A Language Modeling Framework for Resource Selection and Results Merging

Luo Si
School of Computer Science
Carnegie Mellon University
(+1)412-268-3951
lsi@cs.cmu.edu

Rong Jin
School of Computer Science
Carnegie Mellon University
(+1)412-268-4050
rong+@cs.cmu.edu

Jamie Callan
School of Computer Science
Carnegie Mellon University
(+1)412-268-4525
callan+@cs.cmu.edu

Paul Ogilvie
School of Computer Science
Carnegie Mellon University
(+1)412-268-5606
pto+@cs.cmu.edu

ABSTRACT

Statistical language models have been proposed recently for several information retrieval tasks, including the resource selection task in distributed information retrieval. This paper extends the language modeling approach to integrate resource selection, ad-hoc searching, and merging of results from different text databases into a single probabilistic retrieval model. This new approach is designed primarily for Intranet environments, where it is reasonable to assume that resource providers are relatively homogeneous and can adopt the same kind of search engine. Experiments demonstrate that this new, integrated approach is at least as effective as the prior state-of-the-art in distributed IR.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - *retrieval models.* H.3.4 [**Information Storage and Retrieval**]: Systems and Software – *distributed systems*

General Terms

Algorithms, Experimentation

Keywords

Language model, Distributed information retrieval.

1. INTRODUCTION

Language Modeling has a long history of being used successfully in the fields of speech recognition and statistical natural language processing. It has been applied to information retrieval and studies [2,3] have shown its effectiveness in the ad-hoc information retrieval task. Some work has been done to apply language-modeling techniques to do resource selection in distributed retrieval task. However, little work has been done in distributed IR using a single integrated language model framework. The difference between ad-hoc information retrieval and the distributed information retrieval is that ad-hoc information retrieval assumes that all the documents can be copied into a single centralized database for the purpose of indexing and searching while distributed information retrieval targets the cases when documents can't be obtained or stored in a single database. The task of distributed information retrieval can be important when the information is proprietary or access of the information is not free. With the proliferation of online searchable databases on local area networks and the Internet, the significance of the distributed information retrieval is becoming more and more serious [1].

There are various scenarios in distributed information retrieval [15]. In this paper, we focus on the intranet environment, where we can assume each individual database uses the same kind of search engine.

There are three important sub-problems in distributed information retrieval: first, the content of each text database must be represented in a suitable form; second, given an information need (a query), several relevant databases must be selected to do the search; third, the results from all the selected databases have to be merged into a single final list [1]. A lot of research has been done in these three sub-fields. CORI algorithm [1,4] is one of the well-known examples. It assumes that each database uses Inquery search engine. It uses query-based sampling [5] for describing the content of each individual database, the CORI collection selection algorithm for choosing databases most relevant to the user's query and the CORI merging algorithm for fusing results from different databases together. Experiments showed that the CORI distributed information retrieval algorithm has achieved decent performance in many different environments [18].

In this paper, we will present an integrated language modeling approach to the distributed information retrieval problem. In this framework, language modeling is applied to every aspect of the distributed information retrieval problem. When a query is issued, a language model based collection selection algorithm is used for choosing a subset of databases that are most likely to provide documents relevant to the query. Then, within the selected database, a language model based retrieval algorithm is used for finding relevant documents. Finally, with the returned document information from the individual databases, a language model based merging approach is performed to integrate the results.

As for the three steps mentioned above, the language model based retrieval algorithm for a single database has been extensively studied and therefore is not the focus of this paper. The major contribution of this paper is on how to accomplish the phase of resource selection and the phase of results merging using language modeling framework. Compared with the CORI algorithm, this framework tends to be better justified by probability theory. On the standard TREC123 and TREC4 datasets, experiments have shown that the new framework significantly outperforms the well-known CORI distributed information retrieval algorithm when the scores from the individual databases are normalized without cooperation from individual databases and is slightly better or at the same level than the CORI algorithm in case that the operation of score normalization is performed with cooperation [1].

