Cluster-based Language Models For Distributed Retrieval

Jinxi Xu and W. Bruce Croft
Center for Intelligent Information Retrieval
Computer Science Department
University of Massachusetts, Amherst
Amherst, MA 01003-4610, USA
xu@cs.umass.edu croft@cs.umass.edu

Abstract

Effective retrieval in a distributed environment is an important but difficult problem. Lack of effectiveness appears to have three causes. First, collection selection based on word histograms is not appropriate for heterogeneous collections. Second, relevant documents are scattered over many collections and searching a few collections misses many relevant documents. Third, most existing collection selection metrics lack sound theoretical justifications and hence may not be well tuned to the problem. We propose a new approach to distributed retrieval based on document clustering and language modeling. Document clustering is used to organize collections around topics. Language modeling is used to properly represent topics and effectively select the right topics for a query. Based on these ideas, three methods are proposed to suit different environments. We show that all three methods improve effectiveness of distributed retrieval.

1 Introduction

Information has become highly distributed and will be even more so in the future. The World Wide Web, for example, already consists of millions of web sites and the number of sites keeps growing by thousands each day. It is inefficient and may become impossible to construct a central index for such a huge information system. Designing algorithms to efficiently and effectively organize, represent and search distributed collections is one of the most significant challenges facing Information Retrieval (IR) research.

There have been many studies about distributed retrieval in the IR, digital library, database and World Wide Web communities [2, 23, 6, 8, 22]. A critical problem in distributed retrieval is collection selection. Because a distributed retrieval system may consist of a large number of collections, the only way to ensure timely and economic retrieval is to search a small number of collections which are likely to contain relevant documents for a query. Collection selection is critical for retrieval accuracy for the simple reason that searching the wrong collections containing few or no relevant documents will result in retrieval failure for a query. The other problem is how to merge the retrieval results from different collections. In our opinion, result merging, though important, is a secondary issue. Callan *et al* showed that simple normalizations of document scores from different collections can minimize the impact on retrieval performance [2]. In this paper, we avoid the issue of result merging by assuming that searching different collections produces comparable document scores. Distributed retrieval in the context of the World Wide Web is known as meta searching, but current meta search engines such as MetaCrawler [18] typically send a query to a fixed list of popular search engines and do not perform collection selection.

The most common technique for collection selection is to represent a collection as a word histogram, which is usually a list of words that occur in the collection and the associated frequencies. The virtual document representation in [2, 23] is an example. The word histograms are indexed and the resulting data structure is called the collection selection index [23]. Collection selection consists of simply ranking the word histograms against a query in the same way as ranking ordinary documents. Most other techniques are very similar. Such a technique is a simple modification of document retrieval. Documents and collections of documents are, however, very different and techniques that work well for one problem may not work well for the other. A document usually deals with only one topic but a typical collection can deal with many topics. Matching the words in a query with different topics in a collection can cause failure in collection selection. Suppose a collection contains documents about fruits and computers. Matching the query "Apple Computer" against the histogram of the collection will produce a high similarity even though none of the documents are relevant. Heterogeneous collections therefore make the simple technique of matching a query against word histograms ineffective for collection selection. In fact, if all collections are sufficiently heterogeneous, matching a query consisting of a few common words with word histograms can even produce random collection selection because statistically all histograms are almost identical with relation to the query. Xu and Callan showed that ineffective collection selection seriously hurts the performance of distributed retrieval [23].

Past research has shown that distributed retrieval is markedly less effective than centralized retrieval [23]. Our goal is to improve distributed retrieval and make it as effective as centralized retrieval. The new approach we will propose is inspired in part by document clustering [21] and language modeling [19]. The language modeling approach to IR views retrieval as estimating the probability that a language model can generate a query. Language models were originally used in speech recognition to capture statistical regularities of language generation. In IR, word order is less important and therefore a simple language model can be a probability distribution over the words in a vocabulary set. The probability of a word in a language model can be interpreted as the frequency with which a word is used to describe a certain topic.

Document clustering has been extensively studied in IR. The cluster hypothesis, according to van Rijsbergen, is that "Closely associated documents tend to be relevant to the same requests" [21]. A conjecture in document clustering is that searching a few good clusters for a query can be more effective than a full ranking of all documents. The purpose of document clustering in our work is to group documents according to topics. Each cluster is regarded as a topic. Language modeling is then used to model the statistical regularities of word usage in a topic and represent the topic as a language model. We call such a language model a *topic model* to differentiate from language models estimated for single documents in Ponte's work [19]. In Ponte's work, smoothing was used to reduce estimation error caused by insufficient data. In our approach, since all documents about a topic can be used for parameter estimation, topic models can potentially be more accurately estimated. This enables us to determine with high accuracy which topics are appropriate to search for a query and which topics are not relevant. By focusing on a few topics rich in relevant documents, distributed retrieval can potentially be more accurate than centralized retrieval.

