Methodologies for Distributed Information Retrieval

Owen de Kretser Alistair Moffat
Dept. of Computer Science, University of Melbourne, Parkville 3052, Australia
{oldk,alistair}@cs.mu.oz.au

Tim Shimmin Justin Zobel
Dept. of Computer Science, RMIT, GPO Box 2476V, Melbourne 3000, Australia
{tes,jz}@cs.rmit.edu.au

Abstract

Text collections have traditionally been located at a single site and managed as a monolithic whole. However, it is now common for a collection to be spread over several hosts and for these hosts to be geographically separated. In this paper we examine several alternative approaches to distributed text retrieval. We report on our experience with a full implementation of these methods, and give retrieval efficiency and retrieval effectiveness results for collections distributed over both a local area network and a wide area network. We conclude that, compared to monolithic systems, distributed information retrieval systems can be fast and effective, but that they are not efficient.

1 Introduction

The efficient management of large text collections is an important practical problem. With the growth in the use of network services, text collections such as digital libraries are increasingly being distributed. For example, a branch office of an organisation may be responsible for archiving its documents, allow access by the head office, and be able to access information held at head office or any branch. Even at a single site it may be useful for document collections to be distributed over several machines, to simplify update and to balance load.

The change in emphasis from *mono-server* collections to *multi-server* collections changes the manner in which queries are evaluated. One form of query to a document database is Boolean, where, in a distributed environment, independent servers execute the query on each of the subcollections, and the overall result set is simply the union of the individual result sets. Another form of query is *ranking*, in which each document in the collection is assigned a *similarity* score that indicates how closely (according to some heuristic) the document matches the query. Documents are then presented to the user in decreasing similarity order until the user indicates that no further documents are required. In contrast to Boolean queries, there is no precise set of documents that constitute the answers, and thus no "perfect" mechanism for identifying which documents should be returned. However, with a good heuristic ranked queries provide more effective retrieval than Boolean queries in terms of satisfying an information need [16]; and ranked queries can be evaluated in time similar to that required by an equivalent Boolean query [14, 15].

The drawback of ranked queries in a distributed environment is that the most successful heuristics make use of several collection-dependent statistics, including the total number of documents; the number of distinct terms; the number of documents in which each term appears; and the number of times each term appears in each document. In an ideal system query evaluation would involve no more than communication of the query to the subcollections, collation of the answers from each, and communication of them to the user, much as for Boolean queries. However, the use of local rather than global parameters might lead to inaccurate similarity scores, and in such a scheme most of the answers returned will not be presented to the user. In such cases computation of similarity and transmission of unwanted documents are a waste of resources.

In this paper we examine a range of techniques for evaluating ranked queries to distributed text databases, seeking a method that gives good effectiveness—the ability to find relevant answers—while using minimal computational resources. To allow exploration of different approaches we developed the TERAPHIM distributed text database system, based on our mono-server text database system MG [1, 12, 26]. Our experiments with TERAPHIM show that retrieval effectiveness can be maintained in the face of distribution, and that, given a

reasonable network bandwidth, a distributed text collection can be queried as quickly as can a mono-server collection. However, central coordination is necessary for aggregation of the results of the ranking, and distribution leads to much greater overall use of resources.

2 Ranked Retrieval

In principle a ranked query is resolved by using a statistical measure to compute the similarity of each document to the query, then retrieving the documents with the highest similarity. There have been many effective similarity measures presented in the literature [4, 17, 29]. These weight rare terms more highly than common terms, with the "rareness" of term t determined as an inverse function of the number f_t of documents in the collection that contain t; take into account the number of times $f_{d,t}$ that term t appears in document d; and use the length W_d (according to some metric) of document d for normalisation.