2. PREVIOUS WORK

This work is closely related to the language model approaches for ad-hoc information retrieval, including the risk model proposed by Ponte and Croft [3], a simple unigram model by Song and Croft [2] and the two states hidden Markov model by the BBN group [19]. Although the details are different between these approaches, the basic idea is the same. Each document is seen as a sample generated from a special language. Therefore, a language model for each document is estimated beforehand. Then, the relevance for a document to an information need (represented as a query) is computed as how likely the query can be generated from the language model for that document. More specifically, the likelihood for a query Q to be generated from a document D is computed as:

$$P(Q \mid D) = \prod_{q \in Q} (\lambda P(q \mid D) + (1-\lambda) P(q \mid C)) \tag{1}$$

where q is a query item in the query Q, $P(q|D)$ is the probability for the query term q to appear in the document D, $P(q|C)$ is the probability for the term q to be used in the collection C to which the document D belongs, and λ is a weighting parameter between 0 and 1. As seen from Equation (1), the role of term $P(q|C)$ is to smooth the probability for the document D to generate the query term q, particularly when $P(q|D)$ is zero. Furthermore, as pointed out in [20], the idea of smoothing document-based language model with the collection-based language model is similar to the tf.idf term weight scheme [21] used in vector model where 'common' words are discouraged by giving low weights and 'rare' words are emphasized with the high weights.

Unlike the ad-hoc information retrieval task, the distributed information retrieval task assumes that each individual database performs retrieval without the knowledge of other databases. Simply applying Equation (1) to the case of distributed information retrieval will be not feasible because each database has a different word statistics $P(q|C)$. therefore, each database has a different sense of common words and rare words which makes the scores from different databases not directly comparable. One important aspect of applying language modeling to distributed information retrieval is to wipe out the score differences caused by the differing statistics of databases and make the scores comparable. The basic idea is to take advantage of the document sets sampled from the individual databases. Assuming the sampled document sets share the similar word statistics with the original databases, we can gather the approximated word statistics for each individual database and use it as the way to adjust the document scores from the individual databases.

To accomplish this, we need to acquire the resource description for each database. Approaches such as the STARTS protocol and query-based sampling have been used to obtain the resource description for databases [5,6]. The difference between them is that the STARTS protocol requires each database to provide word statistics information directly while query-based sampling only asks the individual search engines to run queries and return a list of documents that are downloadable. In practice, the approach of query-based sampling has been shown to acquire rather accurate resource descriptions using a relatively small number of randomly generated queries (e.g. 75) to retrieve a relatively small number of documents (e.g. 300).

For resource selection, we need to find the subset of the databases that are most relevant to the user's query. There are many successful resource selection algorithms. Among them, gGlOSS [7,8], CORI, and CVV [9] are three best-known resource-ranking algorithms. Gravano et al. proposed GlOSS, the Glossary-of-Servers Server, as an approach to the resource selection for the Boolean IR model and it is generalized as gGlOSS to be used for any IR model. It needs each database to provide the document frequency for each word in its database, and the sum of the term weights in each document of the database. The CVV resource selection algorithm uses a combination of document frequency and cue validity variance information. In this algorithm, document frequency information is used to estimate the importance of a term within a database; the CVV component estimates whether a term is useful for differentiating one database from another. The CORI collection selection algorithm creates a resource selection index in which each database is represented by its terms and their document frequencies. Databases are ranked by the belief of $P(Q|C_k)$ which is determined by the sum of the beliefs of all query items generated by the corresponding database [1]. Previous research has shown that the CORI algorithm is the most stable and effective of the three algorithms [10,11]. But the ranking value, which is the belief, does not have a valid probabilistic explanation. It is hard to incorporate it into a probabilistic distributed information retrieval task.

