes.

To combine pitch and duration similarities using Equation 2, we used ten different w_π/w_δ values: ∞ and $0,1,2,\ldots,9$. The first one is the baseline performance, that is, duration information is ignored ($w_\delta = 0$). The

Baseline MAP value = 0.326.

w_π/w_δ	Scoring Scheme	
	1	2
0	0.016	0.019
1	0.143	0.060
2	0.285	0.240
3	0.339	0.276
4	0.346	0.289
5	0.353	0.332
6	0.353	0.338
7	0.353	0.340
8	0.353	0.341
9	0.353	0.346

Table 2: MAP values for various w_π/w_δ using durations. The best values for each w_π/w_δ are highlighted.

w_π/w_δ	MAP
10	0.353 106 549 666 292
11	0.353 106 844 200 076
12	0.353 107 176 261 353
	⋮
18	0.353 107 351 475 871
19	0.353 107 351 475 871
20	0.353 107 351 475 871

Table 3: MAP values for $10 \leq w_\pi/w_\delta \leq 20$.

baseline performance has a MAP value of 0.326. The results of using other w_π/w_δ values are shown in Table 2. Due to space limitation, we only show the results for the scoring schemes that achieve the highest MAP for at least a value of w_π/w_δ. It can be seen that scoring scheme 1 performs consistently better than the other scoring schemes for various values of w_π/w_δ.

The MAP values for 1 and $w_\pi/w_\delta \geq 5$ appear to be approaching an extreme. Therefore, we performed further experiments with $10 \leq w_\pi/w_\delta \leq 20$ and obtained the results shown in Table 3. To assist us determining up to which w_π/w_δ the MAP value keeps increasing, we use 15 figures behind decimal point. We can see that the MAP values with w_π/w_δ starting from 17 are unchanging. See Figure 4 for the plot of MAP values with scoring scheme 1.

The best obtained MAP value is thus far 0.353. This is slightly higher than the baseline value of 0.326. We analyse further whether the two means are significantly different using a paired t-test as has been done elsewhere [25, 26]. It is found that incorporating duration information using the vector model does *not* lead to significant performance gain ($p > 0.2$).

The best scoring method, scoring scheme 1, implies that the "l" is treated the same as "L", and "s" is treated the same as "S". This is evident as $b = c = d$ and $e = f = g = h = i$. Therefore, if we were to remove the distinction between "much longer" and "a little longer," and also "much shorter" and "a little shorter," we would obtain representations with three distinct symbols (alphabets). Thus, the entropy [37], or the minimum number of bits required to store a symbol, defined as:

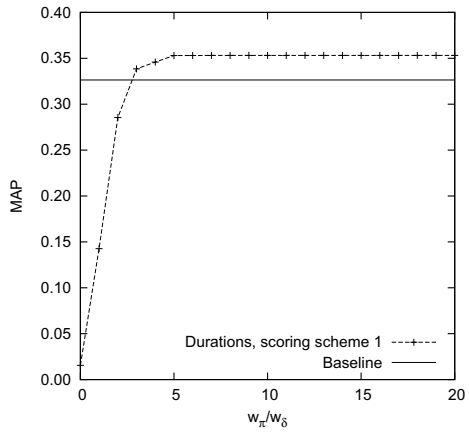


Figure 4: MAP values for pitch and duration matching using scoring scheme 1.

$$H = -\sum_{i=1}^{n} P(i)\log_2 P(i) \tag{3}$$

where $P(i)$ is the probability that the symbol i occurs, would be lower. We performed an informetric analysis as in Downie [3, 4], except that the sequences in our collection were not segmented into n-grams as our experiment assumed a unigram model. With the five-alphabet rhythm standardisation, the entropy of our whole collection is 1.858. With the three-alphabet rhtyhm standardisation, the entropy decreases to 1.491. A decrease in entropy also implies a decrease in information. However, our result shows that with less entropy, the effectiveness of retrieval increases. While Downie [3] believed that a higher information content of n-grams should cause retrieval performance to be better, our informetric analysis of our collection with a unigram model suggests that entropy itself may not be sufficient as an informetric analysis measure of likelihood that target pieces will be ranked higher (that is, high effectiveness). However, we are not certain whether Downie was referring to effectiveness or efficiency. The context hints that it was efficiency. What other measures should be used for effectiveness remains an open question.

We have shown that with the method we propose, duration information does not significantly improve retrieval performance. However, as we shall see shortly, using inter-onset intervals yields a different outcome.

7 Using Inter-Onset Intervals

One advantage of using inter-onset intervals compared to durations is that inter-onset intervals are less susceptible to variations in articulations and are more sensitive to rhythmic variations. As an illustration, let us suppose that we have three melodic fragments as shown in Figure 5.

Our point of interest is the second and third notes. Using durations, the extended duration contour standardisation is "SL" for the three cases. In other words, rhythmic pattern differences are not captured.

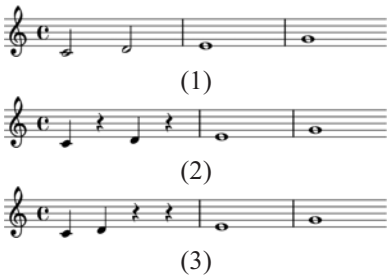


Figure 5: Melodic fragments with different note durations.

Algorithm 2 ALL-MONO-IOI melody extraction algorithm. A note is expressed as a tuple $n = \langle p, d, o \rangle$ where p is the pitch, d is the duration, and o is the onset time. The base index is 0. P is the sequence of the representative bass part. "π_x" is the relational operator for projecting the x attribute.

Require: array of notes **N**
  Sort **N** by ascending onset time as the first sort key and descending pitch as the second sort key.
  {Start taking the highest note at any onset time.}
  for $i = 0 \ldots |\mathbf{N}| - 2$ **do**
    if $(\pi_o n_i \neq \pi_o n_{i+1})$ **then**
      Append $\pi_p n_i$ to P.
    end if
    $d' \leftarrow \pi_o n_{i+1} - \pi_o n_i$
    $n_i \leftarrow \langle \pi_p n_i, d', \pi_o n_i \rangle$
  end for
  Append $\pi_p n_{|\mathbf{N}|-1}$ to P.
  {End.}
  return P

Using inter-onset intervals, the extended duration contour standardisation is "SL" for the first and second cases, and "Ll" for the third case. The difference between the first and second melodies is the articulation of the notes in the first bar, yet they both have the same rhythmic pattern. The difference is successfully picked up by inter-onset intervals. A musically-trained user is less likely to make rhythmic pattern errors when issuing queries. Articulation differences are less often considered as errors. Therefore, inter-onset intervals are more likely to be viable to improve retrieval effectiveness.

We modified the ALL-MONO algorithm so that the durations of a note is replaced by the time interval between itself and the following note. This is done indiscriminatively on the highest note at all onset times (excluding, the last note). Therefore, the difference between ALL-MONO and this algorithm (called ALL-MONO-IOI herethereafter) is that in ALL-MONO-IOI, there is no check whether the time to finish playing a note is after its following note. ALL-MONO-IOI is given as Algorithm 2.

Using ALL-MONO-IOI, we obtained a new set of duration-based representations of the pieces in our query set and collection. We used the same experimental setup outlined in Section 5, with this new

Baseline MAP value = 0.326.

w_π/w_δ	Scoring Scheme			
	1	2	12	13
0	0.025	0.017	0.035	0.022
1	0.176	0.130	0.051	0.051
2	0.322	0.236	0.156	0.207
3	0.318	0.271	0.232	0.281
4	0.320	0.318	0.262	0.307
5	0.319	0.324	0.314	0.313
6	0.319	0.327	0.327	0.353
7	0.319	0.327	0.346	0.356
8	0.319	0.327	0.348	0.356
9	0.319	0.327	0.348	0.356

Table 4: MAP values for various w_π/w_δ using inter-onset intervals. The best values for each w_π/w_δ are highlighted.

w_π/w_δ	**MAP**
10	0.355 544 269 543 587
11	0.355 550 207 447 156
12	0.355 571 651 845 875
	⋮
38	0.355 704 894 395 940
39	0.355 704 894 395 940
40	0.355 704 894 395 940

Table 5: MAP values for $10 \leq w_\pi/w_\delta \leq 40$.

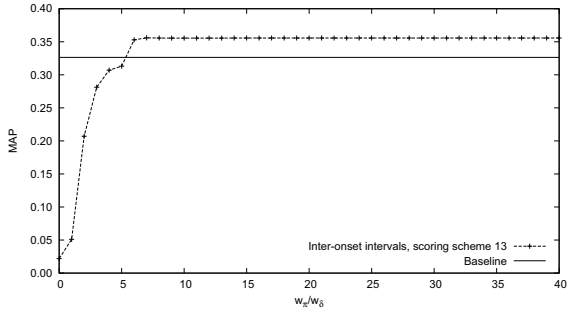


Figure 6: MAP values for pitch and inter-onset interval matching using scoring scheme 13.

set of representations. The MAP scores are given in Table 4.

The MAP values for 13 and $w_\pi/w_\delta \geq 5$ appear to be approaching an extreme. Therefore, we performed further experiments with $10 \leq w_\pi/w_\delta \leq 40$ and obtained the results shown in Table 5. To assist us determining whether there is an asymptotic value, we use 15 figures behind decimal point. We can see that the MAP values with w_π/w_δ starting from 38 are consistent. See Figure 6 for the plot of MAP values with scoring scheme 13.

The best obtained MAP value thus far is 0.356. This is slightly higher than the baseline value of 0.326. We analyse further whether the two means are significantly different using a paired t-test. It is found that incorporating inter-onset intervals using the vector model implies significant performance gain ($p < 0.05$).

8 Summary

We have compared two approaches of using duration-based information to improve retrieval effectiveness in this paper.

The first approach employs the durations of notes in the representative melody as extracted by the ALL-MONO algorithm [29]. Although the use of duration in addition to pitch improves retrieval effectiveness over the use of pitch only, the improvement is not significant. The second approach uses a modified version of ALL-MONO called ALL-MONO-IOI, which is similar to ALL-MONO except that the inter-onset intervals of representative melody notes are calculated. Although the modification is minor, our experimental setup shows that it has a significant impact on retrieval using duration-based information along with pitch. The retrieval effectiveness is improved significantly compared to using pitch only.

Acknowledgements We thank Falk Scholer and the anonymous reviewers for their input.

References

[1] J. C. C. Chen and A. L. P. Chen. Query by rhythm: An approach for song retrieval in music databases. In *Proceedings of IEEE International Workshop on Research Issues in Data Engineering*, pages 139–146, Feb. 1998.

[2] R. B. Dannenberg, W. P. Birmingham, G. Tzanetakis, C. Meek, N. Hu, and B. Pardo. The Musart testbed for query-by-humming evaluation. In Hoos and Bainbridge [9], pages 41–47.

[3] J. S. Downie. Informetrics and music information retrieval. In *Canadian Association for Information Science Proceedings of the 25rd Annual Conference*, pages 295–308. CAIS, June 1997.

[4] J. S. Downie. Informetrics and music information retrieval: An informetric examination of a folksong database. In *Canadian Association for Information Science Proceedings of the 26rd Annual Conference*. CAIS, June 1998.

[5] P. Ferraro and P. Hanna. Optimizations of local edition for evaluating similarity between monophonic musical sequences. In *Proceedings of Recherche d'Information Assistée par Ordinateur 2007*, Pittsburgh, USA, June 2007.

[6] M. Fingerhut, editor. *Proceedings of the Third International Conference on Music Information Retrieval*, Paris, France, Oct. 2002. IRCAM-Centre Pompidou.

[7] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, Cambridge, UK, 1997.

[8] P. Hanna, P. Ferraro, and M. Robine. On optimizing the editing algorithms for evaluating similarity between monophonic musical sequences. *Journal of New Music Research*, 36(4):267–279, Dec. 2007.

[9] H. H. Hoos and D. Bainbridge, editors. *Proceedings of the Fourth International Conference on Music Information Retrieval*, Baltimore, USA, Oct. 2003. Johns Hopkins University.

[10] International Music Information Retrieval Systems Evaluation Laboratory, editor. *Proceedings of the Second Annual Music Information Retrieval Evaluation eXchange*, Oct. 2006. URL `http://www.music-ir.org/mirex2006/`.

[11] T. Kageyama, K. Mochizuki, and Y. Takashima. Melody retrieval with humming. In *Proceedings of International Computer Music Conference 1993*, pages 349–351, 1993.

[12] K. Lemström, P. Laine, and S. Perttu. Using relative interval slope in music information retrieval. In *Proceedings of International Computer Music Conference 1999*, pages 317–320, Beijing, China, Oct. 1999.

[13] K. Lemström, N. Mikkilä, V. Mäkinen, and E. Ukkonen. Sweepline and recursive geometric algorithms for melodic similarity. In International Music Information Retrieval Systems Evaluation Laboratory [10]. URL `http://www.music-ir.org/mirex2006/`.

[14] D. Mazzoni and R. B. Dannenberg. Melody matching directly from audio. In J. S. Downie and D. Bainbridge, editors, *Proceedings of the Second International Symposium on Music Information Retrieval*, pages 17–18, Bloomington, USA, Oct. 2001.

[15] R. J. McNab, L. A. Smith, I. H. Witten, C. L. Henderson, and S. J. Cunningham. Towards the digital music library: Tune retrieval from acoustic input. In *Proceedings of ACM Digital Libraries 1996*, 1996.

[16] C. Meek and W. Birmingham. Johnny can't sing: A comprehensive error model for sung music queries. In Fingerhut [6], pages 124–132.

[17] A. Moles. *Information Theory and Esthetic Perception*. University of Illinois Press, Urbana, US, 1966.

[18] M. Mongeau and D. Sankoff. Comparison of musical sequences. In *Computers and the Humanities*, volume 24, pages 161–175. Kluwer, 1990.

[19] B. Pardo and W. Birmingham. Encoding timing information for musical query matching. In Fingerhut [6].

[20] L. Sitsky. *The Reproducing Piano Roll*. Department of Education, Canberra, Australia, Mar. 1979. ISBN 0-642-90543-6.

[21] I. S. H. Suyoto. Microtonal music information retrieval. Master's thesis, School of Computer Science and Information Technology, RMIT, Melbourne, Australia, 2003.

[22] I. S. H. Suyoto and A. L. Uitdenbogerd. Exploring microtonal matching. In C. L. Buyoli and R. Loureiro, editors, *Proceedings of the Fifth International Conference on Music Information Retrieval*, pages 224–231, Barcelona, Spain, Oct. 2004. Audiovisual Institute Pompeu Fabra University.

[23] I. S. H. Suyoto and A. L. Uitdenbogerd. Effectiveness of note duration information for music retrieval. In L. Zhou, B. C. Ooi, and X. Meng, editors, *Proceedings of the Tenth International Conference on Database Systems for Advanced Applications*, pages 265–275. Springer-Verlag, Apr. 2005. Published as LNCS 3453.

[24] I. S. H. Suyoto and A. L. Uitdenbogerd. Simple efficient n-gram indexing for effective melody retrieval. In International Music Information Retrieval Systems Evaluation Laboratory, editor, *Proceedings of the First Annual Music Information Retrieval Evaluation eXchange*, Sept. 2005. URL `http://www.music-ir.org/mirex2005/`.

[25] I. S. H. Suyoto, A. L. Uitdenbogerd, and F. Scholer. Effective retrieval of polyphonic audio with polyphonic symbolic queries. In J. Z. Wang, N. Boujemaa, A. Del Bimbo, and J. Li, editors, *Proceedings of the 9th ACM SIGMM International Workshop on Multimedia Information Retrieval*, pages 105–114, Augsburg, Germany, Sept. 2007.

[26] I. S. H. Suyoto, A. L. Uitdenbogerd, and F. Scholer. Searching musical audio using symbolic queries. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):372–381, Feb. 2008.

[27] R. Typke, F. Wiering, and R. C. Veltkamp. MIREX symbolic melodic similarity and query by singing/humming. In International Music Information Retrieval Systems Evaluation Laboratory [10]. URL `http://www.music-ir.org/mirex2006/`.

[28] R. Typke, F. Wiering, and R. C. Veltkamp. Transportation distances and human perception of melodic similarity. *ESCOM Musicae Scientiae*, (Discussion Forum 4A-2007):153–181, 2007.

[29] A. Uitdenbogerd and J. Zobel. Melodic matching techniques for large music databases. In D. Bulterman, K. Jeffay, and H. J. Zhang, editors, *Proceedings of the 7th ACM International Conference on Multimedia '99*, pages 57–66, Orlando, USA, Nov. 1999. ACM Press.

[30] A. L. Uitdenbogerd. *Music Information Retrieval Technology*. PhD thesis, School of Computer Science and Information Technology, RMIT, Melbourne, Australia, 2002.

[31] A. L. Uitdenbogerd. Variations on local alignment for specific query types. In International Music Information Retrieval Systems Evaluation Laboratory [10]. URL `http://www.music-ir.org/mirex2006/`.

[32] A. L. Uitdenbogerd. N-gram pattern matching and dynamic programming for symbolic melody search. In International Music Information Retrieval Systems Evaluation Laboratory, editor, *Proceedings of the Third Annual Music Information Retrieval Evaluation eXchange*, Sept. 2007. URL `http://www.music-ir.org/mirex2007/`.

[33] A. L. Uitdenbogerd and Y. W. Yap. Was Parsons right? An experiment in usability of music representations for melody-based music retrieval. In Hoos and Bainbridge [9], pages 75–79.

[34] A. L. Uitdenbogerd and J. Zobel. Music ranking techniques evaluated. In M. Oudshoorn, editor, *Proceedings of the Twenty-Fifth Australasian Computer Science Conference*, pages 275–283, Melbourne, Australia, Jan. 2002.