In our experiments we have used the cosine measure with logarithmic in-document frequency [8], which is one of the most effective similarity measures. In this method the similarity $C(q,d)$ of query q and document d in a collection of N documents is given by

$$C(q,d) = \frac{\sum_{t \in q \cap d} (w_{q,t} \cdot w_{d,t})}{\sqrt{\left(\sum_{t \in q} w_{q,t}^2 \cdot \sum_{t \in d} w_{d,t}^2\right)}}$$

where

$$\begin{aligned} w_{d,t} &= \log(f_{d,t} + 1) \qquad \text{and} \\ w_{q,t} &= \log(f_{q,t} + 1) \cdot \log(N/f_t + 1)\,. \end{aligned}$$

In this formulation the collection-wide statistic $\log(N/f_t + 1)$ applies only to query weights, so that document lengths are not collection-dependent and update is simplified. There are many similar formulations; all are handled in the same manner as far as query evaluation is concerned, and most have similar, good effectiveness [29].

To allow efficient resolution of ranked queries two main structures are used. The first is an inverted file that stores, for each term t that appears in the collection, a list of document numbers d in which t appears together with $f_{d,t}$. While large, the inverted file can be stored compressed, and modern text retrieval systems generate indexes that typically occupy 10% or less of the volume of the text. [14, 26]. The second important structure is a table of document weights computed by $W_d = \sqrt{(\sum_{t \in d} w_{d,t}^2)}$. These are precalculated and stored as part of the database.

To allow measurement of the effectiveness of an information retrieval technique three resources are required: a corpus of text; a set of test queries; and a set of relevance judgements—human evaluations as to which of the sample documents are relevant to which of the queries. The data of the NIST TREC project [6] supplies these three components. The experiments described below were performed on the gigabyte of data on TREC disk two, and two subsets of the queries, 51–200 and 202–250. The queries were split to allow investigation of both long and short queries. In the first group, after simple transformations such as removal of stop-words, the average length is 90.4 terms, whereas in the second group the average length is 9.6 terms; our experiments have been primarily with the second group.

We measure retrieval effectiveness in two ways: by the standard measure of recall-precision, that is, average precision obtained at 11 recall values (denoted in our tables as 11-pt average) over 1000 documents retrieved; and by the number of relevant documents in the top 20 documents returned. For example, if one screen of titles of suggested answer documents contains 20 lines, then the precision at 20 is an important way of quantifying retrieval effectiveness—users will be satisfied if they can fulfil their information need in the first screen of document headers.

3 Distributed Retrieval

In a *multi-server* distributed text retrieval system, there are several independent mono-servers, or *librarians*. Each is responsible for some component of the collection, for which it maintains an index, evaluates queries, and fetches documents. Separate from the librarians are one or more *receptionists*, which interact with the users of the system and communicate user requests to the librarians. Each receptionist may be resident on the same physical machine as one or more librarians, or may be quite separate. In the models of computation we consider, the receptionists may have available global information about each librarian, such as the total number of documents or perhaps a partial (or even full) copy of its index information. (Note that this arrangement is not a typical master-slave or client-server architecture, as a librarian may be in communication with several receptionists; hence our decision to adopt fresh terminology.) A receptionist is essential to ranked query evaluation because it is necessary to collate the results of ranking each subcollection.

In this model, queries evaluation is as follows.

1. A user lodges a query with a receptionist, which examines any global information it has access to and passes the query, with perhaps some of the global information, to a selected set of librarians.

2. Each selected librarian evaluates the query and, making use of any transmitted global information and its own local information, determines a ranking for the local collection—a list of document identifiers and similarity scores.

3. Each ranking is returned to the receptionist, which waits for all the nominated librarians to respond and then merges their rankings to obtain a global collection-wide ranking and identify the top k documents. During the merging process the receptionist may again make use of global information.

4. Each selected librarian is given a list of document identifiers within its domain and is requested to return the text of the corresponding documents to the receptionist for display to the user.

As an optional initial step, the receptionist may converse with the librarians to establish parameters.

