Melbourne University 2004: Terabyte and Web Tracks

Vo Ngoc Anh *Alistair Moffat*

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

{*vo,alistair*}*@cs.mu.oz.au*

Abstract: *The University of Melbourne carried out experiments in the Terabyte and Web tracks of TREC 2004. We applied a further variant of our impact-based retrieval approach by integrating evidence from text content, anchor text, URL depth, and link structure into the process of ranking documents, working toward a retrieval system that handles equally well all of the four query types employed in these two tracks. That is, we sought to avoid special techniques, and did not apply any explicit or implicit query classifiers. The system was designed to be scalable and efficient.*

1 Introduction

The University of Melbourne carried out experiments in the Terabyte and Web tracks of TREC 2004. In the Terabyte Track we used title-only queries, and considered the task to be one of *content search* on web data. In the Web Track, the 225 mixed queries of different types (topic distillation, homepage finding, and named page finding) were processed in our retrieval system without any modifications.

The goals of our investigation in 2004 included:

- to create a system that handles equally well queries of any type, without classification being required;
- to develop further our retrieval approach based on qualitative document-centric impacts [Anh and Moffat, 2003, Anh, 2004], by integrating evidence from the content layout and incoming anchor text;
- to investigate the possibility of combining evidence from URL depth and link structure as document parameters into the retrieval scores established by the impact-based retrieval scheme; and
- to demonstrate a scalable system that operates with large data collections at high throughput rates for both query processing and indexing.

For the Web task, only effectiveness results are presented in this report. On the other hand, the large size of the `GOV2` collection makes it an attractive basis for a discussion of efficiency concerns, as well as effectiveness outcomes.

Our retrieval process is briefly described as follows. During index construction, term statistics from documents and their incoming anchors are employed to assign an integral document-centric impact score to each valid pairing of terms and documents. The triples (term, document, impact) are stored in an impact-sorted inverted index for querying. In addition, document URL depths and authority scores are computed, and similarly quantized to small integers. These weights are also included as components in the index. During query evaluation, the impacts, the URL weights, and

the authority weights are combined in an efficient way to produce a ranking of documents with respect to a query.

The rest of this paper is organized as follows. Section 2 briefly reviews the impact-based ranking scheme for text documents. Next, Section 3 describes the experimental options for combining web-specific evidence from anchor text, URL content, and link structures, with the impact ranking. Section 4 lists and then discusses the results returned from NIST for the Web Track. Section 5 introduces the measures taken to ensure a scalable and efficient implementation, and is followed by Section 6, which analyzes the efficiency and effectiveness of our runs for the Terabyte Track.

2 Impact-Based Document Ranking

This section describes the formulation of document-centric impacts and the general scheme for using them in ranking.

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

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

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

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

where each of the document or query impacts is an integer between 1 and k inclusive, with k a predefined parameter. In this paper, $k = 12$ is used. Document impacts and query impacts might be defined differently, which is why different notion has been used for them. If a term $t \in q$ does not have a corresponding dimension in the n-space, it is ignored.

A two-phase process is used to assign the document impact values $\omega_{d,t}$ for the terms t that appear in d. In the first *sorting* phase, documents are parsed and frequency statistics gathered. Quantities accumulated include the *term frequency* $f_{d,t}$ (the number of times term t appears in document d), and the *document frequency* f_t (the number of documents that contain t). At the end of the sorting phase, the list of distinct terms of each document d is rearranged into decreasing order of the term frequencies $f_{d,t}$, with ties broken by the inverse document frequency $1/f_t$ [Anh and Moffat, 2002b].

In the second *mapping* phase of assigning impacts, the ordered list of terms for each document d is partitioned into k consecutive non-overlapping segments numbered from k down to 1 so that the number of elements x_i in segment i is

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

where

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

and n_d is the number of elements in the list being partitioned – in other words, the number of distinct terms in document d. Finally, each term t is assigned as its impact the index i of the partition it falls in. The result of this step is that a small number of terms are deemed to be of high impact in the document, and a much greater number of terms are deemed to be of low impact.

To complete the process of index construction, the impacts associated with each term are gathered, and sorted into decreasing order of impact score. That is, the inverted list for each distinct term is ordered by decreasing impact, and because there are only k possible different impact scores for each term, can be thought of as a sequence of k (or fewer) blocks, each of which contains document numbers d in which that term has the same impact. The result is an *impact-sorted index* that allows very fast query processing.

