Siemens TREC-4 Report: Further Experiments with Database Merging

Ellen M. Voorhees
Siemens Corporate Research, Inc.
Princeton, NJ
ellen@scr.siemens.com

Abstract

A database merging technique is a strategy for combining the results of multiple, independent searches into a single cohesive response. An isolated database merging technique selects the number of documents to be retrieved from each database without using data from the component databases at run-time. In this paper we investigate the effectiveness of two isolated database merging techniques in the context of the TREC-4 database merging task. The results show that on average a merged result contains about 1 fewer relevant document per query than a comparable single collection run when retrieving up to 100 documents.

1 Introduction

Siemens has used TREC-4 to continue its investigation of the *collection fusion* or *database merging* problem. Informally, the database merging problem is to combine the retrieval results from multiple, independent databases into a single result that has the best possible effectiveness. Such a search is necessary in a variety of distributed IR settings, with the setting determining the kinds of data available to the merging strategies. We assume the merging process is dispatched by an entity that has no control over the individual databases. Therefore, we assume the only information the merging algorithm can obtain from a collection is a ranked list of documents in response to a query. We call merging strategies that have no other access to the individual databases *isolated* merging strategies. In contrast, the methods explored by Callan, Lu, and Croft [3] assume access to particular data items (e.g., word frequencies) within the individual databases. We call these strategies *integrated* merging strategies.

Since integrated strategies have access to more information, they can be expected to be more effective than isolated strategies. While in principle even isolated strategies can produce merged results that are more effective than the result obtained when searching the entire set of documents as a single collection [6], in practice the merged results produced by isolated strategies have been less effective than the single collection run. Our main goal is to minimize this degradation in isolated merging strategies.

TREC-4 contains a database merging track. The track defined the set of component databases to be searched, and stipulated that a single collection run be made to serve as a directly comparable baseline. Our single collection run, `siems1`, is also our ad hoc submission. Siemens runs `siems2` and `siems3` are database merging runs created using two different isolated merging strategies. Each of these runs is described in more detail below. Siemens did not perform any routing runs, nor did it participate in any other tracks.

The results of the experiments show a small degradation in the effectiveness of the merging runs as compared to the single collection run for moderate numbers of retrieved documents. Using the average of the precision after 20 documents are retrieved as the measure of effectiveness, the database merging runs are 10% and 13% less effective than the single collection run. That is, on average the merged runs find approximately 3/4 of a document fewer relevant documents in the top 20 retrieved documents per query than does the single collection run. After 100 documents retrieved, the percentage decreases are 12% and 17%. The average non-interpolated average precision over all relevant documents (with 1000 documents retrieved per query) degrades by 29% and 38%. Since users are generally interested in only a relatively small number of top-ranked documents, these techniques offer a viable solution to the database merging problem.

The next section describes Siemens's retrieval environment in general and the specific settings used for our TREC-4 runs. The following section provides a more detailed comparison between the effectiveness of the merged and single collection runs. Section 4 discusses the size of the data structures required to support the merging algorithms and the resulting efficiency of the merged search: since isolated strategies use no data from the individual collections to select the databases participating in the current search, and since the query is submitted only to those collections from which documents are desired, these merged searches are quite efficient. The final section explores some of the challenges created by the database merging task, and lists areas that still need to be addressed.

2 Merging and Retrieval Methods

Both of the database merging techniques used in this work use relevance assessments from past queries to select the number of documents to request from each database for the current query. If a non-zero number of documents, λ, is to be retrieved from a given database, the (natural language) query is submitted to that database and the λ most highly ranked documents are returned. The following datasets are thus required to test the effectiveness of these database merging methods.

A set of component databases. The database merging track chose the ten document sets contained on TREC disks two and three to be the set of databases to be searched. This choice was motivated by the fact that the union of the component databases is the set of documents to be used for the TREC-4 ad hoc task. The ten databases include: two AP newswire collections (1988 and 1990); a *Federal Register* collection; a set of U.S. patent disclosures; a *San Jose Mercury News* collection; three *Wall Street Journal* collections (1990, 1991, and 1992); and two collections of extracts from Ziff-Publishing's *Computer Selects* disks.

