The Effectiveness of Query Expansion for Distributed Information Retrieval

Paul Ogilvie, Jamie Callan
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891
{pto,callan}@lti.cs.cmu.edu

ABSTRACT

Query expansion has been shown effective for both single database retrieval and for distributed information retrieval where complete collection information is available. One might expect that query expansion would then work for distributed information retrieval when complete collection information is not available. However, this does not appear to be the case. When using local context analysis for query expansion in distributed retrieval with partial information, the most significant reason query expansion does not work is that merging scores of documents retrieved by expanded queries is very difficult. However, we have found that using sampled information for query expansion can give boosts in a single database environment, and that when more information is available, query expansion can work in distributed environments. We also show that most of the benefit of query expansion in distributed retrieval comes from finding good documents, and not from selecting good databases.

1 INTRODUCTION

As the number of databases available online increases, it becomes desirable to provide access to many of these databases through a unified distributed information retrieval system. In order to use many databases efficiently, these systems generally select a few databases to search. These selected databases are then searched using the user's query. Results from separate databases are merged into a single list and presented to the user. Most recent research in distributed information retrieval has focused on new problems introduced by the distributed retrieval environment. Among the new problems are resource description, collection selection, and results merging. However, there is less research on the application of single database techniques to distributed retrieval. For example, there is little work on the use of query operations such as relevance feedback and query expansion in distributed environments [13, 15]. In this paper, we explore the use of query expansion in a practical distributed information retrieval environment.

It has long been thought that the automatic addition of terms to a query, or query expansion, could improve information retrieval performance. Past research has shown that the use of automatic query expansion in a centralized environment can yield good improvement in information retrieval performance. One example is found in [2], which presents massive query expansion. That is, they show that the addition of large numbers of terms to the original query can improve retrieval performance under a number of measures.

Since distributed information retrieval is often compared to and built on top of standard single database information retrieval, one might feel that query expansion could improve performance in the distributed environment. Some preliminary work in a distributed environment suggests that query expansion may indeed be beneficial [15]. However, there are two characteristics of the previous work we feel are unrealistic in some distributed environments. The approach used the complete collection or a very large, representative collection for query expansion. Also, global inverse document frequency (*idf*) information was used for merging documents from separate databases. We feel that in environments where databases are uncooperative, it is unrealistic to assume this information is available or can be easily gathered.

In this paper, we more thoroughly investigate query expansion in distributed information retrieval. We examine local context analysis [14], a technique for query expansion, using smaller collections of documents than previously used. In environments where databases are not cooperative, it is not reasonable to assume that we can gather full information from these databases. We sample documents from each database as an attempt to gather information adequate for distributed information retrieval. The technique we use for sampling collection is query-based sampling [5]. Query-based sampling can be done without requiring the cooperation of databases. When sampling from a database, each document retrieved costs resources, so a goal is to minimize the cost for collecting information from databases. The fewer the documents sampled from a database, the lower the cost of gathering the information. We wish to determine exactly how little information is sufficient for use in query expansion.

Another characteristic that distinguishes this work from [15] is that we investigate distributed retrieval using a more realistic resource merging algorithm. Previous work has used global *idf* information for ranking documents from the different databases. The use of global *idf* information requires all of the databases to cooperate and use the same ranking algorithm, which we feel is unrealistic in many distributed environments. The algorithm we use for merging documents requires much less cooperation between databases.

The following section contains an overview of query expansion in distributed information retrieval, discussing related work and presenting the framework we use. Section 3 presents research

Precision	Title Queries 1 db	Title Queries Multiple db	Description Queries 1 db	Description Queries Multiple db	Concept Queries 1 db	Concept Queries Multiple db
at 5 docs	0.4660	0.4620 (- 0.1%)	0.4680	0.4780 (+ 2.1%)	0.6220	0.5620 (- 9.6%)
at 10 docs	0.4530	0.4260 (- 6.0%)	0.4680	0.4470 (- 4.5%)	0.5930	0.5570 (- 6.1%)
at 15 docs	0.4580	0.4087 (-10.8%)	0.4707	0.4267 (- 9.3%)	0.5753	0.5413 (- 5.9%)
at 20 docs	0.4515	0.3870 (-14.3%)	0.4555	0.4110 (- 9.8%)	0.5670	0.5285 (- 6.8%)
at 30 docs	0.4277	0.3683 (-13.9%)	0.4383	0.3997 (- 8.8%)	0.5503	0.5013 (- 8.9%)
Ave-p	0.1759	0.0804 (-54.3%)	0.1697	0.0824 (-51.4%)	0.2547	0.1228 (-51.8%)

Table 1. This table shows the performance of the original queries in both the single database setting and in distributed retrieval where full information is available for collection selection.

methods and the test suite. Section 4 presents our findings, and we draw conclusions in Section 5.

2 QUERY EXPANSION IN DISTRIBUTED INFORMATION RETRIEVAL

Before going into the details of query expansion, we present the framework we use for distributed retrieval. Distributed information retrieval can be characterized as a set of three problems [4]. Resource description is the process of describing the contents of the individual databases in the distributed environment. Resource selection is where the system must choose which databases to search when presented with an information need. Finally, resource merging requires combining ranked lists returned by searching the selected collections. The result is a single ranked list. For some methods, resource description and resource selection may be combined, as with lightweight probes [10]. These methods assume a greater degree of cooperation than we are willing to use in our experiments.

Query expansion can be used in this framework in two ways. Expanded queries could be used for resource selection, or for searching the individual databases that have been selected. We investigate the use of expanded queries for resource selection (or collection selection) and for searching the individual databases.

