Searching Distributed Collections With Inference Networks

James P. Callan Zhihong Lu W. Bruce Croft

Computer Science Department, University of Massachusetts
Amherst, MA 01003-4610, USA
{callan,zlu,croft}@cs.umass.edu

Abstract

The use of information retrieval systems in networked environments raises a new set of issues that have received little attention. These issues include ranking document collections for relevance to a query, selecting the best set of collections from a ranked list, and merging the document rankings that are returned from a set of collections. This paper describes methods of addressing each issue in the inference network model, discusses their implementation in the INQUERY system, and presents experimental results demonstrating their effectiveness.

1 Introduction

Retrospective document retrieval is usually described as the task of searching a single collection of documents to produce a list of documents ranked in order of relevance to a particular query. The need to search multiple collections in distributed environments is becoming increasingly important as the sizes of individual collections grow and network information services proliferate. Distributed collections can be relatively homogeneous, as in the case where a large single collection is partitioned and distributed over a local network to improve search efficiency. They can also be very heterogeneous in that wide-area network services can make hundreds or even thousands of collections available for searching.

Searching a distributed collection presents a number of unique problems. One approach would be to treat the distributed collections as a single, large, "virtual" collection. Every collection would be searched individually and then the results would be combined or merged to produce a single ranked list. One problem with this approach is how to merge the individual ranked lists. The other problems have to do with the economic aspects of searching. It will generally be too expensive in terms of both computer and communication resources and the user's time to search every collection in a distributed environment. Some systems make this clear by making charges for searching dependent on the number of collections searched.

When many collections are available in distributed environment, therefore, a decision must be made about which of them to search. A retrieval system should provide techniques that can make this decision automatically, because users may be unable or unwilling to make selections by exhaustively examining long lists of the available collections. Having selected the collections to search, the retrieval system must also provide techniques for effectively merging the individual ranked lists of documents that are produced.

This paper describes how these issues can be addressed in a retrieval system based on the inference net, a probabilistic model of information retrieval. In the next section, we describe related work on collection selection and merging of ranked results. In Section 3, we describe how the inference network can be used to rank collections for relevance to a query. Section 4 presents a method for accurately merging the results from different collections based on the collection ranking. The results in Section 5 show that it is possible to select subsets of the available collections for searching without affecting retrieval effectiveness. Section 6 describes several efficiency optimizations for distributed searching. In the final section, we summarize the results and discuss some unsolved problems.

2 Related Work

Users of commercial retrospective information retrieval systems have always faced the *collection selection* problem. The user must either search all collections or choose the subset to be searched. Experienced users, for example librarians acting as intermediaries, may draw upon their past experience or reference aids to help in deciding which collections to search. Many less experienced choose to search all available collections rather than take the time to select a subset by trial-and-error.[1]

Some service providers manually group their collections into sets with common themes, for example newspaper collections or appellate court decisions. Danzig, et al, showed how to automatically maintain similar groupings in distributed environments [3]. They used *broker agents* that maintained centralized indices for particular subjects by periodically querying remote collections. Both of these approaches simplify collection selection for users whose information needs can be anticipated to some extent.

The EXPERT CONIT retrieval system [7] is an early example of automating collection selection. EXPERT decided on a query-by-query basis which collections were most appropriate, albeit for a relatively static set of homogeneous

[1] Personal communication from a commercial retrieval service.

To appear at the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.

collections. It used rule-based inferencing to match the information need to a knowledge-base describing document collections, producing a ranked list of collections.

Voorhees, et al, explored ranking collections using the similarity of a new query to training queries [13]. Relevance judgements for the most similar training queries determine whether, and how much, to retrieve from each collection. This technique may be practical for relatively static collections, but obtaining relevance judgements could be problematic for widely distributed and dynamic collections.

GLOSS [5] estimates the number of potentially relevant documents in collection C for a Boolean AND query Q as $|C| \cdot \Pi_{t \epsilon Q} \frac{df_t}{|C|}$, where t is a term in Q, df_t is the number of documents in C containing t, and $|C|$ is the number of documents in C. The GLOSS approach is easily applied to large numbers of dynamic collections, because GLOSS stores only term frequency information about each collection. It's effectiveness is not known yet, due to limited evaluation and the lack of support for other forms of query.

Moffat, et al, used a centralized index on blocks of B documents in individual collections [8]. For example, each block might be 10 documents concatenated together. A new query first retrieves block identifiers from the centralized index, then searches the highly ranked blocks to retrieve documents. This approach worked well for retrieving small numbers of documents, but caused a significant decrease in precision and recall when 1,000 documents had to be retrieved.

Once a set of collections is chosen, the retrieval system must decide how to combine search results from each collection into a single ranking. This task is simple if the results are an unordered set of documents, but it is more difficult if results are ranked lists of documents. Some have successfully used document scores from the different collections to create a merged ranking [6; 8], but others have had problems with this approach [4].

Voorhees, et al, call this the *collection fusion* problem, and describe two solutions [13]. One solution is to interleave the rankings, in a round-robin fashion. A second solution is uneven interleaving, biased by the expected relevance of the collection to the query. The latter approach was substantially more effective in experiments with the TREC collection.

