# Using Sampled Data and Regression to Merge Search Engine Results

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

## ABSTRACT

This paper addresses the problem of merging results obtained from different databases and search engines in a distributed information retrieval environment. The prior research on this problem either assumed the exchange of statistics necessary for normalizing scores (*cooperative* solutions) or is heuristic. Both approaches have disadvantages.

We show that the problem in uncooperative environments is simpler when viewed as a component of a distributed IR system that uses query-based sampling to create resource descriptions. Documents sampled for creating resource descriptions can also be used to create a sample centralized index, and this index is a source of training data for adaptive results merging algorithms. A variety of experiments demonstrate that this new approach is more effective than a well-known alternative, and that it allows query-by-query tuning of the results merging function.

## Categories & Subject Descriptors:

H.3.3 [**Information Search and Retrieval]:** Distributed Information Retrieval, Results Merging

## General Terms:

Algorithms

## Keywords:

Distributed Information Retrieval, Results Merging, Regression

## 1. INTRODUCTION

The proliferation of online searchable databases on local area networks and the Internet creates a problem of finding information that may be distributed among many text databases (*distributed information retrieval*) [1]. Distributed information retrieval includes three sub-problems. First, information about the contents of each individual database must be acquired. This task is called *resource description* [2,3,4]. Second, a set of text databases must be selected for search [5,8,9]. This task is called *resource selection* or *collection selection*. Third, after results are returned from selected databases, they must be merged into a single ranked list. This task is called *results merging* [1,7,8,10, 11,12,15]. All three tasks are important and interrelated.


*SIGIR '02*, August 11-15, 2002, Tempere, Finland.


Environmental characteristics can simplify or complicate the solution of these problems. In a small local area network the individual search engines might all be of the same type; we call this the *single engine type* case. In a large organization there may be a small number of search engine types, and it may be known which databases use which types of search engine. This knowledge can simplify resource description and results merging. In a very large network or on the Internet there may be many search engine types, and it may not be known which type of engine is associated with each database.

There is also variation in the extent to which search engines *cooperate* in distributed information retrieval. Varying degrees of cooperation have been assumed in prior research, for example assuming that search engines would provide complete and partial resource descriptions [3,10], normalizing statistics [11], normalized document scores [20], or training data [23].

In this paper, we focus on environments containing multiple types of independent, uncooperative search engines. We assume that search engines will run queries and return documents, but they do not provide other information about themselves or their databases. Our goal is solutions that can be applied on wide-area, multi-party, multi-vendor networks where resource providers may not cooperate or may have an incentive to cheat.

The state of the art in distributed IR research is that reasonably good solutions exist for acquiring resource descriptions and for selecting which databases to search, but only weak solutions exist for merging results. Most of the prior research on data fusion and meta-search does not apply, because it assumes considerable overlap in the contents of the databases searched, and hence considerable overlap in the documents returned. But in distributed IR environments, it is more likely that different databases contain, and return, different documents.

The results merging problem is difficult because each engine may use a different ranking algorithm, and may base its ranking on corpus statistics (e.g., idf, average document length) that vary widely. Merging based upon unnormalized ("raw") document scores or document ranks works well when search engines and corpora are very similar, but can be very inaccurate when they differ. Merging based upon weighted document scores [1,2] or ranks [8] has been the state-of-the-art for merging quickly, but is heuristic and prone to unexpected failure. The alternative solution is to download the contents of the retrieved documents and then to re-rank them at the search client [11,12], which produces a consistent ranking but can be very time-consuming.

In this paper we present a fast and effective new solution to the results merging problem. Our new solution uses three different types of information to normalize the document scores produced by

individual databases: Information about the database contents (from the *resource description*), information about how well the database is thought to satisfy the query (the *database score*), and the document score returned by the database. No cooperation is assumed, and there are no assumptions about the type(s) of search engines used. That information is used to build functions that transform document scores from different search engines into a normalized form suitable for merging.

We also present experiments comparing the new approach to the results merging algorithm usually used with the well-known CORI resource selection algorithm [1]. We demonstrate that the new approach is more effective on two distributed IR testbeds, particularly a subject-oriented testbed with skewed vocabulary patterns [20]. We demonstrate its effectiveness in both *single search engine type* and multiple *engine types* environments.

The next section discusses related work. Section 3 describes our new approach to merging results. Section 4 explains our experimental methodology. Sections 5, 6, and 7 present our experimental results for the single search engine type and multiple search engine type cases. Section 8 concludes.

## 2. PRIOR RESEARCH

There has been considerable research on acquiring resource descriptions, ranking resources, and merging results. 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 or most closely related to the new research presented in this paper.