We represent a resource description as a bag of words, storing the frequency of words in the collection. A variety of systems ([16], [9], [4]) use a bag of words for the resource description. Under this model, collection-wide term frequencies are stored for all terms in each collection. As a result of Zipf's law, this is a fairly compact representation of a database, and scales well to large databases. Given complete collection information, one can easily generate a resource description. In some distributed environments, complete information is not available. In this case, we use query-based sampling to approximate the database's true word frequency. Query-based sampling is the process of querying a database using random one-word queries [5]. The initial query is often selected from some large dictionary of terms. Subsequent queries are selected from the documents sampled from the database. All of the documents retrieved from a database are then used to approximate the true word frequencies of the database. This technique has been shown to work reasonably well for providing models to be used for collection selection when sampling 300 or more documents from each database.

For resource selection, we use the CORI database selection algorithm [4]. In both [12] and [6], the CORI algorithm has been found to be fairly robust and perform better than alternatives such as Cue Validity Variance [16] and gGlOSS [9]. The CORI algorithm is an extension of the Bayesian Inference Networks used by Inquery to rank resources. It uses a variant of *tf·idf* adapted for ranking databases. Collections are treated analogously to documents in a database. Following the notation of [4], collection scores for individual terms are computed as follows.

$$T = \frac{df}{df + 50 + 150 \cdot cw / avg_cw}$$

$$I = \frac{\log\left(\frac{C + 0.5}{cf}\right)}{\log\left(C + 1.0\right)}$$

$$p(r_k \mid R_i) = 0.4 + 0.6 \cdot T \cdot I$$

where *df* is the document frequency of term r_k in resource R_i, *cw* is the number of indexing term occurrences in collection R_i, *avg_cw* is the average of *cw* across all collections, *C* is the number of collections, and *cf* is the collection frequency of r_k, or number of collections containing term r_k. The constants 0.4 and 0.6 are derived from the default belief of a term (*x* and 1-*x*, respectively), and can be changed in other systems. $p(r_k|R_i)$ is the score of term r_k for collection R_i. For multiple term queries, we primarily use Inquery's `#sum` and `#wsum` operators. The `#sum` operator treats the words inside the sum as a simple unweighted bag-of-words query. The `#wsum` is similar, but query terms can have weights. The phrase operator `#ow3` is also used in expanded queries, but is converted to a Boolean `and` in collection selection because information needed to support this is not stored in the resource description.

Resource merging, or results merging, is the final stage in distributed information retrieval. Many database systems use corpus-specific document frequency information. If global *idf* information is available across all databases and all of the databases use the same ranking algorithm, then a database can rank its documents using global *idf* instead of using the *idf* statistics for the individual database. This way, consistent *idf* information is used for all databases and merging the results is a simple matter of ordering results based solely on the document score. Alternatively, if the databases do not use database-wide statistics, then raw scores can be used [10]. This approach is unreliable if database-wide statistics are used. Another resource merging technique is to download full or partial documents and re-rank them using *idf* statistics from a reference collection [7]. While this is possible in realistic settings, we feel that the download of documents is an expensive operation that should be avoided. When complete information is not available and the databases use corpus-specific information, another approach is needed. The approach we use is to combine normalized document and collections scores, as in [4]:

Precision	Single no QE	Single, QE on ~270,000 docs	Single, QE on ~135,000 docs	Single, QE on ~67,500 docs	Single, QE on ~34,000 docs
at 5 docs	0.4660	0.4620 (- 0.8%)	0.4790 (+ 2.7%)	0.4570 (- 1.9%)	0.4440 (- 4.7%)
at 10 docs	0.4530	0.4605 (+ 1.6%)	0.4725 (+ 4.3%)	0.4555 (+ 0.5%)	0.4400 (- 2.8%)
at 15 docs	0.4580	0.4567 (- 0.2%)	0.4693 (+ 2.4%)	0.4494 (- 1.8%)	0.4286 (- 6.4%)
at 20 docs	0.4515	0.4577 (+ 1.3%)	0.4695 (+ 3.9%)	0.4517 (+ 0.0%)	0.4315 (- 4.4%)
at 30 docs	0.4277	0.4525 (+ 5.7%)	0.4628 (+ 8.2%)	0.4411 (+ 3.1%)	0.4269 (- 0.1%)
Ave-p	0.1759	0.2144 (+21.9%)	0.2133 (+21.2%)	0.2058 (+16.9%)	0.1958 (+11.3%)

Table 2. Every-n'th sampling was used to collect documents to build a query expansion database. Tests are run in single database environment. This table shows TITLE queries.

Precision	Single no QE	Single, QE on ~270,000 docs	Single, QE on ~135,000 docs	Single, QE on ~67,500 docs	Single, QE on ~34,000 docs
at 5 docs	0.4680	0.4590 (- 1.9%)	0.4100 (-12.3%)	0.4150 (-11.3%)	0.3630 (-22.4%)
at 10 docs	0.4680	0.4450 (- 4.9%)	0.3995 (-14.6%)	0.4055 (-13.3%)	0.3630 (-22.4%)
at 15 docs	0.4707	0.4410 (- 6.3%)	0.3977 (-15.5%)	0.3977 (-15.5%)	0.3604 (-23.4%)
at 20 docs	0.4555	0.4375 (- 3.9%)	0.3955 (-13.1%)	0.3935 (-13.6%)	0.3615 (-20.6%)
at 30 docs	0.4383	0.4277 (- 2.4%)	0.3830 (-12.6%)	0.3860 (-11.9%)	0.3547 (-19.0%)
Ave-p	0.1697	0.2112 (+24.4%)	0.1885 (+11.1%)	0.1898 (+11.8%)	0.1784 (+ 5.0%)

Table 3. Same as Table 2, but shows DESCRIPTION queries.