3 Ranking Collections With Inference Networks

Inference networks are a probabilistic approach to information retrieval [12; 11]. The traditional use of inference networks for document retrieval is a directed acyclic graph in which documents are represented by leaves, and the root node represents an information need (Figure 1).

A major part of the collection selection problem is ranking collections for a given information need. Ranking collections can be addressed by an inference network in which the leaves (the d nodes in Figure 1) represent document collections, and the representation (r) nodes represent the terms that occur in the collection. The probabilities that flow along the arcs can be based upon statistics that are analogous to tf and idf in normal document retrieval; for example, document frequency df (the number of documents containing the term) and inverse collection frequency icf (the number of collections containing the term). We call this type of inference network a *collection retrieval inference network*, or *CORI net* for short, to distinguish it from the more common document retrieval inference networks.

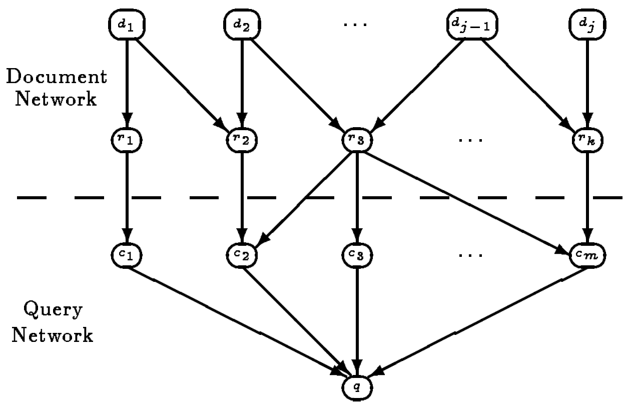


Figure 1: A simple document retrieval inference network.

A CORI net has moderate storage requirements if only document frequency (*df*) and inverse collection frequency (icf) are stored.[2] The CORI net for one 1.2 gigabyte collection (TREC Volume 1) would be about 5 megabytes, assuming simple compression algorithms. In this case, the CORI net is about about 0.4% the size of the original collection.

One advantage of using the inference network for ranking collections is that one system is used for ranking both documents and collections. It is not necessary to design new file organizations or algorithms. Instead, document retrieval becomes a four step process:

1. Use the query to retrieve a ranked list of collections,
2. Select the top group of collections,
3. Search the top group of collections, in parallel or sequentially, and
4. Merge the results from the various collections into a single ranking.

Steps 1 and 3 can be performed by a single algorithm operating on different indices.

To the retrieval algorithm, a CORI network looks like a document retrieval inference network with very big documents; each 'document' is a surrogate for a complete collection. Search complexity is comparable to searching small collections of abstracts. A CORI network for 3,000 document collections is comparable to searching the well-known collection of CACM abstracts. The 'tf' (*df*) and '*idf*' (icf) values are higher, but that does not affect the computational complexity of retrieval. There are also many more inverted lists, but only those that match terms in the query are accessed.

The effectiveness of this approach to ranking collections was evaluated using the INQUERY retrieval system [11; 12; 2] and the 3 gigabyte TREC document collection. The TREC collection is heterogeneous, containing 17 subcollections from different sources and/or periods of time (Table 1). The subcollections vary widely in size, in number of documents, and in average document length. Experiments were conducted with 100 queries developed for TREC topics 51-150 during previous TREC and TIPSTER evaluations [1].

[2]The inference network can incorporate proximity information and operators, but it is impractical to do so for collection ranking.

Table 1: The TREC document collections used for experiments. The TREC volume number is shown in parentheses.

Name	Documents	Words	Megabytes
AP '88 (2)	79,919	21,425,011	249
AP '89 (1)	84,678	22,407,342	267
AP '90 (3)	78,321	21,555,502	249
DOE (1)	226,087	17,201,000	193
Fed. Reg. '88 (2)	19,860	20,068,562	219
Fed. Reg. '89 (1)	25,960	23,444,637	272
Patent (3)	6,711	19,624,651	254
SJM '91 (3)	90,257	36,441,456	301
WSJ '87 (1)	46,448	11,562,767	132
WSJ '88 (1)	39,904	9,738,438	109
WSJ '89 (1)	12,380	3,307,151	38
WSJ '90 (2)	21,705	6,500,181	73
WSJ '91 (2)	52,652	12,418,568	146
WSJ '92 (2)	10,163	2,880,247	35
Ziff 1 (1)	75,180	20,374,002	254
Ziff 2 (2)	56,920	15,637,443	184
Ziff 3 (3)	161,021	44,120,132	362

Table 2: The average optimal rank of the TREC Volume 1 collections for topics 51-100 and 101-150.