[35] A. L. Uitdenbogerd, A. Chattaraj, and J. Zobel. Methodologies for evaluation of music retrieval systems. *INFORMS Journal of Computing*, 18(3):339–347, 2006. ISSN 1091-9856.

[36] R. H. van Leuken, R. C. Veltkamp, and R. Typke. Selecting vantage objects for similarity indexing. In Y. Y. Tang, P. Wang, G. Lorette, and D. S. Yeung, editors, *Proceedings of the 18th International Conference on Pattern Recognition*, pages 453–456, Hong Kong, China, Aug. 2006.

[37] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishing, San Fransisco, USA, second edition, 1999. ISBN 1-55860-570-3.

Extraction of Named Entities from Tables in Gene Mutation Literature

Wern Wong[2], *David Martinez*[1,2], *Lawrence Cavedon*[1]

[1]NICTA Victoria Research Laboratory
[2]Dept of Computer Science and Software Engineering
The University of Melbourne

{*wongwl,davidm,lcavedon*}*@csse.unimelb.edu.au*

Abstract *Information extraction and text mining are receiving growing attention as useful techniques for addressing the crucial information bottleneck in the biomedical domain. We investigate the challenge of extracting information about genetic mutations from tables, an important source of information in scientific papers. We use various machine learning algorithms and feature sets, and evaluate performance in extracting fields associated with an existing hand-created database of mutations. We then show how this technique can be leveraged to improve on existing named entity detection systems for mutations.*

1 Introduction and Background

Biomedical science is a large, fast-paced and rapidly growing field. The volume of papers being written every year presents a serious bottleneck to researchers in the field, both in terms of keeping pace with discoveries and with checking for connections to be made with observations made in laboratories. A large amount of biomedical researchers' and workers' time is spent searching and reading the literature for information salient to a particular experimental result or observation in a clinical or diagnostic laboratory.

Information extraction and *text mining* techniques are garnering much interest in the biomedical space due to their potential for alleviating the information bottleneck experienced by researchers and clinical workers (e.g. [2]). We are interested in applying such methods to aiding the construction of databases of biomedical information, in particular information about genetic mutations. Such databases are currently constructed by hand: a long, involved, time-consuming and human-intensive process. Each paper considered for inclusion in the database must be read, the interesting data identified and then entered by hand into a database.[1]

In this paper, we focus on the task of extracting information from *tables* in biomedical research papers. Tables present a succinct and information-rich format for providing information, and are particularly important when reporting results in biological and medical research papers: the important results in such papers may be reported only in tabular form and not in the main text at all. Table processing presents its own challenges, especially that of detecting tables in text and dealing with structure—see [8] for a survey on table recognition and [3] for a discussion on processing tables in web documents. While interesting approaches to detecting and processing tables have been used in various applications—e.g. Wei *et al* [5] perform question-answering over tables extracted from financial documents—we know of no previous attempt to process tables in biomedical documents.

Our extraction task is grounded in the specific context of the *Mismatch Repair (MMR) Database* compiled at the Memorial University of Newfoundland [7]—a database of known genetic mutations related to hereditary non-polyposis colorectal cancer (HNPCC), a hereditary form of bowel cancer. The MMR Database contains information on genetic mutations known to be related to HNPCC, along with links to the research papers from which the database has been constructed.[2] From the database and its links to papers, we were able to construct a collection of tables related to HNPCC mutations, and then use the MMR database records themselves as a gold standard for evaluating our techniques. As at May 2008, the MMR database contained a total of 5,491 records on mutations that occur on any one of four genes that have been identified as related to colon cancer. An example record from the MMR database is the following:

```
MLH1 | Exon13 | c.1491delG | Yamamoto et al.  | 9500462
```

Respectively, this record contains: the gene; exon; mutation; citation of the paper the information was sourced from;[3] and the paper's PubMedID (PubMedID is a unique identifier assigned to papers whose abstract is contained in the MEDLINE collection). These fields are important because they contain information

[1]Karamis *et al* [4] illustrate how even simple tools can have an impact on improving the database-curation process.

Proceedings of the 13th Australasian Document Computing Symposium, Hobart, Australia, 8 December 2008.

[2]I.e. a team of geneticists manually trawled the biomedical literature for information on HNPCC-related mutation information, and added links to any papers relevant to those mutations in the context of HNPCC.

[3]This field has been abbreviated. We have also omitted fields such as "internal id".

researchers are directly interested in (gene, exon, mutation) and the paper said information was found in. Note that if a gene/mutation pair is referenced in multiple papers, then there are correspondingly multiple entries in the database. Conversely, if a single paper mentions multiple (relevant) genes, then that paper is mentioned in multiple database records. Our goal in this paper is to work towards automatic aids for the curation of this database.

2 Experimental Setting

In this section, we describe the process of creating our experimental dataset and the task design.

2.1 Creating the Dataset

Our collection of tables was extracted via the MMR database, leveraging the MEDLINE collection of biomedical abstracts. We first collected all information available in the hand-curated MMR records, obtaining a total of 5,491 mutations linked to 719 distinct PubMedIDs[4]. We next used a crawler to retrieve the corresponding full-text articles (identified by PubMedIDs stored in the MMR records) by following links from the PubMed interface, and downloaded those papers that had a full-text HTML version, and which contained at least one content table. Tables were then extracted from the full-text HTML files. It is worth noting that the tables were already present as links to separate HTML files rather than being presented as inline tables, making this process easier. Papers that did not contain tables in HTML format were eliminated.

Our final collection consisted of 70 papers from the original 719 PubMedIDs. The articles are linked to 784 MMR records (mutations), which constitutes our gold standard hand-curated annotation. The collection contains 197 tables in all.[5]

The tables in the collection were then pre-processed into a form that more readily allowed experimentation. The tables were split into three parts: column headers, row headers, and data cells. This was done based on the HTML formatting, which was consistent throughout the data set as the tables were automatically generated: cells were replicated when they spanned multiple rows or columns; `img` tags were replaced by the *alternate* text (when available); and `hr` tags were used to separate out column headers from the cells themselves. Row headers were detected by checking if the top left cell of the table was blank, a pattern which occurred in all row-major tables. We acknowledge that this processing may be specific to the vagaries of the particular format of the HTML generation used by PubMed (from which we sourced the tables). However, our whole task is specific to this domain; further, our focus is on the data extraction task rather than the actual detection of row/column headers.

[4]Data was downloaded from the web interface in May 2008.

[5]This collection could be increased in size by more putting more effort into retrieving documents linked to MMR records.

2.2 Task Design

In order to extract mutations from tables, we first performed classification of full columns/rows into relevant entities. Since the content of a column (or row, depending on whether the table was row- or column-oriented) tends to be homogeneous, this allowed us to build classifiers that can identify full vectors of relevant entities in a single step. We refer to this task as *table vector classification*.

We identified the following entities as relevant: Gene, Exon, Mutation, Codon, and Statistic. The first four were chosen directly from the MMR Database. We decided to include "Statistic" after inspecting the tabular dataset, since we found that this provides relevant information about the importance of a given mutation. From the five entities, Mutation is the most informative for our final information extraction goal.

The next step was to hand-annotate the headers of the 197 tables in our collection by using the five entities and the class "Other" as the tagset. Some headers belonged to more that one class, because the entities were collapsed into a single field of the table.

We performed two tasks: vector classification, and mutation extraction. The evaluation for the vector classification step was done using precision, recall and f-score, micro-averaged among the classes. For the machine learning (ML) algorithms, we used stratified 10-fold cross-validation. For mutation extraction we focus on the mutation class, and produce precision and recall against the subset of the hand-curated MMR database.

3 Table Vector Classification

We describe here heuristic and ML approaches to vector classification, along with an analysis of their performance.

3.1 Heuristic Approach

As a baseline method, we approached the task of classifying headers by matching the header string to the names of the classes in a case-insensitive manner. When the class name was found as a substring of the header, the class would be assigned to it. For example, a header string such as "Target Mutation" would be assigned the class "Mutation". Some headers had multiple annotations (e.g. "Gene/Exon").

For better recall, we also matched synonyms for the class "Mutation" (the terms "Variation" and "Missense") and the class "Statistic" (the terms "No.", "Number" and "%"). For the remaining classes we did not identify other obvious synonyms.

Results are shown in Table 1. Precision was very low for the "Mutation" class, illustrating that different types of information are provided under this heading; e.g. the heading "Mutation detected" above a "Gene"

Class	Precision	Recall	FScore
Gene	0.537	0.620	0.575
Exon	0.762	0.615	0.681
Codon	0.850	0.654	0.739
Mutation	0.283	0.301	0.292
Statistic	0.911	0.324	0.478
Other	0.581	0.903	0.707
Micro Avg.	**0.693**	**0.614**	**0.651**

Table 1: Naive Baseline results across the different classes and micro-averaged

Class	Precision	Recall	FScore
Gene	0.537	0.611	0.571
Exon	0.762	0.615	0.681
Codon	0.850	0.654	0.739
Mutation	0.600	0.452	0.515
Statistic	0.911	0.340	0.495
Other	0.579	0.910	0.708
Micro Avg.	**0.715**	**0.633**	**0.672**

Table 2: Results integrating MutationFinder across the different classes and micro-averaged

vector. Recall was low for most classes, suggesting that more sophisticated approaches are required.

Our second step was to build a more informed classifier for the "Mutation" class. We applied the mutation NER tool MutationFinder [1] to the text in cells to identify which table-vectors contained at least one mutation mention. Any such vectors were classified as mutations. The results are shown in Table 2. This approach caused the "Mutation" results to improve, but the overall f-scores leave room for improvement.

3.2 Machine Learning Methods

For the ML experiments we used the Weka [6] toolkit, as it contains a wide selection of in-built algorithms. As a baseline, we applied the majority class from the training data to all test instances. We applied the following ML algorithms from Weka[6]: Naive Bayes (NB), Support Vector Machines (SVM), Propositional Rule Learner (JRip), and Decision Trees (J48).

In order to define our feature sets, we used the text in both the headers and cells of the tables. Other sources of information, such as captions or the running text referring to the table where not employed at this stage, but may also provide valuable information. We used four feature sets:

- **Basic (Basic):** header string, the average and median cell lengths, and a binary feature indicating whether the data in the cells was numeric;
- **Cell Bag-of-Words (C_bow):** Bag of words over the tokens in the table cells;
- **Header Bag-of-Words (H_bow):** Bag of words over the tokens in the header strings;
- **Header + Cell Bag-of-Words (HC_bow):** Bags of words formed by the tokens in headers and cells, represented as different feature types.

[6]We applied a number of other ML algorithms as well, but these showed significantly lesser performance.

Algorithm	Feature Sets			
	Basic	**C_bow**	**H_bow**	**HC_bow**
Maj. Class	0.288			
NB	0.614	0.454	0.678	0.581
SVM	**0.717**	**0.599**	**0.839**	**0.816**
JRip	0.564	0.493	0.790	0.749
J48	0.288	0.532	0.793	0.782

Table 3: Micro-Averaged FScores for ML algorithms. The best results per column are given in bold.

Class	Precision	Recall	FScore
Gene	0.778	0.737	0.757
Exon	0.786	0.707	0.745
Codon	0.833	0.882	0.857
Mutation	0.656	0.679	0.667
Statistic	0.919	0.853	0.885
Other	0.820	0.884	0.850
Micro Avg	**0.839**	**0.841**	**0.839**

Table 4: Results for SVM and the feature set *H_bow* per class and micro-averaged.

The micro-averaged results of the different learning methods and feature sets are shown in Table 3. Regarding the feature sets, we can see that the best performance is obtained by using the headers as bag-of-words, while the content of the cells seems to be too sparse to guide the learning methods. SVM is clearly the best algorithm for this dataset, with JRip and J48 following, and NB performing worst of the four in most cases. Only for the basic feature set (with very few features) does NB outperform JRip and J48.

Overall, the results show that the ML approach is significantly superior to the baselines when relying on the header bag of words[7]; SVM is able to reach a high f-score of 83.9% in predicting the relevant entities.

We break down the results per class in Table 4, using the outputs from SVM and feature-set *H_bow*. We can see that all classes improve over the heuristic baselines. There is a big increase for the classes "Gene" and "Statistic", and all classes except mutation are above 70% f-score. "Mutation" is the most difficult class to predict, but it still reaches 66.7% f-score, which can be helpful for some tasks, as we explore in Section 4.

4 Mutation Extraction

We applied the results of our classifier to a real-world application: the detection of mutations in the literature for the MMR Database project. Table vector classification allows us to extract lists of candidate mutation names from tables to be added to the database. In order to test the viability of this approach, we measured the precision and recall of the system in detecting the existing hand-curated mutations in MMR. Recall is more important in this setting, since it shows the proportion of mutation mentions that we are able to obtain with our technique. Precision will give an indication of the rate of false positives, but note that we also consider as false positives those valid mutations that were not interesting

[7]As shown by a paired *t-test* at 99% confidence.

System	Precision	Recall
MF (full text)	0.01 (6 / 438)	0.01 (4 / 717)
Table Vector classifier	**0.09 (153 / 1702)**	**0.21 (153 / 717)**
Gold standard heads	0.11 (198 / 1847)	0.28 (198 / 717)

Table 5: Mutation detection results, Table Vector classifier in bold.

for MMR, and therefore the reported precision will be artificially low.

As state-of-the-art mutation detection system, we apply MutationFinder (MF) [1] to the full text (including tables) of the journal collection. This allows us to compare the results of our table-processing approach over an existing tool that parses the full text.

The evaluation results are shown in Table 5.[8] We can see that the goldstandard table vector annotation retrieves 28% of the mutations, at a precision of 11%. This means that by looking only at the tables we have an upper-bound of 28% on the percentage of relevant mutations that we can extract. In comparison, MF is only able to retrieve 1% of the mutations by looking at full articles. This happens because MF targets mutation mentions that follow a specific nomenclature, and the mentions that it is able to detect are not the ones covered in the MMR Database. Finally, our automatic table vector classifier is able to retrieve 21% of the gold standard mutations at a precision of 9%, which is 75% of the upperbound.

The precision figures are low for the goldstandard and table vector classifier. The reason for this is that we do not discriminate automatically for mutations of interest for the MMR Database, and valid mutation mentions are often classified as negative. All in all, the vector classifier discriminates 1,702 mutation cells out of a total of 27,700 unique cells in the collection, and it effectively identifies 153 out of the 198 relevant mutations present in the tabular data.

Finally, after the evaluation process we observed that many false mutation candidates could be removed by discarding those that do not contain two consecutive digits or any of the following n-grams: “c.”, “p.”, ’>’, “del”, “ins”, “dup”. This heuristic raises the precision of the system to 15.5% (153 true positives out of 989) with no cost in recall, which would result in greater saved time for database curators in a practical setting.

5 Discussion

Our preliminary results on the task of identifying relevant entities from gene mutation literature show that targeting tables can be a fruitful approach for text mining. By relying on ML methods and simple bag-of-words features, we were able to achieve good performance over a number of selected entities, well above header word-matching baselines. This allowed us to identify lists of mentions of relevant entities with minimal effort, reaching 21% of recall over a hand-curated database.

Another advantage of our approach is that the annotation of examples for training and evaluation is considerably easier, since many entities can be annotated in a single step. This opens the way to faster annotation of other entities of interest in the biomedical domain, which can present a wide variety of forms and non-standard terminology. However, since a table vector of homogeneous information may include representatives of the heterogeneous nomenclature schemes, classification of a whole column or row potentially helps nullify the effect of the terminological variability.

For future work, we plan to study different types of features for better representing the entities targeted in this work. Especially for mutation mentions, we observed that the presence of certain ngrams (e.g. ”del”) can be a strong indicator for this class. Another goal is to increase the size of our dataset of articles by improving our retrieval process, and by hand-annotating the retrieved table vectors for further experimentation.

Acknowledgements NICTA is funded by the Australian government as represented by Dept. of Broadband, Communication and Digital Economy, and the Australian Research Council through the ICT Centre of Excellence programme. Thanks to Mike Woods and his colleagues at the Memorial University of Newfoundland for making the MMR database available to us. Eric Huang wrote several of the scripts mentioned in Section 2 for creating the table collection.

References

[1] J. G. Caporaso, W. A. B. Jr., D. A. Randolph, K. B. Cohen, and L. Hunter. Mutationfinder: A high-performance system for extracting point mutation mentions from text. *Bioinformatics*, 23(14):1862–1865, 2007.

[2] A. M. Cohen and W. R. Hersh. A survey of current work in biomedical text mining : Annual progress in bioinformatics. *Briefings in Bioinformatics*, 6(1), 2005.

[3] M. Hurst. Layout and language: Challenges for table understanding on the web. Technical report, WhizBang!Labs, 2001.

[4] N. Karamanis, R. Seal, I. Lewin, P. McQuilton, A. Vlachos, C. Gasperin, R. Drysdale, and T. Briscoe. Natural language processing in aid of flybase curators. *BMC Bioinformatics*, 9:193–204, 2008.

[5] X. Wei, W. Croft, and D. Pinto. Question answering performance on table data. *Proceedings of National Conference on Digital Government Research*, 2004.

[6] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

[7] M. Woods, P. Williams, A. Careen, L. Edwards, S. Bartlett, J. McLaughlin, and H. B. Younghusband. A new variant database for mismatch repair genes associated with lynch syndrome. *Hum. Mut.*, 28, 2007.

[8] R. Zanibbi, D. Bolstein, and J. R. Cordy. A survey of table recognition. *Int’l J. on Document Analysis and Recognition*, 7(1), 2004.