Precision	Single no QE	Single, QE on ~270,000 docs	Single, QE on ~135,000 docs	Single, QE on ~67,500 docs	Single, QE on ~34,000 docs
at 5 docs	0.6220	0.5850 (- 5.9%)	0.5510 (-11.4%)	0.5160 (-17.0%)	0.4600 (-26.0%)
at 10 docs	0.5930	0.5700 (- 3.8%)	0.5335 (-10.0%)	0.5140 (-13.3%)	0.4690 (-20.9%)
at 15 docs	0.5753	0.5576 (- 3.0%)	0.5284 (- 8.1%)	0.5107 (-11.2%)	0.4787 (-16.7%)
at 20 docs	0.5670	0.5557 (- 1.9%)	0.5188 (- 8.5%)	0.5060 (-10.7%)	0.4715 (-16.8%)
at 30 docs	0.5503	0.5358 (- 2.6%)	0.5045 (- 8.3%)	0.4916 (-10.6%)	0.4607 (-16.2%)
Ave-p	0.2547	0.2514 (- 1.2%)	0.2399 (- 5.8%)	0.2301 (- 9.6%)	0.2133 (-16.2%)

Table 4: Same as Table 2, but shows CONCEPT queries.

$$R_i' = (R_i - R_{\min}) / (R_{\max} - R_{\min})$$
$$D' = (D - D_{\min_i}) / (D_{\max_i} - D_{\min_i})$$
$$D'' = \frac{D' + 0.4 \cdot D' \cdot R'}{1.4}$$

Here, R_i is the score of collection i, $R_{\max}$ is the maximum possible score for a collection, and $R_{\min}$ is the minimum possible score for a collection. Similarly, D is the score of a document, and $D_{\max_i}$ and $D_{\min_i}$ are the maximum and minimum possible scores for any document given collection i and the query. Documents are then ranked according to D''. This merge method has been found to be somewhat sensitive to collections with varying *idf* statistics [11]. However, computation of $D_{\max_i}$ and $D_{\min_i}$ requires cooperation of the databases. This cooperation is contrary to our research goals, so a better merge technique that does not require this cooperation is still a research goal. However, we feel that this merge method is reasonable, and requires a minimal amount of cooperation from a database. It does not require any communication of information between databases; it is an internal constraint on the ranking algorithm of the database. Other methods require a much larger degree of cooperation, and are thus less favorable.

It is on this framework that we explore query expansion. In particular, we are interested in using local context analysis [14]. An advantage of local context analysis over other query expansion techniques is that it combines ideas used in global analysis with those used in local query expansion methods. Local context analysis is a technique that selects nouns and noun compounds (two or more adjacent nouns) to add to a query. Terms are selected according to co-occurrence with the query in *n*-word passages. To generate the co-occurrence measure, the query is evaluated against a database of passages. From the top *x* passages, nouns and noun groups are extracted. Co-occurrence is computed using a variant of *tf·idf*. The database of passages used for query expansion is usually the same as the database searched.

In past research [15], the database used for query expansion in distributed information retrieval has been either all of the documents in all databases or a same sized representative document collection. The first method is not practical in many distributed environments, as it requires access to all documents and the ability to build one large query expansion database. The second approach is also not practical, as we assume that the nature of each database's contents is unknown to us. It is impractical to presume that we could gather a representative collection of documents in size equal to the databases. However, it does seem reasonable that we could gather a smaller, but still representative, collection of documents from each of the databases. That is what query-based sampling is designed to do. *We hypothesize that documents gathered via query-based sampling can be effectively used for query expansion in distributed information retrieval.* That is, we propose that the same documents we gather for collection selection can be combined into a query-expansion database that will improve distributed retrieval performance.

3 EXPERIMENTAL METHODS

In order to test our hypothesis, we must run distributed retrieval tests. We evaluate the performance of using query expansion in distributed information retrieval using a test collection constructed from TREC CD's 1, 2, and 3 [3]. These documents were separated by source and date into 100 databases. Within this test suite, database sizes vary widely in number of documents. The average size of the documents in a database is 33 megabytes. Since the databases are organized by source, the collections tend to be more homogeneous than one central database containing all

Precision	Single no QE	Single, QE on 2900 docs/db (~290000 docs)	Single, QE on 1450 docs/db (~145000 docs)	Single, QE on 725 docs/db (~72500 docs)	Single, QE on 362 docs/db (~36200 docs)
at 5 docs	0.4660	0.4532 (- 2.7%)	0.4412 (- 5.3%)	0.4296 (- 7.8%)	0.4056 (-12.9%)
at 10 docs	0.4530	0.4534 (+ 0.0%)	0.4500 (- 0.6%)	0.4352 (- 3.9%)	0.4202 (- 7.2%)
at 15 docs	0.4580	0.4552 (- 0.6%)	0.4488 (- 2.0%)	0.4396 (- 4.0%)	0.4219 (- 7.8%)
at 20 docs	0.4515	0.4542 (+ 0.5%)	0.4519 (+ 0.0%)	0.4410 (- 2.3%)	0.4233 (- 6.2%)
at 30 docs	0.4277	0.4526 (+ 5.8%)	0.4479 (+ 4.7%)	0.4411 (+ 3.1%)	0.4219 (- 1.3%)
Ave-p	0.1759	0.2124 (+20.7%)	0.2096 (+19.1%)	0.2025 (+15.1%)	0.1910 (+ 8.5%)

Table 5. Query-based sampling was used to collect documents to build a query expansion database. Tests are run in single database environment. This table shows TITLE queries.