In our approach, the task of a distributed retrieval system is first to determine which topics are best for a query and then to direct the searching process to those collections containing the topics. The problem of determining the best topics is the problem of determining which topic models are most likely to generate a query, which is amenable to mathematical treatment. Since collection selection is based on how well a query matches a topic, we eliminate some errors with the old technique that result from matching the words in a query with different topics of a collection. A disadvantage with our approach is that it requires clustering large sets of documents. The computational cost, however, can be made acceptable even on large collections. Our approach is similar to the technique proposed by Weiss *et al* for organizing Web resources [22] but with significant differences. We provide a range of solutions for different retrieval environments. Collection selection in our approach has a well-defined probabilistic interpretation while it is more heuristic in theirs. Furthermore, their approach lacks thorough evaluation using realistic data sets.

In the next section we describe the basic approach in more detail. In section 3, we describe four methods of organizing a distributed retrieval system. One is the old method of distributed retrieval with heterogeneous collections. The other three are new methods proposed in this paper. Experimental results are presented in sections 5, 6 and 7. The three new methods are evaluated on TREC3, TREC4 and TREC6, using the old method and centralized retrieval as baselines. In section 9 we discuss related work. The final section summarizes this work and suggests future work.

2 Cluster-Based Language Models

2.1 Topic Modeling

In this work, a topic model, i.e. a language model for a topic T, is a probability distribution $\{p_1, p_2, ...p_n\}$ over a vocabulary set $\{w_1, w_2, ...w_n\}$, where p_i is the frequency with which word w_i is used in the text of T when observed with an unlimited amount of data. Suppose we have a set of available documents D about T, p_i is estimated as

$$p_i = \frac{f(D, w_i) + 0.01}{|D| + 0.01n}$$

where $f(D, w_i)$ is the number of occurrences of w_i in D, $|D|$ is the size of D in words and n is the vocabulary size. The small value 0.01 prevents zero probabilities as the Kullback-Leibler divergence described below involves logarithms.

We use the Kullback-Leibler divergence to measure how well a topic model for topic T predicts a query Q:

$$KL(Q, T) = \sum_{f(Q,w_i)\neq 0} \frac{f(Q, w_i)}{|Q|} \log \frac{f(Q, w_i)/|Q|}{p_i}$$

where $f(Q, w_i)$ is the number of occurrences of w_i in Q and $|Q|$ is the length of Q in words. Kullback-Leibler divergence is an important metric in information theory. It has been widely used to measure how well one probability distribution predicts another in many applications such as speech recognition, pattern recognition and so forth. It is a distance metric and falls in $[0, \infty]$. The smaller the value, the better the topic model of T predicts Q. Justification for the metric can be found in textbooks on information theory [16].

2.2 Topic Selection can Improve Retrieval Effectiveness

Language modeling can explain why distributed retrieval with topic selection can be more effective than centralized retrieval. We can model the writing of a document about a topic as a random process. How frequently a certain word is used in the process depends on the topic being addressed. For example, if it is about cooking, "oil" and "fry" may have a high frequency. Due to randomness in the process, a relevant document may use the query words less often than a non-relevant document. As a result, a non-relevant document may have a higher rank than a non-relevant document if we rank the documents individually. Because a collection typically has far more non-relevant documents than relevant documents for a query, the problem can substantially affect retrieval accuracy. For an example, suppose a collection has 100 documents about topic T1 and 1000 documents about topic T2, all documents are 100 words long, and we are interested in T1. Without reading the documents, we can assume they are generated according to two topic models M1 and M2. We assume that the frequencies of words a and b are 0.05 and 0.05 in M1 and 0.04 and 0.03 in M2. Since we know a and b are more frequently used in T1, our query will probably be $\{a, b\}$. If we rank the documents individually using simple cosine metric, the expected precision is only 55% (when 10 documents are retrieved) based on our simulations. If we rank each topic as a unit, we are less likely to make mistakes because random factors are cancelled out in a large sample. The probability that T1 is ranked before T2 is almost 100% using the cosine metric.

The ideal case that all relevant documents belong to one topic and all non-relevant documents belong to other topics is very rare in reality. That is, relevant documents for a query may be distributed over several topics which contain both relevant and non-relevant documents. Even so, topic modeling can be still useful to improve retrieval performance. By focusing on a few highly selective topics which contain most relevant documents for a query, we can eliminate from the top retrieved set many of the non-relevant documents which could be ranked highly by a full search without removing many relevant documents. That is the reason we believe distributed retrieval with accurate topic selection can be more effective than centralized retrieval.