The query term impacts for the terms that appear in q are defined during query processing time, and are also formed in two phases, where the first phase is similar to that of the document impacts. Due to the short average query length, the second phase is simply the mapping of the ordered list of terms into the list of integer from k' to 1, where k' is either k or the number of distinct terms in q, whichever is larger. This method of assigning query term impacts is not the best in terms of retrieval effectiveness [Anh, 2004], but is used here because of its simplicity.

Once the impact vectors for a document d and a query q have been formed, they are employed to define $i_{d,q}$, the *impact score* of d with respect to q:

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

In the standard implementation without considering web features, the impact score $i_{d,q}$ is employed as the similarity score $S(d, q)$, that is:

$$S(d, q) = i_{d,q} \,,$$

and serves as the sort key for the ranking process. The similarity score is defined differently when the features of web documents are taken into account, as discussed shortly.

In practical implementations an incomplete version of the $i_{d,q}$ computation is often performed, with only a partial evaluation of the inner product computed, using additional heuristics that are generically referred to as *dynamic pruning* techniques. Anh and Moffat [2002a] describe several pruning methods that can be used with impact-based similarity scores, and the same methods are applied in all of the experiments reported here.

3 Adding Web-Based Evidence

We now consider the types of evidence that can be used in the ranking of web documents, and the ways they can be combined with the impact-based ranking scheme described in Section 2. Four kinds of information can be used in calculating the similarity score between a document d and a query q:

- The appearance of terms in the content part of document d, query q, and in the whole collection. For this type of evidence, we use the statistics $f_{d,t}$, the term frequency of t in d; $f_{q,t}$, the term frequency of t in q; and f_t, the document frequency of t in the collection.

- The appearance of terms in the incoming anchor text of d. The only parameter used is $a_{d,t}$, the *within-anchor frequency* of t in the incoming anchor text of d. This evidence requires a non-trivial amount of preparatory cross-document computation which is not included in any of the timings given here.

- The depth of the URL of d, on the assumption that documents with near the root of a web site are more important than documents deep in subdirectories.

- The link structure of the documents in the collection. The information extracted from the collection is the set of links $d_i \rightarrow d_j$ showing that document d_i contains at least one link to document d_j. As with the incoming anchor text, we suppose that the link information is already available without needing to be computed.

In combining these four types of evidence, the main principle is to keep query processing costs low, by performing as much as possible of the computation during indexing. It is clear that the within-anchor frequencies can be combined with the within-document frequencies when defining document-centric impacts by integrating them during the sorting phase. On the other

hand, the URL depth and link information are invariant for all the terms in a document, so it makes more sense to include these factors separately. We decided to preprocess the URL depth and link information and store them as document-dependent factors. Each of these factors are quantized to integers in range from 1 to 256 to facilitate compact storage.

Balancing the term and anchor frequencies in defining impacts We considered two options for combining the within-anchor frequencies $a_{d,t}$ with the within-document frequencies $f_{d,t}$:

- `Content`: Impacts are defined exactly as described in Section 2.
- `Content+Anchor`: In the standard `Content` implementation, the sorting phase that forms the document impacts uses $f_{d,t}$ values as the primary sort key. In this variant, we combine $f_{d,t}$ with $a_{d,t}$ by supposing that they are equally important, and as the primary sort key employ

$$F_{d,t} = (1 + f_{d,t}) \cdot (1 + a_{d,t}) \ .$$

  Because we are only interested in the relative ordering of terms within the document, taking the product in this way is equivalent to taking the sum of the logarithms of the $(1 + f_{d,t})$ and $(1 + a_{d,t})$ values. Each of the two frequencies contributes separately to the overall score $F_{d,t}$ for term t, with a diminishing return for large frequency values. The set of scores $\{F_{d,t} \mid t \in d\}$ for document d is then ordered into decreasing order, with ties broken by the value of $1/f_t$, and an assignment of integer impacts is carried out, as described in Section 2.

Quantizing URL depth The URL depth is derived for each document and the resulting list of distinct URL depths for the collection is sorted into increasing order. The first 255 elements of this list – typically depths 1, 2, and so on – are assigned the integer weights from 256 down to 2. Any remaining depths – which must be at least 256 – are assigned weight 1. At the end of this process, each document d is assigned a *URL weight* u_d corresponding to its URL depth.