The STARTS protocol is one solution for acquiring resource descriptions [3]. It requires every resource provider to provide accurate resource descriptions to a resource ranking algorithm upon request. It is a cooperative protocol, because it only works when every resource provider supplies accurate vocabulary, frequency, and corpus information. STARTS is a good solution in environments where cooperation can be guaranteed. However, in multi-party environments some resource providers may choose not to cooperate; when they do cooperate, it is impossible to know whether they provide accurate information.

Query-based sampling [4] is an alternative approach to acquiring resource descriptions that does not require explicit cooperation from resource providers. The database selection service constructs its own resource descriptions by sampling individual databases contents via the normal process of running queries and retrieving documents. This solution 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).

There are many resource ranking algorithms; we cannot survey all of them here. gGlOSS [6], CORI [4,1] and CVV [8] are three of the best-known resource ranking algorithms. gGlOSS, is based on the vector space model. It represents a database by the document frequency of each word in the database, and the sum of the term weights in each document in the database. The CVV resource ranking algorithm uses a combination of document frequency and cue validity variance information. The variability of the fraction of documents in a database that contains a specific word is characterized by the cue validity variance. The CORI resource ranking algorithm represents each database by its terms, their document frequencies, and summary corpus statistics such as total word count.

Different researchers using different datasets have shown the CORI algorithm to be the most stable and effective of the three algorithms [9,13]. CVV is often nearly as effective as CORI, but can fail dramatically when its underlying assumptions are violated. CORI and gGlOSS are compatible with resource descriptions acquired by query-based sampling; CVV is not.

We use the CORI algorithm in the research reported here, so we sketch it briefly. The belief $p(r_k \mid db_i)$ in database $db_i$ according to the query term $r_k$ is determined [1,2] by:

$$T = \frac{df}{df + 50 + 150 * cw / avg\_cw} \quad (1)$$

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

$$p(r_k \mid c_i) = b + (1 - b) * T * I \quad (3)$$

where:

df is the number of documents in $db_i$ that contains the $r_k$
cf is the number of databases that contain $r_k$
|DB| is the number of databases to be ranked
cw is the number of words in $db_i$
avg_cw is the average cw of the databases to be ranked
b is the default belief, usually 0.4.

The combination of belief values depends on the query structure. In the simple case relevant to our research, it is the average of the beliefs of the query terms. Generally the 5 or 10 databases with highest belief are selected [1,4,9].

In general, the results merging task is difficult because different databases may have different types of search engines and different corpus statistics. The most accurate solution is to normalize the scores of documents retrieved from different databases, either by using global corpus statistics [10,7], which requires cooperation, or by recomputing document scores at the search client [11], which has high communications costs. A less costly solution is to download only parts of documents [12].

A related approach used for meta-search is to heuristically combine the outputs of search engines, for example, Lee's COMBSUM and COMBMNZ algorithms [14]. The main drawback of these techniques is that they rely on a lot of overlap among the results from different search engines. Javed's Bayesian model is theoretically solid, but it also relies on overlap among the results and also needs relevance information about some queries [21]. Manmatha proposed that a score modeling method might also work well for results merging in distributed IR [15], but our experience is that the modeling technique assumes the presence of many relevant documents in each database, which is often not satisfied in our experiments.

The CORI merging algorithm is a linear combination of the score of the database and the score of the document. It makes no assumption about the number of overlap documents or the number of relevant documents in each database. The intuition is to favor documents from databases with high scores and to enable high-scoring documents from low-scoring databases to be ranked highly. This algorithm has been very effective, but it has not been applied to search engines other than INQUERY.

The CORI results merging algorithm serves as a baseline later in this paper, so we provide its algorithm here. A "normalized" score suitable for merging is calculated as shown below.

$$C_i' = \frac{(R_i - R_{\min})}{(R_{\max} - R_{\min})} \quad (4)$$

$$D' = \frac{(D - D_{\min})}{(D_{\max} - D_{\min})} \quad (5)$$

$$D'' = \frac{D' + 0.4 * D' * C_i'}{1.4} \quad (6)$$

If T in Equation 1 is set to 1.0 for each query term, a score $R_{\max}$ can be computed for each query. If T is set to 0.0 for each query term, a score $R_{\min}$ can be computed for each query. These are the highest and lowest scores that the resource ranking algorithm could potentially assign to a database. So Equation 4 is the normalized database weighting score that can be calculated only from information in the resource selection index. Equation 5 tries to normalize document score from individual search engines to the range between 0 and 1. It 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. Equation 6 calculates the final document scores that can be compared directly.