The problem of merging results with incomparable scores from individual data collections has been studied extensively in the field of distributed information retrieval. Some methods take the approach of normalizing document scores, which needs individual databases to provide their corpus statistics or has to download the documents and recalculate the scores at the client side, which has very high communication and computation costs [10]. Other methods try to avoid either the requirement of the cooperation from individual databases or the cost of downloading the documents and recalculating the scores. For example, simply interleaving the retrieved documents from different databases in the order of their ranks has been used in [12]. CORI merging method [1,4] is based on a linear combination of the score of the database and the score of the document. The "normalized" score suitable for merging is calculated as shown below.

$$C_i' = \frac{(R_i - R_{\min})}{(R_{\max} - R_{\min})} \tag{2}$$

$$D' = \frac{(D - D_{\min})}{(D_{\max} - D_{\min})} \tag{3}$$

$$D'' = \frac{D' + 0.4 * D' * C_i'}{1.4} \tag{4}$$

$R_{\max}$ and $R_{\min}$ are two normalizing parameters that can be calculated by the CORI's resource selection algorithm only from the information in the resource selection index [1]. So equation (2) is the normalized database weighting score. Equation (3) needs the individual search engines to cooperate by providing

$D_{\max}$ and $D_{\min}$. In the absence of cooperation, $D_{\max}$ is set to the maximum document score returned by the search engine and $D_{\min}$ is set to the minimum.

To test the effectiveness of our language model for distributed information retrieval, we used the standard CORI distributed information retrieval system as the baseline system, which incorporate the algorithm of query-based sampling, the CORI resource selection algorithm and the CORI merging method. Previous studies have shown that the CORI system achieved generally good performance on different kinds of testbeds.

3. THE LANGUAGE MODEL APPROACH

In the section, we will give the full description of our language model approach. As already mentioned in the introduction section, the problem of distributed information retrieval comprises of three components, namely a component for acquiring the resource description, a component for selecting most relevant databases and a component for merging results returned from different databases. The following three subsections will focus on each of these three components. Besides these three sub-problems we have made the assumption that all individual databases use a language model search engine, which is a valid assumption for the intranet environment of distributed information retrieval.

3.1 Acquiring Resource Descriptions

As already discussed in the section of related work, one difficulty imposed by the distributed information retrieval problem is that the document scores returned from different database may not be comparable since each database has a different word statistics. One solution to this problem is to acquire an approximate word frequency distribution and adjust the document scores based on the obtained word distribution. Meanwhile, the resource description is critical to the phase of database selection where a subset of databases is chosen based on how similar the resource description is to the query.

In this framework, we adopt the technique of query-based sampling. All the queries used in the query-based sampling were one-term queries. The initial query term was selected randomly from a background language model. Then, the subsequent queries terms were selected randomly by the language model, which was learned from the documents already retrieved from the database by previous queries. The top four documents retrieved by each query were examined to update the resource description. Duplicate documents, which were already retrieved by previous queries, were simply discarded. This results in some queries retrieving less than four documents. Detailed experiments in [1,5] have shown that this method can get an adequate description with only a small amount of queries and a relatively small number of documents from each database.

After three hundred documents were retrieved from each database, a collection based language model $P(q|C)$ will be built for each database C and the retrieved documents of all databases were collapsed together to build a global language model $P(q|G)$. By looking at the difference between the collection based word distribution $P(q|C)$ and the global word distribution $P(q|G)$, we can tell which collection may overestimate the document-query similarity and which collection does the opposite and make the corresponding adjustments. Furthermore, by comparing the collection based word distribution to the query, we are able to tell which collection is more likely to provide relevant documents and which collection will not.

3.2 Resource Selection

In the resource selection phase, we need to select most relevant collections to a query based on the information we got in the resource description phase. Therefore, the key issue here is how to compute the collection-query similarity. To take advantage of the language model used for computing the document-query similarity, we can simply collapse the sampled documents for a database together as one single giant 'document' and perform the similar computation for the document-query similarity. More formally, we need to find the collections that have largest probabilities of $P(Q|C)$, i.e. the probability of generating the query Q from the text collection C. Following the principle of language model, the value of $P(Q|C)$ is calculated in the following way:

$$P(Q \mid C) = \prod_{q \in Q} \left(\lambda P(q \mid C) + (1-\lambda) P(q \mid G) \right) \tag{5}$$

where $P(Q|C)$ is the language models for the collection C and $P(Q|G)$ is language model for the whole collection. Linear interpolation constant λ smoothes the collection-based language model with the global language model. Collections with the largest generation probabilities $P(Q|C)$ will be selected as the most relevant collections.