In this generic description of the model we have specified neither how the rankings are merged nor the internal structure of the librarians and receptionists. In the alternative federated methodologies described below we assume that the librarians and receptionist are similar enough to share information such as vocabulary and index and use the same similarity heuristic, but other arrangements are possible. For example, Voorhees et al. [21, 22, 23, 24] have described strategies for merging the rankings returned by librarians with no knowledge of how they were computed, so long as appropriate training queries are available.

Evaluation Criteria To evaluate performance we need to consider three factors. One is *effectiveness*. Another is *response time*: the delay between issuing the query and return of answers, which depends on the amount of processing involved, the volume of network traffic, and the number of handshaking steps used. The last factor is use of *resources*: CPU time at the receptionist and librarians, volume of data needed by librarians and transmitted over the network, and disk space required by the receptionist. Response time and resource usage are linked, but only loosely. In particular, response time measures the minimum delay a user will experience, even on a lightly loaded system, whereas resource use is an indication (in an inverse sense) of the overall query throughput possible with the system when it is operating at capacity, with multiple users and queries competing for resources.

A key facility that we strived for was transparency: it should be possible for any set of collections to be queried as a single database. We require that each of the subcollections can be accessed without recourse to any central information; and that any subcollection can be a logical component of databases managed by several different receptionists. Note that this independence of librarians eliminates some other possible models for distribution, such as having component files (for example, text, index and dictionary files) at different sites [11] or partitioning the index so that different terms are indexed at different sites [19].

We now describe several specific methodologies for distributed information retrieval. These are not the only possible methodologies—for example, other authors have considered several variants on the "central vocabulary" scheme described below [5, 7, 20], as well as other possibilities [2, 3, 21, 25] that are similar to ours. The methodologies described here are chosen because they are relatively straightforward and lend themselves to efficient query evaluation.

Central Nothing In what we call a *Central Nothing* (*CN*) system the only global information maintained by the receptionist is a list of librarians. When a query is entered every librarian is given the query and prepares a ranking of its k "best" documents, as determined by its index and its values for parameters f_t and N. When these lists have all been returned the receptionist merges them, accepting at face value all supplied similarity values—it has no basis for perturbing either the numeric values or the ordering. For S subcollections the result is a list of kS similarities. The top k are then extracted, and a document request list sent to each librarian. Some of the librarians may not be required in this second phase.

The main advantage of CN operation is that no global information is required; the receptionist is free to choose any subset of librarians. The disadvantage is that much of the power of a ranked query is potentially lost. For example, a term might be common in one subcollection and be assigned a minimal weight, but in the context of the collection as a whole that term might be rare, and documents from the subcollection important; thus the final ranking will be poor. It might also be that effectiveness is dramatically compromised by the use of subcollection weights. Finally, it is possible that unnecessary calculation is performed—the receptionist has no basis for excluding any subcollection, and so every subcollection processes the query in full.

In principle the receptionist could pass the query terms to the librarians and the librarians then return k documents immediately, without the intermediate step of passing back document identifiers and similarities, much as for Boolean queries. Unfortunately this is no more effective than the approach described in the next section [25]; and it is not pragmatic. Document identi-

fiers are only a few bytes each, but documents are much larger; in TREC the average size is over two kilobytes and the largest is over two megabytes. Transmission of kS rather than k documents would severely degrade performance.

Central Vocabulary In a *Central Vocabulary* (or *CV*) system the global information stored by the receptionist is the vocabularies of the subcollections, which allows the receptionist to determine for each term a collection-wide weight. This should allow better ranking, but the preprocessing stage eliminates the spur-of-the-moment choice of subcollections possible in a CN scheme, and storage is required for the collection-wide vocabulary.

Query processing is similar to that in a CN system, with the crucial difference that each query term transmitted to the librarians is accompanied by a weight to be used; a similar approach is used by Walczuch et al. [25]. In our implementation, the librarians still calculate a k-ranking, but the similarity scores computed by the various librarians are exactly the same as for the mono-server alternative. The formation of a global vocabulary means that collections can be completely avoided if they contain none or few of the query terms. There is evidence that the vocabularies of the subcollections can be used to guide the search, allowing it to be focussed on a subset of the subcollections [5, 27, 28].