The research most similar to what we propose in the following sections is based on building logistic regression models for results merging [16]. A key difference is that in the prior research relevance information from additional queries was used to train the models, whereas our new method does not need relevance information. Another is that the same regression model was used for all queries to a given database, but we build different regression models for each query to each database.

# 3. REGRESSION MODEL

Distributed IR algorithms are often evaluated by comparing them to *centralized* solutions, in which all documents are stored in a single search engine. Indeed, solutions that rely upon exchanging corpus statistics are in effect replacing engine-specific document scores with the scores those documents would have gotten if they had been in a single, centralized database.

Results merging is a difficult problem when viewed in isolation, because little information is available for estimating centralized document scores. The CORI results merging algorithm views the problem in the context of the database ranking algorithm, producing "normalized" document scores as a linear combination of database-specific document scores and the scores of the databases they were retrieved from. It is effective, but it is not a general solution because the weighting of database scores and document scores is determined by a constant tuned to produce good average case performance for the CORI database ranking algorithm and the INQUERY search engine.

We can enlarge our view of the distributed IR environment further still, by defining how resource descriptions are obtained. Query-based sampling is a technique in which resource descriptions are created by submitting queries to a search engine and observing the contents of the documents that are returned. Several studies have shown that query-based sampling produces accurate resource descriptions [1,4], but that is not our concern here. The central insight in the research presented here is that results merging is an easier problem if we assume that resource descriptions are created by query-based sampling.

Ordinarily the documents obtained by query-based sampling are discarded after resource descriptions are created, but that need not be the case. Instead, the documents sampled from the various databases can be combined into a single, centralized sample database. This centralized sample database is a subset, perhaps a very small subset, of the complete centralized database often used as a baseline in distributed IR experiments. However, it provides an important source of additional information for results merging.

After the query is sent to the N best databases and different ranked lists are returned, the same query can be used to search the centralized sample database. The result is a set of database-specific document rankings, and a single centralized sample document ranking. Our first hypothesis is that some documents will appear in a database-specific ranking *and* the centralized sample document ranking. This hypothesis might seem unlikely, but recall that resource descriptions are created from the documents in the centralized sample database; a database is only selected for search when its resource description (and hence some documents in the centralized database) matches the query closely. Our second hypothesis is that the scores of documents that appear in both lists can be used to find a function that will map *all* database-specific document scores into their corresponding centralized document scores.

Regression is an efficient and effective mathematical tool for this kind of problem. First we assume the kind of function that is suitable for this problem. Then we try to find the parameter values that can best fit the training data under some criterion. It can be formalized as follows:

$$\lambda = \arg\max_{\lambda} \sum_{i} (f(\lambda, x_i) - y_i)^2 \quad (7)$$

Here, $\lambda$ is the parameter set, $f(\lambda, x_i)$ is some kind of function, and $(x_i, y_i)$ is the ith training data point.

What form of merging function should we choose? Linear functions are simple, can be fit with a small amount of data, and can be solved efficiently. Because we may have only a limited amount of training data for each database, and because the CORI algorithm is evidence that a linear function can be successful, we chose simple linear models for our experiments.

If it is known that all search engines are of the same type (*single engine type* case), then one regression model can be calculated with all the training data. If the search engine types differ or are not known (*multiple engine types* case), different regression models must be learned for each database. Our interest is primarily in the second case, so we describe it first.

## 3.1 Multiple Search Engine Types

The training data for linear regression are pairs of "overlap" document scores $(Dd_{ij}, Dc_{ij})$ obtained from the distributed and centralized sample databases. We assume that a simple linear model $Dd_{ij}' = a_i * Dd_{ij} + b_i$ relates the scores, where $Dd_{ij}'$ is an approximation of $Dc_{ij}$. Regression over all training data from a given database $i$ is expressed in matrix representation as:

$$\begin{bmatrix} Dd_{i,1} & 1 \\ Dd_{i,2} & 1 \\ ..... & 1 \\ Dd_{i,n} & 1 \end{bmatrix} * \begin{bmatrix} a_i & b_i \end{bmatrix} = \begin{bmatrix} Dc_{i,1} \\ Dc_{i,2} \\ ..... \\ Dc_{i,n} \end{bmatrix} \quad (8)$$

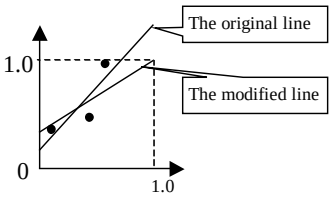


**Figure 1.** The linear model adjustment.

where $a_i$ and $b_i$ are the parameters in the linear formula. Call these matrices *X* (database-specific scores and constants), *W* (parameters) and *Y* (centralized sample scores). Simple algebraic manipulation allows this problem to be expressed as:

$$W = \left(X^T X\right)^{-1}(Y^T X) \qquad (9)$$

Equation 9 is the solution with a Minimum Square Error (MSE) criterion. The result is a set of constants ($a_i$, $b_i$) for each database *i* that allow *all* database-specific document scores to be mapped to close approximations of their corresponding scores in the centralized sample database. These approximations of centralized scores are comparable across databases, and can be used for merging ranked lists returned by different databases.

