Distributed multisearch and resource selection for the TREC Million Query Track

Chris Fallen and **Greg Newby**
Arctic Region Supercomputing Center, University of Alaska Fairbanks, Fairbanks, AK 99775

Kylie McCormick
Mount Holyoke College, South Hadley, MA 01075

Abstract
A distributed information retrieval system with resource-selection and result-set merging capability was used to search subsets of the GOV2 document corpus for the 2008 TREC Million Query Track. The GOV2 collection was partitioned into host-name subcollections and distributed to multiple remote machines. The Multisearch demonstration application restricted each search to a fraction of the available sub-collections that was pre-determined by a resource-selection algorithm. Experiment results from topic-by-topic resource selection and aggregate topic resource selection are compared. The sensitivity of Multisearch retrieval performance to variations in the resource selection algorithm is discussed.

1. Overview

The information processing research group at ARSC works on problems affecting the performance of distributed information retrieval applications such as metasearch [1], federated search [2], and collection sampling [3]. An ongoing goal of this research is to guide the selection of standards and reference implementations for Grid Information Retrieval (GIR) applications [4]. Prototype GIR applications developed at ARSC help to evaluate theoretical research and gain experience with the capabilities and limitations of existing APIs, middleware, and security requirements. The TREC experiments provide an additional context to test and develop distributed IR technology.

Prior TREC Terabyte (TB) and Million Query (MQ) Track experiments performed at ARSC have explored the IR performance and search efficiency of result-set merging and ranking across small numbers of heterogeneous systems and large numbers of homogeneous systems. In the 2005 TREC [5] TB Track [6], the ARSC IR group used a variant of the logistic regression merging strategy [7], modified for efficiency, to merge results from a metasearch-style application that searched and merged results from two indexed copies of the GOV2 corpus [8]. One index was constructed with the Lucene Toolkit and the other index was constructed with the Amberfish application [9]. For the 2006 TREC [10] TB Track [11], the GOV2 corpus was partitioned into approximately 17,000 collections by grouping documents with identical URL host names [12]. Each query was searched against every collection and the ranked results from each collection were merged using the logistic regression algorithm used in the 2005 TB track. The large number of document collections used in the 2006 experiment, coupled with the distribution of collection size (measured in number of documents contained) that spanned five orders of magnitude, introduced significant wall-clock, bandwidth, and IR performance problems. Follow-up work found that the bandwidth performance could be improved somewhat without sacrificing IR performance by

using a simple relation based on collection size to truncate the result sets before merging [13].

The host-name partition of the GOV2 corpus used in 2006 was used again for a distributed IR simulation for the 2007 TREC [14] MQ track [15] but the full set of collections could no longer be searched for every query with the resources available because the resources required to complete the track using a distributed collection approach increase as the product of the number of queries and the number of collections searched rather than just the number of queries alone. Consequently, only a small fraction of the host-name collections, comprising less than 20% of the documents in GOV2, was searched [16]. The collections were chosen according to historical performance in the previous TREC TB Tracks by ranking the collections in decreasing order of total number of "relevant" relevance judgments (qrels) associated with their respective documents and the top 100 collections were searched.

For the 2008 MQ track, the ARSC IR group ran several resource-selection experiments on the host-partitioned GOV2 collection. The purpose of each experiment was similar in motivation to the 2007 MQ experiment: quickly identify the collections most likely to be relevant to the queries, then search only the top 50 relevant collections, and finally merge the retrieved results into a single relevance-ranked list. The union of the largest 50 of 996 host-name collections available in the experiment contains less than 10% of the total documents in GOV2 so each MQ topic search was restricted to about a tenth of the GOV2 collection.

Each collection-ranking technique was based on a virtual "big document" representation of each collection relative to a fixed basis of terms. One of two document relevance-ranking algorithms was modified to relevance-rank the *document collections* or *resources* before sending the MQ track queries to the Multisearch application: the conventional vector space model (VSM) [17] or Latent Semantic Indexing (LSI) [18]. Prior work on resource selection techniques using variations of standard document retrieval algorithms applied to big-document representations of collections or resources is discussed in [2].

2. System description

Our goal for the 2008 MQ track was to efficiently search a large number of document collections using the Multisearch distributed IR research system developed at ARSC. Each document collection provides a search service to the Multisearch portal. The bandwidth cost for each query is approximately proportional to the number of document collections available and this bandwidth cost is split between the transmission of the query to the document collections and the transmission of the response from each document collection. Therefore, reducing the number of collections searched for each query was important. A resource or collection-selection algorithm was developed to restrict each search to a small number of presumably relevant document collections.

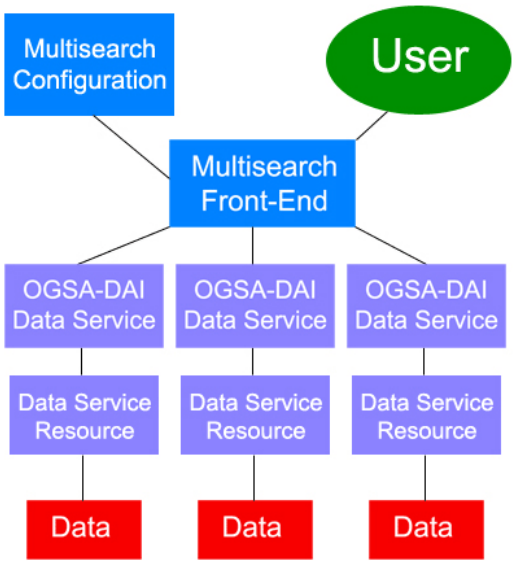


Figure 1. Schematic of the ARSC Multisearch system. The resource selector is contained in the Multisearch front-end block and the host-name document collections are contained in the data blocks.