A set of training queries. We used TREC topics 1–200 as our training queries.

Training query retrieval results. Since the merging algorithms rely on ranked lists of documents, annotated with relevance data, to compute the number of documents to retrieve from a database, the training queries must be run against the individual databases and the retrieved relevant documents marked as such. The TREC collection does not contain relevance assessments for disk3 for topics 1–50 and 151–200, so some collections have more training queries than others. One of the objectives of this study is to investigate the performance of the merging strategies when the set of component databases has differing training data.

We use the SMART retrieval system from Cornell [1] as our underlying retrieval engine. In particular, we used the massive query expansion technique that produced good results for Cornell in TREC-3 [2]. The training query results were created by performing an initial run using 'lnc'-weighted document vectors and 'ltc'-weighted query vectors. The vectors were formed using the standard SMART indexing procedures, and they include both single terms and phrases. The top 15 retrieved documents for each query were *assumed* to be relevant (the actual relevance data was not used in this step), and were used to perform Rocchio feedback on the initial query. During the Rocchio feedback, the initial query was expanded with at most 100 new single terms and at most 10 new phrases; the Rocchio parameters were set to $\alpha = 8$, $\beta = 8$, and $\gamma = 0$. Once created, the newly expanded query was run against the collection to produce the retrieval results used in training.

A set of test queries. TREC topics 202–250 were the test topics.

Test query retrieval results. To form the actual merged result, the test queries must also be run against the individual databases. In this case, however, no relevance data is required (except for evaluation). The test query retrieval results were generated in the same way as the training query results except that a maximum of 500 single terms could be added to query. To conform with the TREC task requirements, 1000 documents were retrieved for each test query.

In addition to expanding the queries in each of the component database, queries 202–250 were expanded (with a maximum of 500 single terms) and run against the collection formed from the entire set of documents. This run is our ad hoc run, `siems1`, and provides a point of comparison for the merged runs.

Siemens run `siems2` was created using the Query Clustering (QC) merging technique, and run `siems3` was created using the Modeling Relevant Document Distributions (MRDD) merging technique. These are the same merging techniques used in our earlier work [4, 5, 6]. While the two techniques require the same basic datasets, described above, they use the training data in different ways. These differences are described in the following subsections.

2.1 Query Clustering

The basic steps in the query clustering merging method are presented in Figure 1. The training phase is depicted in Step 1 of the figure. For each database, the set of training queries that actually have relevance data in that database is clustered. The clusters are produced using Ward's algorithm and the inverse of the number of documents retrieved in common between two queries in the top 1000 documents as the distance metric. Query vectors are created in a vector space formed from the set of training queries, and the vectors of the queries contained within each cluster are averaged to create cluster centroids. (These are *unexpanded* query vectors — there are no documents to expand by.) Each cluster is also assigned a weight that reflects how effective queries in the cluster are on that database. The weight is computed as the average number of relevant documents retrieved by queries in the cluster, where a document is considered to be retrieved if it is in the top 100 documents.

Steps 2 and 3 of Figure 1 depict how the merged result is created for new queries. The cluster whose centroid vector is most similar to the query vector is selected for the query and the associated weight is returned. The set of weights returned by all the collections is used to apportion the retrieved set: when N documents are to be returned to the user and w_i is the weight returned by collection i, $\frac{w_i}{\sum_{i=1}^{C} w_i} * N$ documents are retrieved from collection i. The final ranking of the documents retrieved from each database is produced by a random process. To select the document for rank r, a collection is chosen by rolling a C-faced die that is biased by the number of documents