### *3.1.1 Model Adjustment*

Data sparseness can cause an anomaly in the linear models learned. The model learned from "overlap" training data might produce a document score greater than 1 for unseen documents. A score above 1 is not itself a problem, but it signals that the model is biased high, i.e., has too large a slope, which gives an advantage to documents returned by that database (Figure 1).

We address this case by finding the nearest line that intersects (1,1). If the original line is expressed as y=ax+b and the adjusted line is y=a'x+b', then the problem can be expressed as:

$$(a',b') = \arg\min_{a',b'} \int_0^1 [(a'-a)*x + (b'-b)]^2 dx \qquad (10)$$

By simple mathematical modification, we get:

$$a' = \frac{3-a-3b}{2} \quad \cdot \quad b' = 1-a' \qquad (11)$$

This modification gives slightly better accuracy.

### *3.1.2 Training Data Requirements*

For the multiple search engine types case, as different kinds of search engines have different searching algorithms, a regression model should be calculated for each different database. At least two training data per database are required to fit a linear model, but more data produces a more accurate model. We call a database "bad" if it has less than 3 overlap documents, because we assume that a database with many relevant documents should also have a pretty large number of overlap documents.

If the number of bad databases is less than a threshold, the results from those databases are just discarded. The threshold is assigned empirically as 3. If the number of bad databases exceeds the threshold, there is insufficient data to train the models, so the CORI results merging algorithm is used.

We also must choose how many data points to use in learning the linear regression model for each database. We believe that it is more important to be precise at the top of the merged ranking, so our solution is to limit training to the top 10 "overlap" documents from each database. If a database has fewer than 10 "overlap" documents, all are used. This value was set empirically. Preliminary experiments suggested that it is a reasonable choice.

## 3.2 Single Search Engine Type

In some environments it is known that a single engine is used for all databases. In this environment it might be possible to combine training data from different databases, which might be sufficient to learn a slightly more complex model.

The CORI results merging algorithm is a relatively successful single-engine heuristic that combines the database score $C_i$ and the database-specific document score $Dd_{ij}$ via the model:

$$Dd_{ij}' = \frac{Dd_{ij} + 0.4 Dd_{ij} C_i}{1.4} \qquad (12)$$

This model uses one parameter to control the impact of the database score. Our hypothesis is that training data and linear regression will allow this parameter to be adjusted on a query-by-query basis, which will improve results merging accuracy.

As with the multi-engine type solution, training data are based on pairs of "overlap" document scores $(Dd_{ij}, Dc_{ij})$ obtained from the distributed and centralized sample databases. The database score $C_i$ is also included, as shown above. Regression over the training data from all databases is expressed as:

$$\begin{bmatrix} Dd_{11} & C_1 Dd_{11} \\ Dd_{12} & C_1 Dd_{12} \\ \ldots\ldots & \ldots\ldots\ldots\ldots \\ Dd_{nn_i} & C_n Dd_{nn_i} \end{bmatrix} * \begin{bmatrix} a & b \end{bmatrix} = \begin{bmatrix} Dc_{11} \\ Dc_{12} \\ \ldots \\ Dc_{nn_i} \end{bmatrix} \qquad (13)$$

We call these matrices, X, W, and Y, as in the multi-engine type case (Section 3, Equation 8), and solve as described in Equation 9. The result is a pair of constants (*a, b*) that allow *all* database-specific document scores to be mapped to close approximations of their corresponding scores in the centralized sample database.

$$D'' = a * D' + b * C' * D' \qquad (14)$$

These approximations of centralized scores are comparable across databases, and can be used for merging ranked lists returned by different databases.

The model adjustment required for the multi-engine type case (Section 3.1.1) is not required for the single-engine type case. Even if the learned model is biased, there is only one model (as opposed to one per database), so it affects all documents equally.

### *3.2.1 Training Data Requirements*

In the single search engine type case only one regression model is built because all the databases use the same search algorithm. Because the training data from different databases are merged, it is more likely that there will be sufficient training data. One can back off to the CORI results merging algorithm if there is not sufficient training data, as in the multi-engine type case, but that was never necessary in our experiments.