2.1 Multisearch

Multisearch 2.0 is coded entirely in Java with Apache Tomcat and OGSA-DAI WSI 2.2 as middleware. Tomcat is a servlet container, and it is also a necessary piece of architecture for OGSA-DAI. OGSA-DAI is a middleware for grid services, and it was used because it enables quick, easy launching of literally hundreds of *back-end* web services using Axis. OGSA-DAI also provides methods of searching secured information by having structures for grid credentials, although we did not utilize this ability.

Three different servers were used for the TREC runs. Snowy is a Sun x4500 machine with two dual core Opteron processors at 2.6 GHz with 16 GB of memory and 48 TB of disk space, and it runs Ubuntu Linux 7.10. Pileus is an ASL server with two hyperthreaded Xeon 3.6 GHz CPUs, 8 GB of memory, and 7 TB of disk space. It also runs Ubuntu 7.10. Finally, Nimbus is a Sun v20z with two Opteron processors at 2.2 GHz with 12 GB of memory and 73 GB of disk space. It runs Fedora Core 9. Snowy has the majority of the back ends loaded onto it due to its disk space and speed. It had 873 back end services running, all loaded from the OGSA-DAI architecture. Pileus hosted the remaining 121 back ends. Nimbus ran the front-end client, so all the back ends had to transfer data back and forth over the network.

A query is sent over the network to any number of backend services, as shown in Figure 1. Each service has at least one Data Service Resource (DSR), although some might have multiple resources available. A DSR provides access and to the data within the service, as well as the appropriate methods of getting that data, including a credential system if necessary. In the case of Multisearch, each service was in fact its own search engine. It takes the query and passes it to the DSR which runs the query against the data, compiling a result set to be returned to the requesting query. Multisearch had a minimum result set requirement of five and a maximum result set of twenty. If a result set had too many results, only the first twenty would be taken. If there were not enough results, then none would be returned.

After the result set is completed, it is sent back to the Multisearch front end over the network. There, it is combined with the other result sets from the other back ends. This requires the work of a merge algorithm. In our experiments, we used Naïve Merge (or "log merge" in [12]) to bring results together. Essentially, Multisearch takes the separate result sets and attempts to merge them into one final, reasonable result set. After this, the final set is produced to the user.

2.2 Resource selection

The resource selection component of Multisearch described can relevance-rank many minimally cooperative document collection servers that provide a typical "search" interface that accepts text queries. To build the data set used by the resource selection component, the number of document hits in each collection for each potential query term is required. This data could, at least in principle, be constructed by searching each document collection one *basis term* (defined below) at a time through the search interface provided by the document collection server but in this experiment the indexed document collections were directly accessed to build the resource-selector data.

2.2.1 Vector representation of resources

The search resources, or back ends, in this experiment were Lucene indexes built from nearly 1,000 of the largest host-name document collections in the GOV2 corpus. *Collection size* is defined here as the number of unique documents contained in a document collection. The largest 1,000 host-name collections in GOV2 contain approximately 21 million documents out of a total of 25 million GOV2 documents. For each of the 10,000 topics in the 2008 MQ Track, collections were ranked in decreasing order of relevance to the query. Once the collections were ranked for each query, the ranked lists of collections were read by the Multisearch application and each search was sent to a restricted set of the top 50 relevant collections. Although the ranked

resource lists were pre-computed for all queries before using the Multisearch system to generate TREC result sets, it is straightforward to modify Multisearch to dynamically select resources as each query is received.

A set of Lucene-indexed host-name document collections for each fully qualified host name found in the document location fields of the GOV2 documents was available from prior TREC TB track experiments [12, 16]. A "term-document" matrix was constructed and saved for each of the $n = 996$ Lucene-indexed collections using Matlab, the Distributed Computing Toolkit, and Lucene through the Matlab-JAVA external interface on the *Midnight* Sun Opteron Cluster at ARSC. The row of each matrix corresponded to a document in the collection and each column corresponded to a term in a *term basis*. Matrix entries corresponding to a (document, term) pair were assigned '1' if the term occurred in the document and '0' otherwise.

2.2.2 Term basis

The term basis in one-to-one correspondence with the columns of each term-document matrix was constructed by taking the union of the English terms in the SCOWL wordlist [19] and removing the terms that did not retrieve any document in the n document collections. Note that variations in suffix or punctuation were not filtered from the term basis. The size of the filtered term basis was $p = 401,200$ terms so each document-term matrix corresponding to a document collection had the same number of columns (terms) as the other document-term matrices but the number of rows (documents) of each matrix varied with the number of documents in the corresponding document collection. Finally, the column sum of each matrix was taken to form a single row or *big document* vector for each collection and the row vectors were stacked to form a single *collection-term* matrix. The entries in each big document vector are the number of documents in the collection containing the corresponding term.