	WSJ '87	AP '89	WSJ '88	WSJ '89	DOE	Ziff	FR '89
51-100	2.3	2.6	2.9	4.7	4.9	5.3	5.4
101-150	2.4	2.0	2.9	4.5	4.6	4.9	6.6

The mean-squared error metric was used to compare the effectiveness of variations to the basic collection ranking algorithms. The mean-squared error of the collection ranking for a single query is calculated as:

$$\frac{1}{|C|} \cdot \Sigma_{i \epsilon C} (O_i - R_i)^2$$

where:

O_i = optimal rank for collection i, based on the number of relevant documents it contained (the collection with the largest number of relevant documents is ranked 1, the collection with second largest number of relevant documents is ranked 2, and so on),

R_i = the rank for collection i determined by the retrieval algorithm, and

C = the set of collections being ranked.

The mean-squared error metric has the advantage that it is easy to understand (an optimal result is 0), and it does not require labeling a collection 'relevant' or 'not relevant' for a particular query. The average optimal rank O_i for each collection in TREC Volume 1 is shown in Table 2.

INQUERY's algorithms for ranking documents have been documented extensively [11; 12; 2; 1], so this discussion is confined to the changes necessary to rank collections. The changes were confined initially to replacing tf with df and idf with icf, as discussed above, and with replacing the maximum term frequency in a document statistic (max_tf) with the maximum document frequency in a collection (max_df). Hence, the belief $p(r_k|c_i)$ in collection c_i due to observing term r_k is determined by:

$$T = d\_t + (1 - d\_t) \cdot \frac{\log(df + 0.5)}{\log(max\_df + 1.0)} \quad (1)$$

$$I = \frac{\log\left(\frac{|C|+0.5}{cf}\right)}{\log(|C| + 1.0)} \quad (2)$$

$$p(r_k|c_i) = d\_b + (1 - d\_b) \cdot T \cdot I \quad (3)$$

where:

df	is the number of documents in c_i containing r_k,
max_df	is the number of documents containing the most frequent term in c_i,
$\|C\|$	is the number of collections,
cf	is the number of collections containing term r_k,
d_t	is the minimum term frequency component when term r_k occurs in collection c_i,
d_b	is the minimum belief component when term r_k occurs in collection c_i.

This is a variation of the well-known $tf.idf$ approach to ranking documents, with values normalized to remain between 0 and 1, and further modified by default term frequency (d_t) and default belief (d_b) values. d_t and d_b default to 0.4.

The probabilistic query operators that combine the beliefs accruing from the query terms remained unchanged [11; 12]. The proximity operators were replaced by strict Boolean AND operators, due to the lack of proximity information in CORI nets (discussed in Section 6.2).

This approach was used to rank TREC volume 1 collections for topics 51-100. The mean-squared error, averaged over 50 queries, was 2.3471. The rankings for about 75% of the queries were nearly perfect; the rankings for the other 25% were more disorganized, and accounted for most of the error. No pattern was apparent to explain why some queries yielded such poor rankings.

One possible problem in applying the default formulas for ranking documents to ranking collections is the use of the max_df statistic to scale df. Although we have argued that ranking collections is analogous to ranking documents, there are differences. The reason for ranking collections is *not* to find collections about a particular subject; it is to find collections containing as many documents as possible about the subject. Scaling df by max_df tends to obscure small (and not-so-small) sets of interesting documents in large collections.

Recent experiments with document retrieval suggest that it may be better to scale tf by $tf + K$, for some small K [10]. The analogue for this task would be to scale df by $df + K$, replacing Equation 1 above with Equation 4 below.

$$T = d\_t + (1 - d\_t) \cdot \frac{df}{df + K} \quad (4)$$

When ranking documents, it makes sense to make K a function of document length. However, when ranking collections, it may make more sense to let K be more sensitive to the number, and not the percentage, of documents about the subject. It may also make sense to let K be large, because the df values will generally be large.

We defined $K = k \cdot ((1 - b) + b \cdot cw/\overline{cw})$, where k and b are constants, cw is the number of words in the collection, and $\overline{cw}$ is the mean cw of the collections being ranked. The constant k controls the magnitude of K. Varying b from 0 to 1 increases the sensitivity of K to the size of the collection.

This approach was used to rank TREC volume 1 collections for topics 51-100. Experiments were conducted with