Preprocessing the link structure We preprocess the link structure to extract an authority score for each document using the standard pagerank algorithm of Brin and Page [1998]. At the completion of this step, each document is assigned an *authority score* s_d which is a real number between 0 and 1. Hub scores are not calculated.

The population of authority scores s_d is then transformed to a list of integral *authority weights* a_d ranging from 1 to 256, using a logarithmic transformation and quantization defined as

$$a_d = \left\lfloor 256 \cdot \frac{L \cdot \log_B(s_d/L)}{U - L + \epsilon} \right\rfloor + 1 \, ,$$

where L and U are the minimum and maximum authority scores over the whole collection, ϵ is a small positive constant, and B is computed as

$$B = \left(\frac{U}{L}\right)^{L/(U-L)} \ .$$

Witten et al. [1999] describe other examples of this type of transformation.

Integration during query processing For this year's experiments we took the view that the document-centric impacts alone (that is, not including the URL and authority weights) could be used to define the list of top r answers, where r is 1,000 for the web task, and either 10,000 or 20 for the Terabyte Track. The URL weights and authority weights are then employed to re-rank the elements of that answer list.

Query evaluation proceeds in two stages. In the first stage, the standard impact-based ranking scheme described in Section 2 is applied to the chosen impacts (`Content` or `Content+Anchor`) to determine a list of the top r documents. At the end of this stage, each document d is assigned an impact score $i_{d,q}$, with the list of r documents provided in decreasing order of these scores.

In the second stage of query evaluation, the impact scores $i_{d,t}$, URL weights u_d, and authority weights a_d for the selected subset of documents are integrated in a number of different ways (listed shortly) to produce the final similarity scores for the query. The list of top r documents is then re-sorted in decreasing order of the new combined scores, and presented as the final answers.

The different methods to combine impact scores $i_{d,q}$ with URL weight u_d and authority weight a_d are:

- `Method_A`: Using impact scores only. This is the baseline method, where the URL and authority weights are ignored. That is, the final similarity scores $S(d, q)$ are defined as the corresponding impact scores $i_{d,q}$. However, it should be noted that there are two variants of impacts, as discussed at the beginning of this section.

- `Method_B`: Adding impact scores and scaled URL weights. In this option, the authority weights are ignored. The final similarity score for a document d is defined as

$$S(d, q) = i_d + u_d \cdot (\max\{i_d\} / \max\{u_d\}) \,,$$

  where $\max i_d$ and $\max u_d$ are the maximum value of i_d and u_d, respectively, computed locally within the subset of r documents selected by the first phase.

- `Method_C`: Adding impact scores and scaled authority weights. Here, the URL weights are ignored. The final ranking score for d is computed as the weighted sum of its impact score (with a weight of 2) and authority score (with a weight of 1):

$$S(d, q) = 2 \cdot i_{d,q} + a_d \,.$$

- `Method_D`: Using authority weights to break ties. There is no re-computation of similarity scores $S(d, q)$. Rather, the r top documents are inspected for ties, and the authority scores a_d used to break them.

- `Method_E`: Combining all available evidence. The final similarity score for d is defined as

$$S(d, q) = 2 \cdot (\max\{i_d\} \cdot \max\{u_d\} - (\max\{i_d\} - i_d) \cdot (\max\{u_d\} - u_d)) + a_d \,.$$

  That is, the impact score and URL weight are uniformly combined, and then that combined score is assigned a weight of 2 and added to the authority weight.

- `Method_F`: Sorting based on all evidence. The list of r documents is sorted in decreasing order by a key that is defined as in the case of `Method_B` :

$$i_d + u_d \cdot (\max\{i_d\} / \max\{u_d\})$$

  with ties broken by a_d.

4 Performance in the Web Track

This section reports our runs for the Web Track. The listed effectiveness scores are as advised by the NIST email notifications.

Run tag	Type of impacts	Integration method
MU04web5	Content	Method_A
MU04web2	Content	Method_B
MU04web3	Content	Method_C
MU04web4	Content	Method_E
MU04web1	Content	Method_F

Table 1: *Parameters used in the web runs.*

Experimental settings Five runs were submitted for the Web Track. Each used the collection .GOV and 225 mixed queries, as supplied by the track organizers. The parameters of the runs are listed in Table 1.