2.2.3 Resource ranking

Let $\tilde{\mathbf{A}}$ be the $n \times p$ collection-term matrix described above and $\mathbf{A}$ be the matrix formed by centering and normalizing the columns of $\tilde{\mathbf{A}}$ to unit length. A query $\mathbf{q}$ is represented by a $1 \times p$ row vector by setting the column entries corresponding with the query terms to one and setting all other entries to zero. Then a vector of cosine similarity scores can be calculated for the big-document collections relative to a query analogous to calculating a vector of document similarity scores in the canonical vector space model [17] by forming the vector-matrix product $\mathbf{s}_{\text{VSM}} = \mathbf{q}\mathbf{A}'$.

Following the notation and definitions of similarity scores summarized in [20], only substituting "collection big documents" for "documents," the similarity model can be augmented by introducing the term-correlation matrix $\mathbf{R}$ in the query-document inner product as in the generalized vector space model. The similarity model can be augmented yet again by replacing the term correlation matrix with the reduced rank k term-correlation matrix $\mathbf{R}_k$, as in Latent Semantic Indexing [18], and then by calculating the product $\mathbf{s}_{\text{LSI}} = \mathbf{q}\mathbf{R}_k\mathbf{A}'$. The matrix $\mathbf{R}_k = \mathbf{D}_k\boldsymbol{\Sigma}_k^2\mathbf{D}_k'$ is a product of matrices from the reduced rank-k singular value decomposition of the collection-term matrix $\mathbf{A} = \mathbf{T}_k\boldsymbol{\Sigma}_k\mathbf{D}_k'$ where $\mathbf{T}_k$ is the $n \times k$ orthogonal matrix of left singular vectors, $\boldsymbol{\Sigma}_k$ is the $k \times k$ diagonal matrix of singular values, and $\mathbf{D}_k$ is the $p \times k$ orthogonal matrix of right singular vectors.

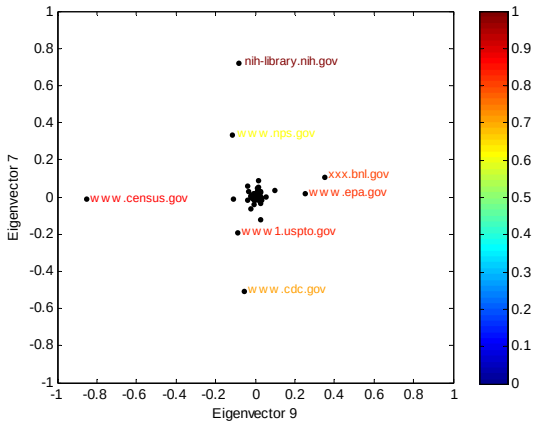


Figure 2. Projection of host-name collection big-document vectors onto eigenvectors 7 and 9 of the collection correlation matrix as calculated by the SVD of the basis term by big-document matrix. The number of documents in each collection is represented by color and a nonlinear color scale is used to enhance contrast.

2.2.4 Resource-level observations

The degree to which redundant information exists at all in the collection-term matrix is unknown, so it is not immediately clear that the dimension-reduction approach in LSI will aid in selecting document collections that are rich with documents relevant to a query. Furthermore, the specialization of web host-name document collections may not be sufficient to use a document-sum of keyword counts to identify collections containing documents relevant to a few query terms. Anecdotal evidence presented in Figure 2, a projection of the collection big-document vectors onto the vector space spanned by two of the largest (in the singular value sense) eigenvectors of the host-name collection correlation matrix, shows that a dimension reduction approach may be useful in classifying document collections using document sums of key words.

In particular note that collections with (presumably) many documents relating to human health, like a National Institutes of Health website *nih-library.nih.gov* and the Centers for Disease Control and Prevention main site *www.cdc.gov*, are the collection-vector components with the largest magnitude coefficients in eigenvector 7; while collections likely to contain many documents on the microbiological effects of chemical compounds, such as a Brookhaven National Laboratory site *xxx.bnl.gov* or the Environmental Protection Agency main site *www.epa.gov*, are among the largest coefficients of eigenvector 9. One important limitation to this approach is also evident: the large coefficient associated with the US Census Bureau site *www.census.gov* on the same axis defined by sites with pages on microbiological research is not readily explained by term associations. The largest coefficients of eigenvectors 5 and 7 of the collection correlation matrix and their associated collection names are listed in Table 1.

Note a few practical considerations when working with matrix factorizations of collection-term matrices similar to the one described here. In this experiment the sparse collection-term matrix **A** was about 10% dense so approximately 40×10^6 integers were required for storage but the full-rank matrices of left and right singular vectors are dense so the minimum storage required is about 4×10^8 floating-point numbers; or if double precision is used about 3.2 GB, near the current maximum available RAM in commodity workstations. The storage required to write the dense reduced-rank matrix factors is linear in the parameter *k*, set to 150 in this experiment, so the matrix factorizations required about 500 MB to store. About one hour of CPU time and 50 GB of RAM on a x4600 node of the Midnight cluster at ARSC were required to calculate the SVD of the collection-term matrix and the term-correlation matrix with Matlab R2007a. This calculation needs to be made only once unless additional documents or document collections are added to the system.