Precision	Single no QE	Single, QE on 2900 docs/db (~290000 docs)	Single, QE on 1450 docs/db (~145000 docs)	Single, QE on 725 docs/db (~72500 docs)	Single, QE on 362 docs/db (~36200 docs)
at 5 docs	0.4680	0.4308 (- 7.9%)	0.3836 (-18.0%)	0.3668 (-21.6%)	0.3388 (-27.6%)
at 10 docs	0.4680	0.4284 (- 8.4%)	0.3840 (-17.9%)	0.3760 (-19.6%)	0.3416 (-27.0%)
at 15 docs	0.4707	0.4187 (-11.0%)	0.3853 (-18.1%)	0.3797 (-19.3%)	0.3457 (-26.5%)
at 20 docs	0.4555	0.4196 (- 7.8%)	0.3873 (-14.9%)	0.3786 (-16.8%)	0.3488 (-23.4%)
at 30 docs	0.4383	0.4133 (- 5.6%)	0.3863 (-11.8%)	0.3735 (-14.7%)	0.3482 (-20.5%)
Ave-p	0.1697	0.2018 (+ 8.9%)	0.1909 (+12.4%)	0.1828 (+ 7.7%)	0.1700 (+ 0.1%)

Table 6. Same as Table 5, but shows DESCRIPTION queries.

Precision	Single no QE	Single, QE on 2900 docs/db (~290000 docs)	Single, QE on 1450 docs/db (~145000 docs)	Single, QE on 725 docs/db (~72500 docs)	Single, QE on 362 docs/db (~36200 docs)
at 5 docs	0.6220	0.5472 (-12.0%)	0.5152 (-17.1%)	0.4844 (-22.1%)	0.4560 (-26.6%)
at 10 docs	0.5930	0.5504 (- 7.1%)	0.5148 (-13.1%)	0.4882 (-17.6%)	0.4464 (-24.7%)
at 15 docs	0.5753	0.5479 (- 4.7%)	0.5103 (-11.3%)	0.4761 (-17.2%)	0.4401 (-23.4%)
at 20 docs	0.5670	0.5400 (- 4.7%)	0.5013 (-11.5%)	0.4698 (-17.1%)	0.4388 (-22.6%)
at 30 docs	0.5503	0.5269 (- 4.2%)	0.4892 (-11.1%)	0.4616 (-16.1%)	0.4338 (-21.1%)
Ave-p	0.2547	0.2480 (- 2.6%)	0.2334 (- 8.3%)	0.2189 (-14.0%)	0.2028 (-20.3%)

Table 7. Same as Table 5, but shows CONCEPT queries.

documents. This causes inverse document frequency (*idf*) statistics to vary widely across databases.

The test topics we use are TREC topics 51-150. We perform experiments with title, description, and concept queries. The queries are created by removing common stop-phrases from the title, description, and concept fields. In all experiments, we use Inquery [1] for single database retrieval. CORI is used for collection selection. Unless specified otherwise, merging is done with normalized collection and document scores as described above. We select 10 of the 100 databases for searching.

For local context analysis, we retrieve 50 passages of 300 terms, selecting the best 70 terms to add to the query. We include noun groups tagged by JTAG, a bigram-based part-of-speech tagger. We stem using kstem and use Inquery's default stop-word list. The expanded query is added to the original query using the following query: `#wsum(1 1 #sum(original query terms) 2 #wsum(added terms))`. All of our parameters for query expansion are the same as those used in [12, 13].

When presenting results, we use as evaluation measures both precision at 5 to 30 documents and average precision. Precision at 5 to 30 documents is useful because it measures performance where the typical user views results. Average precision is useful because it gives an estimate of overall system performance.

4 EXPERIMENTAL RESULTS

In this section, we present experimental results. Section 4.1 describes baseline experiments in both the single database and distributed environments. Section 4.2 presents single database environment experiments using a very simple and representative method of choosing documents for use in query expansion. This serves as a test of how much information is needed to do query expansion. Finally, in Section 4.3, we evaluate the use of query-based sampling for collecting documents to be used for query expansion in the single database and distributed environments.

4.1 Baseline Results

To assist understanding our results, we provide two baselines for each query/data set. These baseline tests are run without query expansion. For each query set, we run our system in both single database mode (search all documents as a single database) and distributed mode (search the highest ranked 10 of 100 collections). The multiple database test is run with full information for collection selection and selecting 10 databases per query. See Table 1 for the baseline performance. In all tables, we show percent change in performance in parentheses.

For single database tests, we hope to achieve better performance than the single database baseline. For distributed searches, we hope to achieve better performance than the distributed baseline. We feel the single database tests are necessary to isolate the performance of query expansion without the influence other factors such as merge algorithms.

4.2 Every N'th Document Sampling

We begin by studying the question of how many documents are required to do effective query expansion. Before we present retrieval experiments using information from query-based sampling, we present experiments where the documents used for expansion were selected by taking every n'th document from the previous data set. We feel that running retrieval experiments from

docs/db	Title Queries		Description Queries		Concept Queries	
	No QE	With QE	No QE	With QE	No QE	With QE
2900	28.6	29.5 (+ 3.1%)	27.2	29.0 (+ 6.6%)	31.6	32.6 (+ 3.2%)
1450	28.1	28.9 (+ 2.8%)	27.2	28.1 (+ 3.3%)	30.6	31.5 (+ 2.9%)
725	27.5	27.3 (- 0.7%)	26.5	26.6 (+ 0.4%)	29.9	29.9 (+ 0.0%)
362	25.7	25.7 (+ 0.0%)	25.3	25.1 (- 0.8%)	28.7	28.5 (- 0.7%)

Table 8. Query-based sampling was used to collect documents to build a query expansion database. This table shows the effects on collection selection using sampled information. Values are percentage of relevant documents available when selecting ten databases to search. This is the $\hat{R}$ value of [8].