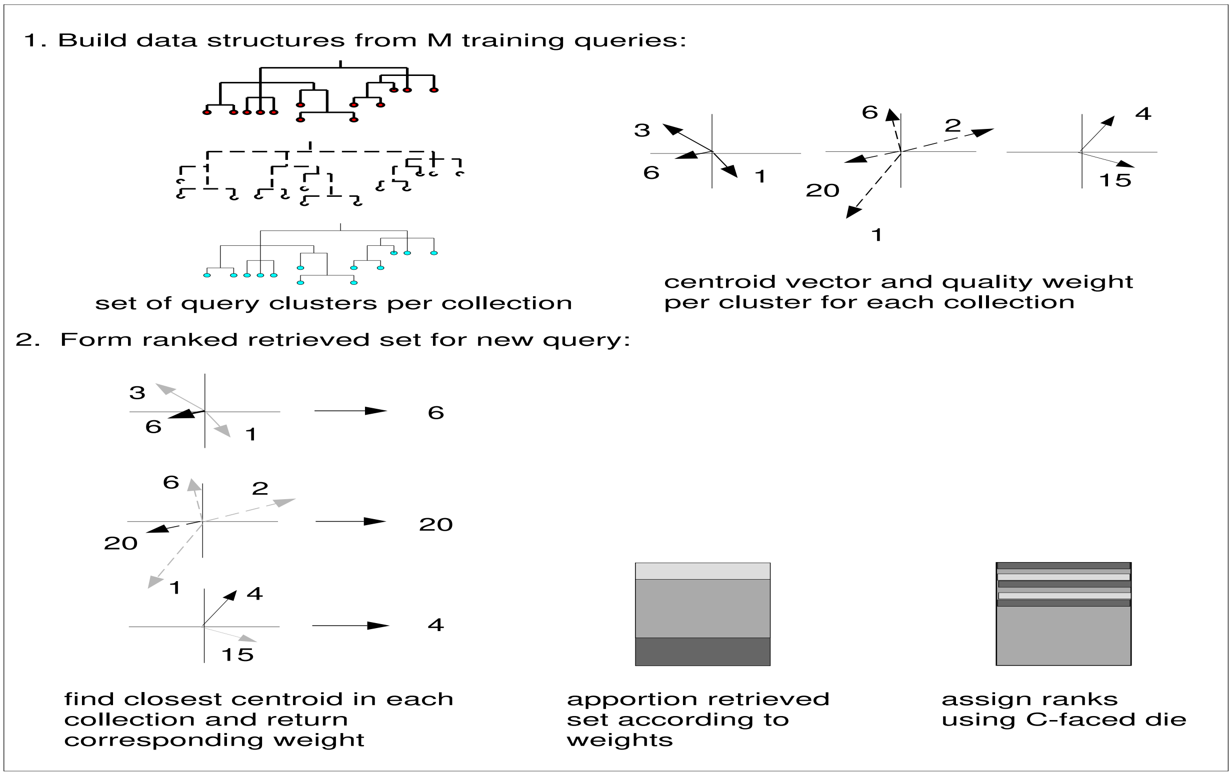

Figure 1: The QC database merging strategy.

still to be picked from each of the C collections. The next document from that collection is placed at rank r and removed from further consideration.

2.2 Modeling Relevant Document Distributions

Figure 2 summarizes the steps of the second database merging technique, modeling relevant document distributions (MRDD). Once again, the first step is a training phase. One set of (unexpanded) query vectors is created in a vector space constructed from all of the training queries. The system also stores an explicit representation of the relevant document distribution for each query in each database. This distribution is equivalent to the ranks of the relevant documents in the top 1000 retrieved.

The first step in processing a new query, q, is to determine the six training queries that are most similar to it. A model of q's retrieval behavior in each collection is constructed by averaging the relevant document distributions of these six nearest neighbors. However, since some databases do not have relevance assessments for all training queries, the average distribution in each database is actually computed over the set of nearest neighbors that have relevance data for that database, which may be less than six. Using the model distributions to predict the number of relevant documents that would be retrieved for q from each of the databases at different cut-off levels, the system computes the number of documents to retrieve from each collection such that the total number of relevant documents that would be retrieved is maximized. The "spill", the number of documents the maximization procedure computes will have no effect on the total number of relevant documents retrieved and may thus come from any collection, is distributed among the databases