Eigenvector 7	Host name	Eigenvector 9	Host name
0.7238	nih-library.nih.gov	0.3486	xxx.bnl.gov
0.3340	www.nps.gov	0.2484	www.epa.gov
0.1064	xxx.bnl.gov	0.0974	www.nal.usda.gov
0.0929	www.fs.usda.gov	0.0511	fnalpubs.fnal.gov
0.0599	www.cr.nps.gov	0.0310	www-library.lbl.gov
…	…	…	…
-0.0306	usinfo.state.gov	-0.0586	www.cdc.gov
-0.0354	www.nlm.nih.gov	-0.0878	nih-library.nih.gov
-0.0621	www.uspto.gov	-0.0913	www1.uspto.gov
-0.1181	catalog.tempe.gov	-0.1121	landview.census.gov
-0.1907	www1.uspto.gov	-0.1175	www.nps.gov
-0.5084	www.cdc.gov	-0.8545	www.census.gov

Table 1. Coefficients of eigenvectors 7 and 9 of the collection correlation matrix calculated from the SVD of the term-collection matrix. The coefficients have been sorted in descending numerical order and listed next to the host-names of the associated vector components.

3. Experiment description and results

The TREC 2008 MQ results reported here are from the first use of the Multisearch demonstration system in a TREC task and are the first significant tests of the Multisearch resource selection component. The document collections were ranked using the resource selection algorithms described above and the ranked document collection lists were sent with the unmodified MQ queries to the Multisearch system for document search and retrieval. Most of the 996 indexed document collections were available to the Multisearch system at the time of the experiments, but the set of collections actually used varied somewhat from experiment to experiment due to minor technical problems.

Two types of experiments were performed: dynamic resource selection for each query and static resource selection for a composite query. In the dynamic resource selection runs, *lsi150dyn* and *vsmdyn*, a ranked list of document collections was generated for each MQ track query that was mapped to a vector in the vector space spanned by the term basis as described above. In the lsi150dyn experiment, the collections were ranked with respect to each query by calculating the latent semantic indexing similarity score $\mathbf{s}_{\mathrm{LSI}}$ for $k = 150$ dimensions and in the vsmdyn experiment the collections were ranked by calculating the canonical vector space similarity score $\mathbf{s}_{\mathrm{VSM}}$. For the static resource selection runs: *lsi150stat*, *vsmstat*, and *vsm07stat*; a composite query formed by mapping each MQ query to a vector in the term basis and taking the vector sum of the query vectors was used to generate a single ranked list of document collections for all of the MQ queries. The collections were ranked with this composite query by forming the respective vector-matrix products and the highest-ranked collections were searched for all MQ queries. In the lsi150stat and vsmstat experiments, the composite query was calculated from the 2008 MQ queries and in vsm07stat, the composite query was calculated using the 2007 MQ queries. Queries from a previous MQ track were used in the vsm07stat experiment to test the sensitivity of IR performance to the resource selection component of Multisearch.

3.1 Multisearch system performance

One issue in a large, distributed system is time required between entering a query and receiving results. Searching one large index can take time, but searching multiple indexes and merging results together can take even more time, causing a delay in retrieving results. In a world where users are used to results in less than a second, this can make GIR impossibly slow. For this reason, Multisearch uses restriction algorithms to limit the number of back-end services to be searched for a particular query. This decreases the time

spent sending information over the network, but we hope that it also reduces noisy data as well.

The TREC runs were all done with a limit of fifty back ends, which were selected by various restriction algorithms relative to the query. After the services were limited to fifty, each query took approximately three to five seconds to run. This includes time sending information back and forth over the network, merging the results, and printing the results to disk.

3.2 Resource selection performance

Estimating the performance of the resource selection component separate from the Multisearch system in a meaningful way is not possible using the estimated AP alone due to the dependence of the metric on the underlying document retrieval algorithm. However, the performance of the resource selection component could be estimated at the relevance judgment (qrel) level, for instance, by treating each ranked list of document collections as "hits" and judging each document-collection hit as relevant if the collection contains a relevant qrel. Then several document-retrieval performance measures such as AP could be extended naturally to include resource-selection performance under the assumption that the resource selection performance will serve as an upper bound of the retrieval performance of the Multisearch system as a whole. (See [16] for an analysis of the distribution of TREC Terabyte track qrels among the GOV2 host-name collections.) Estimating the performance of the resource selector is an area of future work but a few descriptive parameters relevant to the resource selection algorithm used here are summarized below.

Each document collection is represented as a "big document" vector in a fixed vector space of terms so the distribution of the number of nonzero elements in each big-document vector (plotted in Figure , central tendency summarized in Table 2) is important, combined with the distribution of collection sizes [13], in determining how likely a random query will be relevant to any particular collection and how

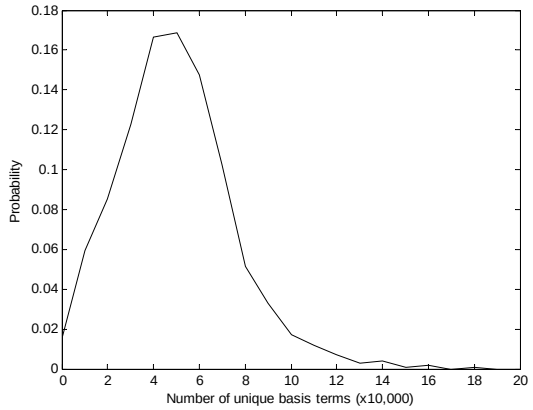


Figure 3. Probability distribution function of the number of unique basis terms drawn from the SCOWL wordlist contained in each host-name document collection.