2.3 K-Means Clustering

In this paper, topics are approximated by document clustering. We run a clustering algorithm on a set of documents and treat each cluster as a topic. The two pass K-Means algorithm is chosen for this purpose [13]. In the first pass, the first k documents are treated as the initial clusters. For each new document, we find the closest cluster and add it to that cluster. The second pass corrects possible mistakes made in the first pass. We take the results of the first pass as the initial clusters and walk through the document set one more time. For each document we find the closest cluster and reassign it to that cluster unless it is already there. The distance metric to determine the closeness of a document d to a cluster c is the Kullback-Leibler divergence with some modification

$$KL(d,c) = \sum_{f(d,w_i)\neq 0} \frac{f(d,w_i)}{|d|} log \frac{f(d,w_i)/|d|}{(f(c,w_i)+f(d,w_i))/(|c|+|d|)}$$

where $f(c, w_i)$ is the number of occurrences of word w_i in c, $f(d, w_i)$ is the number of occurrences of w_i in d, $|d|$ is the size of d and $|c|$ is the size of c. The complexity of the algorithm is $O(nk)$ for a set of n documents when k clusters are created.

3 Four Methods of Distributed Retrieval

Four methods of organizing a distributed retrieval system are used in this paper. One is the old method of distributed retrieval with heterogeneous collections [2, 23]. The other three are new methods based on the basic ideas discussed in the previous section. The three new methods are *global clustering*, *local clustering* and *multiple-topic representation*.

3.1 Baseline Distributed Retrieval

The baseline method represents the typical approach to distributed retrieval in previous studies. A distributed retrieval system using this method consists of a number of heterogeneous collections. Documents in one collection are typically from the same source or were written in the same time period. The collection selection index summarizes each collection as a whole. In our experiments a collection is represented as a language model. This method is used as a baseline in our experiments. It is illustrated by Figure 1.

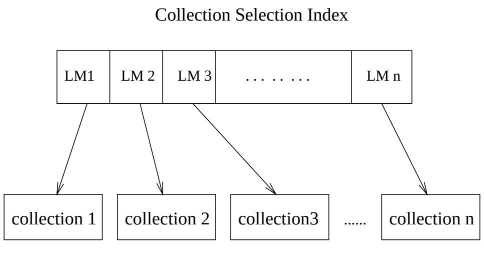

Figure 1: Baseline distributed retrieval

3.2 Global Clustering

Our first new method is global clustering. We assume that all documents are made available in one central repository for us to manipulate. We can cluster all the documents and make each cluster a separate collection. Each collection therefore contains only one topic. Selecting the right collections for a query is the same as selecting the right topics for the query. This method is illustrated by Figure 2.

This method is appropriate for searching very large corpora such as the U.S. Patent and Trademark collection [17] and Internet search engines, where the collection size can be hundreds of Gibabytes and even Terabytes. The size of the collections and the volume of queries to process make efficiency a critical issue. Distributed retrieval can improve efficiency because we do not have to search the whole collections for each query. The baseline method described in section 3.1 would partition a large collection into smaller ones according to attributes such as the sources of the documents and the time of document creation. Such partitions are convenient but undesirable for distributed retrieval. Relevant documents for a query may be scattered in many collections. Therefore we have to either search many collections at the cost of efficiency or search fewer collections at the cost of effectiveness. Furthermore, the resulting collections are often heterogeneous in content and make collection selection difficult. Global clustering produces topic-based collections which are more suitable for distributed retrieval.

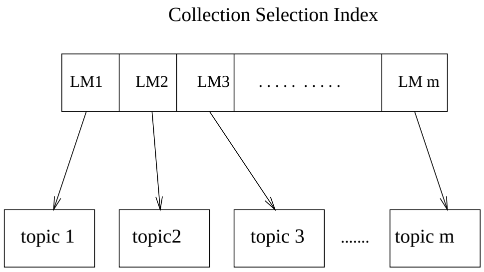

Figure 2: Distributed retrieval with global clustering

Experimental results show that this method can achieve the most effective retrieval. A disadvantage is that creating many clusters can be expensive. The other problem is that it is not appropriate in environments where documents cannot be made available in one place for reasons such as copyright.

3.3 Local Clustering

Our second new method is local clustering. We assume that a distributed system comprises a number of autonomous subsystems. Documents within subsystems are protected and cannot be made available to a central repository. Within a subsystem, however, there is no limitation as how to manipulate the documents. For example, we can imagine a federated retrieval system comprising several for-profit retrieval service providers such as WEST Law, Lexis-Nexis and so forth. Each subsystem can cluster its own documents and make each topic a collection. This method is illustrated by Figure 3.

This method can provide competitive distributed retrieval without assuming full co-operation from the subsystems. The disadvantage is that its performance is slightly worse than that of global clustering.