Notice, our collection selection method is very similar to the Kullback-Leibler (KL) divergence based collection selection method used by Xu and Croft [14]. In their work, the Kullback-Leibler divergence between the word frequency distribution of the query and the database is used to measure how well the content of the database matches with the query. More specifically, the KL divergence between query Q and collection C is computed as:

$$KL(Q,C) = \sum_{q \in Q} P(q \mid Q) \log \left\{ \frac{P(q \mid Q)}{\lambda P(q_i \mid C) + (1-\lambda) P(q_i \mid G)} \right\} \tag{6}$$

Note that collections are ranked using the negative of the KL divergence. Both our method and KL divergence method are using the word distribution as the basis of the similarity measurement. It is not difficult to show that these two methods are actually equivalent by simply taking the logarithm of Equation (5) and noticing that the term $\sum_{q \in Q} P(q \mid Q) \log(q \mid Q)$ in Equation (6) is a query-specific constant.

3.3 Results Merging

The last step of distributed information retrieval is merging the results. It is a difficult task because different databases may use different ranking algorithms and therefore the scores returned from the individual databases may not be comparable. Furthermore, even all the individual databases can be assumed to use the same ranking algorithm; the large variance in corpus statistics for different databases can still make the scores incomparable [15]. In our framework, we rule out the first factor by requiring all the individual databases use the language model based algorithm for retrieving relevant documents. Then, the second factor, i.e. the heterogeneous corpus statistics varying widely from one database to another, becomes the major concern of our results merging algorithm.

With the assumption that all the search engines are using the same language modeling based retrieval algorithm, the document scores returned from the individual search engines can be interpreted as generation probabilities for a given query. More specifically, for a given query Q, the score of the j^{th} document D_{ij} returned by the i^{th} database C_i, should be the probability $P(Q| D_{ij}, C_i)$, i.e. the probability of generating query Q given the document D_{ij} and the database C_i. According to Equation (1), the probability $P(Q| D_{ij}, C_i)$ can be expressed as:

$$P(Q \mid D_{ij}, C_i) = \prod_{q \in Q} \left(\lambda P(q \mid D_{ij}) + (1-\lambda) P(q \mid C_i) \right) \tag{1'}$$

As it can be seen from the above equation, the probabilities $P(Q|D,C)$ returned from individual search engines can be significantly influenced by the heterogeneous corpus statistics $P(q|C)$ for different databases. The task of the merging algorithm is to take these probabilities for documents as inputs and together with all the information of individual databases obtained in the resource description and resource collection phases to effectively erase the bias caused by the corpus statistics and calculate the final comparable scores for all the returned documents.

Since the 'unfairness' within the document scores comes from the factor of database C_i in probability $P(Q|D_{ij}, C_i)$, we would like to compute the 'fair' scores for documents as $P(Q|D_{ij})$ which don't have dependency on database C_i. In order to compute the 'fair' document score $P(Q|D_{ij})$ based on the probability $P(Q|D_{ij}, C_i)$ and the word distribution of individual databases obtained in the phase of acquiring resource description, we need to decompose the probability $P(Q|D_{ij}, C_i)$ into two parts, with one part dependent only on the document D_{ij} and the other part influenced only by the database C_i. A simple solution would be to rewrite the probability $P(Q|D_{ij}, C_i)$ as a linear interpolation of probability $P(Q|C_i)$ and $P(Q|D_{ij})$ as:

$$P(Q \mid C_i, D_{ij}) = \alpha P(Q \mid C_i) + (1-\alpha) P(Q \mid D_{ij}) \tag{7}$$

With the known information of $P(Q|D_{ij}, C_i)$ and $P(Q|C_i)$ (computed in the phase of collection selection), $P(Q||D_{ij})$ can be computed by simply subtracting $P(Q|C_i)$ from $P(Q|D_{ij}, C_i)$. Unfortunately, this simple approach is not feasible because the values for probability $P(Q|C_i)$ and $P(Q|D_{ij})$ are not comparable. A collection usually contains many more words than a document and therefore the word generation probability $P(q|C_i)$ is usually much lower than $P(q|D_{ij})$, which usually results in a significantly smaller value for $P(Q|C_i)$ than $P(Q|D_{ij})$.