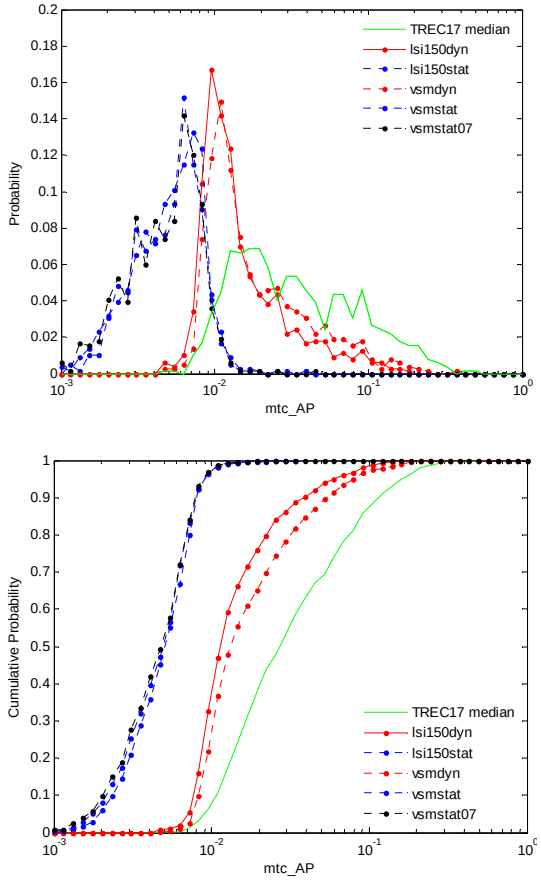


Figure 4. Probability distribution function (top) and cumulative probability distribution function (bottom) of the minimal test collection estimated AP from the ARSC MQ results and TREC participant median results. Plotted points represent centers of the logarithmically spaced and sized AP bins.

likely any two collections will be similar to each other in the sense of the inner product.

About 50,000 terms from the SCOWL wordlist are contained in at least one document in each collection and about 400,000 terms are contained in at least one document in the largest 1,000 host-name collections in GOV2. Consequently, each big-document vector is approximately 10% dense. No attempt was made to stem the wordlist before forming the term basis so the stemmed term counts will be less than reported here but the matrix density is not likely to change significantly.

3.3 Document retrieval performance

The minimal test collection estimated average precision (mtcAP) and (statAP) performance metrics applied to the five experiment runs submitted by ARSC are summarized in Table 3. The probability distribution and cumulative probability distribution functions of mtcAP from the ARSC 2008 MQ experiments are plotted in Figure along with the estimated distributions of the TREC participant median performance. As in prior years, the ARSC system performance is below the TREC median, due in part to the additional difficulty involved in a federated search and also to the relative weakness of the base "demonstration" search application distributed with the Lucene Toolkit.

The main result is that the dynamic resource selection runs, lsi150dyn and vsmdyn, perform significantly better than the static resource selection runs, lsi150stat, vsmstat, and vsm07stat. There is no clear performance advantage to either the latent semantic indexing or the canonical vector space model methods of ranking document collections in either of the dynamic or static resource selection experiments. While the vector space model appears to perform slightly better than latent semantic indexing, the set of document collections used in the two experiments differed by about 10 collections and could result in a slight difference in retrieval performance alone. The dynamic resource selection runs did not perform as well as the historical resource selection run *ffind07d* [16] in the 2007 MQ track, where the collections were ranked according to the number of relevant TB track qrels contained in each collection; however, about twice as many collections and documents were searched in ffind07d than in any of the experiment runs reported here.

A secondary result is that selecting document collections to search from the top of a list of document collections ranked relative to an aggregate of several thousand queries is not effective. Note that the run vsm07stat, where the collections were ranked relative to an aggregate query formed from the MQ 2007 queries performed just as well as the static runs where the aggregate query was formed from the same set of queries that was used to search the collections. This could imply that there is very little variation in the total term counts of the MQ 2007 and the MQ 2008 queries or that the set of collections relevant to the aggregate query is not particularly relevant to any particular query.

	Mean	Median	Standard deviation
Basis terms per collection	49,694	48,665	25,526
Collections containing each term	123	21	221
Documents containing each term	15,183	103	146,250

Table 2. Average number of SCOWL basis terms per host-name document collection used in the experiment. Also listed is the average number of document collections and documents containing each basis term.

Run tag	Mean (x10^{-2})	Median (x10^{-2})	Std. (x10^{-2})	Max (x10^{-2})	Min (x10^{-2})
lsi150dyn	2.052	1.230	2.507	36.860	0.450
Vsmdyn	2.655	1.415	3.228	36.900	0.490
lsi150stat	0.568	0.540	0.294	4.140	0.110
Vsmstat	0.546	0.530	0.292	4.120	0.090
vsm07stat	0.534	5.150	0.297	4.390	0.080
TREC worst	0.317	0.285	0.228	1.150	0.001
TREC median	5.197	2.880	5.785	46.390	0.500
TREC best	10.462	6.780	9.694	58.450	0.830

Table 3. Minimal test collection estimated AP for the ARSC MQ experiment runs and the TREC participant summary statistics.