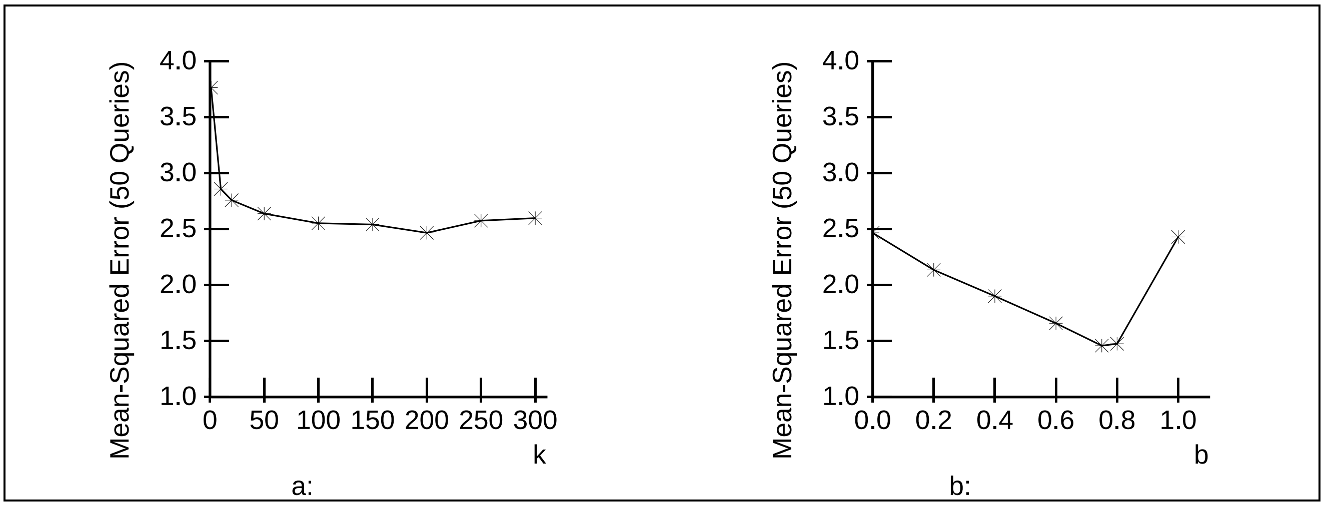


Figure 2: The effect on mean-squared error of varying K (TREC Volume 1 collections, topics 51-100)). In a) $b = 0$ as k is varied. In b) $k = 200$ as b is varied.

values of k ranging from 1 to 300 (Figure 2a), and with values of b ranging from 0 to 1 (Figure 2b). The best combination of values for this set of queries and collections is $k = 200, b = 0.75$.

The mean-squared error, averaged over 50 queries, for this combination was 1.4586, which is 38% better than the mean-squared error obtained when scaling *df* by *max_df*. The collection rankings improved for 30 queries, some quite dramatically. The rankings for 8 queries deteriorated slightly. The rankings for 12 queries did not change.

A mean-squared error of 1.4586 is not perfect, but it is good. Analysis of the results reveals few serious mistakes. Most of the mistakes were due to mixing up the rankings of collections containing nearly equal numbers of relevant documents. These cases were counted as errors, but they would not be noticeable to a user. Indeed, although further improvement is possible, it is likely to yield diminishing returns.

4 Merging Results

Ranking collections is only part of the problem. After a set of collections is searched, the ranked results from each collection must be merged into a single ranking.

If only the document rankings are available, the results from each collection can be *interleaved* [13]. This solution is not satisfying, for it is unlikely that all of the collections have equal numbers or proportions of relevant documents. However, it is difficult to do anything more sophisticated without more information than just the document rankings.

Many IR systems return not only a ranking of documents, but a numeric score that indicates how well each document matches the query. If the scores from different collections are comparable, one can merge the set of rankings based upon the document scores (*raw score* merge).

With some techniques, the scores from different collections may not be directly comparable. For example, although the *idf* weights for many words are relatively consistent across different collections, the *idf* weights of words such as *computer*, *tort* and *cholesterol* will vary widely among technical, legal and medical collections. This can viewed as desirable, because the *idf* represents the term's importance in a particular collection. It can also be viewed as undesirable, because an important query term can behave erratically, rewarding the random mention of a term in one collection and penalizing its common use in another [4].

The problem of incomparable scores can be overcome in some cases by *normalizing* statistics such as *idf* for the set of collections being searched [6]. The intent is to normalize document scores to obtain precisely the same results that would be obtained if the individual document collections were merged into a single unified collection. The difficulty of normalizing document scores for a set of collections depends upon the retrieval algorithms employed. For the inference network architecture, normalizing scores requires a preprocessing step prior to query evaluation. In the preprocessing step, the system obtains from each collection the statistics about how many documents each query term or proximity operator matches. The statistics are merged to obtain a normalized *idf*. The query and the normalized *idf*s are sufficient to then retrieve documents with comparable scores from disparate collections.

Normalizing document scores can entail significant communication and computational costs when collections are distributed across a wide-area network. An alternative to both simple interleaving and normalized scores is merging based on *weighted scores*. Weights can be based upon a document's score and/or the collection ranking information. This approach offers the computational simplicity of simple interleaving while overcoming some of its disadvantages.

The weight w, below, is an example of how one might weight results from different collections. We have used the collection's score, instead of its rank, because we believe that similar collections should have similar weights.

$$w = 1 + |C| \cdot \frac{s - \bar{s}}{\bar{s}}$$

where:

$\lvert C \rvert$	=	the number of collections searched,
s	=	the collection's score, and
$\bar{s}$	=	the mean of the collection scores.

Each document is ranked based upon the product of its score

Table 3: Different techniques for merging results from different collections (TREC Volume 1, topics 51-100).