Precision	2900 docs/db		1450 docs/db		725 docs/db	
	No QE	With QE	No QE	With QE	No QE	With QE
at 5 docs	0.4532	0.4536 (+ 0.0%)	0.4472	0.4456 (- 0.3%)	0.4560	0.4120 (- 9.6%)
at 10 docs	0.4282	0.4338 (+ 1.3%)	0.4184	0.4166 (- 0.4%)	0.4270	0.3862 (- 9.5%)
at 15 docs	0.4100	0.4183 (+ 2.0%)	0.4065	0.3979 (- 2.1%)	0.4092	0.3695 (- 9.7%)
at 20 docs	0.3959	0.4040 (+ 2.0%)	0.3915	0.3856 (- 1.5%)	0.3950	0.3599 (- 8.8%)
at 30 docs	0.3708	0.3756 (+ 1.2%)	0.3682	0.3645 (- 0.9%)	0.3648	0.3380 (- 7.3%)
Ave-p	0.0745	0.0707 (- 5.1%)	0.0702	0.0659 (- 6.1%)	0.0681	0.0568 (-16.5%)

Table 9. Query-based sampling was used to collect documents to build a query expansion database. This table shows distributed retrieval performance for TITLE queries.

Precision	2900 docs/db		1450 docs/db		725 docs/db	
	No QE	With QE	No QE	With QE	No QE	With QE
at 5 docs	0.4856	0.3976 (-18.1%)	0.4760	0.3768 (-20.8%)	0.4820	0.3652 (-24.2%)
at 10 docs	0.4500	0.3814 (-15.2%)	0.4420	0.3608 (-18.3%)	0.4420	0.3488 (-21.0%)
at 15 docs	0.4267	0.3701 (-13.2%)	0.4173	0.3449 (-17.3%)	0.4151	0.3345 (-19.3%)
at 20 docs	0.4167	0.3585 (-13.9%)	0.4063	0.3345 (-17.6%)	0.4037	0.3222 (-20.1%)
at 30 docs	0.3911	0.3331 (-14.8%)	0.3850	0.3091 (-19.7%)	0.3839	0.3029 (-21.0%)
Ave-p	0.0761	0.0598 (-21.3%)	0.0722	0.0533 (-26.1%)	0.0693	0.0468 (-32.4%)

Table 10. Same as Table 9, but shows DESCRIPTION queries.

these data sets is needed because it helps to give an understanding of how much information is needed in order to do query expansion well.

Documents are "sampled" by taking every n'th document from each database. While this really isn't sampling, we call this method sampling for simplicity. When n = 4, about 270,000 documents are used for query expansion. We also present data for n = 8, 16, and 32, resulting in around 135,000, 67,500, and 34,000 documents used for expansion, respectively.

Tables 2, 3, and 4 show the results of the every-n'th sampling technique for query expansion. In the tables, query expansion is abbreviated QE. The query expansion results are averaged over two samples. For all queries, we get no boost for precision at 5-30 documents. In fact, using queries expanded by local context analysis can be detrimental to precision at 5-30 documents. One might think that this is contradictory with previous work in [14] and [15], which showed boosts in retrieval performance when using local context analysis for query expansion. However, [14] and [15] did not present results using precision at 5-30 in the single database environment.

We do see a boost in average precision for both the title and description queries, which agrees with Xu's findings in [14]. The gain is more pronounced for the title queries; local context analysis gives improvement for as few as 34,000 documents used from each of the 100 databases. For description queries, the improvement is clear at 67,500 documents sampled from each database. It is not surprising that the concept queries are degraded by local context analysis, as they are good queries from the start. Expanding them only adds noise to the queries.

4.3 Query-Based Sampling

We now examine the use of documents obtained via query-based sampling for query expansion. The results presented here are averages of scores over five different runs. To obtain the documents, we sampled 2900 documents per database, retrieving four documents per query. In some cases, less than 2900 documents were in the database or readily available using query-based sampling, so we sampled as many documents as were obtainable using 6000 queries per database. For experiments using x = 1450, 725, and 362 documents per database, we took the first x documents from the 2900 document sample.

We first look at the use of sampled documents in single database retrieval. That is, we combine the sampled documents from each of the 100 databases into a single database to expand the queries. We expand the queries on that database, and then run the expanded queries on one central database consisting of all documents in the test set. We do this to examine query expansion independent of distributed factors. Tables 5, 6, and 7 show the results for these experiments. For title queries, using information obtained from query-based sampling works almost as well as information from every-n'th sampling. For description and concept queries, query-based sampling does not work as well as every-n'th sampling. This shows that query-based sampling is not an optimal method of selecting documents to be used by query expansion using local context analysis. As with expanding concept queries using information from every-n'th sampling, using query-based sampling information hurts performance slightly. Since concept queries are difficult to begin with, we will focus the rest of our experiments on title and description queries.

We now look at how query expansion by local context analysis affects distributed information retrieval. Given the above findings, one might expect that query expansion would work well for title queries and perhaps for description queries. We first look

Precision	DIR no QE	2900 docs/db ~290000 docs	1450 docs/db ~145000 docs	725 docs/db ~72500 docs
at 5 docs	0.4480	0.4776 (+ 6.6%)	0.4612 (+ 2.9%)	0.4600 (+ 2.6%)
at 10 docs	0.4130	0.4554 (+10.2%)	0.4472 (+ 8.2%)	0.4346 (+ 5.2%)
at 15 docs	0.3967	0.4392 (+10.7%)	0.4336 (+ 9.3%)	0.4167 (+ 5.0%)
at 20 docs	0.3810	0.4263 (+11.8%)	0.4190 (+ 9.9%)	0.4052 (+ 6.3%)
at 30 docs	0.3577	0.4040 (+12.9%)	0.3970 (+10.9%)	0.3827 (+ 6.9%)

Table 11. Query based sampling was used to collect documents to build a query expansion database. Global *idf* was used for merging, and full information was used for collection selection. This table shows performance for TITLE queries.