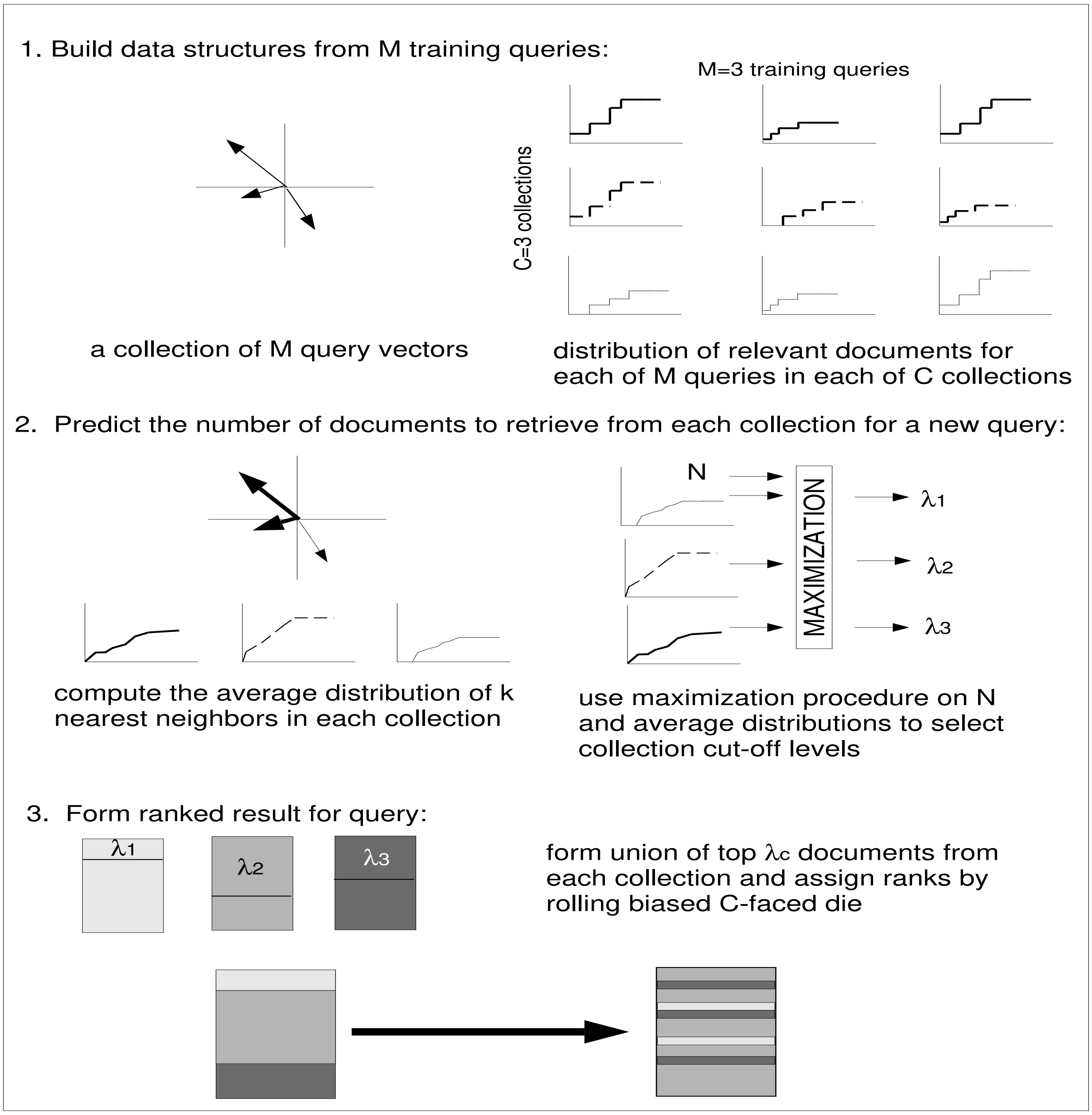

Figure 2: The MRDD database merging strategy.

	Ave. value	# best	# $\geq$ median	# worst
Relevant retrieved in top 100	24.65	0	30	1
Relevant retrieved in top 1000	77.3	7	38	0
Average precision	.2031	0	26	0

Table 1: Effectiveness of the ad hoc run `siems1` as compared to other TREC-4 ad hoc runs.

in proportion to the number of documents that would be otherwise retrieved from it. The final ranking of the retrieved documents is produced by the same procedure as is used in the QC method.