A better solution would be to decompose $P(Q|D_{ij}, C_i)$ into some kind of product. Therefore, according to the Bayesian rule, we have $P(Q|D_{ij}, C_i)$ expanded as

$$P(Q \mid D_{ij}, C_i) = \frac{P(Q \mid D_{ij}) P(C_i \mid Q, D_{ij})}{P(C_i \mid D_{ij})} \tag{8}$$

By assuming that the two evidences in the probability of $P(C_i|Q,D_{ij})$ can be linearly separable, we will have

$$P(C_i \mid Q, D_{ij}) = \alpha P(C_i \mid Q) + (1-\alpha) P(C_i \mid D_{ij}) \tag{9}$$

where α is the parameter that represent the relative importance of two evidence of Q and D_{ij}. Substituting Equation (9) for $P(C_i|Q,D_{ij})$ in Equation (8) and rewriting, we have the expression for $P(Q|D_{ij},C_i)$ as

$$\begin{aligned} &P(Q \mid D_{ij}, C_i) \\ &= (1-\alpha) P(Q \mid D_{ij}) \left(\frac{\alpha P(C_i \mid Q)}{(1-\alpha) P(C_i \mid D_{ij})} + 1 \right) \end{aligned} \tag{10}$$

By taking the logarithm on both sides of the above formula, we will have:

$$\begin{aligned} \log P(Q \mid D_{ij}, C_i) = \log(1-\alpha) + \log P(Q \mid D_{ij}) \\ + \log\left(\frac{\alpha P(C_i \mid Q)}{(1-\alpha) P(C_i \mid D_{ij})} + 1 \right) \end{aligned} \tag{11}$$

Then, the logarithm of probability $P(Q|D_{ij})$, i.e. the final 'fair' document score , can be expressed as:

$$\begin{aligned} \log P(Q \mid D_{ij}) = \log P(Q \mid D_{ij}, C_i) - \log(1-\alpha) \\ - \log\left(\frac{\alpha P(C_i \mid Q)}{(1-\alpha) P(C_i \mid D_{ij})} + 1 \right) \end{aligned} \tag{12}$$

Since α is a constant, term $\log(1-\alpha)$ is a constant contribution to every $\log(P(Q|D_{ij}))$ and therefore can be ignored. Furthermore, since the contents of different databases are already predefined, by assuming the assignment of document D_{ij} to collection C_i is usually correct, we can have $P(C_i|D_{ij})$ approximated as a constant p which is close to 1. With these two considerations, the final expression for the logarithm of probability $P(Q|D_{ij})$ as

$$\begin{aligned} \log P(Q \mid D_{ij}) \propto \ \log P(Q \mid D_{ij}, C_i) \\ - \log(\beta P(C_i \mid Q) + 1) \end{aligned} \tag{13}$$

where β is a constant and is defined as $\frac{\alpha}{(1-\alpha)}$. This is the core formula for our results merging algorithm. As easily seen from Equation (13), the influence of corpus statistics on the document score is represented in the term $P(C_i|Q)$. By subtracting $P(C_i|Q)$ from the original score $P(Q|D_{ij}, C_i)$ returned from the individual databases, we are able to eliminate the factor of different corpus statistics and therefore results in a more 'fair' scores for documents. For term $P(C_i|Q)$, since we have already computed $P(Q|C_i)$ in the phase of resource selection, with the help of Bayesian rule, we can have the following expression for $P(C_i|Q)$

$$P(C_i \mid Q) = \frac{P(Q \mid C_i) P(C_i)}{\sum_i P(Q \mid C_i) P(C_i)} \tag{14}$$

4. EXPERIMENTAL METHODOLOGY

In this section, we describe the experiment design. The detailed description of the testbeds used for the experiment is presented in Section 4.1 and the presentation of experiments setting is given in Section 4.2.