[8] Because of the different mutation nomenclature formats used, comparison to gold standard required manual checking.

Facilitating Biomedical Systematic Reviews Using Ranked Text Retrieval and Classification

David Martinez *Sarvnaz Karimi* *Lawrence Cavedon* *Timothy Baldwin*

NICTA Victoria Research Laboratory
The University of Melbourne
Victoria 3010, Australia

{*davidm,skarimi,lcavedon,tim*}*@csse.unimelb.edu.au*

Abstract *Searching and selecting articles to be included in systematic reviews is a real challenge for healthcare agencies responsible for publishing these reviews. The current practice of manually reviewing all papers returned by complex hand-crafted boolean queries is human labour-intensive and difficult to maintain. We demonstrate a two-stage searching system that takes advantage of ranked queries and support-vector machine text classification to assist in the retrieval of relevant articles, and to restrict results to higher-quality documents. Our proposed approach shows significant work saved in the systematic review process over a baseline of a keyword-based retrieval system.*

Keywords Information Retrieval, Machine Learning.

1 Introduction

The growth and applicability of evidence-based medicine (EBM) has enormous potential for the way medical treatments are applied throughout the world. However, the task of preparing systematic clinical reviews for EBM is currently human workload-intensive. A systematic review is generally formulated against a specific clinical question, such as: *In a pre-hospital setting, what is the effect of intubation vs no intubation in traumatic brain injury*?

As a general framework, systematic reviews are conducted using the following main steps [3]:

1. formulate a high priority problem and develop the *inclusion* criteria (i.e. criteria for judging an article as relevant to the clinical question);
2. *search* through all the relevant published studies or articles. This step involves formulating a complex boolean query and submitting it to a number of databases of medical literature;
3. assess eligibility of retrieved articles, and extract data. The *assessment* involves judgement against the inclusion (and possibly exclusion) criteria;
4. analyse and present findings;
5. interpret results and draw conclusions.

Proceedings of the 13th Australasian Document Computing Symposium, Hobart, Australia, 8 December 2008.

While almost all these steps are labour-intensive, the second and third steps are particularly important, yet time-consuming. General practice in performing this *searching* stage involves developing boolean queries over well-known medical databases, such as Ovid MEDLINE, PubMed, or EMBase. Forming these queries, often as long as sixty lines, is not straightforward, with search experts constantly modifying their queries and seeking for keywords to be augmented to the query to improve recall. Also, boolean queries, even if potentially effective in retrieving most of existing relevant documents, do not rank the retrieved documents, and therefore all the returned articles must be scanned to specify candidates to be studied in detail.

Further, the *assessment* stage is often formulated in two steps:

1. reading the abstracts returned by the boolean query to determine if the associated papers are candidates for the review. This list may number in the tens of thousands;
2. retrieving and reading full-text documents for those which are included, and judging further which of these should be included in the review itself. This collection may number in the hundreds or thousands.

In this paper we investigate this searching problem in the framework of keyword-based text information retrieval (IR) and text classification. We are particularly interested in fully or partially migrating the searching step from boolean to ranked, as implemented in most popular search engines. Differences between retrieval for systematic reviews and standard IR tasks makes this problem challenging. In the searching stage of a systematic review, there is no simple specific topic to be looked up. Queries can be a combination of subject of the review, research questions to be addressed, and inclusion criteria to be observed. Unlike standard IR, recall is of crucial importance to ensure that no important evidence in regard to a clinical question is overlooked.

We investigate a two-stage search system that initiates a search using initial information on a priority research topic, then through a re-ranking scheme based

on text classification assist users to find relevant information more quickly. In particular, we estimate the potential amount of work saved by using such an automated system, as compared to performing manual reading and checking of all results returned by a boolean query, as is current practice.

2 Background

A number of organisations, such as the Cochrane Collaboration[1] and the Agency for Healthcare Research and Quality (AHRQ)[2] publish systematic reviews, as well as associated data for each step in the process: the boolean queries, search results, and inclusion criteria used. The enterprise of producing reviews, as well as updating the existing ones, can be massively time-consuming: a systematic review for a single clinical question may take a number of person-years to compile. Hence, any automated support for the process has the potential to be extremely valuable.

Work has recently been performed in improving the search process so that a higher quality set of documents are retrieved by the first of the search steps. In particular, document classification techniques are used to filter the set returned by the search, and thereby reduce the work of manually reviewing the articles that are candidates for inclusion in the final review. Further, the presence of audit data for existing reviews provides a valuable resource for evaluating performance and effectiveness of the techniques developed.

Cohen *et al* [2] specifically address the issue of reducing workload involved in preparing systematic reviews on specific classes of drugs. They construct a classification system, using a voting perceptron classifier [5], trained on data associated with 15 drug reviews published by AHRQ. The document set for each review topic was the set of MEDLINE abstracts returned by the initial search associated with that review, limited to those which were also contained in the TREC 2004 Genomics Track document corpus (so that full-text papers would be available). The classifier's feature set was constructed from these abstracts. Features included: bags-of-words constructed from title and abstract; MeSH (Medical Subject Headings) terms associated with the abstracts; and MEDLINE publication type. Inclusion and exclusion results for each abstract, as published by AHRQ, were used to create the classification gold standard.

Cohen *et al* evaluate their classifier using 5×2 cross-validation on the document set. They report recall and precision for each of the 15 drug reviews, as well as a measure of *work saved*, which is designed to more closely reflect the actual effectiveness of the classifier in the context of the task. *Work saved* is the percentage of papers that meet the published inclusion criteria which would not have to be manually inspected (because they were filtered out by the classifier). In particular, Cohen *et al* report work saved over random sampling at recall of 95% (WSS@95). This metric is described in greater detail in 4.1.

The actual effectiveness of their technique varied with topic. For 11 of the 15 review topics, WSS@95 was above 10%, which Cohen *et al* considered to be a minimum threshold of the technique adding value; it was estimated that this level would actually result in a saving of a person-week of effort. Three of the review topics resulted in a saving of over 50%.

Other research that uses text classification techniques in the context of EBM do not attempt to directly estimate "work saved". Aphinyanaphongs *et al* (ATSHA) [1] apply a number of techniques — Naïve Bayes, AdaBoost, and Support Vector Machines — to classify documents in various content areas: etiology, prognosis, diagnosis, and treatment. Their evaluation involved comparison with a baseline technique, developed by Haynes *et al* [6], which uses PubMed clinical queries to the above four areas. The EBM source for both baseline and ATSHA's system was the ACP Journal Club[3]. The ACP Journal Club has expert clinicians who categorise articles from a broad set of journals into categories including the ones listed above: this categorisation was the gold standard. ATSHA created filters from a collection of MEDLINE records corresponding to 49 journals referenced by the ACP Journal Club over a given time period, and these were used to filter articles from those journals into the 4 categories. ATSHA found that their machine-learning based categorisation generally outperformed the query-based categorisation[4], and argue that there is a significant reduction in workload over both the manual review method and over the time required to develop the query-filters.

MScanner [7] is a recent, more general-purpose biomedical classifier used to filter search results; it was designed for database creation/curation, rather than creating EBM reviews, with an emphasis on speed. MScanner uses a Naive Bayes classifier and a compact feature representation to support the processing of the whole MEDLINE collection in a reasonable time (approximately 90 seconds). Poulter *et al* [7] describe its effectiveness on a specific classification task as compared to the use of an expert-developed PubMed boolean query: on a task with 3,544 results (1,089 relevant, 2,465 irrelevant), MScanner was comparable to the hand-crafted query in recall and precision up until about 900 results.

3 A Two-Stage Ranking System

We propose a ranking system that pipelines a generic text retrieval search engine, and a classifier that re-ranks the retrieved documents as demonstrated below.

[1] `http://www.cochrane.org/`

[2] `http://www.ahrq.gov/`

[3] `http://www.acpjc.org`

[4] A slight drop in performance was noted for the diagnosis category, attributed to the small number of positive training examples.

3.1 Text Retrieval for Systematic Reviews

As mentioned earlier, common practice in literature review of medical articles is centred around boolean retrieval. Boolean retrieval has two main disadvantages: first, there is no ranking available in its output list, and second, it requires search expertise to formulate effective complicated queries. To facilitate this process for systematic reviews some pre-defined prototype templates have been defined for insertion into boolean queries. For example, if consideration should be restricted to only a particular publication type, such as *randomised controlled trials*, then a pre-formed Ovid format boolean query such as that below can be used as part of the main query [6]:

randomised controlled trial.mp OR
randomised controlled trial.pt

where *.mp* indicates the term should appear in the title, abstract, or MeSH headings[5], and *.pt* indicates *publication type*. For capturing topical features of the review, however, search experts need to specify appropriate keywords, and their arrangement in the query. For example, if a review's focus is pre-hospital intervention, the query might include:

prehospital.tw.
pre-hospital.tw.
paramedic$.tw.
ambulance$.tw.
out of hospital.tw.
emergency rescue.tw.
emergency resus$.tw.
emergency triage.tw.

where *.tw.* indicates the search should be applied to text words of title or abstract. It is worth noting that stemming as is practised in IR systems is forced by explicitly using $ in such queries and it is therefore more easily transferable to a ranked system than other features.

In contrast, a ranked retrieval system provides a ranked list, and querying is easier for inexpert users. However, it does not provide specific features of a boolean system, such as recursive operators (e.g. nested AND and OR), or searching over specific metadata available in the MEDLINE records or other medical or clinical text collections.

A systematic review is built on a priority research question that is generally composed of four information components: prevention, intervention, comparison, and outcome; these are collectively known as PICO. *Prevention* specifies for which group of patients the information is targeted; *intervention* specifies the medical event whose effect is under investigation; *comparison* is the evidence of producing better or worse results against other interventions or no intervention; and *outcome* specifies the effect of intervention. For instance, a question such as "*In the pre-hospital setting, what is the effect of intubation vs. no intubation in traumatic brain injury (TBI)?*" is composed of "traumatic brain injury" as prevention, "prehospital intubation" as intervention, "no intubation" as comparison. Outcome is not specifically listed in the review primary question but would be listed in its inclusion criteria.

Boolean queries normally cover one or two of these question components (mostly prevention and intervention), and the rest are inspected manually in the articles retrieved and marked to be further investigated (we refer to these as the *first tier* judgements). The remaining components are reported in the review as *inclusion criteria* or sometimes as *exclusion criteria* (listing articles that did not comply with some criteria). Examples of such inclusion criteria can be language — as only English languages studies be eligible — or publication date. Such criteria can be specific to the topic as well. For example, an inclusion criteria for the example review topic above could be "mortality" as outcome.

In addition to the main research question targeted in the review, there may be some research subquestion under the main review title. Some of these questions specify disjoint subsets of the final set of articles to be included in the review. They are sometimes covered by different boolean queries, which are reported separately in the review.

All of the above-mentioned characteristics of the search problem in systematic reviews make the query formulation process difficult. Many options require investigation for this domain, both for boolean and ranked queries. To make the process more effective and time efficient, specially if ranked queries are considered as a replacement for long boolean queries, studies on formulating good ranked queries need to be pursued.

In its first stage of ranking, our retrieval system uses a search engine to run selected ranked queries and generate a ranked list of retrieved articles. The main concern, however, is forming ranked queries. Candidate information sources for developing such queries include: review title; PICO; detailed research questions; and inclusion criteria. In our experiments we explore formulating ranked queries and evaluate them in terms of retrieval effectiveness.

3.2 Re-ranking via Text Classification

The complexity of the task explained in the previous section implies a need for either a powerful ranked retrieval procedure that captures all the inclusion criteria in the review; or, if a traditional ranking is used, it must be complemented with other components to help to retrieve as many relevant documents as possible. We are interested in a system which first and foremost has high recall, and as a secondary desideratum, retrieves eligible papers precisely. It also must reduce the workload in systematic reviewing as much as possible. There-

[5]MeSH are the Medical Subject Headings, an taxonomy of medical terms which are manually ascribed to every entry in MEDLINE

fore, we study the effect of document classification in a ranked system framework.

We rely on support vector regression (SVR) [4] to re-rank the output of the text retrieval system. The basic idea of regression algorithms is to find a function that approximates the training instances by minimising the prediction error. The main difference between SVR and other regression methods is that a user-specified parameter ϵ defines the lower limit from where deviations are considered. SVR also tries to maximise the "flatness" of the function at the same time as minimising the error, that is, it tries to fit all training instances within a margin of width $2 \times \epsilon$. There is also a tradeoff with the prediction error, since it may be necessary to allow some training instances to have nonzero error in order to build a better function. This is controlled by the parameter C.

For our experiments, we used the Weka machine learning toolkit [8] with first order polynomial kernels and default parameters ($\epsilon = 0.001$, $C = 1$). In order to train our classifier we took the top-ranked documents (from the retrieval engine) at different cutoffs. The trained model was then applied to the rest of the collection, and finally the documents were ranked according to their weight. Different sizes of training data were tested to better analyse the classifier's effectiveness: the top-ranked 10, 20, or 30 percent of the documents were used.

As our feature representation we chose two feature sets. Our basic feature representation consists of a bag-of-words model including all words occurring in the "abstract" and "references" sections of the paper. For our second set of features, we extended the first set with the MeSH headings of the articles. We performed separate experiments in order to measure the impact of the hand-annotated MeSH terms in the results.

4 Experimental Data

MEDLINE is the most popular database of articles of medical articles freely available to the research community. A recent collection of 16,676,340 abstracts was used in this study as document collection. For every article indexed by MEDLINE, there is an entry — known as a *citation* — which contains the title and abstract of the article, accompanied by metadata. The metadata includes publication date, language, author information, MeSH headings associated with the article at the time of its publication, publication type and venue, and a unique identifier known as a PMID.

We selected 17 systematic reviews from the AHRQ collection (see Section 2) to test our system and form our queries. The selection process was based on the clear provision of included and excluded papers and search strategies; this means that relevance judgements were available for the queries (i.e. the documents returned by the boolean query). A list of included and excluded MEDLINE citations as indicated in the reviews were extracted using their provided PMIDs as the first level of relevance judgements (first-tier). We excluded any listed paper that was not indexed in MEDLINE. The same process generated another set of relevance judgements on documents included in the final review (second-tier). Specifications of these reviews are listed in Table 1.

review	AHRQ identifier, Year	first-tier	second-tier
1	1, 1999	822	184
2	66, 2002	267	67
3	106, 2004	38	12
4	118, 2005	273	92
5	130, 2006	421	198
6	131, 2006	413	83
7	136, 2006	158	117
8	145, 2006	508	104
9	146, 2007	130	34
10	167, 2008	1,103	440
11	138, 2006	796	65
12	57, 2003	647	228
13	11, 1999	329	21
14	100, 2004	535	121
15	103, 2004	932	203
16	110, 2004	2,329	365
17	116, 2005	158	77

Table 1: Specifications of the selected reviews. The third column represents the total number of articles initially considered to be further investigated in details (first-tier), and the fourth column specifies the number of documents chosen to be included in the review (second-tier).

4.1 Evaluation

We evaluated our system on two levels: retrieval that generate the initial results from ranked querying, and the final re-ranked list. For the initial text retrieval evaluation, we use standard IR metrics: precision and recall.

For document classification, we calculate the WSS measure (work saved over sampling) as defined by Cohen et al [2]. WSS measures the number of documents in the collection that the user would need to hand-check in order to reach a fixed recall over the relevant documents. For example, if the fixed recall is 95% and the documents are randomly ordered, an average of 95% of the articles would have to be inspected to reach 95% recall. WSS measures the difference between a given ranking and a random sampling. Its formula is shown below

$$WSS = ((TN + FN)/N) - 1 + (TP/(TP + FN)), \quad (1)$$

where TP, TN, FN, and N represent the number of true positives, true negatives, false negatives, and total number of instances, respectively.

In this paper, we measure WSS at different points in the document ranking, considering the TOP-K as positive and the remaining as negative. For our main results we fix the recall to 95%, in line with the study by Cohen et al. [2]. To calculate the WSS metric, we compute recall for different K values until 95% recall is achieved. At this point, WSS@95 can be computed as shown in Equation 2

$$WSS@95 = ((TN + FN)/N) - 0.05. \quad (2)$$

Note that the number of relevant documents in the ranking (R) will affect the minimum and maximum values of the metric. For WSS@95, since we have a fixed recall of 95%, FN will always be $0.05 \times R$; N will be constant; and TN will determine the value of the ranking in saving work. Low TN/N ratios will produce negative WSS scores, since this indicates that we need to go to the bottom of the list to find all relevant documents, which would be outperformed by a random sampling.

It is worth explaining that ideally we would want to *a priori* approximate the K value that separates positive and negative instances in the classifier's output. This can be a difficult problem, since we cannot be sure of the total number of relevant documents for a given query. One way of dealing with this issue would be to estimate the expected relevant documents by using training or held-out data as a reference.

Apart from WSS, we also calculated the *receiver operating characteristic* (ROC) curves and corresponding *area under curve* (AUC) values for the produced rankings. ROC curves illustrate the trade-off between the true positive rate and the false positive rate, providing a visualisation of the classifier's performance at different cutoff points. The AUC score summarises the curve and the performance of the classifier in a single number, for easier comparison.

For the final results—since the text classification module also requires us to use the top documents for training, for a uniform evaluation over different training splits—we measured the WSS@95 and AUC scores over the full collection, by putting the training documents at the top, and re-ranking the remaining according to the scores of the classifier. The reason for this is that the benefits from re-arranging the training data should not be credited to the classifiers.