Interpolated Recall - Precision Averages (50 queries):	Normalized Merge	Interleaved Merge		Raw Score Merge		Weighted Merge	
0%	88.37	78.95	(−10.7)	83.14	(−5.9)	85.92	(−2.8)
10%	65.99	38.67	(−41.4)	60.08	(−9.0)	65.15	(−1.3)
20%	57.49	29.42	(−48.8)	52.49	(−8.7)	58.21	(+1.3)
30%	51.23	24.76	(−51.7)	45.88	(−10.4)	51.84	(+1.2)
40%	45.63	20.13	(−55.9)	40.53	(−11.2)	45.92	(+0.6)
50%	38.65	14.64	(−62.1)	34.23	(−11.4)	39.50	(+2.2)
60%	32.95	10.74	(−67.4)	27.57	(−16.3)	32.83	(−0.4)
70%	26.25	6.81	(−74.1)	21.83	(−16.8)	26.28	(+0.1)
80%	17.31	4.29	(−75.2)	15.24	(−12.0)	18.22	(+5.3)
90%	8.36	1.43	(−82.9)	6.71	(−19.7)	10.64	(+27.3)
100%	1.22	0.33	(−73.0)	1.18	(−3.3)	1.25	(+2.5)
Average precision (non-interpolated) over all rel docs							
	37.76	17.54	(−53.5)	33.67	(−10.8)	38.18	(+1.1)
Precision:							
5 docs:	69.60	52.80	(−24.1)	63.60	(−8.6)	67.60	(−2.9)
10 docs:	65.00	43.20	(−33.5)	59.20	(−8.9)	64.20	(−1.2)
15 docs:	62.93	38.00	(−39.6)	59.47	(−5.5)	62.00	(−1.5)
20 docs:	62.10	37.30	(−39.9)	56.20	(−9.5)	61.60	(−0.8)
30 docs:	59.00	35.40	(−40.0)	53.67	(−9.0)	59.33	(+0.6)
100 docs:	46.76	29.82	(−36.2)	43.66	(−6.6)	47.76	(+2.1)
R-Precision (precision after R (= num_rel for a query) docs retrieved):							
Exact:	41.96	25.42	(−39.4)	38.67	(−7.8)	42.46	(+1.2)

and the weight w for its collection. This algorithm favors documents from collections with high scores, but also enables a good document from a poor collection to be ranked highly.

These four approaches to merging results, *interleaving*, *raw scores*, *normalized scores* and *weighted scores*, were compared in a series of experiments using the collections and queries described in Section 3. Experiments were conducted with two query sets (51-100 and 101-150) on TREC Volume 1 (7 collections), Volume 2 (6 collections), Volume 3 (4 collections), Volumes 1+2 (13 collections), and Volumes 1+2+3 (17 collections). Results are shown for Volume 1, and Volumes 1+2+3. The *normalized scores* approach was treated as the baseline, because it is equivalent to the "single database" paradigm that has been the norm in information retrieval. The experimental results are contained in Tables 3-5.

Both 11 point recall/precision and absolute precision at various cutoffs are included, to give a clear picture of behavior at both low and high recall. Recall and precision were calculated by the TREC `trec2_eval` program using just the top 1000 documents from the merged ranking. This approach treats all documents above rank 1000 as ranked last, giving a very pessimistic view of high recall behavior.

In each experiment, simple interleaving of document rankings was extremely ineffective, producing dramatic losses in average precision. This result is not surprising, because interleaving has the effect of boosting the rankings of random documents from collections with few relevant documents.

Merging based on raw document scores from each collection was significantly worse than ranking based on normalized document scores, causing losses from 10-20% in average precision. The only difference between the normalized and unnormalized scores was the *idf* component. This result confirms previous research suggesting that unnormalized *idf*s can give misleading results [4].

Ranking based on document scores and collection-specific weights was about as effective as ranking based on normalized scores. It produced small improvements in most levels of recall for queries 51-100 on the TREC 1 collections; small losses in most levels of recall for queries 101-150 on the TREC 1 collections; and small changes in precision at most levels of recall for queries 51-100 on the TREC 1+2+3 collections. In general, our experience with weighted rankings was small changes at low recall, and occasionally erratic behavior at high recall.

We view these results as extremely encouraging. They suggest that it is possible to get the accuracy of rankings based on normalized scores without the computational effort.

5 Collection Selection

In the experiments described above, all collections were searched, and their results used to create the final document rankings. However, in distributed environments, one rarely has the resources to search every collection. It is more likely that one searches an index of collections, obtains a ranking, and then searches the top few collections for interesting documents.

There are any number of ways to decide how far down the collection rankings to go. One could choose the top n, any collection with a score greater than some threshold, or the top group as defined by some clustering method. We investigated the latter approach.

A single-pass algorithm [9] was used to cluster the collection rankings for each query. Collections were clustered on the basis of their scores, as determined by the collection ranking algorithm (Section 3). The cluster difference threshold was low (0.0012), creating clusters that tended to be smaller than necessary. The seven TREC Volume 1 collections produced an average of 4.04 clusters for topics 51-100 and 4.06 clusters for topics 101-150. It was rare for a collection with a significant number of relevant documents to be excluded from the top 2 clusters.

Table 4: Different techniques for merging results from different collections (TREC Volume 1, topics 101-150).