4. Summary and Future Work

The Multisearch system is currently performing at three to five seconds per query, but this is still a long wait for results. Furthermore, the searches were over 50 services, rather than the possible 996 services. One possible exploration is the best number at which to limit the searches. Multisearch is also tied down to Tomcat and has been for quite a while. Multisearch 3.0 aims to be separate from web services, although include them as an additional (but not required) middleware. Additional back end search capability is needed, as the Lucene demonstration search application is a significant limiting factor of the Multisearch retrieval performance.

To improve IR performance, the Indri search engine, part of the Lemur toolkit, will be added Multisearch back-end search service. The size of the term basis used for the resource selection component adversely affects memory and compute time usage when performing matrix factorizations and computations on the term-collection matrix or the term correlation matrix. Stemming the term basis, compiled from the SCOWL wordlist, is the first step to making the singular value decomposition of the collection-term matrix more accessible to typical commodity workstations. Additional theoretical work could be pursued on incorporating collection-ranks in the document result-set merging step. The sensitivity of document retrieval performance to the dimension parameter k in Latent Semantic Indexing similarity ranking will be explored.

5. References

1. Meng, W., C.T. Yu, and K.-L. Liu, *Building efficient and effective metasearch engines.* ACM Computing Surveys, 2002. **34**(1): p. 48-49.
2. Si, L., *Federated Search of Text Search Engines in Uncooperative Environments*, in *Language Technology Institute*. 2006, Carnegie Mellon University.
3. Thomas, P. and D. Hawking. *Evaluating Sampling Methods for Uncooperative Collections*. in *SIGIR*. 2007. Amsterdam, The Netherlands: ACM.
4. Gamiel, K., G.B. Newby, and N. Nassar. *Grid Information Retrieval Requirements (GFD.27)*. in *Global Grid Forum*. 2003. Lamont, IL.
5. Voorhees, E., *Overview of TREC 2005*, in *The Fourteenth Text REtrieval Conference*. 2005, NIST Special Publication: Gaithersburg, Maryland, USA.
6. Clarke, C.L.A. and F. Scholer, *The TREC 2005 Terabyte Track*, in *The Fourteenth Text REtrieval Conference*. 2005, NIST Special Publications: Gaithersburg, Maryland, USA.
7. Calvé, A.L. and J. Savoy, *Database merging strategy based on logistic regression.* Information Processing and Management, 2000. **36**(3): p. 341-359.
8. Fallen, C.T. and G.B. Newby, *Logistic Regression Merging of Amberfish and Lucene Multisearch Results*, in *The Fourteenth Text REtrieval Conference*. 2005, NIST Special Publications: Gaithersburg, Maryland, USA.
9. Nassar, N., *Amberfish at the TREC 2004 Terabyte Track*, in *The Thirteenth Text REtrieval Conference*. 2004, NIST Special Publications: Gaithersburg, Maryland, USA.
10. Voorhees, E. *Overview of TREC 2006*. in *15th Annual Text REtrieval Conference*. 2006. National Institute of Standards and Technology: NIST.
11. Buttcher, S. and C.L.A. Clarke. *The TREC 2006 Terabyte Track*. in *15th Annual Text REtrieval*

Conference (TREC 2006). 2006. National Institute of Standards and Technology: NIST.
12. Fallen, C.T. and G.B. Newby, *Partitioning the Gov2 Corpus by Internet Domain Name: A Result-set Merging Experiment*, in *The Fifteenth Text REtrieval Converence*. 2006, NIST Special Publications: Gaithersburg, Maryland, USA.
13. Fallen, C.T. and G.B. Newby, *Distributed Web Search Efficiency by Truncating Results*, in *JCDL '07*. 2007, ACM: Vancouver, British Columbia, Canada.
14. Voorhees, E. *Overview of TREC 2007*. in *16th Annual Text REtrieval Conference*. 2007. National Institute of Standards and Technology: NIST.
15. Allan, J., et al. *Million Query Track 2007 Overview*. in *16th Annual Text REtrieval Conference*. 2007. National Institute of Standards and Technology: NIST.
16. Fallen, C.T. and G.B. Newby. *Collection Selection Based on Historical Performance for Efficient Processing*. in *16th Text REtrieval Conference (TREC2007)*. 2007. National Institute of Standards and Technology: NIST.
17. Wong, S.K.M., W. Ziarko, and P.C.N. Wong, *Generalized vector spaces model in information retrieval*, in *Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval*. 1985, ACM: Montreal, Quebec, Canada.
18. Deerwester, S., et al., *Indexing by Latent Semantic Analysis.* Journal of the American Society for Information Science, 1990. **41**(6): p. 391-407.
19. Atkinson, K. *Kevin's Word List Page*. 2007 Aug, 2008]; Available from: http://wordlist.sourceforge.net/.
20. Efron, M., *Eigenvalue-based model selection during latent semantic indexing.* Journal of the American Society for Information Science and Technology, 2005. **56**(9): p. 969-988.