Central Index In a *Central Index* (or *CI*) system, the receptionist has full access to the indexes of the subcollections, so it can perform all the index processing and request from each librarian the documents required to make a global ranking of length k. In this case the preprocessing involves merging the subcollection vocabularies and indexes, and the need for storage space on the part of the receptionist is relatively large.

To save some of the central index space the receptionist can collect adjacent documents into *groups* and then index the groups as if they were single documents [13]; space is saved because the number of groups containing each term is less than the number of documents, reducing index size. For example, in our earlier work [13], where we explored the effect of group size on effectiveness, we showed that use of groups of ten documents approximately halves index size. Suppose then that a grouped index has been formed with groups of size G. To process a query the receptionist first examines its own index, and determines a list of k' highly ranked groups, where $k' \geq k/G$ is a function of both G and k. The k' group identifiers are then expanded into k' ranges, each containing G document identifiers, and the subcollections owning each of the $k'G$ document identifiers is instructed to consult its local index to determine a similarity value for that document. With a mechanism that allows similarity values for some documents to be computed without processing the index lists in full [14], processing at the librarians can be considerably faster than for CN or CV operation. The $k'G$ similarity values so calculated are then sorted, and a final listing of the k highest scoring documents determined and requested from the various librarians.

Compared with a CV system, the advantage is that each librarian must consult only a fraction of its index. The potential disadvantage is that highly relevant documents that are (by bad luck) grouped with non-relevant documents may not be retrieved. In our earlier work with grouped indexes, which used a simulated implementation, we measured the relationship between G, k', and k—how far the number of groups can be reduced before retrieval effectiveness declines. The performance questions we sought to answer in our full implementation were the size of the central index, the cost of processing the central index, the extent to which the librarians could be protected from redundant computation, and how overall costs compare to other approaches.

Implementation To test our proposals we developed a prototype distributed text-retrieval engine, TERAPHIM. It is built on the research prototype MG [1, 12, 26], a mono-server text database system that uses compression for text and index, and in doing so provides an attractive combination of fast query processing and modest storage overheads. In TERAPHIM, each librarian-to-receptionist session is an ordinary MG process, so that the individual subcollections can be queried independently, and the receptionist is a dataless MG process. MG and TERAPHIM have had several years of development and we are confident that performance is as good as or better than production text retrieval systems.[1] We have used MG to investigate large numbers of similarity heuristics [29], and the one used in these experiments is competitive with the best known measures.

4 Results

Effectiveness We first investigated effectiveness, using, as discussed above, TREC disk two, which contains about one gigabyte of text in four distinct collections: AP, FR, WSJ, and ZIFF. The systems investigated were a MS database of the full one gigabyte of text, and

[1] MG and TERAPHIM are written in C. A publicly available version of MG (but not, at this stage, TERAPHIM) can be fetched from *ftp://munnari.oz.au/pub/mg*. An on-line description of MG can be found at *http://www.mds.rmit.edu.au/mg*.

Mode	11-pt average %	Relevant docs. in top 20
Long queries (51–200)		
MS and CV	23.07	8.2
CN	24.35	8.6
CI, $k' = 100$	10.49	7.2
CI, $k' = 1000$	21.10	8.5
Short queries (202–250)		
MS and CV	15.67	4.7
CN	16.21	4.9
CI, $k' = 100$	14.01	5.3
CI, $k' = 1000$	16.81	5.0

Table 1: *Retrieval effectiveness using two query sets: 11-point average recall-precision at 1000 documents retrieved; and average number of relevant documents in the 20 documents most highly ranked.*

distributed systems built from the four subcollections using the CN, CV, and CI paradigms, and using the formulation of the cosine measure described earlier. In the CI method we set the groupsize G to be 10 based on our earlier experiments [13], and undertook different runs with different values for the parameter k', the number of groups to be expanded. For example, with $k' = 100$ and $G = 10$ a total of $k'G = 1000$ similarity values are generated; and it is unsurprising that for this combination the 11-point retrieval effectiveness is very low, since in the TREC methodology the 11-point average is based upon a ranking of 1000 documents.