The maximization procedure used above is an NP-complete optimization problem. In previous experiments with MRDD, a simple exhaustive search was efficient enough because the optimization entailed a small number of documents to be retrieved and/or a small number of databases. Unfortunately, with the TREC-4 submission deadline looming, we discovered that 1000 documents to be retrieved from a subset of 10 databases was prohibitively time-consuming to run. (And the same time pressures prevented an implementation of a more efficient optimization procedure.) Thus run `siems3` was produced by telling the optimization procedure that 50 documents were to be retrieved and then multiplying the resulting number of documents to retrieve from each database by 20 to obtain a total of 1000 documents. Note that this is unlikely to seriously degrade the performance of the MRDD method. Previous experiments have shown that the model distributions are most accurate in the range of 20–100 documents to be retrieved [4, 5].

3 Retrieval Effectiveness

This section reports on the effectiveness of our retrieval runs. The single collection run `siems1` meets the requirements of the TREC-4 ad hoc task (Topics 202–250 run against Disks 2 and 3), so the first subsection compares its effectiveness to the other ad hoc runs. The remainder of the section explores the effectiveness of the merged runs by comparing their effectiveness to that of `siems1` and a variety of other baseline merging techniques.

3.1 Single Collection Results

Our ad hoc run `siems1` is a completely automatic, single collection run that expanded queries as described above. The use of query expansion was motivated by our desire to have the searches of individual databases be as effective as possible, and to make comparisons meaningful we used the same technique for the single collection run. As the SMART group demonstrated in TREC-3 [2], query expansion improves retrieval performance by providing much more context in the queries. The context is created by adding the terms that occur in the documents retrieved in an initial phase to the newly expanded query vector.

The effectiveness of `siems1` as compared to the other TREC-4 ad hoc runs is summarized in Table 1. The first column in the table gives the value obtained by `siems1` averaged over the 49 ad hoc queries. The remaining columns give the number of queries for which `siems1` obtained the best, the worst, and an above-median score.

In general, `siems1` is an effective run, being at or above the median for a majority of the queries for all three effectiveness measures. (The one worst score was a query for which no relevant documents were retrieved when the median was one relevant document retrieved.) Unsurprisingly,

	Prec(20)		Prec(100)		Prec(1000)		Average Precision	
Single Collection (`siems1`)	.3265	—	.2465	—	.0773	—	.2031	—
QC Merging (`siems2`)	.2949	–10%	.2167	–12%	.0671	–13%	.1433	–29%
MRDD Merging (`siems3`)	.2847	–13%	.2053	–17%	.0536	–31%	.1253	–38%

Table 2: The effectiveness of the QC and MRDD merged results as compared to the single collection results.

the results demonstrate that the massive query expansion is a recall-oriented procedure: the queries tend to retrieve many relevant documents, but those documents are not always highly ranked. The `siems1` run retrieved the most relevant documents in the top 1000 documents for seven queries, but the non-interpolated average precision was always much closer to the median.

Some queries were adversely affected by the automatic expansion procedure. This happened when short, non-relevant documents contained a key term of the query and were thus ranked highly in the initial set of retrieved documents. The automatic expansion based on these documents led the subsequent search in the wrong direction. For example, the retrieval performance of Query 248 in `siems1` is much worse than the median performance. The text of Topic 248 is *What are some developments in electronic technology being applied to and resulting in advances for the blind.* Unfortunately, the *Wall Street Journal* has a number of very short earning reports for the company Electronic Technology. As a result, the final query had far more to do with finance than with blindness.

3.2 Database Merging Results

The single collection run provides one benchmark for the effectiveness of the database merging runs. As mentioned above, our goal is to have the effectiveness of the merged results match those of the single collection run. Table 2 gives the effectiveness of the single collection and merged runs averaged over the 49 queries. Effectiveness is measured in terms of the precision after 20, 100, and 1000 documents have been retrieved as well as the non-interpolated average precision. For the merged runs, the percentage difference over the single collection run is also given. As the non-interpolated average precision figures demonstrate, the merged runs are clearly less effective at ranking documents when large numbers of documents are retrieved than is the single collection run. The total number of relevant documents retrieved is much less severely degraded. Fortunately, the effectiveness is best at the smaller numbers of retrieved documents, which is the area most likely to be of concern to the typical user.