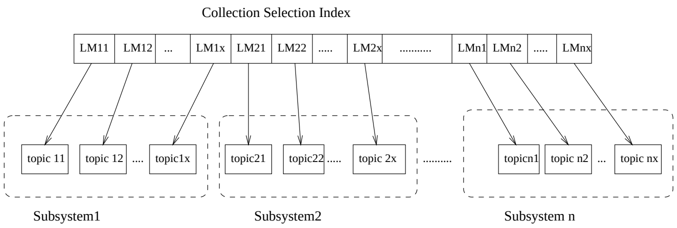

Figure 3: Distributed retrieval with local clustering

3.4 Multiple-topic Representation

Our third new method is multiple-topic representation. In addition to the constraints in local clustering, we assume that subsystems do not want to physically partition their documents into several collections. A possible reason is that a subsystem has already created a single index and wants to avoid the cost of re-indexing. However, each subsystem is willing to cluster its documents and summarize its collection as a number of topic models for effective collection selection. With this method, a collection corresponds to several topics. Collection selection is based on how well the best topic in a collection matches a query. This method is illustrated by Figure 4.

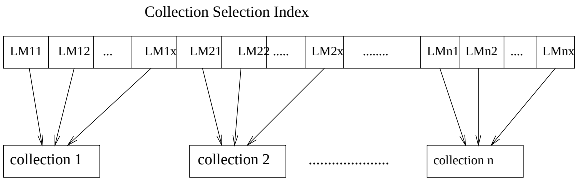

Figure 4: Distributed retrieval with multiple-topic representation

The advantage with this approach is that it assumes minimum co-operation from the subsystems. The disadvantage is that it is less effective than global clustering and local clustering. Experiments show, however, that it is still more effective than the baseline method which represents a heterogeneous collection as a whole.

4 Experimental Setup

Experiments were carried out on TREC3, TREC4 and TREC6. The TREC3 queries consist of words from the title, description and narrative fields, averaging 34.5 words per query. The TREC4 queries have 7.5 words per query. The TREC6 queries in this study only use the title words, averaging 2.6 words per query. The purpose of using 3 sets of queries of different lengths is to ensure the generality of our results. Table 1 shows the statistics of the test sets.

The two baselines used are centralized retrieval and the baseline distributed retrieval method, which creates collections based on document sources and represents a collection as a whole. The sets of collections used for baseline distributed retrieval are:

- TREC3-100col-bysource: 100 collections created according to the sources of the documents in TREC3. Each collection has roughly 7,418 documents. The number of collections for a source is proportional to the total number of documents in the source. For example, the source DOE has 226,087 documents and is split into 30 collections.
- TREC4-100col-bysource: 100 collections created similarly for TREC4. Each collection has roughly 5,675 documents.
- TREC6-100col-bysource: 100 collections created similarly for TREC6. Each collection has roughly 5,560 documents.

The sets of collections created by global clustering are: TREC3-100col-global, TREC4-100col-global, and TREC6-100col-global. Each set was created by running the K-Means algorithm on the corresponding document set and has 100 collections.

The set of collections created by local clustering is TREC4-100col-local. The K-Means algorithm was run on each of the six TREC4 sources separately. The total number of collections is 100. The number of collections for a source is proportional to the number of documents in the source.

The set of collections used in the multiple-topic representation experiments is TREC4-10col-bysource. The ten collections are AP88, AP90, FR88, U.S. Patent, SJM91, WSJ90, WSJ91, WSJ92, ZIFF91 and ZIFF92. These are the natural collections in TREC volumes 2 and 3. Each collection is represented as several topic models. The number of topic models for a collection is proportional to the number of documents in the collection. The total number of topic models is 100.

Collections are indexed and searched by the INQUERY retrieval system [1]. A problem encountered in the experiments is IDF. IDF is intended to give rare terms, which are usually important terms, more credit in retrieval. When collections are created by topics, however, IDF can achieve the opposite effect. Frequent terms in a topic are often important in distinguishing the topic from other topics. To avoid the problem, we modified INQUERY so that it uses global IDF in retrieval. Given a test set (e.g. TREC3), the global IDF of a term is calculated based on the total number of documents in the set that contain the term. By doing so we also avoided the tricky issue of result merging which is not the primary concern of this study. Global IDF was used in all experiments in this paper.

The steps to search a set of a distributed collections for a query are (1) rank the collections against the query, (2) retrieve 30 documents from each of the best n collections, (3) merge the retrieval results based on the document scores. Precision is calculated at document cut-offs 5, 10, 15, 20 and 30, for the reason that few people are interested in more than a small number of documents in a realistic environment.