Name	Query Count	Size (GB)	Number of documents			Megabytes (MB)		
			Min	Avg	Max	Min	Avg	Max
Trec123	50	3.2	752	10782	39713	28.1	32	41.8
Trec4_kmeans	50	2.0	301	5675	82727	3.9	20	248.6

Table1: Testbed statistics

4.1 Testbeds

Testbeds play a very important role in distributed information retrieval experiments since the performance of distributed information retrieval systems is highly influenced by the testbed characteristics. Two different testbeds were used in our experiments. The goal was to test the effectiveness of our algorithm in different degrees of heterogeneity and different types of queries.

"By source" testbed: trec123
In this testbed, there were altogether 100 databases created from TREC CDs 1,2 and 3. The databases were organized by source and publication date [1,5], and are somewhat heterogeneous. 50 short queries were created from the title fields of TREC topics 51-100.

"By subject" testbed: trec4_kmeans
For this testbed, there were altogether 100 databases created from TREC 4 data. A k-means clustering algorithm was used to cluster the databases by topic automatically [22], so the databases are homogenous and the word distributions are very skewed. 50 longer queries were created from the description fields of TREC topics 201-250.

The characteristics of these two testbeds are shown in Table 1. Meanwhile, the characteristics of their corresponding queries are shown in Table 2.

Name	TREC Topic Set	TREC Topic Field	Average Length (Words)
Trec123	51-100	Title	3
Trec4_kmeans	201-250	Description	7.2

Table2: Query set statistics

4.2 Experiment Settings

To test the effectiveness of our language model approach, we compared our algorithm to the CORI algorithm. To make the comparison fair, we try to make the setup for both algorithms as close as possible.

In order to reduce the cooperation needed from each individual search engines, query-based sampling technique was used to acquire the database description for both our algorithm and CORI algorithm. 300 documents from each of the 100 databases of both testbeds were sampled. All the documents from the same database were merged together to make the description of the corresponding database. For the part of resource selection, the top 10 databases were selected for each query, which has been a common choice in many previous researches [1,14,15]. At most 100 documents were retrieved from each database. The documents returned by the selected databases were merged into the final result lists. This results list is fed into an evaluation program for computing precision.

The language modeling based search engine for each database is implemented using the Lemur toolkit [17]. All the weighting parameters λ used in Equation (1) are set to 0.5. The parameter β in Equation (13) is empirically set to be 19 for both testbeds.

Searching only a fraction of the databases will surely cause some relevant documents to be missed. Precision at high recall will suffer. But in most environments, users are only concerned with the top several dozen documents. Therefore, precision at top 5, 10, 15, 20 and 30 are the most important evaluation measures in distributed information retrieval [1,14,15], which is the measure used in our experiments.

5. EXPERIMENTAL RESULTS – COMPARISON WITH CORI

CORI algorithm is a well-known framework that has been shown to work very effectively in different environments. In the section for related work, we have already briefly covered the three important components for CORI algorithm, i.e. the component for acquiring resource description, the component for resource selection and the component for the results merging. More detailed descriptions can be found in [1,4]. Experiments in this

Precision at Doc Rank	CORI without Cooperation	CORI with Cooperation	LM for Distributed IR
5 docs	0.4280	0.4480 (+ 4.67%)	0.4680 (+ 9.35%) (+ 4.46%)
10 docs	0.3840	0.4220 (+ 9.90%)	0.4420 (+15.10%) (+ 4.74%)
15 docs	0.3933	0.4053 (+ 3.50%)	0.4413 (+12.20%) (+ 8.88%)
20 docs	0.3720	0.3820 (+ 2.69%)	0.4240 (+13.98%) (+10.99%)
30 docs	0.3593	0.3627 (+ 0.95%)	0.3940 (+ 9.66%) (+ 8.63%)