Precision	DIR no QE	2900 docs/db ~290000 docs	1450 docs/db ~145000 docs	725 docs/db ~72500 docs
at 5 docs	0.4640	0.4536 (- 2.2%)	0.4116 (-11.2%)	0.3928 (-15.3%)
at 10 docs	0.4420	0.4362 (- 1.3%)	0.3984 (- 9.8%)	0.3724 (-15.7%)
at 15 docs	0.4207	0.4213 (+ 0.1%)	0.3874 (- 7.9%)	0.3627 (-13.7%)
at 20 docs	0.4085	0.4031 (- 1.3%)	0.3745 (- 8.3%)	0.3522 (-13.7%)
at 30 docs	0.3837	0.3840 (+ 0.0%)	0.3571 (- 6.9%)	0.3384 (-11.8%)

Table 12. Same as Table 11, but shows DESCRIPTION queries.

Precision	Single no QE	2900 docs/db ~290000 docs	1450 docs/db ~145000 docs	725 docs/db ~72500 docs
at 5 docs	0.4660	0.4776 (+ 2.5%)	0.4612 (- 1.0%)	0.4600 (- 1.3%)
at 10 docs	0.4530	0.4554 (+ 0.5%)	0.4472 (- 1.3%)	0.4346 (- 4.0%)
at 15 docs	0.4580	0.4392 (- 4.1%)	0.4336 (- 5.3%)	0.4167 (- 9.0%)
at 20 docs	0.4515	0.4263 (- 5.6%)	0.4190 (- 7.2%)	0.4052 (-10.3%)
at 30 docs	0.4277	0.4040 (- 5.5%)	0.3970 (- 7.2%)	0.3827 (-10.5%)

Table 13. Same as Table 11, but the baseline here is single database retrieval with original queries.

at the effects of local context analysis on collection selection. Table 8 shows the percentage of relevant documents that are available when selecting ten databases to search per query. This table suggests that query expansion provides only small improvements in collection selection. Using full information for collection selection, we see very similar results (Table 16). This may be because the resource description and collection selection algorithm we use does not support phrases, thus losing some of the benefits of local context analysis. In [15], there are experiments using words and phrases for collection selection, but only modest improvement in retrieval effectiveness was reported. [15] did not examine the effects on collection selection independent of retrieval, which we feel would be necessary to draw conclusions in our distributed environment.

Tables 9 and 10 show the performance of query expansion on document retrieval in a distributed information retrieval setting. Using local context analysis does not help retrieval performance. For title queries, as we use less data for query expansion and collection selection, performance rapidly drops when using less than 1450 documents per database (about 1/8 collection size). We do not report results for 362 documents per database due to lack of space in the tables. Results for description and concept queries are worse than for title queries. This is somewhat surprising and seems contradictory at first glance with work reported in [15].

There are several possible reasons query expansion worked in [15] but not in these experiments. Xu used full information for collection selection and query expansion for most experiments, and also used global *idf* information for merges. In one experiment, Xu used a database of size equal to the test database for query expansion and also used full information for collection selection. While this database was separate from the test database, it was as large as the test database and its contents were representative of the test database. In the experiments reported here, we do not use full information for either collection selection or query expansion.

If we use full information for collection selection and merge using global *idf* information, we get much better results (Tables 11 and 12). Since we are primarily concerned with explaining the performance of precision at 5-30 documents, and we are interested in saving time when running experiments, we do not present average precision in these or later experiments. This is similar to the experiments done in [15], except that we use documents from query-based sampling for query expansion. These results do not provide as much of a boost as reported in [15]. One reason is that our query/data set's performance did not degrade as much when moving from single database to distributed retrieval. If we compare the results of the title queries for 2900 documents sampled per database, we see that the performance is about that of using one central index. This agrees with observations in [15]. The other reason is that we are using partial information for query expansion. The results are best for title queries, but there is no clear gain for description and concepts queries. While Xu used both title and description fields in his queries, most of the emphasis was placed on the title field, so our finding is largely in agreement with [15]. As seen in the single database setting, when we reduce the sample size, performance using query expansion decreases. We are confident that using other methods of sampling will also reduce performance. See Tables 17 and 18 for confirmation of this when using every-n'th document sampling.

If we turn off full information for collection selection, we see little difference in the effectiveness of query expansion for the title queries (Tables 14 and 15). However, the performance of expanded description queries drops off from the original query performance less quickly. This is somewhat surprising; it is in part due to the fact that the performance of the original description queries falls off when using less information for collection selection. On the other hand, the absolute performance of the expanded description queries is similar regardless of whether full or partial information is used for collection selection. Interestingly, we do not observe this behavior for original

Precision	2900 docs/db No QE	2900 docs/db With QE	1450 docs/db No QE	1450 docs/db With QE	725 docs/db No QE	725 docs/db With QE
at 5 docs	0.4448	0.4716 (+ 6.0%)	0.4432	0.4596 (+ 3.7%)	0.4412	0.4492 (+ 1.8%)
at 10 docs	0.4196	0.4526 (+ 7.8%)	0.4188	0.4460 (+ 6.4%)	0.4160	0.4372 (+ 5.0%)
at 15 docs	0.4101	0.4369 (+ 6.5%)	0.4051	0.4313 (+ 6.4%)	0.3996	0.4164 (+ 4.2%)
at 20 docs	0.3926	0.4238 (+ 7.9%)	0.3865	0.4174 (+ 7.9%)	0.3819	0.3992 (+ 4.5%)
at 30 docs	0.3669	0.4047 (+10.3%)	0.3631	0.3933 (+ 8.3%)	0.3549	0.3781 (+ 6.5%)

Table 14. Query based sampling was used to collect documents to build a query expansion database. Global *idf* was used for merging, and sampled information was used for collection selection. This table shows performance for TITLE queries.