Distributed multisearch and resource selection for the TREC Million Query Track

Chris Fallen and **Greg Newby**
Arctic Region Supercomputing Center, University of Alaska Fairbanks, Fairbanks, AK 99775

Kylie McCormick
Mount Holyoke College, South Hadley, MA 01075

Abstract

A distributed information retrieval system with resource-selection and result-set merging capability was used to search 1000 subsets of the GOV2 document corpus for the 2008 TREC Million Query Track. The GOV2 collection was partitioned into host-name subcollections and the largest collections were distributed to multiple remote machines. The Multisearch demonstration application restricted each search to a fraction of the available subcollections that was predetermined by a resource-selection algorithm. Experiment results from topic-by-topic resource selection and aggregate topic resource selection are compared. The sensitivity of Multisearch retrieval performance to variations in the resource selection algorithm is discussed.

TREC 17 | 18-21 November 2008 | NIST | Gaithersburg, Maryland

ARSC Multisearch 2008 System

Interactive or batch input of search topics and restrictions of data resource sets

Single search interface to an inhomogeneous collection of data search services

1000 data resources available

Each data resource is a host-name subcollection of the GOV2 corpus

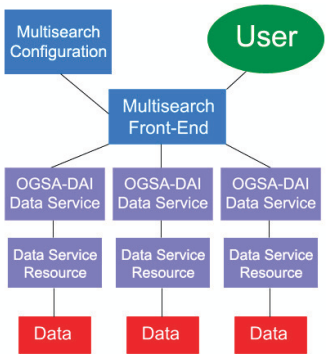


Figure 1. Schematic of the ARSC Multisearch system. The resource selector is contained in the Multisearch front-end block and the host-name document collections are contained in the data blocks.

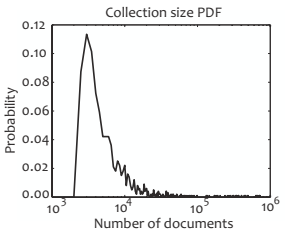


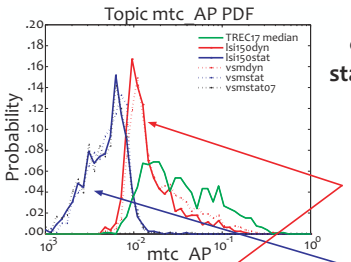
Figure 3. Probability distribution function of the number of documents in each data collection accessible from Multisearch. GOV2 host-name subcollections containing fewer than about 2000 documents were not included.

Documents are represented as row vectors in a fixed vector space of about 600,000 terms drawn from the SCOWL English word list. All of the documents in a subcollection are represented as a matrix of document row vectors.

$$\begin{pmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \\ \vdots \\ \mathbf{d}_{n_{c_j}} \end{pmatrix} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & 0 \end{pmatrix} \xrightarrow{\text{sum column entries}} \begin{pmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_j \\ \vdots \\ \mathbf{c}_m \end{pmatrix} = \begin{pmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \cdots & \mathbf{t}_p \\ * & * & \cdots & * \\ * & * & \cdots & * \\ 11 & 10 & \cdots & 79 \\ * & * & \cdots & * \\ * & * & \cdots & * \end{pmatrix} =: \mathbf{A} \xrightarrow[\text{find the rank-}k\text{ SVD}]{\text{center and normalize columns,}} \mathbf{T}_k \mathbf{\Sigma}_k \mathbf{D}_k'$$

A search query is also represented as a row vector in the vector space of terms.

$$\mathbf{q} = \begin{pmatrix} 1 & 0 & \cdots & 1 \end{pmatrix}$$



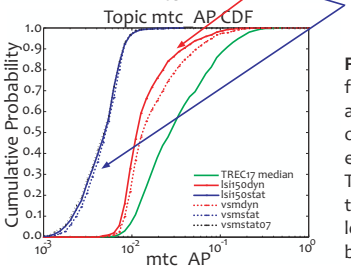


Figure 2. Probability distribution function (top) and cumulative probability distribution function (bottom) of the minimal test collection estimated AP from the ARSC MQ results and TREC participant median results. Plotted points represent centers of the logarithmically spaced and sized AP bins.

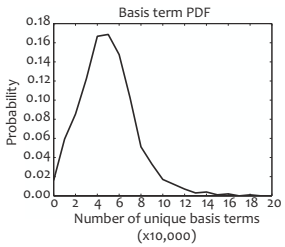
A document subcollection is represented as the sum of its document vectors. All of the subcollections accessible to Multisearch are represented as a matrix of subcollection row vectors. Document subcollections are relevance ranked for each query or for an aggregate of many queries.

Aggregate IR performance comparison of dynamic and static "big document" resource selection

Dynamic (topic-by-topic) resource selection

Static (fixed average topic) resource selection

Figure 4. Probability distribution function of the number of unique basis terms drawn from the SCOWL word list contained in each document collection.