Interpolated Recall - Precision Averages (50 queries):	Normalized Merge	Interleaved Merge		Raw Score Merge		Weighted Merge	
0%	84.05	80.36	(−4.4)	82.91	(−1.4)	79.64	(−5.2)
10%	61.62	42.65	(−30.8)	57.25	(−7.1)	60.76	(−1.4)
20%	54.66	33.35	(−39.0)	49.94	(−8.6)	53.35	(−2.4)
30%	49.54	27.72	(−44.0)	45.16	(−8.8)	48.19	(−2.7)
40%	44.76	22.75	(−49.2)	38.72	(−13.5)	41.86	(−6.5)
50%	38.18	18.26	(−52.2)	32.71	(−14.3)	37.03	(−3.0)
60%	32.68	14.57	(−55.4)	27.82	(−14.9)	30.92	(−5.4)
70%	26.13	8.54	(−67.3)	22.05	(−15.6)	25.30	(−3.2)
80%	19.01	4.62	(−75.7)	16.46	(−13.4)	18.47	(−2.8)
90%	10.66	1.66	(−84.4)	9.40	(−11.8)	9.70	(−9.0)
100%	1.47	0.35	(−76.2)	1.39	(−5.4)	1.15	(−21.8)
Average precision (non-interpolated) over all rel docs							
	36.98	20.48	(−44.6)	33.22	(−10.2)	35.60	(−3.7)
Precision:							
5 docs:	63.20	54.40	(−13.9)	60.80	(−3.8)	62.00	(−1.9)
10 docs:	58.80	50.00	(−15.0)	55.80	(−5.1)	58.20	(−1.0)
15 docs:	59.13	43.73	(−26.0)	54.00	(−8.7)	56.00	(−5.3)
20 docs:	57.30	41.00	(−28.4)	52.30	(−8.7)	55.50	(−3.1)
30 docs:	55.60	38.47	(−30.8)	50.27	(−9.6)	53.80	(−3.2)
100 docs:	43.92	29.06	(−33.8)	40.04	(−8.8)	43.34	(−1.3)
R-Precision (precision after R (= num_rel for a query) docs retrieved):							
Exact:	39.85	26.50	(−33.5)	36.96	(−7.3)	38.47	(−3.5)

The effect on recall and precision was noticeable, but not significant at low recall. We summarize the results, rather than providing complete recall/precision tables, due to space limitations. With topics 51-100, the difference between using the two best clusters of collections (an average of 4.04 collections) and using all 7 collections was less than −5% at all document cutoffs from 5 to 1,000. The difference in exact R-precision was −0.9%. The difference in average 11 point precision was −2.2%. The effect on topics 101-150 was more noticeable, with differences > 5.0% at and above the 200 document cutoff. The difference in exact R-precision was −5.6%, and the difference in average 11 point precision was −9.1%, both reflecting significant deterioration in high recall results.

It is not surprising that eliminating collections reduces recall. There will occasionally be errors in the collection rankings, or a few relevant documents in marginal collections. In our opinion it is more significant, and encouraging, that precision at low recall is relatively unaffected when the number of collections searched is reduced by 43%. Few interactive users will care about a drop in recall at rank 200. Those that do can choose to search more collections.

6 Optimization

In this section we discuss several decisions that can affect the efficiency of collection ranking and merging rankings.

6.1 Represent a Collection With a Subset of Terms

One could create a smaller collection retrieval inference net by keeping only a subset of the terms that appear in the collection. An example would be to keep only the most frequent. An experiment with TREC Volume 1 and topics 51-100 investigated the effect of building inference nets with different percentages of the most frequent terms in the collection. Figure 3a shows the mean-squared error, averaged over 50 queries. Figure 3b shows the effect on average 11 point precision. The results suggest that it is necessary to store at least the 20% most frequent terms, and that there is some advantage to storing all of the terms.

6.2 Proximity Information

The INQUERY system [2], which is based on the inference network model, extends the inference network formalism to include proximity operators. One could also use term proximity information for collection ranking, but it would require that the location of each term in each document in each collection be stored in the CORI index. Although it is possible to do so, the CORI net for one collection would become about 30% the size of the original collection.

6.3 Retrieving Fewer Documents

If a user wants to retrieve n documents from a set of C collections, one could retrieve n documents from each collection, merge the rankings, and then discard everything below rank n. However, this approach is costly if collections are distributed widely across networks, because $(C-1)\cdot n$ documents are retrieved, sent across the network, and then discarded without a user seeing them [13]. This cost raises the question of whether it is possible to safely retrieve fewer than n documents from collections with low ranks or scores.

We have experimented with a heuristic that uses the collection ranking information to decide how much to retrieve from each collection. The number of documents R retrieved from the i'th ranked collection is:

$$R(i) = M \cdot n \cdot \frac{1+C-i}{\Sigma_{j=1}^{C} j} = M \cdot n \cdot \frac{2\cdot(1+C-i)}{C\cdot(C+1)}$$

where $M \epsilon [1.0, \frac{C+1}{2}]$, and $M \cdot n$ is the number of documents to be retrieved from all collections.

This heuristic is a linear function that allocates a predetermined number of document retrievals $(M \cdot n)$ across C collections. If $C = 5$ and $M = 2$, the first collection retrieves

Table 5: Different techniques for merging results from different collections (TREC Volumes 1, 2 and 3, topics 51-100).

