Distributed Information Retrieval With Skewed Database Size Distributions

Luo Si, Jie Lu, and Jamie Callan
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
lsi@cs.cmu.edu, jielu@cs.cmu.edu, callan@cs.cmu.edu

Abstract

The proliferation of government information on local area networks and the Internet creates the problem of finding information that may be distributed among many disjoint text databases (*distributed information retrieval* or *federated search*). A distributed information retrieval system is composed of three components: Resource representation, resource selection and result merging. Previous research suggested that the CORI algorithm is one of the most effective resource selection algorithms, but its effectiveness in environments containing a wide range of database sizes was not studied thoroughly. This paper shows that the CORI algorithm does not work well in environments with a skewed distribution of database sizes. We present a new resource selection algorithm based on estimating the distribution of relevant documents among the online databases. This new algorithm selects resources more accurately than the CORI algorithm, which can lead to improved document rankings.

1. Introduction

Individual government departments and agencies increasingly create and maintain digital libraries of text documents, which has caused a proliferation of government search engines. For example, the Government Printing Office (GPO) provides online access to more than 2,200 databases of federal documents created by Congress, the White House, and about 130 other departments and agencies [6]. This proliferation of document databases makes information hard to find unless one knows where to look. The goal of *distributed information retrieval* or *federated search* systems is to provide a single, uniform search engine interface that provides access to all of the available digital libraries, guiding each information need to the appropriate digital libraries, and merging what is returned into a single, integrated list of results. This problem involves three subproblems: i) acquiring a description of the contents of each document database (*resource representation*), ii) choosing which databases to search to satisfy a particular information need (*resource selection*), and iii) merging the results retrieved from each database into a single integrated list (*result merging*) [3]. All three of these subproblems have received considerable attention in recent years.

The goal of the resource selection task is to select a small a set of document databases that are likely to contain many (ideally, most) of the relevant documents. The CORI resource selection algorithm is a well-known example; it essentially models each document database as a very large document, and ranks each database using a modified version of a well-known document ranking algorithm. Previous research has demonstrated that the CORI algorithm is one of the most effective and robust resource selection algorithms developed so far [1, 4], but the previous research was confined to environments in which database sizes did not vary widely. In some "real world" environments, such as the U.S. government, allowing digital libraries to be created and maintained independently results in a distribution of database sizes that varies over a large range and is skewed (e.g., many small DBs, a few massive DBs).

In this paper, we show that the CORI algorithm does not work well in environments containing a mix of "small" and "very large" document databases. A new resource selection algorithm, Relevant Document Distribution Estimation (ReDDE), is more effective. The ReDDE algorithm estimates the distribution of relevant documents across the databases for each user query and ranks databases according to this distribution. Experiments demonstrate that the new algorithm selects resources more accurately than the CORI algorithm in environments containing a wide range of database sizes.

The next section reviews prior research on distributed information retrieval. Section 3 describes the Relevant Document Distribution Estimation (ReDDE) algorithm in more detail. Section 4 discusses the subtle relationship between resource selection accuracy and document retrieval accuracy and proposes the modified version of the ReDDE algorithm for better retrieval performance. Section 5 explains our experimental data. Sections 6 and 7 present the experimental results. Section 8 concludes.

2. Prior Research

There has been considerable research on all the three subproblems of distributed information retrieval during the last decade. We survey related work in this section to provide an overview of the state of the art, focusing on the research that is the most relevant to the research reported in this paper.

In uncooperative environments perhaps the simplest and most reliable way to determine the contents of a document database is to submit queries and examine what is returned, a process called *query-based sampling* [3]. Prior research demonstrated that query-based sampling acquires relatively accurate resource descriptions with only a small number of queries that retrieve a relatively small number of documents. For example, 75-100 queries, each retrieving 4 documents, is sufficient to create accurate descriptions of databases containing up to 1,000,000 documents [3].

A variety of resource selection algorithms have been investigated in prior research [5, 3, 1]. The CORI resource selection algorithm has been shown by different researchers to be one of the most stable and effective resource selection algorithms [1, 4], so it was used as a baseline in the research reported here. The CORI algorithm operates on resource descriptions that consist of vocabulary, term frequency, and corpus statistics for each document database. It is based on a Bayesian Inference Network model of information retrieval in which each resource is ranked by the belief $P(q|C_j)$ that query q is satisfied given that resource C_j is observed (searched). The belief $P(q|C_j)$ in the simple case relevant to the research reported here is the average of the beliefs $P(r_k|C_j)$ for all query terms r_k in q [3].