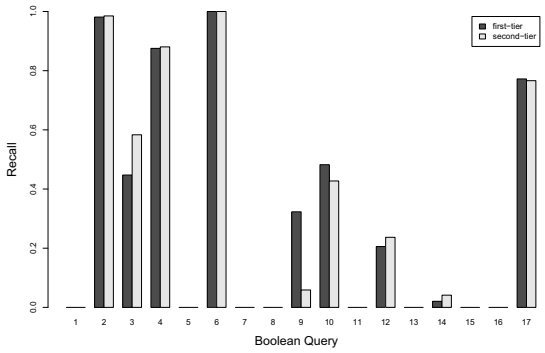
5 Experimental Results

We first measure the performance of the existing boolean queries over MEDLINE. We then evaluate different retrieval strategies that rely on different sources of query-words. For our final experiment we rely on a text classification algorithm for improving the initial ranked results.

5.1 Boolean Retrieval

Each systematic review contains search strategies which list all the finalised boolean queries – for different databases – used to retrieve potential relevant articles. We extracted Ovid MEDLINE boolean queries from our selected 17 reviews and re-ran them against Ovid MEDLINE that indexes articles from 1950 to the first week of October 2008. We then extracted PMIDs of the retrieved articles and matched them against the relevance judgements we created from the reviews as reported in Table 1. Surprisingly however not all these

(a) Replicated boolean queries



(b) TREC ranked queries (TOP-10K)

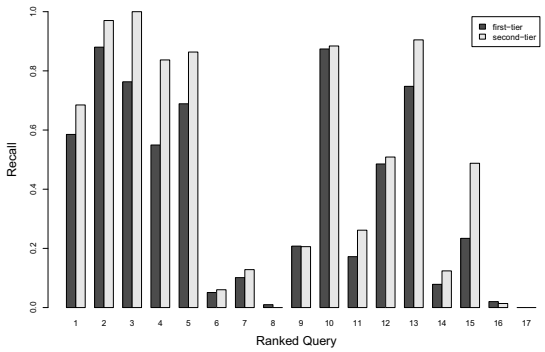


Figure 1: Recall of replicated boolean queries and ranked queries based on the relevance judgements reported in the reviews. Note 7 boolean queries (1, 5, 7, 11, 13, 15, 16) are just shown for easier comparison of the two graphs and they do not represent any data.

queries were in the state of reproducibility and led to errors when running them in Ovid. Examples of these errors were missing query lines in the reported queries, non-matching MeSH headings, and query lines referring to specific partial results in the query that made it dependant on the time it has been first executed. Figure 1 (a) shows recall values of 10 out of 17 queries only, as the remaining were not replicable. As presented in the figure, in contrast to our expectation, not all the relevant documents were retrieved by these queries, with some of them showing very low recall (query 8 and query 14 had 0.0 and 0.02 recall, respectively, based on the first-tier judgements). This clearly illustrates the limitations of the present approach for documenting systematic review strategies based on boolean queries.

5.2 Ranked Retrieval

Ranked retrieval is the first stage of our architecture, as explained in Section 3.1, which retrieves an initial set of documents to feed to the second stage, classification.

We created ranked queries from the 17 systematic reviews available from AHRQ based on three main in-

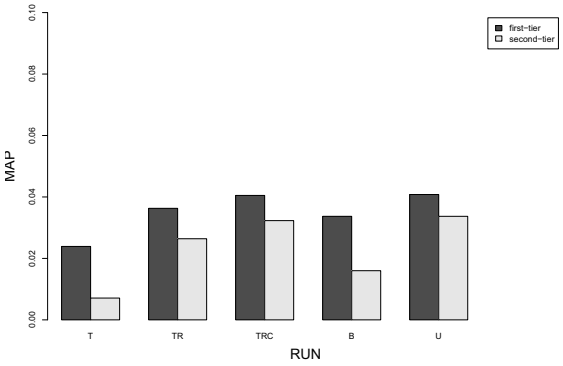


Figure 2: Retrieval effectiveness of different types of ranked queries on MEDLINE citations (TOP-10K).

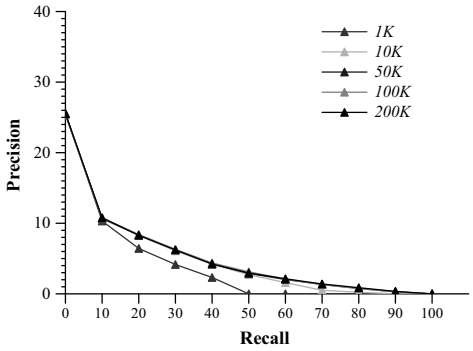


Figure 3: Precision at eleven recall points for different cutoffs when using queries composed of title, research questions, and inclusion criteria (TRC). Judgements were first-tier articles.

formation components they are built on: title or major research question (T), other research questions (R), and inclusion criteria (C). For each of these query types, we used the exact string that appeared in the corresponding review. We also flattened boolean queries to make ranked queries by removing metadata indicators such as *.tw.* or *.pt.* (B). Finally, we ran some combinations of these queries against the MEDLINE collection. In our experiments, we used Zettair[6] search engine with its default setting for Okapi BM25 ranking scheme.

Figure 2 shows the mean average precision (MAP) for the five categories of title only (T), title and research questions (TR), title and research questions and inclusion criteria (TRC), ranked queries made from boolean queries (B), and using unique words of C with T and R (U). All categories of the queries work poorly, with MAP scores less than 0.1 using each of the first-tier and second-tier judgements. The third group, TRC, achieved a slightly better MAP score of 0.0405 in comparison to others, and we will consider this method as the baseline for the classification experiments. We also show recall values for each query in Figure 1 (b) as a comparison to their boolean counterparts where available. Interestingly, ranked queries are more effective based on the second-tier judgements, where only included articles are considered as relevant (twelve out of seventeen queries had higher recall in the second-trier level than the first-trier level). Also, comparing the recall values of the boolean and ranked queries in the second-trier level, five ranked queries had higher recall than boolean queries, with boolean queries winning only for four queries.

Recall values specify a maximum threshold on our system effectiveness in finding relevant documents. That is, our classifier can improve the ranking of these documents in the list, but no new relevant document that might have been missed in the initial retrieval can be added to the pool.

Systematic reviews are expected to cover all the relevant studies and therefore recall is crucial. In order to choose an experimentally sound cutoff point in our ranking list that covers most relevant documents, we calculated precision at eleven recall points for the different cutoffs of TOP-1K, TOP-10K, TOP-50K, and TOP-100K, averaged over 17 queries (Figure 3). There was little difference between recall values when more than ten thousand documents were retrieved. We therefore chose TOP-10K results as input to the classifier. This number of documents also reflects the amount of articles that researchers commonly manually check after running boolean queries, which can be in the order of tens of thousands.

5.3 Re-ranking via Text Classification

The goal of the re-ranking step is to improve the retrieval rankings by moving relevant documents towards the top of the list. As mentioned above, achieving very high recall is crucial, and the value of a ranked list will be dependant on the number of documents required to reach a given recall (95% in our case), which can be shown by the metrics WSS@95 and AUC (see Section 4.1).

For our classification experiments we rely on the top 10,000 documents from the text retrieval step, using the TRC query strategy, which attained the highest recall. As relevance judgements, we focused on the documents included in the final reviews (second-tier); the main reasons being that: (i) they are the ones chosen for the final review, and (ii) the retrieval step showed that the recall is proportionally higher for these.

Table 2 shows the WSS results and TOP-K value for our first experiment relying on the basic feature set (no MeSH terms), with different amounts of training data. The TRC column shows WSS@95 without re-ranking; the average WSS score for these is 16.4%, which would be the amount of work saved over random sampling of the documents. The classifier is able to clearly improve these values for different amounts of training data, obtaining the best performance for all but three of the queries. The results using 10% of data are significantly improved, according to the paired t-test, and we save

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

	Baseline		Classifier 10%		20%		30%	
Query	WSS	K	WSS	K	WSS	K	WSS	K
1	8.9	8608	19.4	7564	9.0	8599	16.9	7812
2	19.4	7562	53.3	4169	44.7	5030	38.4	5660
3	11.0	8398	56.5	3847	56.7	3830	55.8	3925
4	17.9	7710	67.9	2707	41.5	5348	42.4	5259
5	45.1	4986	67.7	2728	67.6	2742	59.4	3558
6	42.1	5288	42.1	5288	42.1	5288	63.2	3179
7	8.7	8631	69.6	2537	54.9	4010	59.8	3517
8	-	-	-	-	-	-	-	-
9	59.0	3597	33.0	6195	18.4	7655	13.6	8135
10	1.6	9344	26.6	6835	33.4	6163	31.2	6375
11	2.7	9230	2.2	9281	25.2	6977	-3.0	9799
12	13.4	8165	7.6	8736	5.8	8916	5.0	9003
13	8.7	8628	5.4	8958	0.7	9426	-2.3	9729
14	2.5	9247	-2.3	9731	24.0	7096	16.0	7896
15	4.5	9048	7.7	8728	4.5	9054	2.0	9296
16	-0.1	9506	-0.1	9506	13.7	8133	63.5	3146
17	-	-	-	-	-	-	-	-
Avg.	16.4	7863.2	30.5†	6454.0	29.5†	6551.1	30.8	6419.3
Wins	3		6		4		2	

Table 2: Comparison of the baseline (TRC) and the two-stage system with classifier using WSS metric with basic features. K: number of documents that are returned as positive. *Wins*: number of queries for which a particular approach attains the best results. †: paired t-test with 95 percent confidence level (over baseline).

more than 30% of the work on average. Notice also that for queries 8 and 17 there are no results, since the TRC query-strategy is not able to return any relevant document.

As the training data gets bigger we can see that the WSS score remains very similar on average. The main reason for this is that even if the classifier should get more accurate with additional training data, the top of the ranking will be given by TRC, and the pool to re-rank will be smaller, forcing the classifier to do a better job to obtain the same WSS score. We can see that the results for different training splits change depending on the query, but the averages remain the same. The paired t-test shows that there are differences between 10% and 20%, but not between 20% and 30%. These figures illustrate that 10% of training data is enough to benefit from the text classification system.

We also calculated the more known ROC curves and corresponding AUC values for this experiment. The results are given in Table 3, and they show a similar behaviour, with the 10% classifiers obtaining slightly better curves than other training splits. In this case the differences are always significant according to the t-test.

Our next step was to add MeSH terms as features. A summary of the WSS@95 and AUC results is given in Table 4, with the baseline feature results for reference. We can see that MeSH terms improves the results when using 20% of the data for training. This indicates that the classifier improves considerably in ranking the remaining 80% of data.

The WSS results for each query are shown in Figure 4, when using MeSH terms. Here, we can see that there are big differences depending on the query, with cases where the classifiers make a huge contribution

Query	Baseline	Classifier 10%	20%	30%
1	69.6	75.9	73.4	74.6
2	85.2	89.7	89.6	88.2
3	89.8	92.3	93.6	93.5
4	86.7	91.2	88.9	87.9
5	90.6	93.1	92.4	91.5
6	66.4	66.4	66.4	74.7
7	68.9	90.8	83.4	79.8
8	-	-	-	-
9	86.6	83.7	77.7	74.4
10	61.3	79.4	75.9	71.3
11	56.2	75.3	74.1	66.3
12	66.3	72.1	72.3	69.6
13	84.9	85.7	86.5	84.9
14	68.2	71.6	80.7	72.6
15	71.2	75.3	74.8	74.3
16	57.5	57.5	67.7	76.1
17	-	-	-	-
Avg.	73.9	80.0‡	79.8‡	78.6†

Table 3: Comparison of the baseline (TRC) and two-stage system that uses classifier based on AUC metric. †, ‡: paired t-test with 95 and 98 percent confidence levels respectively (over baseline).

Metric	Baseline	Feats.	Classifier 10%	20%	30%
WSS	16.4	Base	30.5	29.5	30.8
		MeSH	31.5	34.3†	31.2
AUC	73.9	Base	80.0	79.8	78.6
		MeSH	80.5‡	80.6‡	79.0‡

Table 4: Comparison of average WSS and AUC between Base features and adding MeSH headings to the Base ones. †, ‡: paired t-test with 95 and 98 percent confidence levels respectively (MeSH over baseline).

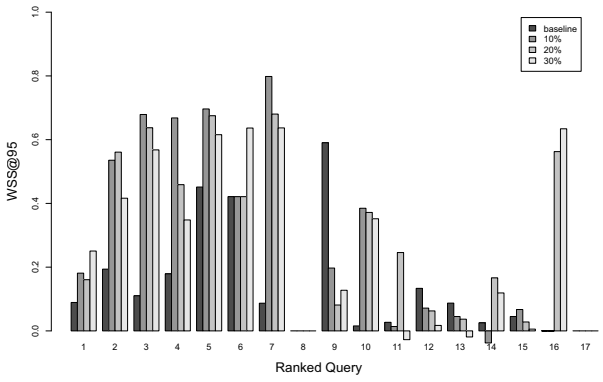


Figure 4: Word saved over sampling (WSS@95) per query when MeSH terms are used along with text of articles as classifier training features.

(e.g. queries 7 and 16), and others where the scores go down (e.g. queries 9, 12 and 13). The results for query 9 are particularly interesting; this is the query that has the highest baseline, but the performance of the re-ranking system clearly drops. On the other hand, for query 16 the baseline ranking is as low as random sampling, but the classifier is able to obtain remarkable results. We think that there are a number of parameters affecting the final performance, such as the baseline score or the number of relevant documents in the collection, which would give us a better indication of the expected performance per query. Overall, we can see that the re-ranking approach is beneficial for most queries.

6 Conclusions and Future Work

We addressed the search needs for building systematic clinical reviews for EBM, an increasingly growing area that targets the way medical care is provided. This problem is specifically difficult to solve with standard search strategies. We illustrated some of the main problems of the current approach, which relies on boolean queries: they are time-consuming to formulate and maintain, they are difficult to execute without expert knowledge, and they do not provide a ranking of documents. Furthermore, when replicating existing search strategies from the AHRQ collection (publicly available on the web) we found that some are poorly documented, and those that can be replicated do not lead to the set of documents that was used in the construction of the final review.

Thus, we explored the use of ranked queries and text classification for better retrieval of the relevant documents. We found that different keyword-search strategies can reach recall that is comparable and sometimes better than the costly boolean queries. Use of ranked queries for systematic reviews was not explored before in the previous studies. In our next step we found that these retrieval rankings can be re-organised using machine learning to significantly reduce the amount of work required to find most of the relevant documents. These results show good potential for the migration from boolean queries towards ranked systems, which are easier to maintain and provide means to prioritise the document analysis.

For future work our aim is to integrate our experimental findings into a new tool to aid in the construction of systematic reviews, focusing on the *search* and *assessment* steps. This tool would benefit from the user's feedback to dynamically re-rank the documents remaining to be analysed, and reduce the time to generate and maintain the reviews. Another important issue that we are exploring is the way to estimate the total number of documents to be checked to reach the required recall. We plan to address this issue by using similarity thresholds between the relevant documents already identified and the remaining candidates. Finally, we also want to enrich the features of our text classifier by adding different types of information, such as citation contexts.

Acknowledgements NICTA is funded by the Australian government as represented by Department of Broadband, Communication and Digital Economy, and the Australian Research Council through the ICT centre of Excellence programme.

References

[1] Y. Aphinyanaphongs, I. Tsamardinos, A. Statnikov, D. Hardin and C. F. Aliferis. Text categorization models for high-quality article retrieval in internal medicine. *Journal of the American Medical Informatics Association*, Volume 12, Number 2, pages 207–216, 2005.

[2] A. M. Cohen, W. R. Hersh, K. Peterson and P. Y. Yen. Reducing workload in systematic review preparation using automated citation classification. *Journal of the American Medical Informatics Association*, Volume 13, Number 2, pages 206–219, 2006.

[3] The Cochrane Collaboration. Cochrane handbook for systematic reviews of interventions, version 5.0.0, `http://www.cochrane.org/resources/handbook/`, 2008.

[4] H. Drucker, C. J. Burges, L. Kaufman, A. Smola and V. Vapnik. Support vector regression machines. *Advances in Neural Information Processing Systems*, Volume 9, pages 155–161, 1997.

[5] Y. Freund and R.E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, Volume 37, pages 277–296, 1999.

[6] B. Haynes, K. A. McKibbon, N. L. Wilczynski, S. D. Walter and S. R. Werre. Optimal search strategies for retrieving scientifically strong studies of treatment from Medline: analytical survey. *British Medical Journal*, Volume 330, Number 7501, pages 1179–1182, 2005.

[7] G. Poulter, D. L. Rubin, R. B. Altman and C. Seoighe. Mscanner: a classifier for retrieving medline citations. *BMC Bioinformatics*, Volume 9, Number 1, 2008.

[8] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

Parameter Sensitivity in Rank-Biased Precision

Yuye Zhang *Laurence A. F. Park* *Alistair Moffat*

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

{*zhangy, lapark, alistair*}*@csse.unimelb.edu.au*

Abstract *Rank-Biased Precision (RBP) is a retrieval evaluation metric that assigns an effectiveness score to a ranking by computing a geometricly weighted sum of document relevance values, with the monotonicly decreasing weights in the geometric distribution determined via a persistence parameter p. Despite exhibiting various advantageous traits over well known existing measures such as Average Precision, RBP has the drawback of requiring the designer of any experiment to choose a value for p. Here we present a method that allows retrieval systems evaluated using RBP with different p values to be compared. The proposed approach involves calculating two critical bounding relevance vectors for the original RBP score, and using those vectors to calculate the range of possible RBP scores for any other value of p. Those bounds may then be sufficient to allow the outright superiority of one system over the other to be established. In addition, the process can be modified to handle any RBP residuals associated with either of the two systems. We believe the adoption of the comparison process described in this paper will greatly aid the uptake of RBP in evaluation experiments.*

Keywords Rank-Biased Precision, Evaluation, System Comparison

1 Introduction

Effectiveness evaluation focuses on allocating scores to retrieval systems, allowing researchers to compare pairs of systems, and argue that one or the other has the better effectiveness. When using a non-parameterized metric, systems are simply compared by the effectiveness score computed for each system's set of retrieved relevance vectors. However, the task of comparing systems is complicated when adjustable effectiveness-scoring parameters are introduced, as it is difficult (or simply meaningless) to compare systems evaluated using different parameter values. This presents a problem for measures where there are no conventionally agreed values for those parameters, and results in a lack of any clear baselines being established as the basis for future improvements.