For the single search engine type case, we use only the 20 top documents to build the regression model for all the databases. This value was determined empirically.

# 4. Experimental Methodology

## 4.1 Testbeds

**Table1:** Testbed statistics.

| Testbed | Size (GB) | Number of documents | | | Size (MB) | | |
|---|---|---|---|---|---|---|---|
| | | Min | Avg | Max | Min | Avg | Max |
| Trec123 | 3.2 | 752 | 10782 | 39713 | 28 | 32 | 42 |
| Trec4 | 2.0 | 301 | 5675 | 82727 | 4 | 20 | 249 |

Testbed characteristics have been shown to highly influence the performance of distributed retrieval systems. Two different testbeds were used in our experiments (Tables 1 and 2). Our goal was to test our solutions with testbeds that have different degrees of heterogeneity and different types of queries.

**Trec123-100col-bysource:** 100 databases created from TREC CDs 1, 2 and 3. They are organized by source and publication date [1, 4, 17], and are somewhat heterogeneous. 50 queries were created from the Title fields of TREC topics 51-100.

**Trec4-kmeans:** 100 databases created from TREC 4 data. A k-means clustering algorithm was used to organize the databases by topic [18], so the databases are homogenous and the word distributions are very skewed. 50 queries were created from the Description fields of TREC topics 201-250.

## 4.2 Search Engines

We used three different search engines in our experiments: INQUERY, the Lemur toolkit unigram language model [22], and an in-house version of SMART using "ltc" weighting [19]. These three search engines were selected because they are each effective, but based on very different retrieval models. INQUERY and SMART are each well-known and have been described many times in the research literature, so we do not describe them in detail. Language models are a somewhat newer approach to retrieval, so we sketch the details briefly and refer the reader elsewhere for more information.

The basic idea is to estimate a language model for each document and to rank documents by the likelihood of the query according to the language model:

$$P(Q \mid d) = \prod_{i=1}^{m} P(t_i \mid d) \qquad (15)$$

where Q is the query, d is a specific document, and $t_i$ tis i'th word in the query. Jelinek-Mercer smoothing was used to generate the document language model. It is just a linear combination of the maximum likelihood ($p_{ml}$ (w|d)) and collection ($p_c$ (w|c)) language models. A coefficient $\lambda$ controls the influence of each model.

$$P(w \mid d) = (1-\lambda)\, p_{ml}(w \mid d) + \lambda p_c(w \mid c) \qquad (16)$$

In our experiments, $\lambda$ is set to 0.5.

## 5. Results: "Overlap" Documents

Our first hypothesis was that a sufficient number of documents would appear in both the centralized and database-specific rankings. These "overlap" documents are crucial to our second hypothesis, because the pairs of centralized and database-specific document scores ($Ds_{ij}$, $Cs_{ij}$) are training data. We know that some minimum amount of training data is required to calculate the regression model accurately. Are there enough?

**Table2:** Query set statistics.

| Name | TREC Topic Set | TREC Topic Field | Average Length (Words) |
|---|---|---|---|
| Trec123 | 51-100 | Title | 3 |
| Trec4 | 201-250 | Description | 7.2 |

Many factors influence the number of data points, such as the number of documents obtained by query-based sampling and the characteristics of the query and the databases. Generally, the number of overlap documents increases based on i) the number of documents sampled from each database; ii) the length of the query; and iii) the degree of topic homogeneity in each database.

A set of experiments with two testbeds investigated the average number of "overlap" documents. Experiments were conducted with two 100-database testbeds: trec123 (organized by source and date) and trec4_kmeans (organized by topic). In each testbed databases were assigned randomly to INQUERY, SMART, and Language Modeling search engines. Database resource descriptions were obtained as described in [4], by repeatedly submitting one-word random queries to a database and examining the top 4 documents returned until 300 unique documents were seen. During overlap evaluation, the CORI resource-ranking algorithm ranked databases, and the top 10 were considered selected for search. 1000 documents were retrieved from each selected database. The INQUERY search engine was also used to search the centralized sample database, and all retrieved documents from this database were used.

Figures 2 and 3 show the average number of overlap documents per query in each testbed. For most queries on the trec123 testbed there are more than 50 overlap documents. All of the queries for the trec4-kmeans testbed have more than 200 overlap documents. The different numbers of overlap documents in the two testbeds is not surprising, because the trec4-kmeans testbed has longer queries and homogenous databases.