Interpolated Recall - Precision Averages (50 queries):	Normalized Merge	Interleaved Merge		Raw Score Merge		Weighted Merge	
0%	86.50	77.15	(−10.8)	84.41	(−2.4)	86.54	(+0.0)
10%	59.90	33.81	(−43.6)	54.04	(−9.8)	60.09	(+0.3)
20%	50.96	24.51	(−51.9)	45.88	(−10.0)	50.30	(−1.3)
30%	43.32	16.28	(−62.4)	38.91	(−10.2)	42.04	(−3.0)
40%	33.78	10.06	(−70.2)	29.35	(−13.1)	35.04	(+3.7)
50%	24.67	7.01	(−71.6)	19.71	(−20.1)	23.47	(−4.9)
60%	16.12	4.44	(−72.5)	12.95	(−19.7)	16.61	(+3.0)
70%	12.19	2.04	(−83.3)	10.86	(−10.9)	11.53	(−5.4)
80%	7.44	0.41	(−94.5)	5.55	(−25.4)	6.45	(−13.3)
90%	3.20	0.33	(−89.7)	2.53	(−20.9)	2.76	(−13.8)
100%	0.00	0.00	(Inf)	0.00	(Inf)	0.00	(Inf)
Average precision (non-interpolated) over all rel docs							
	28.28	12.34	(−56.4)	25.11	(−11.2)	27.91	(−1.3)
Precision:							
5 docs:	68.80	60.00	(−12.8)	64.40	(−6.4)	68.00	(−1.2)
10 docs:	67.00	53.40	(−20.3)	61.60	(−8.1)	66.20	(−1.2)
15 docs:	64.67	42.00	(−35.1)	59.47	(−8.0)	64.53	(−0.2)
20 docs:	63.40	40.90	(−35.5)	57.60	(−9.1)	62.80	(−0.9)
30 docs:	61.27	39.80	(−35.0)	57.93	(−5.5)	61.33	(+0.1)
100 docs:	53.80	32.76	(−39.1)	49.80	(−7.4)	54.40	(+1.1)
R-Precision (precision after R (= num_rel for a query) docs retrieved):							
Exact:	36.99	22.86	(−38.2)	34.54	(−6.6)	36.66	(−0.9)

$0.67 \cdot n$ documents, the second retrieves $0.53 \cdot n$ documents, the third retrieves $0.40 \cdot n$, the fourth retrieves $0.27 \cdot n$, and the fifth retrieves $0.13 \cdot n$. More documents are retrieved as M is increased, allowing the user or system to trade cost for safety.

In experiments with TREC volume 1 and topics 51-150, the total number of documents retrieved to produce a final list of 1000 was reduced from 4,050 to 2000 (a 51% savings), with only minimal impact on recall and precision. There was almost no change in which documents were ranked 1 to 500 for each query. The average 11 point precision changed −0.1% for topics 51-100, and +1.0% for topics 101-150.

We also explored the effect of retrieving different numbers of documents by varying M from 1 to 3 by 0.5. For the TREC volume 1 collections, all values produced nearly identical results for ranks 1 to 200. At $M = 2$, they were identical down to rank 500. $M = 3$ produced slightly better results between ranks 500 and 1,000, on both query sets, than smaller values of M. Similar results were obtained, with slightly higher values of M (e.g. $M = 3$ instead of $M = 2$), for the full set of 17 TREC collections.

These results are encouraging, but they are based on relatively accurate methods of ranking and selecting collections for a query. When accurate methods are available, computation and communication costs can be minimized by retrieving fewer documents from each collection. When there is doubt about the collection ranking, it may be worth the added cost to retrieve more documents from each collection.

7 Conclusion

As information retrieval systems are applied in networked environments to widely distributed document collections, the systems will need to provide *collection ranking*, *collection selection* and *results merging* capabilities. It is desirable to provide these capabilities efficiently and transparently, so that users can, if they choose, maintain the illusion of a single 'virtual' collection returning a single, coherent set of results.

This paper describes efficient algorithms for providing these capabilities in systems based on the inference network model of information retrieval. They enable a system to automatically and efficiently rank collections for relevance to a query, select a subset of them, search the subset efficiently, and then accurately merge the results. The effectiveness of the algorithms is demonstrated in experiments with the INQUERY information retrieval system and the TREC set of document collections.