5 Global Clustering

5.1 Results

Tables 2, 3 and 4 compare the baseline results and global clustering on TREC3, TREC4 and TREC6. Ten collections were searched per query in the experiments. Each collection was represented as a language model. The collection selection metric is the Kullback-Leibler divergence. The results show that the baseline distributed retrieval with heterogeneous collections is significantly worse than centralized retrieval, around 30% at all document cut-offs on all three test sets. But when collections are created based on topics, the performance of distributed retrieval is close to centralized retrieval on all three test sets. On TREC4, global clustering is even better than centralized retrieval at document cut-offs 5 and 10. The t-test [12] shows the improvement over

collection	query count	size (GB)	document count	words per query	words per document	rel docs per query
TREC3	50	2.2	741,856	34.5	260	196
TREC4	49	2.0	567,529	7.5	299	133
TREC6	50	2.2	556,077	2.6	308	92

Table 1: Test collections statistics. Stop words are not included.

centralized retrieval is statistically significant at document cutoff 10 (p_value=0.05).

	TREC3 centralized	TREC3 100col-bysource	TREC3 100col-global
5 docs:	0.6760	0.5000 (-26.0)	0.6680 (-1.2)
10 docs:	0.6080	0.4520 (-25.7)	0.6080 (+0.0)
15 docs:	0.5840	0.4067 (-30.4)	0.5707 (-2.3)
20 docs:	0.5490	0.3770 (-31.3)	0.5320 (-3.1)
30 docs:	0.5120	0.3240 (-36.7)	0.4920 (-3.9)

Table 2: TREC3: comparing centralized retrieval, baseline distributed retrieval and global clustering

	TREC4 centralized	TREC4 100col-bysource	TREC4 100col-global
5 docs:	0.5918	0.4245 (-28.3)	0.6204 (+4.8)
10 docs:	0.4918	0.3816 (-22.4)	0.5163 (+5.0)
15 docs:	0.4612	0.3469 (-24.8)	0.4639 (+0.6)
20 docs:	0.4337	0.3122 (-28.0)	0.4194 (-3.3)
30 docs:	0.3714	0.2769 (-25.4)	0.3707 (-0.2)

Table 3: TREC4: comparing centralized retrieval, baseline distributed retrieval and global clustering

	TREC6 centralized	TREC6 100col-bysource	TREC6 100col-global
5 docs:	0.4440	0.3360 (-24.3)	0.4520 (+1.8)
10 docs:	0.3920	0.2940 (-25.0)	0.3820 (-2.6)
15 docs:	0.3573	0.2560 (-28.4)	0.3533 (-1.1)
20 docs:	0.3330	0.2350 (-29.4)	0.3220 (-3.3)
30 docs:	0.2907	0.1973 (-32.1)	0.2780 (-4.4)

Table 4: TREC6: comparing centralized retrieval, baseline distributed retrieval and global clustering

5.2 Discussion

One reason that global clustering improves distributed retrieval is that it makes the distribution of relevant documents more concentrated. Figure 5 plots the distribution of relevant documents in TREC4-100col-global and in TREC4-100col-bysource. We ranked the collections according to how many relevant documents a collection has for a query. This is named the *optimal* ranking because it is the best a collection selection algorithm can attain. The X-axis shows the collection ranks and the Y-axis shows how many relevant documents a collection at a rank position has for a query on average. That is,

$$y(i) = (\sum_{Q_j} \text{number of relevant documents in } \; c_{ji} \; \text{ for } \; Q_j)/49$$

where c_{ji} is the ith ranked collection for query Q_j according the optimal ranking. We can see that relevant documents are significantly more densely distributed in TREC4-100col-global than in TREC4-100col-bysource. The TREC4 queries have 133 relevant documents per query on average. With TREC4-100col-global, the top 10 collections for a query have 119 relevant documents on average. By comparison, the number is only 65 with TREC4-100col-bysource. The dense distribution of relevant documents increases the potential for effective distributed retrieval. The only remaining question is whether we can select the right collections.

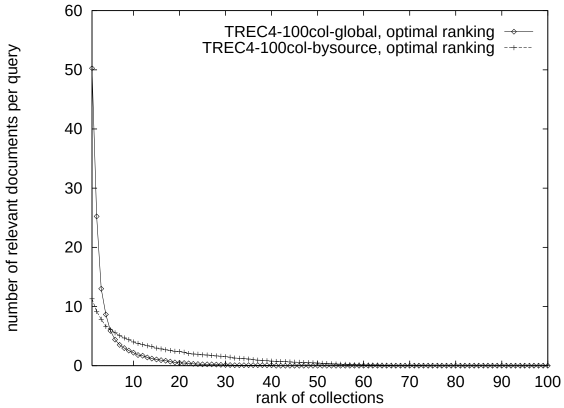

Figure 5: Comparing the distributions of relevant documents in TREC4-100col-global and TREC4-100col-bysource