Precision	2900 docs/db No QE	2900 docs/db With QE	1450 docs/db No QE	1450 docs/db With QE	725 docs/db No QE	725 docs/db With QE
at 5 docs	0.4660	0.4448 (- 4.5%)	0.4480	0.4072 (- 9.1%)	0.4440	0.4016 (- 9.5%)
at 10 docs	0.4286	0.4232 (- 1.2%)	0.4156	0.3982 (- 4.1%)	0.4198	0.3906 (- 6.9%)
at 15 docs	0.4079	0.4107 (+ 0.6%)	0.3965	0.3845 (- 3.0%)	0.3996	0.3783 (- 5.3%)
at 20 docs	0.3918	0.3979 (+ 1.5%)	0.3838	0.3740 (- 2.5%)	0.3839	0.3675 (- 4.2%)
at 30 docs	0.3617	0.3793 (+ 4.8%)	0.3563	0.3597 (+ 0.9%)	0.3538	0.3474 (- 1.8%)

Table 15. Same as Table 14, but shows DESCRIPTION queries.

description queries when using normalized document and collection scores for merging. We have not been able to find good explanation for this behavior, but we do not feel that this detracts from the accuracy of our results. In our experiments, the concept queries exhibit similar behavior when switching from full information for collection selection to sampled information.

It appears that most of the difference between the results in [15] and our results comes from the use of global *idf* information. The use of global *idf* information for computing the scores of documents simplified the merging problem. This is not the first time that normalized scores have been found to perform worse than global *idf*. In [11], experiments show that when the data is topically organized, the performance found using heuristically normalized scores might fall short of that of using global *idf* information. Our results are different from this other work in that we only see this behavior with the expanded queries; the performance of the original queries is nearly identical when using global *idf* or normalized scores. While we do not know what causes the difficulty in merging, we do know that the merge problem is non-trivial to solve. Better known merge algorithms require more information, which is not practical in many situations.

5 CONCLUSIONS

In this paper, we explored the use of query-based sampling for local context analysis in a practical distributed information retrieval environment. We find that query expansion fails in this environment for several reasons. Regardless of the method used for sampling documents, as fewer documents are used for expansion, local context analysis performs worse. Also, independent of the method used for selecting documents for expansion, query expansion makes merging document scores more difficult without assuming large amounts of cooperation between databases. Query-based sampling is not the best method for selecting documents to use for expansion. The first and last issues affect both the collection selection and resource merging stages of retrieval, and the second issue affects the merging stage only. It appears that the score merging issue degrades the retrieval performance the most, and the deficiencies of query-based sampling degrade performance the least.

We have two guesses as to why merging the results of expanded queries is difficult when using normalized scores. One reason could be that as queries get longer, it is harder to merge the results using the normalized and document collection scores. The other reason could be that local context analysis selects terms that have more skewed *idf* scores across databases, making the normalized scores merge method less accurate. Preliminary experiments testing these hypotheses have been inconclusive.

Given the past performance of local context analysis in single database retrieval, it is somewhat disheartening that query expansion does not work well in all distributed environments. However, we have some positive results, and we have some understanding as to why query expansion fails. We have found that using sampled information can yield improvement for average precision when operating in a single database environment. When working in a distributed environment, we find that using sampled information for query expansion can improve retrieval performance when using highly effective merging techniques. It is primarily when we use merge techniques that require less cooperation among databases that query expansion fails. When operating under conditions where query expansion does work, most of the benefit comes from finding good documents, not from choosing good databases.

These results raise several research issues for distributed information retrieval. How can the results of expanded queries be effectively merged without requiring the download of documents or large amounts of cooperation between databases? Why does the merging fail? In order to answer the first question, the second question may need to be answered. If we have effective merging, we have shown that for some types of queries using query-based sampling for local context analysis can boost performance.

We would like to investigate if it is possible to improve the collection selection phase of distributed retrieval. If we can do much better at collection selection, then we can bypass the document ranking merge issue by using expanded queries for collection selection only. We were surprised to see that local context analysis only gave negligible boosts to collection selection when sampling around one quarter of each collection. Perhaps much of the power of local context analysis for query expansion lies primarily in its ability to add good phrases to queries. If so, its failure here would be less surprising, because the resource description used by the CORI algorithm does not support phrases.

6 ACKNOWLEDGEMENTS

We thank Victor Lavrenko and the CIIR for providing us with Victor's local context analysis programs. We also thank Jinxi Xu for clarifications

of his algorithm and for providing us his data. This work was sponsored in full by the Advanced Research and Development Activity in Information Technology (ARDA) under its Statistical Language Modeling for Information Retrieval Research Program. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors, and do not necessarily reflect those of the sponsor.

7 REFERENCES

[1] J. Allan, M.E. Connell, W.B. Croft, F-F Feng, D. Fisher, and X. Li. INQUERY and TREC-9. In Proceedings of the Ninth Text Retrieval Conference (TREC-9), 2000.