Three out of fifty queries on the trec123 testbed backed off to the default CORI results merging algorithm because there were too many (4 or more) databases that had too few (less than 4) overlap documents to be confident of learning accurate models. None of the trec4_kmeans queries backed off to the CORI results merging algorithm, i.e., all had sufficient training data.

These results suggest that there are usually enough overlap documents for training a linear regression model.

## 6. Results: Multiple Engines

A series of multi-engine type experiments was conducted to test the effectiveness of the new approach to results merging. Two 100-database testbeds with different document distribution patterns were used (Section 4.1), and three different types of search engines were used (Section 4.2) in several different combinations. Our goal was to determine whether the new approach is robust under a variety of different conditions.

One test examined an environment in which 100 databases were randomly assigned to one of three search engines ("three-engine" test). A second set of tests examined an environment in which 100 databases were randomly assigned to one of two search engines ("two-engine" tests"). "Two-engine" tests were conducted with each pair of engines, so there are three "two-engine" tests. In each experiment the CORI algorithm ranked databases, and the top 10 were considered selected for search

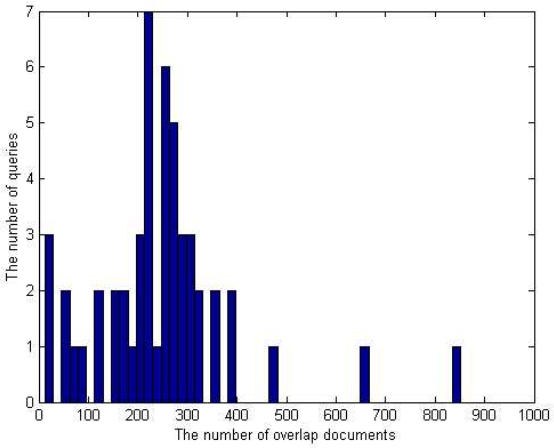


**Figure 2.** The distribution of overlap documents for 50 "Title" queries on the trec123 testbed.

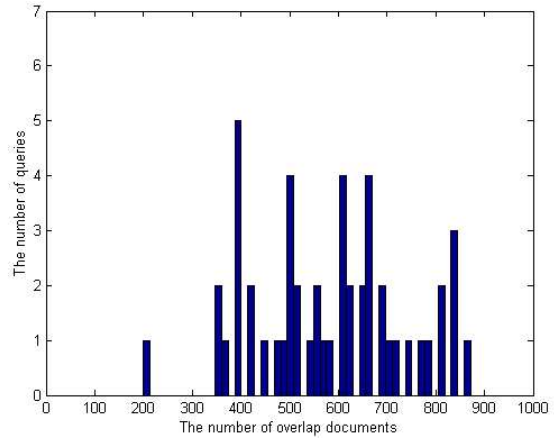


**Figure 3.** The distribution of overlap documents for 50 "Description" queries on the trec4 testbed.

Tests were also conducted with the CORI results merging algorithm, which we consider representative of the state-of-the-art in this area, and it is used as a baseline for comparison in the results below. Note that the CORI results merging algorithm is tuned to work well with the INQUERY search engine but would not necessarily be expected to work well with other search engines, or when the search engine is not known.

The "two engine" results are summarized in Table 3 (INQUERY and language model engines), Table 4 (INQUERY and SMART), and Table 5 (SMART and language model engine). Each table contains results averaged over 50 queries. Regression results shown in bold typeface are significantly better than the corresponding CORI performance (paired Wilcoxon test at p=0.05).

The trec123 testbed was the easiest for the results merging algorithms, because it has a relatively even distribution of vocabulary across databases. The regression algorithm was always as good as or better than the CORI results merging algorithm, but the largest difference was about 10%, for the INQUERY and SMART combination. The trec4_kmeans data was more difficult, because it has a very skewed vocabulary distribution. The regression algorithm handled this testbed far more effectively than the CORI results merging algorithm.

**Table 3.** Precision in "two search-engine type" environment (INQUERY and language model). Bold typeface indicates Regression significantly better than CORI.

| Docs | Trec123 | | Trec4_kmeans | |
|---|---|---|---|---|
| Rank | CORI | Regression | CORI | Regression |
| 5 | 0.3600 | *0.4080* (+13.3%) | 0.2640 | **0.4040** (+53.0%) |
| 10 | 0.3460 | *0.3820* (+10.4%) | 0.1900 | **0.3780** (+98.9%) |
| 15 | 0.3480 | *0.3560* (+2.3%) | 0.1840 | **0.3400** (+84.8%) |
| 20 | 0.3400 | *0.3440* (+1.2%) | 0.1800 | **0.3130** (+73.9%) |
| 30 | 0.3247 | 0.3200 (-1.4%) | 0.1647 | **0.2740** (+66.4%) |