Another reason is that collection selection is more accurate with topic-based collections than with heterogeneous collections. Figures 6 and 7 show how well collection selection works with topic-based collections (TREC4-100col-global) and heterogeneous collections (TREC4-100col-bysource). Collections were ranked in two ways, by optimal ranking (as done in Figure 5) and by Kullback-Leibler divergence. For TREC4-100col-global (Figure 6), Kullback-Leibler closely fits the optimal curve. When 10 collections are selected for each query by each method, the optimal ranking finds 119 relevant documents per query and Kullback-Leibler finds 90. This represents a 76% (90/119) accuracy. This shows that with topic-based collections, collection selection is very accurate. For TREC4-100col-bysource (Figure 7), Kullback-Leibler does not fit the optimal curve as well. The accuracy is only 54% (35 Kullback-Leibler /65 optimal).

Table 3 shows that global clustering is more effective than centralized retrieval when 10 documents are retrieved

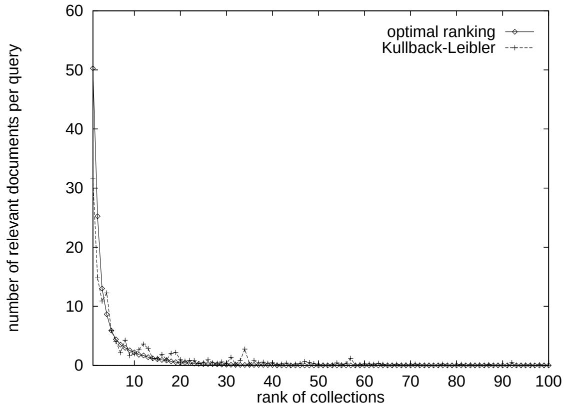

Figure 6: TREC4-100col-global: optimal collection ranking vs ranking by Kullback-Leibler

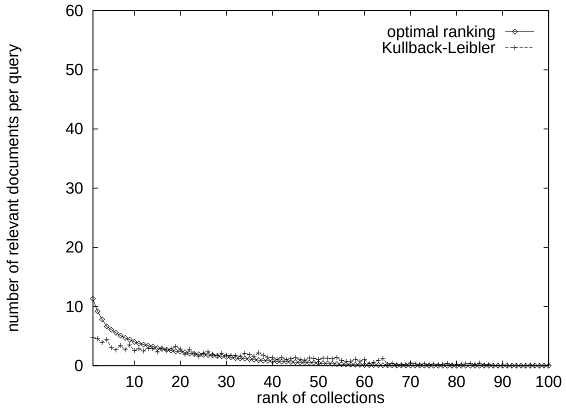

Figure 7: TREC4-100col-bysource: optimal collection ranking vs ranking by Kullback-Leibler

per query on TREC4. The reason for the improvement is that the small number of collections we selected contain most of the relevant documents. Therefore we are able to exclude many non-relevant documents from the top ranked set without removing many relevant documents. If we divide the 490 documents (10 documents per query * 49 queries) retrieved by centralized retrieval in two categories, those that were included and those that were excluded by distributed retrieval, 56% (217 relevant/387 total) of the included are relevant while only 23% (24/103) of the excluded are relevant.

The collections in TREC4-100col-bysource have roughly the same number of documents. The collections in TREC4-100col-global, however, have very different numbers of documents, ranging from 301 to 82,727. One might have the concern that our collection selection method may simply choose the largest collections. To make sure our technique is immune to this problem, we calculated the average number of documents per collection for the collections we searched (10 collections per query). The number is 5,300, which is even slightly smaller than the average (5,675) for the whole set TREC4-100col-global.

In a dynamic environment where new documents arrive regularly, one pass clustering is more appropriate. We found that when one pass K-Means clustering was used, the retrieval performance is only 3% worse than two pass clustering on TREC4. It means that our technique would also work well in a dynamic environment.

5.3 Collection Selection Metrics

The INQUERY retrieval function is very effective for document retrieval, as shown by past TREC results [9]. It is, however, less effective than Kullback-Leibler for collection selection. Table 5 compares the retrieval performance on TREC4-100col-global when the INQUERY retrieval function instead of Kullback-Leibler was used for collection selection. (The INQUERY-style collection selection is described in [2, 23].) Ten collections were selected per query by each metric. Using INQUERY for collection selection resulted in a drop in precision at all cutoff levels, with an average drop of 7.6%. The collections selected by INQUERY have an average of 82 relevant documents per query while the ones selected by Kullback-Leibler have 90 per query. The results show that collection selection is different from document retrieval. A good metric for one problem is not necessarily good for the other.