Table 1 shows the behaviour of the various retrieval modes on the long and short TREC query sets, and illustrates the effect that k' has upon the 11-point average. Note how the precision values in the last column are relatively insensitive to the value of k'. These results show that, for high-precision retrieval in applications such as web search engines, small values of k' may be used without the usefulness of the result being substantially eroded.

It is possible that the good performance may be to some extent a result of the small set of subcollections we used, and of their uniform size (although, however, most of the relevant documents were in AP and TREC). To explore this issue further we also examined effectiveness when TREC disk two is broken into 43 subcollections (using a standard division into subcollections developed for TREC). The impact on effectiveness was surprisingly small. For the short queries and CN, for example, the effectiveness was only marginally poorer than in Table 1. However, the variation in subcollection size was relatively small—from just over 1000 to just under 10,000 documents—and it is quite possible that with greater variation that effectiveness would decline. That is, for these large collections the size means that the statistics can be used for reliable retrieval, but in general the CN method is likely to be less robust than the other approaches.

Efficiency Efficiency and response time in a distributed text database system depend on many factors, including the power of the machines on which the text is resident, bandwidth of the network, network traffic, and point-to-point network transmission time. In order to obtain indicative results, we decided to evaluate the various modes of operation in several configurations:

Mono-Disk: On a single machine with the data on a single disk, which is almost certainly a worst case; not only is disk bandwidth a bottleneck for librarian query processing but the librarians interfere with each other by repositioning the disk head unpredictably.

Multi-Disk: On a single machine with the data distributed across three locally mounted disk drives and two NFS mounted drives (four librarians plus one receptionist). In both this case and the one above the machine used was a Sun SPARC 10 with four processors, thus allowing a modest amount of CPU parallelism.

LAN: On three machines (a four-processor SPARC 10, running the receptionist and the FR database; a dual-processor SPARC 10, running the AP and WSJ collections; and a two-processor SPARC 20, running the ZIFF collection) on a common 10 megabit ethernet cable. This configuration was designed as typical of a local area network.

WAN: On machines at geographically separated sites, namely Melbourne (the receptionist); Canberra (Australia, running the ZIFF collection); Brisbane (Australia, running the AP collection); Hamilton (New Zealand, running the FR collection); and Tel Aviv (Israel, running the WSJ collection). This configuration was designed to emulate a typical arrangement using a wide area network.

The connectivity of the various remote sites used for the WAN experiments is summarised in Table 2. Note the relatively high cost of communicating with both New Zealand and Israel; in the first case the link is relatively direct, but of modest bandwidth; and in the other the link traverses the United States. An obvious but nevertheless crucial consequence is that handshaking should be kept to an absolute minimum. For exam-

Location	Network hops from Melbourne	Avg. "ping" time (sec)
Waikato	13	0.76
Canberra	14	0.18
Brisbane	16	0.14
Israel	28	1.04

Table 2: *Network communication costs: network hops; and average round-trip communication time for one packet using "ping" at approximately GMT0100 on a Wednesday (12 noon local Australian time).*

Mode	Configuration			
	mono-disk	multi-disk	LAN	WAN
Short queries (202–250)				
MS	1.07	—	—	—
CN	1.11	0.91	0.91	4.21
CV	1.17	0.90	0.82	4.20
CI	1.55	1.42	1.25	4.86

Table 3: *Elapsed time (sec) in each configuration per query, indexing processing time only (steps 1, 2, and 3 of the method listed in Section 3),* $k = 20$ *and* $k' = 100$.