Effectiveness The performance of our five runs is summarized in Figure 1. The figure shows that combining URL and authority weights with the impact scores improves effectiveness for the topic distillation and homepage finding, but not for the named page finding task. For the named page finding task, the baseline approach is the best choice (of our runs) in terms of average precision.

Run MU04web1, which combines evidence from all of content, anchor text, URL depth, and link structure using Method_F, performs best. It gives the best results for the distillation and homepage finding task, and reasonable effectiveness for the named page finding task. This combination should be considered to be the best achievable with our approach at this time.

Compared to other groups, the MU04web1 run was around the median of the best runs from each group for the homepage finding and distillation tasks, but was below the median for the named page and the mixed query tasks. In that sense, we have not achieved our goal of excellent performance, and it may well be that the difference between what we have attained and the higher performance of other systems is a consequence of our integration of several different types of evidence into a single impact score.

5 Scalable and Parallel Processing

The main challenge for the Terabyte Track this year was the large size of the input document collection. To manage this volume of data, the retrieval mechanism must be scalable, effective, and fast in terms of both indexing and querying. This section reports the techniques we used to handle this large collection.

Scalable indexing We indexed the collection by dividing it in to manageable sub-collections, indexing them separately, and then merging the partial indexes. If required, the merging process can be performed in a hierarchy of levels, depending on the number of sub-collections and the number of files the system is allowed to open. In our experiments, we used a 10-way merge, and limited the sub-collection size to 2 GB or 100,000 documents. The merging process can run in an interleaved fashion if restricting the number of temporary index files is important.

Construction of an impact-sorted index consists of three components: primary indexing (the total time spent indexing sub-collections); merging (the total time of the merge sequence); and sorting (the time required to impact-sort the index).

Scalable querying When the collection is large, it may not be feasible to hold the vocabulary in main memory. It might also be necessary to allow for inverted lists that are longer than the available memory. To avoid the first problem, we employed the standard technique of blocking the

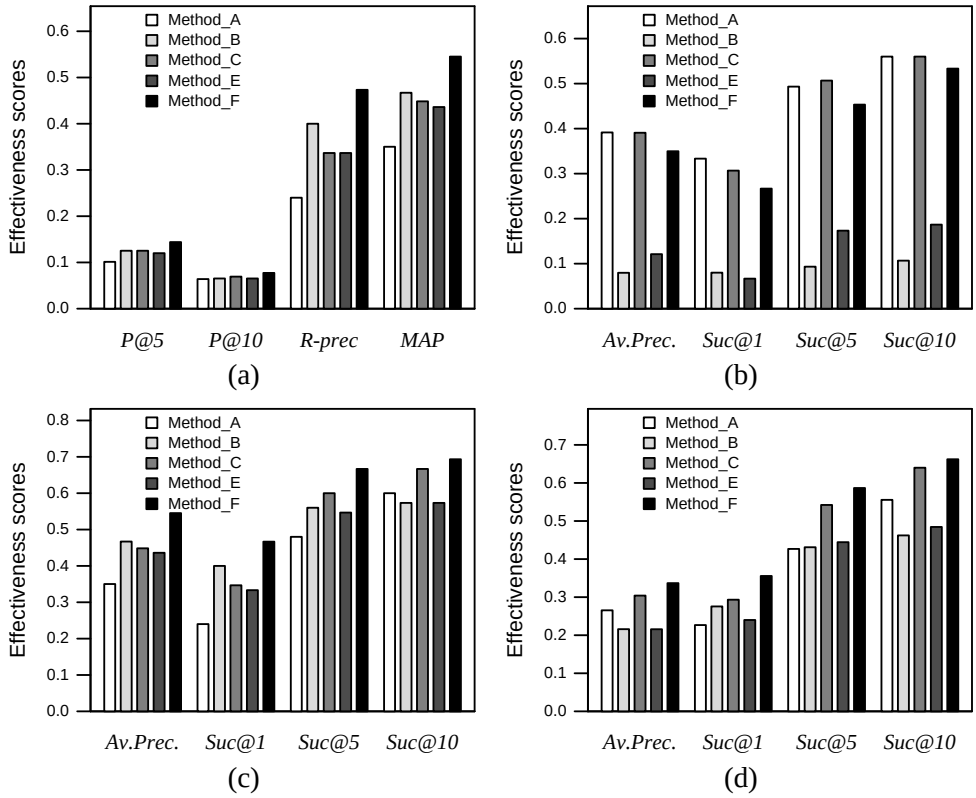

Figure 1: *Performance of the runs.* Retrieval effectiveness for: (a) the topic distillation queries; (b) the named page finding queries; (c) the homepage finding queries; and (d) all 225 mixed queries. Note that the metrics used for topic distillation differ from the metrics used in the other three quadrants.

lexicon [Witten et al., 1999]. To address the latter problem, we process inverted lists sequentially using fixed-length buffers.