	Kullback Leibler	INQUERY	
5 docs:	0.6204	0.5510	(-11.2)
10 docs:	0.5163	0.4633	(-10.3)
15 docs:	0.4639	0.4272	(- 7.9)
20 docs:	0.4194	0.3980	(- 5.1)
30 docs:	0.3707	0.3565	(- 3.8)

Table 5: Comparing Kullback-Leibler and INQUERY for collection selection on TREC4-100col-global

6 Local Clustering

In environments where subsystems are autonomous, local clustering is appropriate. Since the number of clusters created is small for each subsystem, the method also scales well. Table 6 shows that the retrieval performance of local clustering is only slightly worse than centralized retrieval. Ten collections were searched for each query. Performance at document cutoff 5 is even somewhat (2.1%) better than centralized retrieval. Local clustering is substantially better than the baseline distributed retrieval method. The results demonstrate that some extra work from participating subsystems can significantly improve the performance of a distributed retrieval system. Compared to global clustering (Table 3), local clustering is slightly worse in performance. In our opinion, local clustering is a reasonable tradeoff between retrieval effectiveness and implementation complexity.

7 Multiple-Topic Representation

Table 7 shows the performance of distributed retrieval with multiple-topic representation. The set of collections used is TREC4-10col-bysource, which has 10 collections. The baseline distributed retrieval method represents a collection as one topic model while the new method represents a collection as several topic models, as discussed in section 3. The average number of topic models is 10 per collection with the new method. The rank of a collection was determined by the best topic model of that collection for a query. Two

	TREC4 centralized	TREC4 100col-bysource	TREC4 100col-local
5 docs:	0.5918	0.4245 (-28.3)	0.6041 (+2.1)
10 docs:	0.4918	0.3816 (-22.4)	0.4857 (-1.2)
15 docs:	0.4612	0.3469 (-24.8)	0.4381 (-5.0)
20 docs:	0.4337	0.3122 (-28.0)	0.4020 (-7.3)
30 docs:	0.3714	0.2769 (-25.4)	0.3476 (-6.4)

Table 6: TREC4: comparing centralized retrieval, baseline distributed retrieval and local clustering

collections were searched per query. The retrieval performance of multiple-topic representation is noticeably better than the baseline distributed retrieval at all cutoffs. The improvement is statistically significant at all cutoffs (t-test, p_value < 0.01). The improvement at document cutoff 5 is a substantial 18%. The results show that representing a heterogeneous collection as a number of topics can significantly improve collection selection.

One problem with multiple-topic representation is that relevant documents are still sparsely distributed. Even though we are able to rank the collections more accurately, searching a few collections miss many relevant documents. Therefore the retrieval performance is significantly worse than centralized retrieval and the other two techniques. In fact, even with perfect collection selection based on the number of relevant documents in a collection, the performance is still worse than centralized retrieval (Table 7).

8 Efficiency

It took about 6 hours on an Alpha workstation to run the two pass K-Means algorithm on TREC4 when 100 topics were created. The speed is acceptable for 2 GB collections. Memory usage was around 100 MB and can be reduced with more careful implementation. When local clustering was performed on the six document sources of TREC4 individually, the time to create 100 topics was 2 hours.

For very large collections (e.g. 100 GB), we probably need to create significantly more topics. Global clustering would be too slow in such cases. One solution is to use faster clustering algorithms. The other solution is to partition large collections into chunks of appropriate size (e.g. 2 GB per chunk) and cluster each chunk separately. The results of local clustering suggests that the second solution shall be close to global clustering in performance.

9 Related Work

Distributed retrieval has been studied under a variety of names, including server selection [10], text database resource discovery [8] and collection selection [2]. A popular technique for representing collections is to use word histograms [8, 2, 23]. Other techniques include manually describing the content of a collection [14] and knowledge-based techniques [3].

Kosmynin proposed an approach to distributed retrieval based on shared interests among users [15]. Danzig proposed an approach to organize a document space around retrieval results for a set of common queries [5]. Dolin proposed an approach to server selection based on term classification to address the vocabulary mismatch between user requests and actual documents [6]. Weiss *et al* proposed an approach to organizing web resources based on content and link clustering [22]. Hawking *et al* proposed a distributed retrieval technique based on lightweight probe queries [10]. Xu and Callan demonstrated that properly expanded queries can improve collection selection [23]. French *et al* discussed issues in evaluating collection selection techniques [7].

Document clustering has been extensively studied in IR as an alternative to ranking-based retrieval and as a tool for browsing [21]. Recent trends in document clustering include faster algorithms [4, 20] and clustering query results [11]. The K-Means algorithm was described in [13].

Ponte showed that language modeling approach to retrieval can produce very effective retrieval results [19]. His results were corroborated by groups using similar techniques at TREC7 [9]. Language modeling was used by Yamron *et al* for text segmentation [24]. Topic models in this study are similar to the ones in that work.

10 Conclusion and Future Work

This paper proposed a new approach to distributed retrieval based on document clustering and language modeling. Under the general approach, we proposed three methods of organizing a distributed retrieval system. All three methods can improve the results of distributed retrieval.