The database representation and the resource selection components can be combined into a database recommendation system that suggests which online databases are best for a specific query. This is sufficient when someone wants to search the databases manually. If the person wants a federated search system to also search the selected databases and merge the document rankings that are returned from each database, a result-merging component is needed. The Semi Supervised Learning (SSL) result merging algorithm [7] uses the documents acquired by query-based sampling as training data to build query-specific, database-specific regression models that map document scores returned by individual databases into normalized document scores. The SSL algorithm is very accurate and can be used with both the CORI and ReDDE resource selection algorithms, so it was used in the experiments reported below.

3. The Relevant Document Distribution Estimation (ReDDE) algorithm

The goal of resource selection is to identify a few databases that each contain many relevant documents. It is typically approached as a problem of ranking databases, with the top few databases (e.g., 3, 5, or 10) automatically selected for search. A *relevance based ranking* (RBR), in which databases are ranked by the number of relevant documents they contain, is typically considered the best a resource selection algorithm could do.

The ReDDE algorithm tries to estimate the distribution of relevant documents among the available databases. For simplicity, we assume here that each database publishes the number of documents it contains (its *size*).[1] The number of documents in database C_j that are relevant to query q is estimated as:

---

[1] This value can also be estimated without explicit cooperation from the database provider [8].

$$\hat{Re1\_q}(j) = \sum_{d_i \in C_j} P(rel \mid d_i) * P(d_i \mid C_j) * N_{C_j} \tag{1}$$

where N_{C_j} is the number of documents in database C_j, $P(d_i / C_j)$ is the probability of selecting document d_i from C_j, and $P(rel / d_i)$ is the probability that document d_i is relevant. If one had complete access to all databases the probability $P(d_i / C_j)$ would be $1/N_{C_j}$ and the probability $P(rel / d_i)$ could be estimated by a document ranking algorithm. However, complete access is rare when search engines are independent.

In uncooperative environments the database resource descriptions can be created by submitting queries to the database and examining the documents that are returned (*query-based sampling*) [3]. Query-based sampling is known to be an effective method of creating resource descriptions; we show here that the documents obtained by query-based sampling can be used to solve other problems, too.

Given a representative sample of documents from a database, Equation 1 can be rewritten as:

$$\hat{Re1\_q}(j) = \sum_{d_i \in C_{j\_samp}} P(rel \mid d_i) * \frac{N_{C_j}}{N_{C_{j\_samp}}} \tag{2}$$

where C_{j_samp} is the set of documents sampled from database C_j and $N_{C_{j\_samp}}$ is its size.

We define the *centralized complete database* as the union of all the documents in all the databases. Ideally, we want to calculate the probability $P(rel \mid d_i)$ by utilizing the rank information of the document in the centralized complete database with an effective retrieval method. This means that if a document ranks in some specific top percentage of the centralized complete database, the probability will be some query dependent positive constant otherwise it is zero. Formally, we have:

$$P(rel \mid d_i) = \begin{cases} C_q & \text{if } Rank\_central(d_i) < ratio * N_{all} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $Rank\_central(d_i)$ is the rank of a document in the centralized complete database, and N_{all} is the number of documents in all the databases. The $ratio$ is set to 0.003 in our experiments, which means we only consider as partially relevant the top 3,000 document in a testbed of 1,000,000 documents.

It is impractical to build a centralized complete database. However, it is very practical to combine the documents obtained during query-based sampling of each database into a *centralized sample database* that approximates a centralized complete database. The rank of a document in the centralized complete database can be estimated from its rank within the centralized sample database. The rank of a document in the centralized complete database is estimated as:

$$Rank\_\hat{Central}(di) = \sum_{\substack{d_j \\ Rank\_Samp(dj) < Rank\_Samp(di)}} \frac{N_{c(dj)}}{N_{c(dj)\_samp}} \tag{4}$$

where $Rank\_samp(d_i)$ is the rank of the document in the centralized sample database. By plugging this estimator into Equation 3, we can estimate the number of relevant documents within each database. As these values still contain unknown query dependent constants from Equation 3, we can normalize them and get the distribution of relevant documents among the databases as:

$$Dist\_\hat{Rel\_q}(j) = \frac{\hat{Rel\_q}(j)}{\sum_i \hat{Rel\_q}(i)} \tag{5}$$

Testbed	Size (GB)	Num of documents			Size (MB)		
		Min	Avg	Max	Min	Avg	Max
Trec123	3.2	752	10782	39713	28	32	42

Table1: Summary statistics for the trec123-100col testbed.