Mode	Configuration			
	mono-disk	multi-disk	LAN	WAN
Short queries (202–250)				
MS	1.43	—	—	—
CN	1.33	1.31	1.33	15.04
CV	1.49	1.37	1.27	14.71
CI	2.00	2.08	1.63	10.71

Table 4: *Elapsed time (sec) in each configuration per query, total time including index processing and retrieval of answers (steps 1 to 4 of the method listed in Section 3),* $k = 20$ *and* $k' = 100$.

ple, documents should be bundled into blocks by the librarians rather than transferred individually.

Reasonable efforts were made to run these experiments on idle machines, but this was not within our control for the WAN configuration, and we were unable to predict network load—depending upon the time of day and time of week, the cost of running the WAN queries varied by as much as a factor of one hundred. (As a consequence the times reported for the WAN are representative rather than accurate.) Finally, as a base case we measured the performance of a mono-server MG system with all the data files on a single disk drive.

Tables 3 and 4 show the response time measured with the various modes of operation and network configurations for the short TREC queries. (Network problems prevented us from completely trialling the long queries; based upon the trials we did complete, we would expect to see the same trends.) Table 3 shows the component of the response time caused by index processing, from the moment the query is issued until the list of answer documents is determined, but excluding the actual cost of fetching those documents. As can be seen, the multi-disk versions operate slightly faster than do the mono-disk configurations, and all but the WAN arrangement operate at speeds comparable with the MS system. Table 4 shows the response time for the same queries, but including the cost incurred through the librarians returning the answer documents back to the receptionist. Fetching the documents adds to the cost, but, in all but the WAN case, by a relatively small amount.

Analysis Some clear trends emerge from the experiments recorded in the previous sections. In particular, distribution need not have any impact on effectiveness: with vocabularies held at the receptionist, effectiveness is identical to that of a MS system. The space required by the combined vocabularies is moderate (less than 10 Mb for the gigabyte of text), and saves a preliminary round of communication between receptionist and librarians, and a central vocabulary may allow receptionists to neglect some librarians [5, 27, 28]. The CN method was similarly effective and efficient. However, it is not clear that it is robust, because small, topical collections are likely to have highly distorted statistics. The results showed that little was gained by storing a full central index, which in this case occupied around 40 Mb: elapsed times were greater because of the sequential processing of the central index. However, the CI method does have the potential to save net index processing time, because only part of the index at each librarian need be inspected.

Network bandwidth and round-trip times are crucial to efficiency. In the WAN experiments the cost is dominated by the latency of the Internet. However there is some scope for improvement in the current implementation; with prefetching and blocking the cost of fetching documents from the remote sites can probably be reduced. Compression can also help: a traditional solution to the problem of network costs is to compress data prior to transmission, a solution that is facilitated in TERAPHIM since all documents are stored compressed [1]. However, performance was still poor compared to the MS approach. A further refinement would be to only send part of each document, such as a header (on the assumption that users will inspect

only a few documents in full, and these can be fetched on demand), a solution that reduces costs but requires knowledge of document structure.

Also, for the CI system the CPU cost can be reduced: in these experiments we did not employ our "skipping" mechanism [14], and we expect that, with skipping, when the number k' of groups to be processed is small the CPU cost at the librarians would decrease by a factor of two or more.

With regard to overall efficiency, all of the distributed methods were poor. In some distributed applications, each site can operate fairly independently and elapsed times can be greatly reduced; but in this case, significant central coordination is needed, limiting possible savings. Furthermore, one of the major costs of query evaluation for text databases is accessing the vocabulary and fetching the inverted lists, and this operation is repeated at each librarian. Although the individual lists are shorter, overall many more lists must be fetched. Thus only a small speed increase is available through distribution of a text database, and total resource costs are greater than for a traditional mono-server implementation. These problems become more acute as the number of collections is increased—the small gain in response is at the cost of a great deal of additional processing. Net savings are possible only if, given a query, it can be reliably determined that many of the subcollections can be neglected. As a management measure, however, distribution has other benefits, such as faster update, allowing data to be controlled independently at local sites, and allowing text data from different sources to be stored separately.