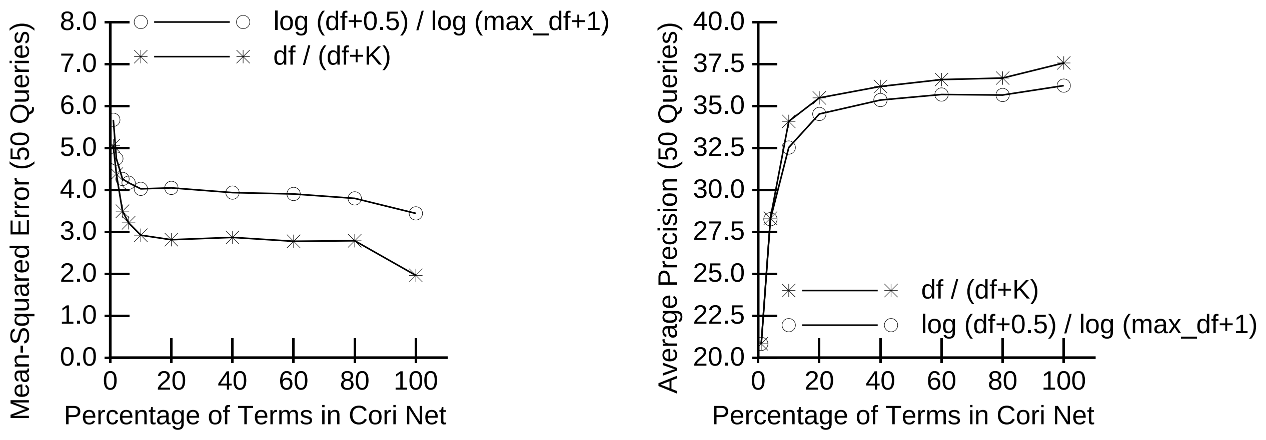
The experimental results are encouraging because simple methods were quite effective with collections that varied widely in size and content. However, they must be viewed as preliminary, because only 17 collections were involved. These methods will be equally efficient with hundreds or thousands of collections, but it is not known whether they will be equally effective.

Distributed collections present several other problems that we have not addressed. In our work, all of the collections used the same stemming algorithm, stopword list, and query processing techniques. When these differ, as they will in collections on wide-area networks, several problems become more difficult. It is less obvious how to represent each collection in the collection retrieval inference network. It is not clear how to provide information to the user about how the information need was transformed into a structured query, because the transformation may be collection-specific. Finally, it is unclear how to perform relevance feedback when the set of relevant documents is scattered across a set of collections with different representations.

The work described in this paper, although positive and encouraging, is merely the first step.

Acknowledgements

This research was partially supported by the NSF Center for Intelligent Information Retrieval at the University of Massachusetts, Amherst.



a: Accuracy of collection rankings.

b: Accuracy of merged document rankings.

Figure 3: The effect of building CORI nets from only the most frequent terms (TREC Volume 1 collections, topics 51-100).

References

[1] J. P. Callan, W. B. Croft, and J. Broglio. TREC and TIPSTER experiments with INQUERY. *Information Processing and Management*, (in press).

[2] J. P. Callan, W. B. Croft, and S. M. Harding. The INQUERY retrieval system. In *Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 78–83, Valencia, Spain, 1992. Springer-Verlag.

[3] P. B. Danzig, J. Ahn, J. Noll, and K. Obraczka. Distributed indexing: A scalable mechanism for distributed information retrieval. In A. Bookstein, Y. Chiaramella, G. Salton, and V. V. Raghavan, editors, *Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 220–229, Chicago, IL, October 1991. Association for Computing Machinery.

[4] S. T. Dumais. Latent semantic indexing (LSI) and TREC-2. In D. K. Harman, editor, *The Second Text REtrieval Conference (TREC-2)*, pages 105–115, Gaithersburg, MD, 1994. National Institute of Standards and Technology, Special Publication 500-215.

[5] L. Gravano, H. García-Molina, and A. Tomasic. The effectiveness of GLOSS for the text database discovery problem. In *Proceedings of SIGMOD 94*, pages 126–137. ACM, September 1994.

[6] K. L. Kwok, L. Grunfeld, and D. D. Lewis. TREC-3 ad-hoc, routing retrieval and thresholding experiments using PIRCS. In D. K. Harman, editor, *The Third Text REtrieval Conference (TREC-3)*, Gaithersburg, MD, (in press). National Institute of Standards and Technology, Special Publication 500-225.

[7] R. S. Marcus. An experimental comparison of the effectiveness of computers and humans as search intermediaries. *Journal of the American Society for Information Science*, 34:381–404, 1983.

[8] A. Moffat and J. Zobel. Information retrieval systems for large document collections. In D. K. Harman, editor, *The Third Text REtrieval Conference (TREC-3)*, Gaithersburg, MD, (in press). National Institute of Standards and Technology, Special Publication 500-225.

[9] Edie Rasmussen. Clustering algorithms. In Frakes and Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, chapter 16, pages 419–442. Prentice Hall, 1992.

[10] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In D. K. Harman, editor, *The Third Text REtrieval Conference (TREC-3)*, Gaithersburg, MD, (in press). National Institute of Standards and Technology, Special Publication 500-225.

[11] H. R. Turtle and W. B. Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222, 1991.

[12] Howard R. Turtle and W. Bruce Croft. Efficient probabilistic inference for text retrieval. In *RIAO 3 Conference Proceedings*, pages 644–661, Barcelona, Spain, April 1991.

[13] E. M. Voorhees, N. K. Gupta, and B. Johnson-Laird. The collection fusion problem. In D. K. Harman, editor, *The Third Text REtrieval Conference (TREC-3)*, Gaithersburg, MD, (in press). National Institute of Standards and Technology, Special Publication 500-225.