**Table 4.** Precision in "two search-engine type" environment (INQUERY and SMART). Bold typeface indicates Regression significantly better than CORI.

| Docs | Trec123 | | Trec4_kmeans | |
|---|---|---|---|---|
| Rank | CORI | Regression | CORI | Regression |
| 5 | 0.2840 | *0.3280* (+15.5%) | 0.2280 | **0.3360** (+47.3%) |
| 10 | 0.2700 | *0.3040* (+12.6%) | 0.1660 | **0.3080** (+85.5%) |
| 15 | 0.2640 | *0.2947* (+11.6%) | 0.1613 | **0.2760** (+71.1%) |
| 20 | 0.2610 | *0.2820* (+8.0%) | 0.1610 | **0.2620** (+62.7%) |
| 30 | 0.2487 | **0.2867** (+15.3%) | 0.1487 | **0.2240** (+50.6%) |

**Table 5.** Precision in "two search-engine type" environment (language model and SMART). Bold typeface indicates Regression significantly better than CORI.

| Docs | Trec123 | | Trec4_kmeans | |
|---|---|---|---|---|
| Rank | CORI | Regression | CORI | Regression |
| 5 | 0.2880 | *0.2960* (+2.8%) | 0.2120 | **0.3400** (+60.4%) |
| 10 | 0.2680 | *0.2860* (+6.7%) | 0.1800 | **0.3120** (+73.3%) |
| 15 | 0.2693 | *0.2840* (+5.5%) | 0.1680 | **0.2720** (+61.9%) |
| 20 | 0.2680 | 0.2750 (+2.6%) | 0.1620 | **0.2530** (+56.2%) |
| 30 | 0.2620 | *0.2687* (+1.6%) | 0.1507 | **0.2333** (+51.4%) |

The "three engine" results are summarized in Table 6. As in the "two engine" experiments, the regression model has a clear advantage over the CORI results merging algorithm, and that advantage is more pronounced on the trec4_kmeans testbed.

## 6.1 Tuning the CORI Parameter

There are two factors in the merging problem: document score and database weight. The CORI results merging algorithm used as a baseline assigns a constant value for the database weight for all the queries. The power of our regression model is that it can automatically adjust the database weight, and the weight can be different for different queries.

It is an open question how well the CORI baseline could do if it had a better database weight parameter in the merging formula. The formula can be expressed as shown below:

$$D'' = \frac{D' + k * D' * C_i'}{1 + k} \qquad (17)$$

**Table 6.** Precision in "three search-engine type" environment (INQUERY, language model and SMART). Bold typeface indicates Regression significantly better than CORI.

| Docs | Trec123 | | Trec4_kmeans | |
|---|---|---|---|---|
| Rank | CORI | Regression | CORI | Regression |
| 5 | 0.3240 | *0.3520* (+8.6%) | 0.2560 | **0.3640** (+42.2%) |
| 10 | 0.3020 | *0.3400* (+12.6%) | 0.1920 | **0.2940** (+54.4%) |
| 15 | 0.3013 | *0.3280* (+8.9%) | 0.1787 | **0.2760** (+43.5%) |
| 20 | 0.2960 | **0.3290** (+11.1%) | 0.1770 | **0.2540** (+47.4%) |
| 30 | 0.2947 | *0.3200* (+7.0%) | 0.1560 | **0.2300** (+46.0%) |

**Table 7.** Precision in "one search-engine type" environment (INQUERY). Bold typeface indicates Regression significantly better than CORI.

| Docs | Trec123 | | Trec4_kmeans | |
|---|---|---|---|---|
| Rank | CORI | Regression | CORI | Regression |
| 5 | 0.4480 | *0.4680* (+4.5%) | 0.4240 | *0.4520* (+6.6%) |
| 10 | 0.4220 | *0.4360* (+3.3%) | 0.3860 | *0.4060* (+5.2%) |
| 15 | 0.4053 | *0.4107* (+1.3%) | 0.3400 | *0.3573* (+4.6%) |
| 20 | 0.3820 | *0.3840* (+0.5%) | 0.3140 | *0.3230* (+2.9%) |
| 30 | 0.3627 | *0.3647* (+0.6%) | 0.2753 | **0.2913** (+5.8%) |

The parameter *k* controls the effect of the database score in the results merging formula. A series of experiments using two testbeds was conducted to examine the effects of varying *k*.
When *k* was set to a constant value, average precision at rank r remained relatively constant over a range *k* values (Figures 4 and 5). However, the regression results show that setting k on a query-by-query basis for different databases provided a large improvement over any constant value of k. The ability to tune the results-merging formula on a query-by-query basis is the real power of the regression approach.