[2] C. Buckley, G. Salton, and J. Allan. The Effect of Adding Relevance Information in a Relevance Feedback Environment. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 292-200, 1994.

[3] J. Callan. Distributed IR testbed definition: trec123-100-bysource-callan99.v2a. Technical report, Language Technologies Institute, Carnegie Mellon University, 2000. Available at http://www.cs.cmu.edu/~callan/Data/.

[4] J. Callan. Distributed Information Retrieval. In W. B. Croft, editor, *Advances in Information Retrieval*, 127-150, 2000. Kluwer Academic Publishers.

[5] J. Callan, M. Connell. Query Based Sampling of Text Databases. In *ACM Transactions of Information Systems*, 97-130, 19 (2), 2001.

[6] N. Craswell, P. Bailey, and D. Hawking. Server Selection on the World Wide Web. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, 37-46, 2000.

[7] N. Craswell, D. Hawking, and P. Thistlewaite. Merging Results from Isolated Search Engines. In *Proceedings of the Tenth Australian Database Conference*, 1999.

[8] J. French, A. Powell, J. Callan, C. Viles, T. Emmitt, K. Prey, and Y. Mou. Comparing the Performance of Database Selection Algorithms. In *Proceedings of the Twenty-Second Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 238-245, 1999.

[9] L. Gravano, H. Garcia-Molina, and A. Tomasic. GlOSS: Text-Source Discovery over the Internet. In *ACM Transactions on Database Systems*, vol. 24, no. 2, Jun. 1999.

[10] D. Hawking and P. Thistlewaite. Methods for Information Server Selection. In *ACM Transactions on Information Systems*, vol. 17, no. 1, Jan. 1999.

[11] L. Larkey, M. Connell, and J. Callan. Collection Selection and Results Merging with Topically Organized U.S. Patents and TREC Data." In *Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM)*, 282-289, 2000.

[12] A. Powell, J. French, J. Callan, M. Connell, and C. Viles. The Impact of Database Selection on Distributed Searching. In *Proceedings of the Twenty-Third Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 232-239, 2000.

[13] A. Smeaton and F. Crimmins. Relevance Feedback and Query Expansion for Searching the Web: A Model for Searching a Digital Library. In *Research and Advanced Technology for Digital Libraries, First European Conference on Digital Libraries*, 99-112, 1997.

[14] J. Xu and W.B. Croft. Query Expansion Using Local and Global Analysis. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 4-11, 1996.

[15] J. Xu and J. Callan. Effective Retrieval with Distributed Collections. In *Proceedings of the Twenty-First Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 112-120, 1998.

[16] B. Yuwono, and D. L. Lee. Server Ranking for Distributed Text Retrieval Systems on the Internet. In *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications*, 41-49, 1997.

	Title Queries		Description Queries		Concept Queries	
docs/db	No QE	With QE	No QE	With QE	No QE	With QE
2900	31.4	31.2 (- 0.6%)	31.6	31.8 (+ 0.6%)	34.3	34.8 (+ 1.5%)
1450	31.4	30.5 (- 2.9%)	31.6	30.9 (- 2.2%)	34.3	35.0 (+ 2.0%)
725	31.4	30.8 (- 1.9%)	31.6	29.5 (- 6.6%)	34.3	34.0 (- 0.9%)
362	31.4	28.7 (- 9.4%)	31.6	28.5 (- 9.8%)	34.3	33.5 (- 2.3%)

Table 16. Query-based document sampling was used to collect documents to build a query expansion database. This table shows the effects on collection selection using full information. Values are percentage relevant documents available when selecting ten databases to search.

	n = 4 ~270000 docs		n = 8 ~135000 docs		n = 16 ~67500 docs	
Precision	No QE	With QE	No QE	With QE	No QE	With QE
at 5 docs	0.4620	0.4200 (- 9.0%)	0.4620	0.4410 (- 4.5%)	0.4620	0.4300 (- 6.9%)
at 10 docs	0.4260	0.4110 (- 3.5%)	0.4260	0.4235 (- 0.5%)	0.4260	0.3950 (- 7.2%)
at 15 docs	0.4087	0.3943 (- 3.5%)	0.4087	0.4096 (+ 0.2%)	0.4087	0.3793 (- 7.1%)
at 20 docs	0.3870	0.3810 (- 1.5%)	0.3870	0.3952 (+ 2.1%)	0.3870	0.3612 (- 6.6%)
at 30 docs	0.3683	0.3625 (- 1.5%)	0.3683	0.3660 (- 0.6%)	0.3683	0.3347 (- 9.1%)

Table 17. Every-n'th sampling was used to collect documents to build a query expansion database. Full information was used for collection selection. Table shows distributed retrieval performance of TITLE queries.

	n = 4 ~270000 docs		n = 8 ~135000 docs		n = 16 ~67500 docs	
Precision	No QE	With QE	No QE	With QE	No QE	With QE
at 5 docs	0.4780	0.3960 (-17.1%)	0.4780	0.4030 (-15.6%)	0.4780	0.3680 (-23.0%)
at 10 docs	0.4470	0.3760 (-15.8%)	0.4470	0.3810 (-14.7%)	0.4470	0.3385 (-24.2%)
at 15 docs	0.4267	0.3650 (-14.4%)	0.4267	0.3684 (-13.6%)	0.4267	0.3226 (-24.3%)
at 20 docs	0.4110	0.3515 (-14.4%)	0.4110	0.3522 (-14.2%)	0.4110	0.3105 (-24.4%)
at 30 docs	0.3997	0.3297 (-17.5%)	0.3997	0.3240 (-18.9%)	0.3997	0.2893 (-27.6%)

Table 18. Same as Table 17, but shows DESCRIPTION queries.