Table 3. The distributed retrieval results for trec123 testbed by CORI algorithm, scores normalized without cooperation from individual databases, scores normalized without cooperation from individual databases and Language Model for distributed information retrieval. (The first baseline in CORI with cooperation and Language Model for distributed information retrieval is the performance of CORI algorithm without cooperation; the second baseline in Language Model for distributed information retrieval is the performance of CORI algorithm with cooperation)

Precision at Doc Rank	CORI without Cooperation	CORI with Cooperation	LM for Distributed IR
5 docs	0.3600	0.4240 (+17.78%)	0.4320 (+20.00%) (+ 1.89%)
10 docs	0.3540	0.3860 (+ 9.04%)	0.3800 (+ 7.34%) (- 1.55%)
15 docs	0.3187	0.3400 (+ 6.68%)	0.3413 (+ 7.09%) (+ 0.38%)
20 docs	0.2900	0.3140 (+ 8.28%)	0.3270 (+12.76%) (+ 4.14%)
30 docs	0.2600	0.2753 (+ 5.88%)	0.2953 (+13.58%) (+ 7.26%)

Table 4. The distributed retrieval results for trec4_kmeans testbed by CORI algorithm, scores normalized without cooperation from individual databases, scores normalized without cooperation from individual databases and Language Model for distributed information retrieval. (The first baseline in CORI with cooperation and Language Model for distributed information retrieval is the performance of CORI algorithm without cooperation; the second baseline in Language Model for distributed information retrieval is the performance of CORI algorithm with cooperation)

section were designed to compare the performance of our language model distributed information retrieval algorithm with CORI.

As described in Section 2, there are two versions of the CORI merging algorithm. As indicated by Equation (4), in order to calculate the normalized document score, we need the maximum possible document score and minimum possible document score for the individual database. In the cooperation case, CORI merging algorithm assumes that it can get these scores from the individual search engines. If it is not possible, it simply uses the maximum and minimum document scores returned by the search engine. Notice that our language model algorithm does not need this kind of cooperation from individual database to provide the normalization scores.

Table 3 and Table 4 show the retrieval results by three distributed information retrieval algorithms, namely CORI algorithm without cooperation (normalization), CORI algorithm with cooperation (normalization) and the language model algorithm for distributed information retrieval, carried both on trec123 and trec4_kmeans testbeds. The CORI algorithm without cooperation from databases is used a baseline. It can be seen that both CORI algorithm with cooperation and language model algorithm always outperform CORI without cooperation. Someone may argue that the cooperation needed by CORI can be implemented as an internal procedure in individual Inquery search engines, but it still needs to modify the original Inquery retrieval algorithm [1,18]. On the trec4_kmeans testbed, the language model algorithm is at the same level as CORI algorithm with cooperation. On the trec123 testbed, Language Model for distributed information retrieval has a notable improvement over the CORI with cooperation.

6. CONCLUSIONS

This paper presents a language modeling based framework for distributed information retrieval task in intranet environment, where all databases use a language model based search engine. Under this framework, the three sub-problems within the distributed information retrieval have been viewed as components of an integrated task. In the resource description, it uses query-based sampling to acquire language models description of each individual database. In the resource selection, it ranks the databases based on how likely a given query can be generated from the language model of every database. In the results merging, it computes the 'fair' scores for documents by removing the bias within the original document scores caused by the different corpus statistics.

Our experiments carried on two testbeds of different characteristics in Section 5 demonstrate that the language model algorithm always outperforms the CORI algorithm without normalization cooperation from individual databases. It is better than the CORI algorithm with normalization cooperation on one testbed and at the same level on the other testbed. Therefore, we tend to conclude that the simple language model for distributed information retrieval is an effective approach for finding relevant documents in the intranet distributed information retrieval environment.