Another potential problem with querying large collections that we have not yet rectified in our implementation is the use of accumulators. Currently, a 4-byte accumulator is used for each document of the collection, regardless of whether it is assigned a value. With 1 GB of memory for accumulators, that means an upper bound of 250 million documents.

Index compression Indexes should be small while allowing fast processing. To that end we employed the word-aligned compression scheme described by Anh and Moffat [2005]. No term positional information is stored in our indexes.

Another possibility is to use static pruning, and simply reduce the number of pointers stored, perhaps through the use of a stoplist, for example. Anh and Moffat [2002a] describe a static pruning scheme for use with impact-sorted indexes, but no static pruning is employed here, and the listed index sizes can be compared with other systems.

Parallel processing A Beowulf-style cluster was used in our experiments, consisting of a server and eight additional nodes. The server is a Dual 2.8 GHz Intel Xeon with 2 GB of memory, running Debian GNU/Linux. The server supports a 73 GB SCSI disk for system files, and twelve 146 GB SCSI disks in a RAID-5 configuration for data. Each of the nodes is a 2.8 GHz Intel Pentium-4 with 1 GB of memory and 250 GB of local SATA disk.

Our objective was to achieve fast processing of individual queries. To this end, the data collection was evenly distributed across the cluster nodes in a partitioned *local index* arrangement. Each

Run tag	Type of impacts	Intergration method	Hardware
MU04tb1	Content	Method_A	Cluster
MU04tb2	Content+Anchor	Method_A	Cluster
MU04tb4	Content	Method_D	Single
MU04tb5	Content+Anchor	Method_D	Cluster
MU04tb3	Content+Anchor	Method_F	Cluster

Table 2: *Parameter settings for the Terabyte Track.*

of the eight nodes constructed an index for its sub-collection, and executed queries against that index. When querying, the server played the role of a receptionist, and received queries, broadcast them to the nodes, and received back the results lists. The final result listing was then prepared by the receptionist by merging the node result sets, using the local similarity scores supplied by the nodes. There was no exchange of global information between the nodes and the receptionist, and all scoring and ranking was on the basis of local information.

6 Performance in the Terabyte Track

This section reports the settings and outcomes of our experiments in the 2004 Terabyte Track.

Hardware configuration Two alternative hardware configurations were explored:

1. `Cluster`: The cluster of eight nodes plus the server, as described in Section 5 is employed.
2. `Single`: The server only is employed.

Efficiency metrics The efficiency metrics reported consist of:

- *IndexTime*: The elapsed time for index construction. Only three time components were included: primary indexing time, merging time, and sorting time. The time required to transfer the original collection to the nodes, and for preprocessing the anchor text, URL weights, and authority weights, was not included. In particular, the time spent preprocessing anchor text would have been large if the extra information supplied by NIST (URL-id and link files) had not been available.
- *IndexSize*: The total size of the index, summed across all nodes. In calculating the index size, all index components (inverted file, lexicon, URL weight file, authority weight file) were taken into account. The document representation and the mapping from document number to the document location were not counted.
- *AverageQueryTime*: Queries were processed sequentially, with a single query active in the system at any given time. Each query is considered as being "finished" once the result list of document identifications is finalized, without any time being spent to actually retrieve the answer documents. The elapsed time was measured from the moment when the first query arrives, until the last query has been processed, not including the initialization costs associated with loading a range of memory-resident files. The total time is then divided by the number of queries in the stream to obtain *AverageQueryTime*.