Collection	Num of documents	Size (MB)
LDB1	231,271	665
LDB2	199,690	667

Table2: Summary statistics for the two large databases in the trec123-2ldb-60col testbed.

Our ReDDE resource selection algorithm uses these values to rank all the databases.

4. The Modified ReDDE Algorithm For Retrieval

For a database recommendation system, the database ranking given by the resource selection algorithm is the final result. A complete distributed information retrieval or federated search system requires the additional steps of searching the selected databases and merging the results returned by each.

Our experience is that better resource selection performance does not always produce better retrieval accuracy. Accuracy is measured by Precision at specified document ranks, for example the top 5 to 100 documents. The top 100 documents are only about 0.0001 percent of a testbed with 1 million documents. A database that contains more relevant documents does not always return more relevant documents within the top ranked documents. Retrieval accuracy is improved only when a resource selection algorithm maximizes the number of relevant documents that appear in the top ranks, which represents a much smaller percentage than the ratio 0.003 used in resource selection. Relevance at the very highest ranks can be emphasized by using a much smaller ratio, for example 0.0005. However, because query-based sampling only samples a rather small percentage of the documents from each database, a small ratio causes a lot of databases to have the estimated value of 0 percent of the relevant documents. Our solution is to use two ratios: a smaller one and a larger one. For databases that have large enough estimation values with the smaller ratio, they are sorted by these values. For all the other databases, they are sorted by the estimation values with a larger ratio. This modified ReDDE algorithm is formalized as follows:

1. First rank all the databases that have a large value with the smaller ratio: $DistRel\_r1j >= backoff\_Thres$
2. For all the other databases rank them by the values with the larger ratio: $DistRel\_r2j$

The two ratios were set to 0.0005 and 0.003 in our experiments and the backoff_Thres is set to 0.05.

5. Experimental Data

Two testbeds were created to evaluate the performance of ReDDE resource selection algorithm.

Trec123-100col-bysource: 100 databases were created from TREC CDs 1, 2 and 3. They were organized by source and publication date [3]. The sizes of the databases are not skewed. The characteristics of this testbed are shown in Table 1. This testbed has been used often in prior research (e.g., [3]).

Trec123-2ldb-60col: The databases in Trec123-100col-bysource were sorted alphabetically. Every fifth database, starting with the first, was collapsed into one large "representative" database called LDB1. Every fifth database, starting with the second, was collapsed into another large database called LDB2. The other 60 databases were left unchanged. LDB1 and LDB2 are about 20 times larger than the other databases but have about the same densities of relevant documents as the other databases. Details are shown in Table 2.

50 queries, containing an average of 3 words, were created from the title fields of TREC topics 51-100.
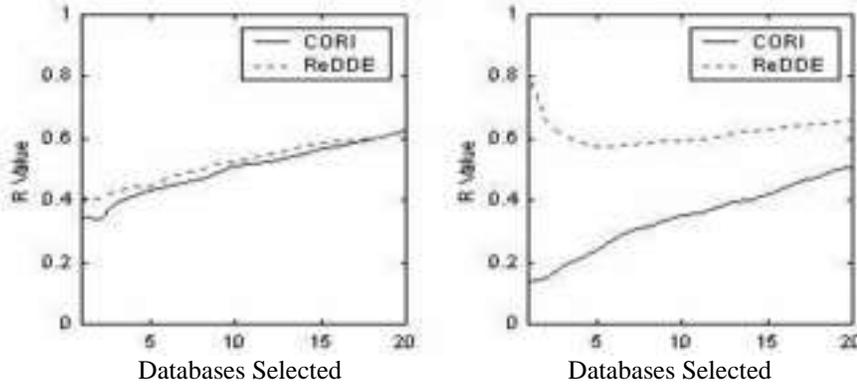
Figure 1. Resource selection accuracy on the trec123-100col (left) and trec123-2ldb-60col (right) testbeds. A perfect system would have a value of 1.0 for all values of *x*.