Unfortunately, many popular metrics such as NDCG [Järvelin and Kekäläinen, 2002] and BPref [Buckley and Voorhees, 2004] make use of such parameters, and it is often the case that the parameter needs to be adjusted to the experimental context or be chosen based on prior knowledge of the dataset properties. This problem also extends to metrics such as Average Precision, for which the depth of the evaluation is clearly a parameter that must be set by the experimental designer, and might play a large part in determining the numeric value of the scores that are achieved.

Rank-Biased Precision (RBP) [Moffat and Zobel, 2009] is an evaluation measure which assigns relevance weights based on the geometric distribution for a given parameter $0 \leq p < 1$ or *persistence*, where a smaller p value places greater emphasis on documents that appear early in the ranking, and a larger p spreads the weight further down the document ranking, but in both cases with all documents in the ranking contributing to the final score.

Despite the merits of RBP, there is no single "best" p that can be used for experimentation, as p is by its very nature something that is varied across different types of experiment. Here we present a method to compare RBP scores computed using differing p values, based on the bounding binary relevance vectors obtained by inverting the RBP score calculation.

Our methodology uses a three step process:

1. For a given p and RBP score, calculate the lexicographically greatest and least relevance vectors which might have generated that score (at some floating point precision).

2. Using these two vectors, calculate the range of possible RBP values for the target p value.

3. Finally, based on the target RBP value, we can deduce whether one RBP score is outright greater than the other.

Additionally, this method can be modified to work for RBP values with non-zero residuals, or known

Proceedings of the 13th Australasian Document Computing Symposium, Hobart, Australia, 8 December 2008.

imprecision. Our investigation into these properties shows that changing p results in interesting behaviors in RBP based on the source values, and that it is possible for outright comparisons to be performed on two RBP scores computed using different parameters. Our outcomes suggest that RBP can be employed more extensively in evaluation experiments than it is currently, with reduced concerns over incomparable results between researchers.

Section 2 introduces the RBP metric and other related work in the general research area. Section 3 describes the method to generate relevance vectors from an initial RBP score and associated p value, in particular the process to obtain the lexicographically greatest and least relevance vectors. Section 4 demonstrates how to obtain a range of RBP values using these two vectors and the interpretation needed to form a clear system comparison. Section 5 presents a modified process for handling the presence of RBP residuals. Section 6 examines some peculiarities and limitations of the comparison process. Finally, Section 7 concludes the paper and discusses possibilities for future investigation.

2 RBP and related metrics

Rank-Biased Precision (RBP) is based on the monotonically decreasing values in a geometric sequence. It has the form:

$$\text{RBP}(R, p) = (1 - p)\sum_{i=1}^{|R|} r_i p^{i-1}$$

where p is a abstraction of the user's searching persistence, expressed as a parameter between 0 and 1, R represents the relevance vector to be evaluated, and r_i indicates the relevance of the document ranked in position i within the ranking [Moffat and Zobel, 2009]. Unlike some other metrics, RBP does not utilize the global number of relevant documents for the query and is formulated in such a manner that a relevant document at any given rank contributes a set value to the overall score, meaning that potential contributions from unjudged documents can also be calculated and incorporated as they become available. Consequently, RBP is always bounded between zero and one, with a score of one only achievable when the length of an "every document is relevant" ranking vector approaches infinity.

As an example of how RBP is calculated, suppose that a user has persistence of $p = 0.5$, meaning that there is a 50:50 chance that the user will progress from one document in the ranking to the next. If a system returns the relevance vector $R = \{11010001\}$, where 1 denotes a relevant document and 0 denotes an irrelevant document, then the RBP of this system is computed as: $(1 - 0.5) \times (0.5^0 + 0.5^1 + 0.5^3 + 0.5^7) = 0.816$.

Figure 1 depicts the effect of three different values of p on the RBP contribution for a set of ranks. As p increases, the contribution from early ranked documents decreases, and later ranked documents increase their weight in the final RBP score. The specific properties of the geometric sequence imply that it is possible to determine the ranking depth required when evaluating to a given accuracy for any predetermined value of p. For example, when $p = 0.5$ the sum of contributions from rank 12 onwards is 4.88×10^{-4}. Therefore, when evaluating to a precision of three decimal places (0.001), it is possible to do so using relevance judgments up to just rank 11 (and no further), as 4.88×10^{-4} rounds to less than 0.001. We make use of this property later.

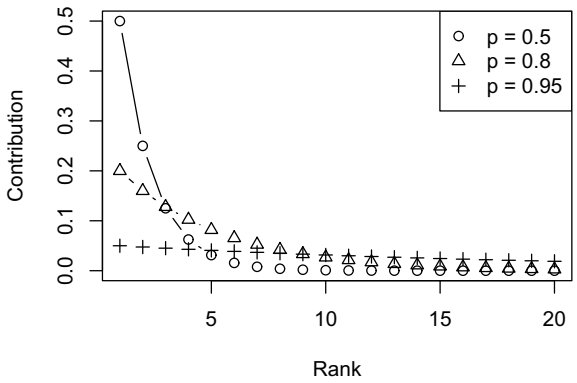


Figure 1: Contribution of each rank towards the RBP total for three values of p. Increasing the value of p shifts the emphasis to documents further down the ranked list.

Previous studies [Park and Zhang, 2007] suggest that for web search a p value of 0.8 is an appropriate value. In practice, values as high as 0.95 are used in experiments with higher pooling depths such as TREC [Voorhees and Harman, 2000], matching the deep evaluations provided by Mean Average Precision (MAP) and NDCG. Moffat and Zobel [2009] argue that this flexibility in choosing p works to RBP's advantage, allowing it to (mimic as required) the characteristics of other common evaluation metrics, such as Reciprocal rank, Precision@10, and so on.

Other recent studies have also examined RBP. Park and Zhang [2007] describe a method for selecting p based on session and click-through data from a large Microsoft query log. Moffat et al. [2007] investigate methods for reducing the number of relevance judgments required when performing system comparisons, and evaluated their methods in the context of RBP. Webber et al. [2008] describe a process for standardization using existing experimental results to modify evaluation metrics to shift reliance away from collection specific parameters. Compared to other evaluation metrics, Discounted Cumulative Gain (DCG) [Järvelin and Kekäläinen, 2002] bears many similarities to RBP, although overall DCG scores are unbounded as ranked lists grow longer, and the approach employed in Normalized Discounted Cumulative Gain (NDCG) requires prior knowledge of the relevance data. The BPref metric [Buckley and Voorhees, 2004] and the Q-measure metric [Sakai,

2004] are examples of more complex measures that make use of query-specific relevance information and the sequencing of relevant documents within the result list being evaluated.

Like all other metrics, RBP scores are easily comparable when the systems in question use the same parameter values, or supply the original rankings (runs), in which case scores using new parameters can be calculated. However, since not all experiments utilize identical parameter values, and published results typically only include summary data rather than details of the runs, alternative methods are required to compare systems evaluated using different p values.

3 Generating relevance vectors

As mentioned previously, it is possible to compare RBP scores computed using different p values as long as the original relevance vector is available. Our key contribution is the observation that it is possible to reconstruct a set of relevance vectors which could have given rise to any given RBP score.

Since the contribution of a relevant document at each rank is fixed for a given p, we can take a straightforward constraint-based approach to determine the values of ranks in the relevance vector. Given a retrieval system, with RBP score S obtained using persistence p, our goal of generating relevance vectors can be formally defined as calculating some set of relevance vectors $\mathcal{R}$, such that each relevance vector $R = \{r_1, r_2 \cdots\} \in \mathcal{R}$ satisfies the equation:

$$\text{RBP}(R, p) = S\,.$$

In our scenario of generating vectors R that satisfy S using p, we have no knowledge of any $r_i \in R$ and it is our goal to determine their values. We now define the following constraints:

Constraint 1 *Given R and p where r_j is determined for $0 < j < i$ and RBP$(R,p) < S$, if setting $r_i = 1$ causes the calculated RBP(R,p) to become greater than S, then either $r_i = 0$, or one of the earlier r_j values is incorrect.*

Constraint 2 *Given R and p where r_j is determined for $0 < j < i$ and RBP$(R,p) < S$, if setting $r_k = 1$ for $k > i$ still results in RBP$(R,p) < S$, then either $r_i = 1$, or one of the earlier r_j values is incorrect.*

An example demonstrates the use of the two constraints when deriving the required relevance vector. Suppose that the target score is $S = 0.4$, that $p = 0.8$, and that $R = \{1\ 0\ 0\ 0\ 1\ ?\ ?\ ?\}$, where ? represents a ranking with some unknown values. In this configuration, RBP$(R,p) = 0.2812$. If the options for r_6 are then considered, setting it to 1 does not break constraint 1, but setting $r_6 = 0$ breaks constraint 2, because the remaining unknown document judgments can only contribute at most 0.0943, which is not enough to allow the target of $S = 0.4$ to be reached (because $0.2812 + 0.0943 < 0.4$). Therefore $r_6 = 1$, giving $R = \{1\ 0\ 0\ 0\ 1\ \underline{1}\ ?\ ?\}$. The search can then continue.

	10^{-2}	10^{-4}	10^{-8}
$p = 0.5$	8	15	28
$p = 0.7$	15	28	54
$p = 0.8$	24	45	86
$p = 0.9$	51	94	182
$p = 0.95$	104	194	373
$p = 0.99$	528	986	1,001

Table 1: Number of significant ranks at given floating point precision as a function of p.

To formalize this process, let con(i) represent the contribution for a relevant document at rank i. Then, assuming binary relevance and some fixed p:

$$\text{con}(i) = p^{i-1} \times (1-p)\,,$$

and the contributions of all remaining documents from rank i onwards can be represented as:

$$\text{rem}(i) = \sum_{k=i+1}^{|R|} \text{con}(k)\ .$$

The constraints can now be expressed as:

$$r_i \in R = \begin{cases} 0 & \text{if acc}(i) + \text{con}(i) > S \\ 1 & \text{if acc}(i) + \text{rem}(i) < S \end{cases}$$

where:

$$\text{acc}(i) = \begin{cases} 0 & \text{if } i = 1 \\ \sum_{j=1}^{i-1} r_j \cdot \text{con}(j) & \text{otherwise} \end{cases}$$

In this approach, we evaluate the values of r_i sequentially by following the constraints, starting at r_1. Technically R can reach lengths up to infinity, but we bound this value by specifying a precision at which we stop the calculation. Hence, $|R|$ is simply the rank at which the rounded value of rem(i) becomes less than the precision. Table 1 depicts the number of results which are significant in terms of precision for various p and precision values.

For cases when neither constraint is satisfied, there is a choice. To obtain the full set of vectors $\mathcal{R}$, both possible values for r_i would be made at these *choicepoints*, and the search would then continue along both paths. However, we will take special note of two possible R vectors with useful properties: the one with the most relevant documents at the top of the ranking, obtained by always assigning $r_i = 1$ at choicepoints and denoted as R_G; and the one with the most irrelevant documents at the top of the ranking, obtained by always assigning $r_i = 0$ at choicepoints, denoted as R_L.

Both R_G and R_L are useful in that they depict the extremes of possible relevance combinations, being respectively the lexicographically greatest and lexicographically least. For our proposed method of RBP comparisons, simply making use of these two vectors is sufficient, meaning there is no need to generate all of $\mathcal{R}$.

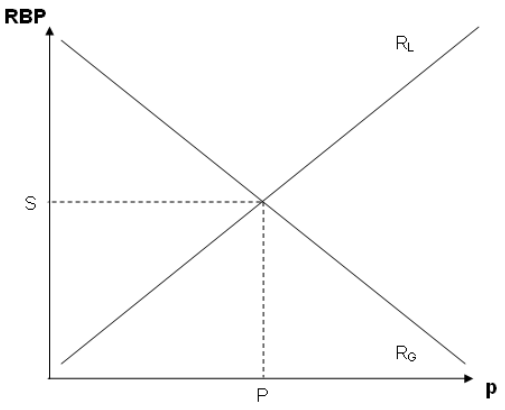


Figure 2: Changes in RBP value for R_G and R_L vectors for varying p values. As p increases, greater emphasis is placed on later ranked documents, pushing up the score assigned to the R_L vector, and decreasing the value assigned to the R_G vector. When p decreases, the converse is true. This figure illustrate a case in which R_L and R_G are divergent.

4 Comparing RBP values

The generated relevance vectors then allow calculation of the (range of) RBP scores that might have arisen if a different value of p had been used.

4.1 Bounding RBP scores for varying p

For a given initial p and score S, upper and lower limits on the RBP score can be computed for all other values of p, using the R_L and R_G vectors. These two values represent the extremes that can arise from any other members of $\mathcal{R}$, and yield the largest variations in RBP score as p is varied.

Recall that a floating point precision was specified to limit the length of the relevance vectors, with the implication that for a fixed precision and RBP score, a higher p' value (where $p' > p$) requires more ranks to be fully represented. This creates a undesirable situation where r_i of significant ranks at p' are unavailable, meaning we cannot use the R_G and R_L vectors in their current state as $|R|$ is too short to fully represent the RBP score at the required precision.

However, as we are primarily interested in the upper and lower bounds for possible RBP values, designating $r_i = 1$ with R_G and $r_i = 0$ with R_L for the extended rank positions in p' provides the greatest and least possible values for RBP respectively. When moving to a lower p value, the number of significant ranks decreases, meaning that $r_i = 0$ is appropriate for the extended ranks.

Figure 2 shows a representation of the range of possible RBP values obtainable from an initial p and S pairing. Intuitively, the range of possible RBP scores expands as p increases. All possible values in the range $[0, 1]$ eventually become obtainable as p asymptotically approaches 1. That is, as the effect of later ranked documents is accentuated, the score from the R_L vector increases, and the score from the R_G vector decreases.

On the other hand, when the value of p approaches 0, greater emphasis is placed on early ranked documents, until only the first ranked document is significant in the RBP computation. Indeed, below $p = 0.5$ the first document in the ranking dominates the sum of all of the other rank positions. This property allows us to easily predict the behavior of the R_G and R_L scores for smaller values of p: as p tends to zero, if $r_1 = 1$ in R_L, both scores converge to 1. Otherwise, if $r_1 = 0$ in R_G, both R_L and R_G scores will converge to 0. Furthermore, given the initial p and S, we can easily determine the value r_1, as $\text{con}(1) = (1-p) \times p^{1-1} = (1-p)$ and $\text{rem}(1) = 1 - \text{con}(1) = p$. Therefore, if $S \leq (1-p)$, r_1 must equal 0. Conversely, if $S \geq p$, then r_1 must equal 1. Figure 3 depicts this occurrence for nine different combinations of p and S. Note that one of the nine combinations in the matrix of possibilities is infeasible, and has not been plotted.

4.2 System comparison with RBP

We now have a process to handle our experimental scenario: suppose we have two retrieval systems (A & B) which executed the same query on identical datasets. The author of system A reported a RBP score of S_A calculated using p_A. Similarly, the author of system B reported a RBP score of S_B calculated using $p_B > p_A$. We need to determine, if possible, whether system A outperforms system B on that query or vice versa, using one or the other of the two values of p. Using the methods outlined above, we can accomplish this task by generating the R_G and R_L vectors of one system and comparing the range of possible RBP scores to those of the other at that system's p.

Although we have the choice of generating relevance vectors for either system (and thus attempting the comparison at either of the two values of p), it is prudent to use the system with the higher p value, and compare the range of RBP scores to the system with the lower p value. This is because moving from a higher p to a lower p places greater emphasis on contributions of early ranks, and later ranks are more likely to be uncalculated due to initial precision specification. The same effect was noted earlier when recalculating RBP for higher values of p.

With these considerations in mind, we now have a complete process for comparing systems evaluated using RBP with different p values:

1. For evaluation systems A and B, assuming $p_B > p_A$, generate R_G and R_L using p_B and S_B at the required level of accuracy.
2. For $p < p_B$, crop $|R|$ to the significant ranks at the given level of accuracy. Calculate $\text{RBP}(R_G, p)$ and $\text{RBP}(R_L, p)$.
3. For $p > p_B$, append $r_i = 1$ to R_G and $r_i = 0$ to R_L until the significant rank at the given level of accuracy is reached. Calculate $\text{RBP}(R_G, p)$ and $\text{RBP}(R_L, p)$.