Term correlation matrix $\mathbf{R}_k = \mathbf{D}_k \mathbf{\Sigma}_k^2 \mathbf{D}_k'$

Classic Vector Space Model (VSM) $\mathbf{s}_{\text{VSM}} = \mathbf{q}\mathbf{A}'$

Latent semantic indexing (LSI) $\mathbf{s}_{\text{LSI}} = \mathbf{q}\mathbf{R}_k\mathbf{A}'$

Eigenvector representation of document *collections*

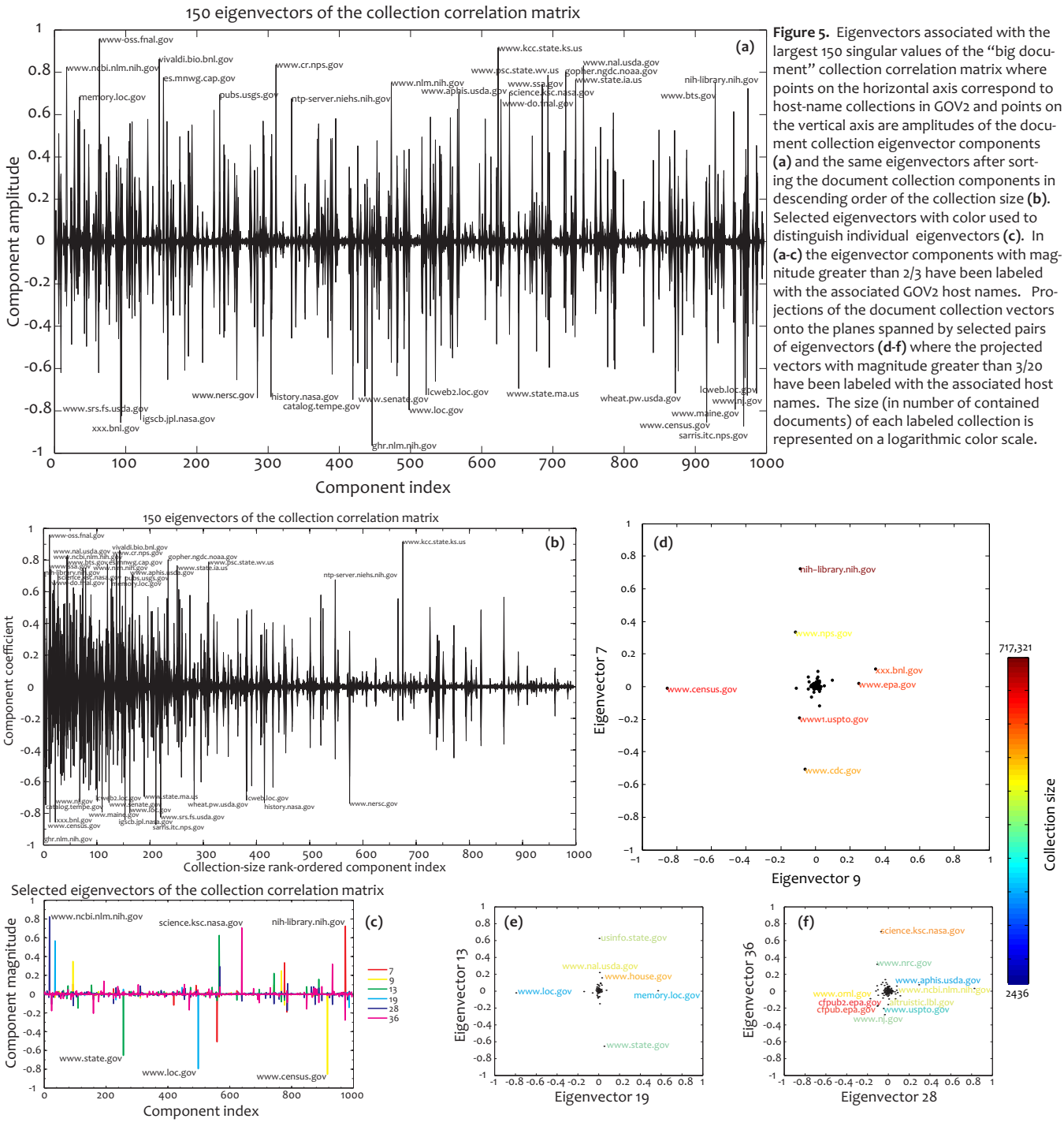


Figure 5. Eigenvectors associated with the largest 150 singular values of the "big document" collection correlation matrix where points on the horizontal axis correspond to host-name collections in GOV2 and points on the vertical axis are amplitudes of the document collection eigenvector components **(a)** and the same eigenvectors after sorting the document collection components in descending order of the collection size **(b)**. Selected eigenvectors with color used to distinguish individual eigenvectors **(c)**. In **(a-c)** the eigenvector components with magnitude greater than 2/3 have been labeled with the associated GOV2 host names. Projections of the document collection vectors onto the planes spanned by selected pairs of eigenvectors **(d-f)** where the projected vectors with magnitude greater than 3/20 have been labeled with the associated host names. The size (in number of contained documents) of each labeled collection is represented on a logarithmic color scale.

TREC 17 18-21 November 2008 NIST Gaithersburg, Maryland

Summary

Vector representation of document collections may be used to improve minimally cooperative federated search efficiency by selecting a fraction of available collections likely to contain relevant documents.

Retrieval performance of federated search that is constrained to a fraction of available collections for each query may be improved by restricting each search using variants of the vector space similarity model (VSM) or latent semantic indexing (LSI).

Targeted federated search using VSM or LSI to relevance-rank document collections to each query yields comparable retrieval performance surpassing the performance of federated search on a set of document collections relevance-ranked to a single aggregate query.

ARSC
HPC Modernization Program
UAF UNIVERSITY OF ALASKA FAIRBANKS