One area for future work is to determine how many topics are appropriate for a collection. Related to this issue is to determine when to create a new topic in a dynamic application. The second area is to explore techniques to improve the quality of clustering. One approach under investigation is to characterize the natural structure of a collection by using the average link algorithm on a random sample of the collection. The resulting clusters are used as seeds when we run the K-Means algorithm on the whole collection. The third area is to explore the usability of faster clustering algorithms.

Acknowledgements This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623. It is also supported in part by Defense Advanced Research Projects Agency/ITO under ARPA order number D468, issued by ESC/AXS contract number F19628-95-C-0235. Any opinions, findings and conclusions or recommendations expressed in this material are the authors and do not necessarily reflect those of the sponsors.

References

[1] J. Broglio, J. P. Callan, and W.B. Croft. An overview of the INQUERY system as used for the TIPSTER project. In *Proceedings of the TIPSTER Workshop*. Morgan Kaufmann, 1994.

[2] J. P. Callan, Z. Lu, and W.B. Croft. Searching distributed collections with inference networks. In *Proceedings of 18th ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 21–28, 1995.

[3] A. Chakravarthy and K. Hasse. NetSerf: Using semantic knowledge to find Internet information archives. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, 1995.

	TREC4 centralized	TREC4 10col-bysource baseline		TREC4 10col-bysource multiple-topic		TREC4 10col-bysource perfect	
5 docs:	0.5918	0.4163	(-29.7)	0.4898	(-17.2)	0.5469	(- 7.6)
10 docs:	0.4918	0.3673	(-25.3)	0.3980	(-19.1)	0.4673	(- 5.0)
15 docs:	0.4612	0.3347	(-27.4)	0.3646	(-20.9)	0.4463	(- 3.2)
20 docs:	0.4337	0.2969	(-31.5)	0.3255	(-24.9)	0.4041	(- 6.8)
30 docs:	0.3714	0.2537	(-31.7)	0.2850	(-23.3)	0.3503	(- 5.7)

Table 7: TREC4: comparing centralized retrieval, baseline distributed retrieval, multiple-topic representation and perfect collection selection

[4] D. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/gather: a cluster-based approach to broswing large document collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1992.

[5] P. Danzig, J. Ahn, J. Noll, and K. Obraczka. Distributed indexing: A scalable mechanism for distributed information retrieval. In *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 220–229, 1991.

[6] R. Dolin, D. Agrawal, L. Dillon, and A. El Abbadi. Pharos: a scalable distributed architecture for locating heterogeneous information sources. Technical Report TRCS96-05, Computer Science Department, University of California, Santa Barbara, 1996.

[7] J. French, A. Powell, C. Viles, T. Emmitt, and K. Prey. Evaluating database selection techniques: A testbed and experiment. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 121–129, 1998.

[8] 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.

[9] D. Harman, editor. *TREC7 Proceedings*. NIST, To Appear.

[10] D. Hawking and P. Thistlewaite. Methods for information server selection. *ACM Transactions on Office Information Systems*, 17(1):40–76, January 1999.

[11] M. Hearst and J. O. Pedersen. Reeaxming the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 76–84, 1996.

[12] D. Hull. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 329–338, 1993.

[13] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[14] B. Kahle and A. Medlar. An information system for corporate users: Wide Area Information Servers. Technical Report TMC199, Thinking Machines Corporation, 1991.

[15] A. Kosmynin. From bookmark managers to distributed indexing: an evolutionary way to the next generation of search engines. *IEEE Communications Magzine*, 35(6):146–151, June 1997.

[16] S. Kullback, J.C. Keegel, and J.H. Kullback. *Topics In Statistical Information Theory*. Springer-Verlag, 1987.

[17] L. Larkey. Some issues in the automatic classification of U.S. patents. In *Learning for Text Categorization. Papers from the 1998 Workshop. AAAI Press*, pages 87–90, 1998.

[18] MetaCrawler. http://www.metacrawler.com.

[19] J. Ponte. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, 1998.

[20] C. Silverstein and J. O. Pedersen. Almost constant-time clustering of arbitrary corpus subsets. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 60–66, 1997.

[21] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, second edition, 1979.

[22] R. Weiss, B. Velez, M. Sheldon, C. Namprempre, P. Szilagyi, A. Duda, and D. Gifford. Hypursuit: A hiearchical network search engine that exploits content-link hypertext clustering. In *Proceedings of the 7th ACM Conference on Hypertext*, 1996.

[23] J. Xu and J.P. Callan. Effective retrieval with distributed collections. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 112–120, 1998.

[24] J. Yamron. Topic detection and tracking segmentation task. In *Proceedings of the Topic Detection and Tracking Workshop*, 1997.