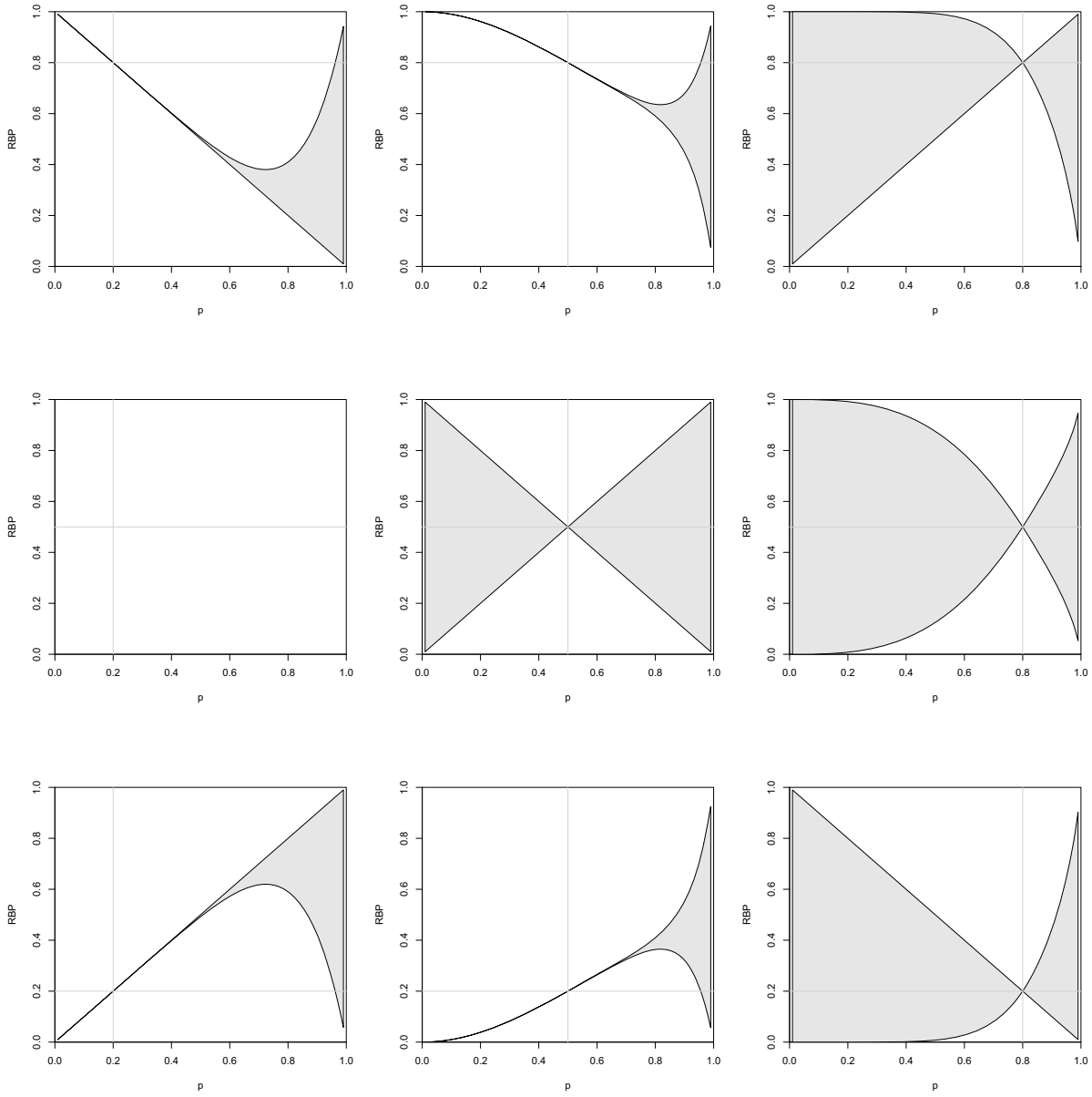


Figure 3: Convergence behavior using combinations p and S (RBP) values drawn from $\{0.2, 0.5, 0.8\}$. The bottom left corner shows $p = 0.2, S = 0.2$, and the top right corner shows $p = 0.8, S = 0.8$. Intersections occur at (p, S) in each graph, corresponding to the supplied arguments for generating R_G and R_L. The full divergent range of RBP scores in $[0, 1]$ becomes obtainable for values of p tending to 1, when the unspecified tail of the ranking has the power to completely change the score. Convergence of R_G and R_L for small values of p can be easily validated with be comparing p and S: both will converge to 1 if $S \geq p$ or both will converge to zero if $S \leq (1 - p)$. The graph of $p = 0.2, S = 0.5$ is missing because a score of 0.5 cannot arise when $p = 0.2$, as all valid RBP scores must be either greater than 0.8, or less than 0.2. Note that all of these bounds are based solely on the S and p values. If the actual ranking is available, RBP with a reduced p can always be calculated to at least the same accuracy as it was using the initial value of p.

4. System B outperforms system A at $p = p_A$ if and only if $S_A < \text{RBP}\,(R_L, p_A)$.

5. System A outperforms system B at p_A if and only if $S_A > \text{RBP}\,(R_G, p_A)$.

6. Otherwise, there is no clear outcome as to which system is superior.

The first two outcomes are fairly straightforward: if the RBP score of system A fails to reach lowest possible RBP score of system B, then system A is inferior. The second outcome is simply the mirrored case. However, when the RBP of system A lies in the range of system B, there is no clear evidence of superiority either way: out of all possible relevance vectors generated from system B (which fall between the RBP bounds marked out by system B's R_G and R_L), a non-empty subset of those vectors results in a higher RBP score at p_A, while others result in a lower score.

5 Integrating RBP residuals

Although the proposed method is generally applicable, retrieval experiments are often run with limited relevance judgments due to resource constraints, meaning that evaluation with exhaustive relevance judgments is impossible. In the case of RBP, this uncertainty is handled by summing the contributions of all unjudged documents to form an error bound, or *residual* [Moffat and Zobel, 2009]. The RBP residual (ε) can be described as:

$$\varepsilon(R, p) = (1 - p) \sum_{i=1}^{\infty} \text{unjudged}(i) \cdot p^{i-1}$$

where $\text{unjudged}(i) = 1$ if and only if r_i is unknown.

In this sense, the RBP score S of a ranking is the lower bound of RBP (if all unjudged documents are irrelevant), and $S + \varepsilon$ gives the upper bound, achieved if all unjudged documents are relevant. Ideally, both S and ε, along with the p employed, are reported when effectiveness evaluation results are being disseminated.

5.1 Relevance vectors with residuals

Residuals present a challenge when reconstructing the relevance vectors used, in that it is no longer valid to select r_i freely when choicepoints are encountered. This is because unlike previously discussed, the generated relevance vector R must be able to produce $S+\varepsilon$ should some subset of $r_i = 0$ positions be switched to $r_i = 1$, but still give S if they all remain unaltered. These modifications apply to both the R_G and R_L vectors.

Fortunately, we are able to incorporate these conditions into the original calculation process. We still need to abide by the original constraints so that the relevance vectors sums to the required S, but also have to integrate additional rules such that R_G and R_L is capable of satisfying $S + \varepsilon$ which certain positions are altered. Therefore, we integrate some new rules that affect the selection of r_i when we encounter a choicepoint.

For the R_G vector, we want to set $r_i = 1$ at choicepoints as long as the $r_i = 0$ positions (decided or otherwise) can contribute the equivalent of the residual. Furthermore, we have to take into account for ranks that haven't been decided, some subset of r_i that must be allocated to fulfilling the lower bound S:

$$r_i \in R_G = \begin{cases} 0 & \text{if } \text{acc}'(i) + \text{rem}\,(i) < \\ & \qquad S - (\text{con}\,(i) + \text{acc}\,(i)) + \varepsilon \\ 1 & \text{otherwise}\,, \end{cases}$$

where:

$$\text{acc}'(i) = \begin{cases} 0 & \text{if } i = 1 \\ \sum_{j=1}^{i-1} (1 - r_j) \cdot \text{con}\,(j) & \text{otherwise}\,. \end{cases}$$

For the R_L vector, we want to set $r_i = 0$ at choicepoints as long as the remaining contributions can reach the upper bound $S + \varepsilon$:

$$r_i \in R_L = \begin{cases} 1 & \text{if } \text{acc}\,(i) + \text{rem}\,(i) < S + \varepsilon \\ 0 & \text{otherwise}\,. \end{cases}$$

After integration of these rules, both R_G and R_L fit the criteria which allows some subset of their $r_i = 0$ judgments to be altered to reach the upper bound of the RBP score.

5.2 Positions of unjudged ranks

To supplement R_G and R_L, we determine the positions at which unjudged documents may occur in these vectors, taking

$$r_i = \begin{cases} \text{NA} & \text{if } \text{acc}\,(i) + \text{rem}'(i) < \varepsilon \\ 0 & \text{otherwise}\,, \end{cases}$$

where:

$$\text{rem}'(i) = \begin{cases} 0 & \text{if } i = 1 \\ \sum_{k=i+1}^{d} (1 - r_k) \cdot \text{con}\,(k) & \text{otherwise}\,. \end{cases}$$

Note that this calculation is only applied on ranks where $r_i = 0$, since we cannot change positions which have been previously decided to be relevant. Choicepoints can also be handled in a similar manner to determine the possible combinations of unknown judgment ranks, meaning we can consistently select $r_i = \text{NA}$ to shift unjudged documents towards the earlier ranks, or $r_i = 0$ to shift them to later ones.

Using the R_G and R_L vectors as a base, we now have four possible vector combinations. However, because we are most interested in the extremes of the relevance combinations, we let R'_G represent the lexicographically greatest vector with unknown judgments shifted towards the top of the ranking, and R'_L represent the lexicographically least vector with unknown judgments shifted towards the bottom of the ranking.

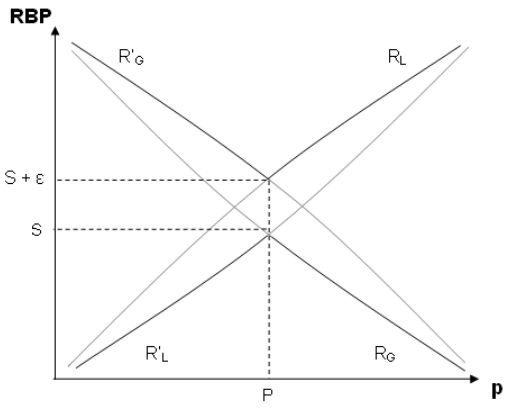


Figure 4: Change in RBP values for relevance vectors when p is varied. The darker lines indicates the upper and lower bounds for possible RBP values at any given p. This stylized figure illustrates the change in RBP bounds; and actual RBP bounds will vary.

5.3 System comparison with residuals

Using our previous definitions of S_A, p_A, S_B and $p_B > p_A$, we now introduce ε_A and ε_B to represent the RBP residual reported by each system. Following the convention of generating relevance vectors for system B as it has the higher p value, we now have bounds of $\{S_B, S_B + \varepsilon_B\}$ as the set of possible RBP values at p_B.

Firstly, we will deal with range of RBP scores possible at the lower bound of the initial RBP score. In this case, r_i = NA ranks in both R'_G and R'_L must be 0 for the score of S_B to be obtained. This implies the positions of the unjudged documents are not of importance, as only $r_i = 0$ positions were altered in creating R'_G and R'_L. We can simply calculated the possible RBP scores using the unaltered R_G and R_L vectors.

At the upper bound of the initial RBP score, all unjudged ranks in both R'_G and R'_L must be changed to 1 for the score of $S_B + \varepsilon_B$ to be obtained. We must now set r_i = NA to $r_i = 1$ in both vectors, and plot these in a similar manner. Figure 4 depicts the RBP bounds for a non-zero RBP residual.

We can see that in the case of a RBP residual being present, the upper and lower bounds on the possible RBP values are dictated by the magnitude of the residual at the initial p values. For larger values of p, the range of possible RBP values is dictated by R'_L and R_G, while for smaller values of p it is dictated by R_L and R'_G. Using these observations, our outcomes for the experimental scenario can be updated as follows:

1. System B outperforms System A if and only if $(S_A + \varepsilon_A) < \text{RBP}\,(R'_L, p_A)$.
2. System A outperforms System B if and only if $S_A > \text{RBP}\,(R'_G, p_A)$.

The interpretations for these outcomes are similar to the situation when no residual is present: one system outperforms the other only it's lowest possible RBP score at the given p is higher than the highest possible RBP score for the other system. In the case of overlap, it is still impossible to determine whether one system is better due insufficient knowledge about the generated relevance vectors.

6 Discussion

Although our method for comparing the RBP scores of different retrieval systems is relatively straightforward in terms of the processes involved, there remains a number of inherent characteristics (some intrinsic to the design of RBP itself) which should be taken into consideration.

Firstly, the initial process of generating relevance vectors is applicable for all values of p, although the number of possible relevance vectors generated varies:

- For $p = 0.5$, for all ranks $\text{con}\,(i) = \text{rem}\,(i)$ meaning there will always be exactly two vectors generated for all RBP values with either recurring 0 or 1 at the tail. These are the R_G and R_L vectors respectively. The range of possible RBP values as p is shifted is $[0, 1]$, as shown in Figure 3.
- For $p < 0.5$, some values of RBP are impossible to obtain: consider the case when $p = 0.2$ and $S = 0.5$. In this case, $\text{con}\,(1) = (1 - 0.2) \times (0.2)^0 = 0.8$ and $\text{rem}\,(1) = 0.2$ (assuming $d = \infty$), meaning it is impossible to obtain any RBP score between 0.2 and 0.8, as r_i cannot be assigned in any manner. In all other cases, there is a single unique vector. The range of possible RBP values is $[0, p]$ and $[1 - p, 1]$, with similar (and so on recursively) gaps within these two ranges.
- For $p > 0.5$, all possible values of RBP are obtainable. At a given level of accuracy, higher values of p will generate more potential vectors in $\mathcal{R}$ as there is a smaller variation in $\text{con}\,(i)$. The range of possible RBP values is $[0, 1]$.

Furthermore, because the generation process is based around determining potential contributions of individual ranks in the output vector, RBP scores obtained using non-binary relevance judgments are incompatible. This is because the additional variable introduced by the scaled judgments further confounds the range of choicepoints, and instead of seeking binary representations using a fractional radix, we are seeking n-ary ones.

Earlier we mentioned that when ranges of possible RBP values overlap, no unambiguous conclusion can be made in terms of relative superiority of systems. However, in the case when no residual is present in either system, supposing we have generated all possible relevance vectors $\mathcal{R}$ (instead of just R_G and R_L), calculating their values at the target p value will give a discrete set of possible RBP values instead of a range. Using basic probability measures it is tempting to determine a crude percentage chance that either system is superior.

Unfortunately, the shortcomings of this approach outweigh the benefits. The number of calculations required increases greatly for larger initial values of p, and the presence of RBP residuals in either system exacerbates the problem. The computation may be manageable for small residuals, but the overall expenditure does not justify the (arguably) limited usefulness of the information obtained. A simpler approach would be to calculate the numeric overlap of the two RBP regions, although this conveys even less information as $\mathcal{R}$ can vary greatly with different values of p, meaning the probability estimate will be quite likely to be skewed in some manner.

Finally, despite the fact that it is possible to calculate the range of RBP values at all values of p using the R_G and R_L vectors, the knowledge of upper and lower bounds of RBP for p greater than the original has limited usefulness. Although it was included in our illustrations for completeness, the upper and lower bounds represent extreme cases when the trailing documents are either all completely relevant or completely irrelevant. As such, these bounds should be used as guidelines only, rather than an indication that the comparison is possible.

7 Conclusion

We have presented a method comparing two systems evaluated using the Rank-Biased Precision effectiveness measure with different values for parameter p. This is achieved by generating the lexicographically greatest and least relevance vectors which can give rise to the original RBP score at the specified p, and using those two vectors to model the upper and lower bounds of possible RBP values at all other values of p. Furthermore, the generation and modeling process can be modified to handle RBP residuals, which will almost certainly be present in real world evaluation experiments.

By utilizing the processes we outlined, it may be possible for direct conclusions to be drawn regarding the superiority of one system over another, even though they have been scored using different values of p. This is a significant improvement from the current situation where there is no process to compare systems evaluated using varying values of p, and may aid in the uptake of RBP as a standard experimental metric. It is also possible that, with appropriate amendment, our proposed method for system comparison can be applied to other evaluation metrics that have fixed relevance contributions for each given position in the document ranking.

In terms of possible improvements, currently our method for performing system comparisons fails to deliver a clear outcome when there is an overlap between the RBP bounds of both systems at the required p value. It is not clear how this situation can be handled due to the limited information available when generating relevance vectors, and finding better approaches to this problem is a topic currently under investigation.

Finally, to get a sense of how often RBP bounds do in fact overlap when performing comparisons, it may be possible to utilize runs from existing TREC datasets and recalculate RBP scores using different p values. We can then compare different systems and observe how often one system is outright superior to the other, and establish a general idea of the applicability for our comparison method in its current form.

Acknowledgements This work was supported by the Australian Research Council. National ICT Australia (NICTA) is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

References

Chris Buckley and Ellen M. Voorhees. Retrieval evaluation with incomplete information. In *Proc. 27th Ann. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 25–32, Sheffield, United Kingdom, 2004. ACM Press, New York.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

Alistair Moffat and Justin Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems*, January 2009. To appear.

Alistair Moffat, William Webber, and Justin Zobel. Strategic system comparisons via targeted relevance judgments. In *Proc. 30th Ann. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 375–382, Amsterdam, The Netherlands, 2007. ACM Press, New York.

Laurence A. F. Park and Yuye Zhang. On the distribution of user persistence for rank-biased precision. In *Proc. 12th Australasian Document Computing Symp.*, pages 17–24. School of Computer Science and Information Technology, RMIT University, Australia, December 2007.

Tetsuya Sakai. Ranking the NTCIR systems based on multi-grade relevance. In *Proc. Asian Information Retrieval Symp.*, volume 3411, LNCS, pages 251–262. Springer, Berlin/Heidelberg, October 2004.