Type of impacts	Hardware	*IndexSize*		*IndexTime*	*AverageQueryTime*
		in GB	% of collection	(minutes)	(seconds per query)
`Content+Anchor`	`Cluster`	7.18	1.68	153	0.08
`Content`	`Cluster`	7.06	1.65	104	0.08
`Content`	`Single`	5.60	1.31	785	0.36

Table 3: *Efficiency outcomes for the Terabyte Track.*

Run tag	MAP	R-prec	P@5	P@10	P@20	Recip.Rank
Content-only:						
`MU04tb1`	0.2657	0.3318	0.6000	0.5857	0.5378	0.7556
Content and link structure:						
`MU04tb4`	0.2679	0.3336	0.6122	0.5878	0.5449	0.7599
Content, anchor text, and link structure:						
`MU04tb2`	0.0634	0.1210	0.3633	0.3408	0.2898	0.5253
`MU04tb3`	0.0434	0.0924	0.3388	0.2735	0.2429	0.4776
`MU04tb5`	0.0639	0.1218	0.3755	0.3408	0.2939	0.5616

Table 4: *Effectiveness outcomes for the Terabyte Track.*

Run settings Five runs were submitted to the Terabyte Track. Of the five, four runs used the full power of the cluster, and one used the server only. In terms of similarity computation, two runs used the `Content` impacts, and three others the `Content+Anchor` impacts. Three variants of evidence integration were tested: using impacts only (`Method_A`); using authority weights to break ties on impact scores (`Method_D`); and combining all evidence (`Method_E`). The details of the combinations tested are presented in Table 2.

Efficiency Table 3 shows our achievements in terms of efficiency. Using a single machine, we can index the `GOV2` collection at a rate of approximately 33 GB per hour, and process queries at the rate of approximately three per second. When all nine machines are engaged, indexing throughput increases by a factor of 7.5, and querying throughput by a factor of 4.5. The latter is smaller than the former (and also smaller than the ideal factor of 8) because we do not support intra-query parallelism. For example, when the receptionist is engaged merging the result lists from the nodes, the nodes are always idle. The use of threading to allow multiple queries to be active at any given moment would increase the throughput of the `Single` system by a non-trivial amount and of the `Cluster` system by a greater factor. In all scenarios the cost of storing the compressed index is very small.

Effectiveness Effectiveness scores for our Terabyte runs are shown in Table 4. The runs that employ the incoming anchor text failed completely, the result of a programming error, and the last three rows of the table demonstrate only that buggy programs remain the bane of these experiments. Hence, our discussion focuses on the results for the content-only and content-and-link-structure runs.

Relative to the title-only runs lodged by the TREC participants, the performance of `MU04tb1` and `MU04tb4` is good, especially in terms of P@5, P@10, P@20 (where one or the other of the two runs holds the first position over all title-only submitted runs) and Recip.Rank (where the two runs obtain the second and third positions over all title-only submitted runs).

On the other hand, Table 4, also shows that although the use of structure link information

to break the ties in defining impact order did give some positive results, the gain is marginal. Considering the large effort spent for collecting data and calculating pagerank scores, the modest gain represents a very small return on effort spent. It is also possible that using the pagerank information in a different way may result in a better gain.

7 Future Directions

We intend to continue our development of integral impact-based scoring mechanisms. Their speed during querying, and compact index requirements, make them an attractive proposition for web retrieval, and have not yet exhausted their potential. For example, we plan to incorporate hub weighting, elements of the document markup and structure, and terms extracted from the URL string. In addition, we plan to examine the possibility of incorporating all evidence at indexing time, to further reduce the query-time costs. Extension to collections beyond the current terabyte is also of considerable interest.

Acknowledgement This work was supported by the Australian Research Council, the ARC Center for Perceptive and Intelligent Machines in Complex Environments, and the NICTA Victoria Laboratory.

References

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

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

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

V. Anh and A. Moffat. Robust and web retrieval with document-centric integral impacts. In E. Voorhees and D. Harman, editors, *The Twelth Text REtrieval Conference (TREC 2003)*, Gaithersburg, MD, November 2003. National Institute of Standards and Technology (Special Publication 500-255). URL `http://trec.nist.gov/pubs/trec12/t12_proceedings.html`.

V. Anh and A. Moffat. Inverted index compression using word-aligned binary codes. Source code available from `www.cs.mu.oz.au/~alistair/carry/`, January 2005.

S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proc. 7th International World Wide Web Conference*, pages 107–117, Brisbane, Australia, April 1998. URL `http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm`.

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