An important difference in these results is that the QC method is more effective than the MRDD method (the reverse was true in previous experiments). The MRDD method makes use of much more of the training data than the QC method: it stores and exploits the entire rankings of the training queries rather than summarizing their performance in a set of weights. Theoretically, this should lead to better performance for MRDD, and indeed that had been true [4]. However, such reliance on the training queries makes the method more susceptible to differences between the training and test queries. The topics in TREC-4 were much shorter than in previous years, and the subject matter of some topics did not always have corresponding training queries. In these cases, any test query words that just happened to be in training queries caused the resulting query-query similarity to be relatively large. For example, the text of Topic 224 is *What can be done to lower*

	Prec(20)		Prec(100)		Prec(1000)	
Single Collection (`siems1`)	.3265	—	.2465	—	.0773	—
Uniform	.2235	–32%	.1624	–34%	.0662	–14%
Optimal	.5663	+73%	.3478	+41%	n/a	—
AP Only	.3255	0%	.2173	–12%	.0458	–41%
QC Merging (`siems2`)	.2949	–10%	.2167	–12%	.0671	–13%
MRDD Merging (`siems3`)	.2847	–13%	.2053	–17%	.0536	–31%

Table 3: The effectiveness of the merged runs in comparison to a variety of benchmarks.

blood pressure for people diagnosed with high blood pressure? Include benefits and side effects. The most similar query to 224 matched only on *effect* and *includ*; the next most only on *diagnos*; and the third most similar on *side*, *high*, and *people*. As might be expected, none of these queries had anything to do with blood pressure. The query about diagnosis was vaguely medicine related, asking about computer programs that aided in medical diagnosis. As a result, MRDD retrieved 360 documents from the Ziff3 collection, which contains no relevant documents. The QC merging technique's more crude representation of topic areas makes it more robust against these types of errors.

There are several other benchmarks the merged runs can be compared against to obtain a fuller understanding of their effectiveness. Effectiveness measures for these baselines are given in Table 3. The *optimal* run uses relevance information to compute the best possible merged result given the retrieval results for the individual collections. As in previous experiments, the optimal merged run is significantly more effective than the single collection run. The *uniform* run retrieves an equal number of documents from each collection. This is essentially a straw-man benchmark. The uniform strategy is the best strategy to use in the absence of any training data; a viable merging strategy should be more effective than the uniform run. Since the uniform run is approximately as effective as the merged results after 1000 documents are retrieved, the behavior of the merging strategies at that large of number of retrieved documents is probably meaningless. The *AP-only* run retrieves half its documents from the AP88 collection and half from AP90. In previous experiments with the TREC collection, the queries exhibited a large bias towards the AP collection [6]. The results in Table 3 demonstrate that this bias exists for the TREC-4 queries as well.

This bias complicates the interpretation of the retrieval results. Learning to retrieve a majority of documents from the AP collection is a relatively simple thing to do, and the QC method learned to do just that. The QC method retrieved a majority of its documents from the AP collections for a sizable majority of the 49 queries. It also learned to completely ignore the patent and *Federal Register* collections. In these collections, the training queries clustered into a single large cluster that was assigned a weight of 0. New queries could therefore never retrieve documents from these collections. Such a strong bias against these collections is perfectly understandable given the relevance assessments for Topics 1–200.

4 Efficiency of Merging Techniques

The database merging track definition requires participants to report the size of data structures built from training data and the amount of data from the component databases that is used at run time to decide how many documents to retrieve from each database. We assume there is no interaction with the databases at run time to decide how many documents to retrieve, so the latter amount is zero for both the QC and MRDD merging strategies.

The QC merging technique must store the cluster centroids and the weight assigned to each cluster for each database. The cluster weights are completely dominated by the size of the centroid vectors. In our experimental environment, we do not store the centroid vectors themselves, but instead store the query vectors and recompute the centroids each time. The SMART vectors for queries 1-200 are approximately 800,000 bytes per database.