Ellen M. Voorhees and Donna Harman. Overview of the Ninth Text REtrieval conference (TREC-9). In *Proc. 2000 TREC Text Retrieval Conf.* National Institute of Standards and Technology, November 2000. `http://trec.nist.gov/pubs/trec9/papers/overview_9.pdf`.

William Webber, Alistair Moffat, and Justin Zobel. Score standardization for inter-collection comparison of retrieval systems. In *Proc. 31st Ann. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 51–58, Singapore, July 2008. ACM Press, New York.

Querying Linguistic Annotations

Sumukh Ghodke and Steven Bird

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

{sghodke,sb}@csse.unimelb.edu.au

Abstract *Over the past decade, a variety of expressive linguistic query languages have been developed. The most scalable of these have been implemented on top of an existing database engine. However, with the arrival of efficient, wide-coverage parsers, it is feasible to parse text on a scale that is several orders of magnitude larger. We show that the existing database approach will not scale up, and speculate on a new approach that leverages proximity search in the context of an IR engine. We also propose a simple syntax for querying linguistic annotations, avoiding the usability problems with existing tree query languages.*

Keywords Information Retrieval, Natural Language Techniques and Documents, XML Document Standards

1 Introduction

High quality part-of-speech taggers and syntactic parsers are able to annotate large quantities of English text [5]. With suitable indexing methods, it should be possible for users to express queries that are sensitive to this additional information, and support more focussed search.

In some cases, the part-of-speech of a word may disambiguate the primary senses of the word, e.g. *wind*, *park*. A user could easily select the intended POS-tag using a query format like: `wind/N` or `park/V`. In other cases we want to do a proximity search but need to constrain the syntax of the intervening material, e.g. `give NP up` will find instances of "give up" wrapped around a noun phrase (NP), and won't include results which have other intervening material.

Expecting users to annotate their queries adds a significant burden, without guaranteeing that the result will be sufficiently improved to justify the effort. However, we hypothesise that a specialised query engine can cluster results from a conventional ad-hoc query using extra information present in the linguistic annotations. Exemplars from each cluster could be presented to the user, together with the annotations that characterise each cluster. In this way, users are educated about the relevant linguistic properties and can start to annotate their own queries.

Proceedings of the 13th Australasian Document Computing Symposium, Hobart, Australia, 8 December 2008.

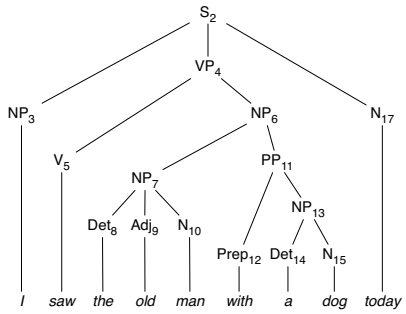


Figure 1: Syntax Tree

For example, an ad hoc query for documents concerning acquisitions of QANTAS mostly returns documents concerning acquisitions *by* QANTAS. The linguistic difference between these two cases is the grammatical role of QANTAS relative to the main verb (subject vs object). If this difference could be discovered, a user would be able to look through results of the refined query: `"acquire QANTAS/NP-OBJ"`. Similar analysis could detect differences in tense and cluster the results as pertaining to the past or the future.

This paper explores the suitability of existing work on linguistic tree query as the basis for such a query engine. The paper is organised as follows. First we give an overview of existing work on linguistic tree query (§2) and XML indexing (§3). Our scaling experiments are presented in §4. Our negative conclusions about scaling lead to a more speculative discussion (§5) on the prospects for using an IR engine for performing the desired query tasks.

2 Linguistic Tree Query

The problem of representing and querying linguistic annotations has been an active area of research for several years [3, 7]. It has grown out of work on curating large databases of annotated text such as *treebanks* [10] for use in developing and testing language technologies. Figure 1 gives an example of a parsed sentence; it represents the constituent structure of a sentence, and involves non-terminal nodes for noun phrases, verb phrases, prepositional phrases, and so on. Trees also encode relationships between these constituents, and permit us, amongst other things, to discover the subject and object noun phrases corresponding to a particular verb.

At least a dozen linguistic tree query languages have been developed for interrogating treebanks (see [8] for a survey). One of these languages, called LPath, extends XPath [4] with extra navigational operators tailored to the needs of linguistic tree query [2].

The syntax of such languages is arcane, and we would need to provide a more accessible, high-level syntax for use by a non-specialised audience. For instance, a high-level query for the word *wind* used as a noun, `wind/N`, could be automatically translated to the LPath expression `//N[@lex="wind"]` (and then compiled into SQL for execution). The high-level query `"acquire QANTAS/NP-OBJ"` could be translated into the LPath expression `//_[@lex="acquire"] -> _[@lex="QANTAS"]\\NP-OBJ`.

A more serious issue is scalability. Treebanks typically contain millions of words; the Penn Treebank, for instance, has 4.5 million words [10]. The scale of data on the Web is several orders of magnitude larger again. We will explore the scalability of this approach to querying linguistic annotations using the Penn Treebank, using multiple copies when necessary in order to simulate larger data sizes.

3 Indexing Hierarchical Data in Databases

Many approaches to storing hierarchical data have been carefully analysed in recent years. In the past, relational databases were preferred to specialised semi-structured database approaches, since the relational formalisms were more mature and offered superior performance [14].

However, more recently, several features found in relational databases have been incorporated into native XML databases, and indexes have been designed specifically for XML data. Commercial relational databases such as Oracle and DB2 now support native XML storage and retrieval, albeit with widely varying indexing and query evaluation techniques. Oracle uses a hybrid approach to store XML within relational tables [9], while DB2 allows XML data to be stored natively and builds value and full text indexes over them [12]. Both offer varying levels of XQuery support, and a primitive XML datatype called XMLType which can be used across all queries.

Yet another native XML database is the eXist open source database. It builds three primary indexes on XML data: the structure, range and full-text indexes [11]. A structure index is similar to an inverted index of all nodes and attributes of XML documents along with their document and node ids. The node ids help in identifying hierarchical and sibling relationships without tree traversal. Range indexes permit comparisons based on typed values while full-text indexes support queries over sequences of words or tokens.

Hierarchical data can be decomposed into sets of relations and stored in a relational database. In our experiments, we store all the elements and attributes in a single node relation, as it is best suited for a diverse structure such as that present in annotated linguistic data. A common feature linking such a relational representation and the eXist database representation is the evaluation of path expressions by decomposing them into smaller components.

Each path expression is treated as sequence of elements interleaved with operators such as *parent*, *child*, *ancestor*, or other navigational constructs. These expressions are evaluated by converting the path sequence into one or more binary expressions involving a single operator. Indexes are used to search for matching elements on either side of the binary expression and the resulting sets are reduced using a join based on the operator. This join operation is termed the structural join and several optimisations have been proposed to improve the efficiency of such joins [1].

4 Experiments

In this section we describe the experimental setup used to evaluate performance of databases for linguistic queries. We ran the experiments on an Intel core 2 Duo 2.4 GHz processor with 2 GB of RAM, running openSUSE Linux 11.0. The task of choosing the right database system and optimising parameter settings for each of the experiments introduce multiple variables within the experiment. As our study attempts to highlight the scalability of particular systems rather than compare relative performance, we feel that fine tuned optimisations will not drastically change our observations on scalability. Instead we focus on scalability by varying the size of the datasets.

Annotated texts from the Wall Street Journal section of the Penn Treebank corpus were used in our experiments. This corpus contains around 50,000 sentences annotated with POS and syntactic tags. In order to study the variation of performance with the size of the datasets we either selected a subset of the corpus, or used multiple copies to simulate larger datasets.

Tests using the relational approach use the Oracle 11g Standard Edition, while the eXist XML database ver-1.2.2 is used in the native XML approach. The relational database schema contains a node relation storing all elements and attributes of the treebank. This schema is similar to the one used by LPath to query treebanks and more details can be found in Bird et al.'s work on LPath [2]. For the XML database approach we store each sentence as a separate XML document.

The LPath queries used during evaluation are listed in Table 3. These queries were converted to SQL for the relational database and into XQuery for the XML database. Some of these queries include highly selective nodes, while others search for commonly occurring terms. Queries 3–6 evaluate the effects of a simple join on nodes with varying selectivity. Queries 7 and 8 find the occurrences of words within the treebank irrespective of their POS tag. Other queries include features like scoping, edge alignment, and negation; all

Table 1: Query execution time in Native XML DB

	Dataset sizes					
	500	5k	∼25k	∼50k	∼100k	∼200k
1	.06	.51	**.99**	**1.99**	**4.59**	**16.29**
2	.01	.12	.60	.73	1.23	3.13
3	.08	.24	.95	1.90	5.80	20.58
4	.08	.11	.86	2.32	7.98	29.81
5	.01	.01	.50	1.01	1.99	2.63
6	.03	.05	.40	.87	7.20	23.62
7	.38	.83	**3.98**	**9.38**	**33.26**	**156.60**
8	.10	.63	**4.17**	**9.31**	**29.93**	**116.36**
9	.19	.82	2.16	3.73	31.24	97.72
10	.82	2.36	9.24	17.43	43.28	86.90
11	.17	.54	2.06	4.23	17.48	76.28
12	.64	3.20	**14.85**	**27.99**	**58.19**	**160.28**
13	.03	.17	1.19	2.43	10.45	37.95

Table 2: Query execution time in Relational DB

	Dataset sizes					
	500	5k	∼25k	∼50k	∼100k	∼200k
1	.10	.32	1.29	2.48	4.80	9.46
2	.07	.07	.08	.07	.07	.11
3	.08	.18	.63	1.20	2.35	4.94
4	.10	.35	1.47	**2.83**	**5.54**	**12.15**
5	.07	.08	.07	.07	.08	.13
6	.07	.09	.16	.24	.40	.13
7	.18	1.07	5.04	9.94	19.93	6.79
8	.07	.07	.08	.07	.07	.12
9	.09	.16	.49	**.90**	**1.65**	**12.33**
10	.08	.14	.41	.76	1.30	2.75
11	.10	.29	1.17	**2.15**	**4.12**	**12.11**
12	.08	.12	.30	.55	.90	2.09
13	.08	.08	.09	.10	.10	.18

features commonly found in linguistic queries. Adjacency is another common linguistic query operator and is represented by queries 12 and 13.

In the Oracle setup, the buffer cache and shared pool of the database were cleared after every query, but the first run always took the greatest time to execute. Subsequent queries had stable and repeatable query times. Table 2 lists the minimum time (in seconds) taken by each query over a sample of 3 runs. The eXist queries seemed to perform more uniformly between runs, but random slowdowns were observed in some cases. Each value in Table 1 corresponds to the minimum execution time (in seconds) of 3 consecutive runs of each query, on the eXist database. The minimum execution time was chosen instead of an average to avoid including random slowdown times in the measurement.

One of the main observations in the eXist experiment was that the rate of increase in execution time increases with dataset size; especially for larger datasets. For some queries, when we double the size of the dataset from 100k to 200k sentences, the time taken by eXist increases drastically; see the results for queries 1, 7, 8 and 12 in Table 1, where values of interest appear in boldface. However, these queries do not display such a trend in the Oracle setup. On

Table 3: Test Queries

	LPath query
1	`//NP`
2	`//PP_LOC_MNR`
3	`//NP/NP`
4	`//NP//NP`
5	`//RRC/PP_TMP`
6	`//VP/PP_TMP`
7	`//_[@lex=saw]`
8	`//_[@lex=rapprochement]`
9	`//VP{//VB-->NN}`
10	`//VP//NP$`
11	`//NP[not(//JJ)]`
12	`//NP=>NP`
13	`//ADVP=>ADJP`

further inspection, we can see that queries 7 and 8 probably involve simple index lookup. It is unclear why these queries exhibit such an increase in spite of using the full-text index in eXist. The poor scaling behaviour of Query 1 and 12 could be attributed to the low selectivity of NP.

Queries 6 and 9 are exceptions to the pattern mentioned above; their execution times grows by a factor of 8–10 between 50k and 100k sentence datasets, but drops to a factor of 3 for larger datasets. A plausible explanation for this behaviour could be the thrashing of memory, caused by intermediate object creations. Query 5 is very similar to query 6 in its construction but does not exhibit such a phenomenon as it is composed of high selectivity elements. Overall, the query times in eXist almost always seem to increase by a factor of 3–5 from 100k to 200k sentences.

A linear increase in time was observed in fewer Oracle tests when compared to eXist. High selectivity queries in Oracle displayed almost constant execution time, indicating optimal use of indexes. However, query 4, 9 and 11 (in boldface), show an increase in execution time with the size of the dataset.

5 Searching Linguistic Annotations Using an IR Engine

From our experiments we observe that the database approach using structural joins does not scale for queries containing low selectivity elements. To address this shortcoming we intend to evaluate systems where paths are indexed in their entirety and not as a combination of element pairs. One such approach has been proposed by Cooper et. al., where paths are inserted into an indexing data structure called Index Fabric [6]. Paths from the root to each of the leaf nodes of every tree are treated as string sequences and are inserted into the data structure.

Patricia tries form a core component of Index Fabric. They are unbalanced structures and not very efficient for main memory operations. Hence, in Index Fabric, access to different fragments of a trie is broken down into multiple layers. Pointers are created at each node to navigate to deeper tree fragments within lower

layers. A multi-layered approach balances the overall data-structure and results in a constant number of I/O look-ups for all searches. The Patricia trie is also known to use an aggressive key compression algorithm, making it a scalable architecture for large number of keys and for paths of varied lengths. The only drawback of this approach is that it suits queries where the paths are defined from the root to the leaves or for selected predefined paths, and not for arbitrary partial expressions.

An alternative approach – using a combination of IR and database systems – has been developed by Park et al. and is known as XIR. Here, paths are treated as sentences, and individual elements in a path form the words within the sentence [13]. They identify paths as representations of the document schema and hence are indexed independently from the data, which is considered to occupy only the leaf nodes. Each unique path is stored in a path table, while individual elements comprising the paths are indexed in an inverted index. Every unique element appears in the inverted index with a postings list containing information regarding the element's occurrence in different paths, an offset indicating its relative position within the path and the total length of the path. The data and element information is stored in a separate table with document and node identifiers.

The two key contributions of the XIR system include the concept of using inverted indexes to search path expressions and the conversion of path queries into equivalent IR style proximity searches. For instance, a query such as `VP/NP/NNP`, which searches for a singular proper noun phrase (`NNP`) in the specified hierarchical relationship with a noun phrase (`NP`) and verb phrase (`VP`), could be converted into an IR query where the *near* operator specifies the proximity relation: `VP near(1) NP near(1) NNP`. Similarly, a descendent query `VP//NP`, could be rewritten as `VP` near(∞) `NP`, indicating that the second element can appear anywhere after the first element in the path string. Once the set of paths is known from the inverted index, a select query on the data and element information table retrieves the final results.

For linguistic queries, sequential navigation is as significant as hierarchical navigation. We expect that by extending the XIR approach, we could create independent indexes for hierarchical and sequential relationships in a document. Queries containing hierarchical and sequential expressions would be converted into appropriate proximity queries and the resulting nodes could be reduced by a join operation. This method would essentially reduce the number and cardinality of joins, as they would occur only when the expression changes from path to sequence or vice-versa and not at every element in the expression.

As a part of ongoing work in this area, we would also like to compare the performance of such a system with a pure database implementation containing an indexing algorithm similar to Index Fabric.

Acknowledgements

We gratefully acknowledge the support of Dr A. Kumaran and Microsoft Research India.

References

[1] Shurug Al-Khalifa, H. V. Jagadish, Nick Koudas, Jignesh M. Patel, Divesh Srivastava and Yuqing Wu. Structural joins: a primitive for efficient XML query pattern matching. In *ICDE '02: Proceedings of the 18th International Conference on Data Engineering*, page 141, Washington, DC, USA, 2002. IEEE Computer Society.

[2] Steven Bird, Yi Chen, Susan B. Davidson, Haejoong Lee and Yifeng Zheng. Designing and evaluating an XPath dialect for linguistic queries. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, page 52, Washington, DC, USA, 2006. IEEE Computer Society.

[3] Steven Bird and Jonathan Harrington (editors). *Speech Communication: Special Issue on Speech Annotation and Corpus Tools*, Volume 33 (1–2). Elsevier, 2001.

[4] James Clark and Steve DeRose. *XML Path language (XPath)*. W3C, 1999. `http://www.w3.org/TR/xpath`.

[5] Stephen Clark and James R. Curran. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computational Linguistics*, Volume 33, Number 4, pages 493–552, 2007.

[6] Brian Cooper, Neal Sample, Michael J. Franklin, Gisli R. Hjaltason and Moshe Shadmon. A fast index for semistructured data. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, pages 341–350, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[7] Stephan Kepser. Finite structure query: a tool for querying syntactically annotated corpora. In *Proceedings of the Tenth Conference of the European Chapter of the Association for Computational Linguistics*, pages 179–186, 2003.

[8] Catherine Lai and Steven Bird. Querying and updating treebanks: A critical survey and requirements analysis. In *Proceedings of the Australasian Language Technology Workshop*, pages 139–146, 2004.

[9] Zhen Hua Liu, Muralidhar Krishnaprasad and Vikas Arora. Native XQuery processing in oracle XMLDB. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 828–833, New York, NY, USA, 2005. ACM.

[10] Mitchell P. Marcus, Beatrice Santorini and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, Volume 19, Number 2, pages 313–30, 1993.

[11] Wolfgang Meier. *Web, Web-Services, and Database Systems*, Volume Volume 2593/2008, Chapter eXist: an open source native XML database, pages 169–183. Springer Berlin / Heidelberg, 2008.