5 Related Work

With compression, inverted indexes are only a small fraction of the size of the indexed data. Nonetheless, having a combined index of all stored data and duplicating it for each receptionist is clearly impractical. It was for this reason that we introduced the concept of grouping, to reduce index size while allowing query evaluation to proceed. However, it is not the only way of reducing index size. Another possibility is to apply a threshold based on $f_{d,t}$ values; for term occurrences that can only make a small contribution to similarity values, because both $f_{d,t}$ and w_t are small, not using these values in ranking may have little impact on effectiveness. Persin et al. [15] describe experiments with per-query thresholding that show that the volume of index information processed can be reduced by a factor of five without reducing effectiveness. In preliminary experiments, applying thresholds that only reduced index size by a third severely degraded effectiveness; nonetheless we plan to further investigate this option in future work. Smeaton et al. [18] have considered thresholding in a mono-server context.

Finally we note that the INQUERY system that has been developed over many years at the University of Massachusetts is being considered as a platform for distributed collections, and there are several common elements between their architecture and our own. Cahoon and McKinley [2] describe the result of simulated experiments on the distributed INQUERY architecture. Using observed behaviour for a mono-server implementation they derive likely performance figures for a distributed implementation, showing it to be scaleable.

6 Conclusions

We have described three alternative methodologies for practical distributed information retrieval, each based on a common architecture in which subcollections are managed independently by librarians and queries are brokered to librarians by receptionists. The methodologies are differentiated by the kind of data that must be held by the receptionist, varying from no more than a list of valid subcollections (central nothing) to a merged vocabulary (central vocabulary) to a full index of stored data (central index).

To test these methodologies we implemented a prototype distributed information retrieval system, TERAPHIM, and used it to index and query one gigabyte of text. These experiments showed that distribution need have no impact on effectiveness, and it is possible to obtain faster response times to queries. However, overall resource usage rises significantly as a consequence of distribution.

The main performance bottleneck was network delay. We had hypothesised that achievement of good performance would require that the number of communication steps between librarians and receptionists be kept low; this supposition was confirmed by our experiments, which showed that—with even the minimal number of communication steps required by our architecture—network delay was the dominant factor in response for wide-area distribution. Our experiments show that addressing the costs of handshaking, data transmission, and unnecessary access to subcollections are the major problems that must be addressed in construction of a practical system for distributed text retrieval.

Acknowledgements

We thank Dave Hawking (Australian National University, Canberra), Tomi Klein (Bar-Ilan University, Israel), Rodney Topor (Griffith University, Brisbane), and Ian Witten (Waikato University, New Zealand) for

arranging access to their machines. We also thank Daryl D'Souza and Ross Wilkinson. This work was supported by the Australian Research Council.

References

[1] T.C. Bell, A. Moffat, I.H. Witten, and J. Zobel. The MG retrieval system: compressing for space and speed. *Communications of the ACM*, 38(4):41–42, April 1995.

[2] B. Cahoon and K.S. McKinley. Performance evaluation of a distributed architecture for information retrieval. In H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, editors, *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 110–118, Zurich, Switzerland, 1996.

[3] J.P. Callan, Z. Lu, and W.B. Croft. Searching distributed collections with inference networks. In E.A. Fox, P. Ingwerson, and R. Fidel, editors, *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 21–28, Seattle, Washington, July 1995.

[4] W.B. Frakes and R. Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, Englewood Cliffs, New Jersey, 1992.

[5] L. Gravano and J.H. Garcia-Molina. Generalising GlOSS to vector-space databases and broker hierarchies. In U. Dayal, P.M.D. Gray, and S. Nishio, editors, *Proc. International Conference on Very Large Databases*, pages 78–89, Zurich, Switzerland, September 1995.

[6] D. Harman. Overview of the second text retrieval conference (TREC-2). *Information Processing & Management*, 31(3):271–289, May 1995.