One problem left in the new framework is the parameter β, which was empirically set to 19. To test the sensitiveness of our algorithm to the value of the parameter β, we varied it from 1 to 99 and repeated the experiments over testbeds trec123 and trec4_kmenas. The results are quite similar on both testbeds. Therefore, the performance of the new framework appears to be insensitive to the setup of the parameter β. More investigation needs to be done over different environments. We hope that a regression algorithm could be used set this parameter automatically [15].

7. ACKNOWLEDGEMENTS

This material is based on work supported by NSF grants EIA-9983253, IIS-0118767, and DUE-0085834, NSF Cooperative Agreement IRI-9817496, and Advanced Research and Development Activity (ARDA) contract MDA908-00-C-0037. Any opinions, findings, conclusions or recommendations expressed in this material are the authors', and do not necessarily reflect those of the sponsor.

8. REFERENCES

[1] J. Callan. Distributed information retrieval. In W.B. Croft, editor, *Advances in information retrieval,* chapter 5, pages 127-150. Kluwer Academic Publishers, 2000.

[2] F. Song and W. B. Croft. A general language model information retrieval. In *Proc of the 22nd Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval,* 1999.

[3] J. Pone and W. B. Croft. A language modeling approach to information retrieval. In *Proc of the 21st Annual Int'l ACM*

SIGIR Conference on Research and Development in Information Retrieval, 1998.

[4] J. Callan, Z. Lu, and W. B. Croft. Searching Distributed Collections with Inference Networks. In *Proc. of the 18th Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval,* 1995.

[5] J. Callan and M. Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems, 19(2): 97-130,* 2001.

[6] L. Gravano, C. Chang, H. Garcia-Molina, and A. Paepcke. STARTS: Stanford Proposal for Internet Meta-Searching. *In Proc. of the ACM-SIGMOD Int'l Conference on Management of Data,* 1997.

[7] L. Gravano, H. Garcia-Molina, and A. Tomasic. The Effectiveness of GlOSS for the Text Database Discovery Problem. *In Proc. of the ACM-SIGMOD Int'l Conference on Management of Data,* 1994.

[8] L. Gravano and H. Garcia-Molina. Generalizing GlOSS to Vector-Space Databases and Broker Hierarchies. *In Proceedings of the 21st International Conference on Very Large Databases (VLDB),* 1995.

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

[10] N. Craswell, D. Hawking, and P. Thistlewaite. Merging Results from Isolated Search Engines. *In Proc. of the Tenth Australasian Database Conf., pages 189--200,* 1999

[11] J.C. French, A.L. Powell, J. Callan, C.L. Viles, T. Emmitt, K.J. Prey, and Y. Mou. Comparing the performance of database selection algorithms. In *Proc of the 22nd Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval,* 1999.

[12] A F. Smeaton and F. Crimmins Using A Data Fusion Agent for Searching the WWW. *In Proc of The Sixth International World Wide Web Conference,* 1997.

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

[14] J. Xu and J. Callan. Cluster-based Language Models For Distributed Retrieval. In *Proc of the 22nd Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval,* 1999.

[15] L. Si and J. Callan. Using Sampled Data and Regression to Merge Search Engine Results. In *Proc of the 25th Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval,* 2002.

[16] L. Larkey, M. Connell, and J. Callan. Collection selection and results merging with topically organized U.S. patents and TREC data. *In Proceedings of Conference of Information and Knowledge Management,* 2000.

[17] Lemur Toolkit http://www.cs.cmu.edu/~lemur

[18] J. Callan, W.B. Croft, and J. Broglio, TREC and TIPSTER experiments with INQUERY. *Information Processing and Management, 31(3):327-343,* 1995.

[19] D R. H. Miller, T. Leek, and R. M. Schwartz. A Hidden Markov Model Information Retrieval System. In *Proc of the 22nd Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval,* 1999.

[20] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proc of the 24th Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval,* 2000.

[21] C. Buckley, A. Singhal, M. Mitra, and G. Salton, New retrieval approaches using SMART. *In Proceedings of 1995 Text REtrieval Conference (TREC-3).* National Institute of Standards and Technology, special publication.