[12] Matthias Nicola and Bert van der Linden. Native XML support in DB2 universal database. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 1164–1174. VLDB Endowment, 2005.

[13] Young-Ho Park, Kyu-Young Whang, Byung Suk Lee and Wook-Shin Han. Efficient evaluation of partial match queries for XML documents using information retrieval techniques. *Database Systems for Advanced Applications*, pages 95–112, 2005.

[14] Jayavel Shanmugasundaram, Kristin Tufte, Chun Zhang, Gang He, David J. DeWitt and Jeffrey F. Naughton. Relational databases for querying XML documents: limitations and opportunities. In *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, pages 302–314, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

Using Collaboratively Constructed Document Collections to Simulate Real-World Object Comparisons

Karl Grieser,[♡] *Timothy Baldwin,*[♠] *Fabian Bohnert*[♣] and *Liz Sonenberg*[♡]

♡ DIS
University of Melbourne
VIC 3010 Australia
kgrieser@csse.unimelb.edu.au
l.sonenberg@unimelb.edu.au

♠ CSSE
University of Melbourne
VIC 3010 Australia
tim@csse.unimelb.edu.au

♣ Faculty of IT
Monash University
VIC 3800 Australia
fabianb@csse.monash.edu.au

Abstract *While the layout of a museum exhibition is largely prescribed by the curator, visitors to museums view connections between exhibits in ways unique to themselves. With the assistance of a large-scale survey of museum visitors we identify that the view taken by museum visitors of a collection of exhibits can be represented by similarity over documents associated with each exhibit. We show that even when using a basic document similarity measure there is a correlation between document similarity and visitors' judgements of relatedness of exhibits aligned to these documents.*

Keywords User Studies Involving Documents, Web Documents, Cognitive Aspects of Documents.

1 Introduction

Recently there has been a move towards providing visitors to museums and Cultural Heritage (CH) spaces with personalised tours. These tours can be created explicitly by a visitor prior to entering the collection, or tailored to a visitor while browsing a collection. In order to create a dynamic tour for a given visitor, there is the need to (1) model a visitor's preferences (Zukerman and Albrecht [14]), and (2) have knowledge about the content of individual exhibits and connections between pairs of exhibits (Aroyo et al. [1], Bohnert et al. [3], Cox et al. [4], and Grieser et al. [7]). The focus of this paper is on the second of these requirements, using web documents to represent museum exhibits, and document similarity to model similarities between them.

Museum exhibitions are generally designed around a common theme (e.g., *Melbourne* or *marine life*), and professionally curated so that exhibits are organised in a coherent fashion relevant to that theme with closely-related objects in close physical proximity of each other (e.g., artefacts from the same era or of the same function are often presented together). The task of tour personalisation can be seen as one of matching the interests of a visitor to the themes represented in the museum.

Proceedings of the 13th Australasian Document Computing Symposium, Hobart, Australia, 8 December 2008.

However, visitors to a museum or cultural heritage site can categorise the museum space in a way particular to the context of their visit (e.g., preferring to visit more tactile exhibits to entertain small children, or choosing to visit all exhibits from a particular location or era, irrespective of theme). That is, they often have their own opinions on the degree of relatedness of exhibits, independent of the themed design of a gallery or exhibition.

While various computational methods have been developed to identify relationships between documents or words (e.g., Rubenstein and Goodenough [13] and Ponzetto and Strübe [12]), there is currently no standard method of identifying the manner in which people view the relationships between real-world objects. The objective of this research is to test the portability of document-based similarity methods to the task of estimating museum exhibit relatedness. In particular, we map each of 41 exhibits from Melbourne Museum to the most closely associated Wikipedia article, and perform a simple pairwise cosine similarity calculation over the weighted document vector for each document.

We compare our calculated document similarities against two data sets: (1) real-world relatedness estimates of pairs of exhibits in Melbourne Museum provided by over 500 museum visitors, and (2) a calculation of the physical distance between each pairing of exhibits. In the first instance, we calculate how well document similarity models the museum visitors' notions of exhibit relatedness. In the second instance, we determine how closely our similarity estimates mimics the physical layout of the museum. We also compare physical distance with the visitors' ratings to gauge how faithful the ratings are to the prescriptive theming of the museum space.

2 Related Research

Cultural Heritage spaces such as historical sites and museums are providing greater access to their collections through mobile computing (e.g., Benelli et al. [2] and Oppermann et al. [10]) and the web (Aroyo

et al. [1]). This has enabled museums to reach wider audiences and to better communicate the importance of their collections. The personalisation of content through digital collections has been a major focus of many CH projects, and multiple methods have been used to tailor content or tours to the visitor (Aroyo et at. [1], Benelli et al. [2], and Cox et al. [4]). Previous approaches such as the ones in Aroyo et al. [1] and Cox et al. [4] used the attributes of previously rated exhibits to identify other exhibits that the visitor may find interesting. Grieser et al. [7], on the other hand, used common attributes of exhibits in the current visit to predict future exhibits the visitor may visit, while Bohnert et al. [3] used the amount of time a visitor spent viewing exhibits to infer a visitor's interests and pathways. The content-based models explored in Grieser et al. [7] and the collaborative models proposed by Bohnert et al. [3] have the advantage of being non-intrusive, as they do not require explicit exhibit ratings. However, these techniques suffer from the so-called cold-start problem in the initial stages of a visit.

The identification of reasons for visitors finding commonality between exhibits is a key step in personalising a tour. Previous studies have used common attributes to align exhibits and identify similarities (e.g., same artist, same style of jewelery, as used in Cox et al. [4]). This has lead to the use of ontological frameworks as a basis for these comparisons (e.g., The Getty AAT, Iconclass, and the CIDOC Conceptual Reference Model). For CH sites such as art galleries where all exhibits have the same attributes, this method is appropriate. However, for CH sites that have exhibits of differing backgrounds (e.g., natural history museums or national parks) this method does not adequately account for the diversity of the exhibit structure. We aim to address this gap by using an alternative semi-structured data source that is able to identify relationships between concepts within its hierarchy.

Estes [5] showed that for concepts that do not have a common conceptual frame (or physical structure), people relate concepts using a process of integration. An integrative relationship is the interaction that occurs between two concepts. This is different from attributive comparison, where concept attributes are compared in order to determine similarity. This indicates that for CH sites with diverse collections, highly structured data sources and ontologies are unable to sufficiently identify the interactions between exhibits that visitors will make when considering them. Their key failure is that they do not simulate the thought process that the average museum visitor will go through (often an integrative relationship), but rather focus on the organisational hierarchy designed by the collection's curator.

For this study, we will use a non-expert data source that provides relationships between highly different entities, and that is able to represent the information at a common visitor level: Wikipedia. In recent years, Wikipedia has been used increasingly in document processing tasks, due to its sheer size, multilinguality, and domain diversity. Extensive conceptual similarity experiments have been performed in other studies, such as the ones discussed in Gabrilovich and Markovitch [6] and Milne et al. [9]). Particularly interesting is the category hierarchy, as each article must be a member of at least one category. This hierarchy has been investigated by Ponzetto and Strübe [11, 12] as a parallel to other existing hierarchies such as WordNet. In Wikipedia, the articles are created with the intention of being understandable to all users, and even its place in the category hierarchy is reached through discussion and consensus, meaning that an article's content and its organisation is designed to make sense to the majority of people viewing it. This collaboratively constructed social nature of Wikipedia (Mathes [8]) is the reason for choosing it as a data source for this research.

3 Museum Visitor Survey

For the purposes of this research, we identified 41 exhibits from Melbourne Museum which could readily be aligned with Wikipedia documents as per Wikipedia guidelines[1] – some trivially as named entities (e.g., *Phar Lap*), others less convincingly via more general articles (e.g., *gold mining* for a diorama of a Ballarat gold mine). We then designed a web survey drawing heavily on the psycholinguistic research of Rubenstein and Goodenough [13] on lexical similarity. In their research, subjects were presented with a standardised set of word pairs (presented in random order), and asked to rate their relatedness on a discrete scale of 0 to 4. In our case, rather than words, we present the subject with images of two exhibits from Melbourne Museum, and ask them to rate their relatedness on a scale of 0 to 4, keeping with the standardised scale defined by Rubenstein and Goodenough [13]. We also asked for a justification of the rating.

Subjects were presented with 15 exhibit pairs in random order, 3 of which were common to all respondents and the remaining 12 of which were chosen randomly. The images were presented adjacent to each other in a web browser, again in randomised order.

In order to ensure that the ratings were relative to actual visits to Melbourne Museum, we targeted Museum Victoria members exclusively, and asked respondents to indicate how frequently they had visited the museum in the preceding 12 months (as well as other demographic and profiling data which is irrelevant to this current research). We received over 500 responses over a three-week period, and recorded at least one rating for every exhibit pair. Of these, we filtered out a small number where the same relatedness value was given for all 15 exhibit pairs. We then calculated the mean of the relatedness values for a given exhibit pair, and use this as our gold-standard relatedness data.

[1] `http://en.wikipedia.org/wiki/Wikipedia:Neutral_point_of_view`

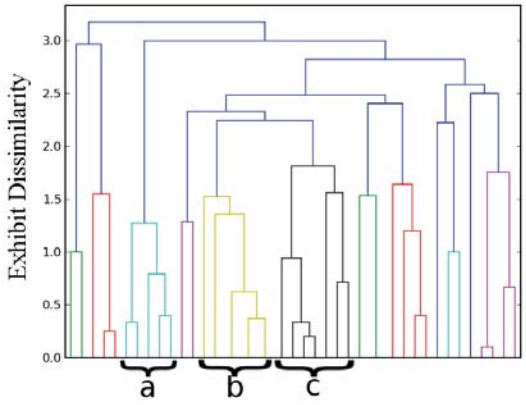


Figure 1: Clustering of exhibits based on the pairwise relatedness ratings

In order to carry out preliminary analysis of the ratings from the museum members, we performed agglomerative hierarchical clustering over the survey results, and identified distinct groupings of exhibits. The hierarchy created from the relatedness scores (translated into *dis*similarity ratings by subtracting from the maximum relatedness value, i.e., 4) is shown in Figure 1.

Encouragingly, we found that the results followed broad thematic boundaries, with cluster (a) revolving around geology, cluster (b) revolving around birds and trees, and cluster (c) revolving around prehistoric animals and fossils, for example. However, in many cases, these clusters do not correspond to the thematic/physical layout present in the museum. We will discuss this further in Section 5.

4 Exhibit Comparison

The arrangement of exhibits within a museum exhibition is often planned around a central theme. This can be a rather broad theme such as *science*, or a more specific one such as documenting the growth of a city over time. In the more specific case, the arrangement of the exhibits is key to an exhibition's interpretation. However, this interpretation may not be the interpretation that a visitor considers when identifying relationships between exhibits.

At this early stage of the research, we calculate simple cosine similarity between the term vectors of the Wikipedia documents to estimate the relatedness of a given pairing of exhibits, weighting terms with a basic tf·idf model. In addition to the document similarity model, we explore the hypothesis that physical walking distance between exhibits is inversely proportional to their degree of relatedness, i.e., closely-related exhibits should be in close physical proximity, and less-related exhibits should be further apart from each other. This derives simply from the careful theming of the museum space by curatorial staff. We calculate the physical distance between exhibits via an SVG image of the museum space, mapped onto a graph structure which preserves the physical layout of the museum (i.e., preventing paths from passing through walls or ceilings).

Pairing of methods	ρ-value	p-value
Human & Physical	$+0.196$	1.5×10^{-8}
Human & Document	$+0.157$	6.6×10^{-6}
Physical & Document	$+0.038$	2.3×10^{-1}

Table 1: Two-tailed Pearson correlation and p-value between the different methods (Human Judgements, Physical Distance and Document Similarity)

Identifying which of these measures most closely aligns with the ratings provided in the survey may provide an indication of which viewpoint the average visitor takes: the expert view of the curator, or the more common interpretation supported by the socially-constructed documents.

5 Results and Discussion

We evaluated the relative "fit" between each pairing of human judgements, walking distance and document similarity via the ρ-value of a two-tailed Pearson correlation test over the corresponding lists of exhibit pairings.[2] The results are presented in Table 1.

The highest correlation (at level of statistical significance, $p \ll 0.01$) was obtained for the pairing of physical distance between exhibits with the human judgements. The most obvious explanation for this result is that the visitors' view of exhibits mirrors that intended by the curators to a certain degree. Preliminary analysis of the justifications for relatedness from the web survey supports this observation, with a number of respondents citing physical proximity as the reason for a higher relatedness value.

The second highest correlation was achieved for the pairing of human judgements and document similarities (again at a level of statistical significance), indicating that our document similarity model was moderately successful at capturing exhibit relatedness, despite our relatively simple approach. Note that as the documents were sourced from Wikipedia, there is nothing specific to Melbourne Museum in them, and no indication of how the exhibits are interpreted in the museum space. In this sense, the results are highly encouraging.

We get very low correlation between the physical distance and document similarity (not at a level of statistical significance). When combined with the above two results, this indicates that the document similarity model is modelling something removed from the physical layout of the museum, and yet agrees with our human subjects. Hence, it appears to be picking up on cross-gallery relatedness, and complementing the physical distance model.

[2]Note that we reverse the sign of the ρ-value in the case that we are comparing a similarity with a distance (i.e., similarity vs. dissimilarity).

It is unclear at this point exactly what the degree of influence of the museum layout was on the survey responses. We intend to carry out further analysis of the survey data to clarify this point.

That a basic document similarity measure over a single set of documents could achieve these results is highly encouraging. Clearly there is much more that can be done. Areas of future research we are interested in are analysing cross-article links and the category hierarchy in Wikipedia, and combining these with the document similarity model (inspired in part by the work of Ponzetto and Strübe [11]). We are also interested in exploring a broader range of term weighting, feature selection, and similarity metrics in the document similarity model, as well as different document sets (including documents from the Melbourne Museum website). We anticipate that this will provide a more thorough picture of the way in which museum visitors conceptualise relationships between exhibits.

6 Conclusions

When identifying relationships between exhibits within a museum, previous methods have used highly structured methods often created by a curator highly familiar with the collection. We have proposed a document similarity model which makes use of content authored by non-experts to overcome the curator-centric design as well as to identify associations between exhibits.

Through comparison of the conceptual design of the museum in its physical layout (based on exhibit locality within the museum), the content of exhibits (represented by collaboratively-constructed documents relating to the exhibit content), and the ratings of museum visitors (obtained through a web survey), we have shown the following: (1) visitors' impressions of exhibit relatedness is affected by the physical layout of the museum, although less than might have been expected; (2) a basic document similarity model is surprisingly effective at capturing visitor ratings of exhibit relatedness, in a manner largely orthogonal to the relatedness derived from the museum layout.

Acknowledgements Thanks to the staff at Melbourne Museum for assistance and access to their members, and in particular Carolyn Meehan for helping with the survey design and distribution.

References

[1] L Aroyo, R Brussee, L Rutledge, P Gorgels, N Stash and Y Wang. Personalized museum experience: The Rijksmuseum use case. In *Proceedings of Museums and the Web*, page Online proceedings, San Francisco, United States, 2007.

[2] G Benelli, A Bianchi, P Marti, D Sennati and E Not. HIPS: Hyper-interaction within physical space. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pages 1077–1078, Florence, Italy, 1999.

[3] F Bohnert, I Zukerman, S Berkovsky, T Baldwin and L Sonenberg. Using collaborative models to adaptively predict visitor locations in museums. In *Proceedings of the 5th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 42–51, Hanover, Germany, 2008.

[4] R Cox, M O'Donnell and J Oberlander. Dynamic versus static hypermedia in museum education: an evaluation of ILEX, the intelligent labelling explorer. In *Proceedings of the Artificial Intelligence in Education Conference*, pages 181–188, Le Mans, France, 1999.

[5] Z Estes. A tale of two similarities: comparison and integration of conceptual combination. *Cognitive Science*, Volume 27, Number 6, pages 911–921, 2003.

[6] E Gabrilovich and S Markovitch. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 1606–1611, Hyderabad, India, 2007.

[7] K Grieser, T Baldwin and S Bird. Dynamic path prediction and recommendation in a museum environment. In *Proceedings of the Workshop on Language for Cultural Hetitage Data*, pages 49–56, Prague, Czech Republic, 2007.

[8] A Mathes. *Folksonomies - Cooperative Classification and Communication Through Shared Metadata*. University of Illinois Urbana-Champaign, `http://adammathes.com/academic/computer-mediated-communication/folksonomies.pdf`, 2004. Unpublished Paper, Retrieved on 11 Aug, 2008.

[9] D N Milne, I H Witten and D M Nichols. A knowledge-based search engine powered by Wikipedia. In *Proceedings of the 16th ACM conference on Conference on Information and Knowledge Management*, pages 445–454, Lisbon, Portugal, 2007.

[10] R Oppermann and M Specht. A context-sensitive nomadic information system as an exhibition guide. In *Proceedings of the Handheld and Ubiquitous Computing Second International Symposium*, pages 127–142, Bristol, United Kingdom, 2000.

[11] S P Ponzetto and M Strübe. An API for measuring the relatedness of words in Wikipedia. In *Proceedings of the ACL Demo and Poster Sessions*, pages 49–52, Prague, Czech Republic, 2007.

[12] S P Ponzetto and M Strübe. Deriving a large scale taxonomy from Wikipedia. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 1440–1445, Vancouver, Canada, 2007.

[13] H Rubenstein and J B Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, Volume 8, Number 10, pages 627–633, 1965.

[14] I Zukerman and D.W. Albrecht. Predictive statistical models for user modeling. *User Modeling and User-Adapted Interaction*, Volume 11, Number 1-2, pages 5–18, 2001.