6. Resource Selection Results

The ReDDE and CORI algorithms were compared using the two testbeds described above. Resource descriptions were built by query-based sampling, using 75 queries to obtain 300 documents per database. INQUERY [2] ranked the documents in the centralized sample database (Equation 4).

Different resource selection algorithms are typically compared using the recall metric R_n [3, 4]. Let us denote B as a baseline ranking, which is often the RBR (relevance based ranking), and E as a ranking provided by a resource selection algorithm. And let B_i and E_i denote the number of relevant documents in the ith ranked database of B or E. Then R_n is defined as shown below.

$$R_k = \frac{\sum_{i=1}^{k} E_i}{\sum_{i=1}^{k} B_i} \qquad (6)$$

Usually the goal is to search only a few databases, so in our experiments the resource selection algorithms were only evaluated over the top 20 databases. The experimental results were averaged over 50 queries.

The ReDDE algorithm outperformed the CORI algorithm in both of the multi-database test environments (Figure 1). The difference was small on the trec123-100col testbed, which has relatively uniform database sizes; the difference was large on the trec123-2ldb-60col testbed, which has a skewed database size distribution. CORI did poorly on this testbed because it rarely ranked the two large databases in the top 10. This experiment indicates that the CORI algorithm has a strong bias against large databases, a characteristic that was not known previously. The new ReDDE algorithm appears not to have this bias.

7. Retrieval Results

The experiments reported above demonstrate improved resource ranking, but do not measure whether a better set of documents is returned. An additional set of experiment addressed this issue.

The three best databases identified by each resource ranking algorithm were searched by the INQUERY search engine [2]; this design models an organization that buys software from a single vendor but allows each part of the organization to manage its databases independently. The SSL result merging algorithm was used to merge search results into a final ranked list. Results were averaged over 50 queries.

The modified ReDDE algorithm led to slightly more accurate results on the trec123-100col testbed and much more accurate results on the trec123-2ldb-60col testbed (Table 3). These results confirm that the modified ReDDE algorithm is a more effective platform for federated search than the CORI algorithm.

Document Rank	Trec123-100col Testbed		Trec123-2ldb-60col	
	CORI	Modified ReDDE	CORI	Modified ReDDE
5	0.3760	0.4520 (+20.21%)	0.3720	0.4440 (+19.4%)
10	0.3660	0.3880 (+6.0%)	0.3680	0.4420 (+20.1%)
15	0.3453	0.3627 (+5.0%)	0.3440	0.4107 (+19.4%)
20	0.3140	0.3340 (+6.4%)	0.3240	0.3960 (+22.2%)
30	0.2873	0.3080 (+7.2%)	0.2853	0.3713 (+30.1%)
100	0.1750	0.1976(+12.9%)	0.1692	0.2683(+58.6%)

Table 3. Precision at different document ranks using the CORI and ReDDE resource selection algorithms. All the databases used the INQUERY search engine. 3 databases were selected to search for each query. SSL algorithm was used to merge the results. Results were averaged over 50 queries.

8. Conclusion

Distributed information retrieval / federated search capabilities are becoming increasingly important as search engines proliferate. When different government agencies manage their document databases independently, and particularly when agencies such as the Government Printing Office act as aggregators of information from other agencies, the distribution of database sizes becomes skewed. In this paper we demonstrate that this skew can be a problem for a database selection algorithm that has been considered the state-of-the-art. A new resource selection algorithm is proposed that tries to explicitly estimate the distribution of relevant documents across the document databases. Experiments show that this new resource selection algorithm produces more accurate database rankings than the well-known alternative in environments with skewed and homogeneous database size distributions. Experiments also show that a modified version of the new resource selection algorithm results in more accurate document rankings.

Acknowledgement

References

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

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

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

[4] 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.

[5] L. Gravano, H. Garcia-Molina, and A. Tomasic. "GlOSS: Text-Source Discovery over the Internet." *ACM Transactions on Database Systems,* 24(2), 1999.

[6] http://www.access.gpo.gov/

[7] L. Si and J. Callan. "Using Sampled Data and Regression to Merge Search Engine Results." In *Proc. of the 24th Annual International ACM SIGIR Conference on Research and Development in Information* Retrieval, 2002.

[8] L. Si and J. Callan. "Relevant document distribution estimation method for resource selection." In *Proc. of the 26th Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval*, in press.