[7] D. Harman, W. McCoy, R. Toense, and G. Candela. Prototyping a distributed information retrieval system using statistical ranking. *Information Processing & Management*, 27(5):449–460, 1991.

[8] D.K. Harman. Ranking algorithms. In Frakes and Baeza-Yates [4], chapter 14, pages 363–392.

[9] D.K. Harman, editor. *Proc. Third Text REtrieval Conference (TREC–3)*, Gaithersburg, MD, November 1994. National Institute of Standards and Technology Special Publication 500-225.

[10] D.K. Harman, editor. *Proc. Fourth Text REtrieval Conference (TREC–4)*, Gaithersburg, MD, November 1995. National Institute of Standards and Technology Special Publication 500-236.

[11] I.A. Macleod, T.P. Martin, B. Nordin, and J.R. Phillips. Strategies for building distributed information retrieval systems. *Information Processing & Management*, 23(6):511–528, 1987.

[12] A. Moffat and J. Zobel. Compression and fast indexing for multi-gigabyte text databases. *Australian Computer Journal*, 26(1):1–9, February 1994.

[13] A. Moffat and J. Zobel. Information retrieval systems for large document collections. In Harman [9], pages 85–93.

[14] A. Moffat and J. Zobel. Self-indexing inverted files for fast text retrieval. *ACM Transactions on Information Systems*, 14(4):349–379, October 1996.

[15] M. Persin, J. Zobel, and R. Sacks-Davis. Filtered document retrieval with frequency-sorted indexes. *Journal of the American Society for Information Science*, 47(10):749–764, 1996.

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

[17] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[18] A.F. Smeaton, F. Kelledy, and R. O'Donnell. TREC-4 experiments at Dublin City University. In Harman [10], pages 373–389.

[19] A. Tomasic and H. Garcia-Molina. Performance of inverted indices in shared-nothing distributed text document information retrieval systems. In M.J. Carey and P. Valduriez, editors, *Proc. 2nd International Conference On Parallel and Distributed Information Systems*, pages 8–17, Los Alamitos, CA, January 1993. IEEE Computer Society Press. Conference held in San Diego.

[20] C.L. Viles and J.C. French. Dissemination of collection wide information in a distributed information retrieval system. In E.A. Fox, P. Ingwersen, and R. Fidel, editors, *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 12–20, Seattle, WA, 1995.

[21] E.M. Voorhees. Siemens TREC-4 report: Further experiments with database merging. In Harman [10], pages 121–130.

[22] E.M. Voorhees, N.K. Gupta, and B. Johnson-Laird. The collection fusion problem. In Harman [9], pages 95–104.

[23] E.M. Voorhees, N.K. Gupta, and B. Johnson-Laird. Learning collection fusion strategies. In E.A. Fox, P. Ingwersen, and R. Fidel, editors, *Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 172–179, Seattle, WA, 1995.

[24] E.M. Voorhees and R.M. Tong. Multiple search engines in database merging. In R.B. Allen and E. Rasmussen, editors, *Proc. ACM Digital Libraries*, pages 93–102, Philadephia, Pennsylvania, 1997.

[25] N. Walczuch, N. Fuhr, M. Pollmann, and B. Sievers. Routing and ad-hoc retrieval with the TREC-3 collection in a distributed loosely federated environment. In Harman [9], pages 135–144.

[26] I.H. Witten, A. Moffat, and T.C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Van Nostrand Reinhold, New York, 1994.

[27] B. Yuwona and D.L. Lee. Server ranking for distributed text retrieval systems on the Internet. In R. Topor and K. Tanaka, editors, *Proc. International Conf. on Database Systems for Advanced Applications*, pages 41–49, Melbourne, Australia, 1997.

[28] J. Zobel. Collection selection via lexicon inspection. In P. Bruza, editor, *Proc. Australian Document Computing Conference*, pages 74–80, Melbourne, Australia, 1997.

[29] J. Zobel and A. Moffat. Exploring the similarity space. *SIGIR Forum*. To appear.