The MRDD merging technique has greater space requirements. The MRDD method must store an inverted file and dictionary for the collection consisting of the queries plus a list of the ranks of the relevant retrieved documents for each training query in each database. Our experimental setup (accidentally) uses a much larger than necessary dictionary and inverted file for the query collection. However, the inverted file for the queries contains 8,612 entries, so, assuming a 16 byte entry size, the inverted file would require at least 137,792 bytes. The dictionary contains 5828 terms, so its size would need to be at least $(5828 * 8) +$ *sum of the lengths of the character strings* bytes. The size of the data structure containing the ranks of the relevant retrieved documents obviously depends on the number of queries for which there is training data and the number of relevant documents per training query. The size of the relevance data for AP88 — a database that has relevance assessments for all 200 training queries and a larger than average number of relevant documents — is approximately 115,000 bytes.

The MRDD method requires more processing time than the QC method in addition to having greater space requirements. MRDD must solve an optimization problem each time it executes a query, while the QC method only needs to do a simple best match search in each collection to find the appropriate centroid and then a few arithmetic operations on the returned weights to compute the final number of documents to retrieve. However, since neither method has to communicate with the component databases to decide how many to retrieve, and since the computation of how many documents to retrieve will likely eliminate most databases from consideration, both methods are likely to be sufficiently quick in practice.

5 Conclusion

TREC-4 provided an opportunity to test our two database merging strategies on a new set of queries and, for the first time, in an environment where there were different amounts of training data for different databases. As in previous experiments, the effectiveness of the merged results was within 15% of the effectiveness of a single collection run when evaluated at moderate numbers of retrieved documents. The lack of relevance assessments for some queries in some databases had no obvious effect on the performance of the merged runs, although such an effect might be difficult to discern.

The merging strategies we use are isolated merging strategies in that they require no data from the component databases at runtime to decide how many documents to retrieve from each database. This makes the strategies efficient and suitable for use in environments where there is no central authority. A 15% degradation only amounts to approximately one fewer relevant document retrieved per query, and is thus quite reasonable when other circumstances prevent a single collection search.

These experiments raise issues that the results do not address and that therefore need further investigation. A major open issue for the isolated merging techniques is how the available training data affects the merging behavior. In settings other than TREC, one would expect many more training queries, but each query would have relevance data for only a few collections. The MRDD strategy may be more practical in such an environment since it is more dependent on quality training data. A possible alternative to the current strategy of using all available training data would be to select (by hand) a smaller number of exemplar queries. This would increase the efficiency of both the QC and MRDD methods, although its effect on the quality of the searches is unclear. Finally, Topics 202–250 are quite short as compared to previous TREC topics, and the MRDD method appears to have some difficulty with them. Could MRDD cope if all the training queries were as short?

A second issue involves the kinds of distinctions among databases a practical isolated merging strategy can be expected to learn. In TREC-4, the set of documents was divided into databases such that several databases were from the same source (e.g., WSJ90, WSJ91, and WSJ92). That is, the criteria that were used to classify documents into databases included considerations other than subject matter, which is likely to occur in other environments as well. While this does not appear to have been much of an impediment to the merging strategies in TREC-4, there may be an effect that is masked by the AP bias.

References

[1] Chris Buckley. Implementation of the SMART information retrieval system. Technical Report 85-686, Computer Science Department, Cornell University, Ithaca, New York, May 1985.

[2] Chris Buckley, Gerard Salton, James Allan, and Amit Singhal. Automatic query expansion using smart: Trec 3. In Donna K. Harman, editor, *Overview of the Third Text REtrieval Conference (TREC-3) [Proceedings of TREC-3.]*, pages 69–80, April 1995. NIST Special Publication 500-225.

[3] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching distributed collections with inference networks. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28, July 1995.

[4] Geoffrey Towell, Ellen M. Voorhees, Narendra K. Gupta, and Ben Johnson-Laird. Learning collection fusion strategies for information retrieval. In *Proceedings of the 12th Annual Machine Learning Conference*, July 1995.

[5] Ellen M. Voorhees, Narendra K. Gupta, and Ben Johnson-Laird. The collection fusion problem. In Donna K. Harman, editor, *Overview of the Third Text REtrieval Conference (TREC-3) [Proceedings of TREC-3.]*, pages 95–104, April 1995. NIST Special Publication 500-225.

[6] Ellen M. Voorhees, Narendra K. Gupta, and Ben Johnson-Laird. Learning collection fusion strategies. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 172–179, July 1995.