## 7. Results: Single Engine

A series of single-engine type experiments was conducted to test the effectiveness of the new approach to results merging. In each case the search engine was INQUERY, chosen in part because it is well-tuned for use with the baseline CORI results merging algorithm. The combination of CORI and INQUERY has been the state-of-the-art in this research area. Two 100-database testbeds with different document distribution patterns were used (Section 4.1). As in the multi-engine experiments, the CORI algorithm was used to rank databases for each query, and the top 10 were considered selected for search.

The experimental results are summarized in Table 7. They show that the regression model performs about the same as the CORI results merging algorithm when the vocabulary is distributed relatively evenly across the testbed (trec123), but that it has a small advantage when the vocabulary distribution is skewed (trec4_kmeans). The CORI results merging algorithm is clearly well-tuned for the INQUERY search engine, but even in a "best case" scenario for the CORI results merging algorithm, the new regression approach to results merging has a small advantage.

## 8. Conclusion

This paper presents a new solution for merging retrieval results returned by different databases in a distributed information retrieval

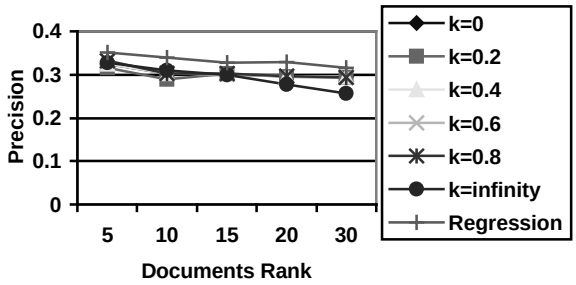


**Figure 4.** How varying the k parameter in the CORI results merging algorithm affects Precision. trec123 testbed.

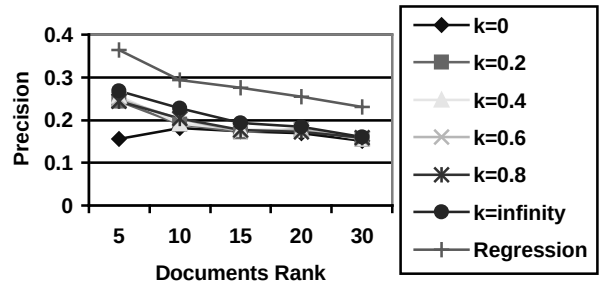


**Figure 5.** How varying the k parameter in the CORI results merging algorithm affects Precision. trec4_kmeans testbed.

environment. A common approach is to view results-merging in isolation from the rest of the distributed retrieval system, which conveniently bounds the problem, but which also limits the information available for solving the problem. By viewing it in the context of a complete distributed retrieval solution, other options become available. In particular, when resource descriptions for database selection are constructed by query-based sampling, a set of documents becomes available that can serve as training data for adaptive merging methods.

In this paper we present a particular adaptive solution based on linear regression and simple models for transforming database-specific scores into approximations of scores that would be obtained if all of the documents were in a single, centralized database. This is a more principled approach to normalizing document scores returned from different databases and different search engines than most of the prior research on this problem. Experiments with two rather different 100-database testbeds and in single-engine type and multiple-engine type configurations demonstrate that this approach is more effective than a well-known and relatively effective heuristic solution.

Our experiments demonstrate that our particular adaptive solution is effective in large part because it can tune the transformation on a query-by-query basis for different databases. There is no single parameter setting of the models we explored that provides similar effectiveness. Although one might expect that query-by-query tuning would be effective, we are not aware of any prior solutions that enable such tuning automatically.

We make no claims that the particular models and approach to regression presented here are the best. They are effective, but one can imagine other models and methods. Exploration of other models, features, and learning methods is an important topic for future research. The problem is challenging, because the amount of training data (the number of "overlap" documents) is often very

small for a single query. However some queries produce large amounts of training data, so it may make sense to use different algorithms depending on the amount of training data available for a particular query.

We began this paper by arguing that in many interesting distributed retrieval environments one cannot rely upon any kind of cooperation from search engines beyond their traditional service of accepting queries and returning documents. Our experiments show that a results merging algorithm that assumes no cooperation has better performance than a well-known heuristic algorithm. Solutions designed for completely cooperative environments will always have an advantage, but our results suggest that the performance difference will begin to shrink, and that partially cooperative solutions such as the CORI results merging algorithm may find limited applicability.

Results-merging in uncooperative environments has been a major open problem in distributed IR research for seven or eight years. It is not yet a solved problem, but the research reported here represents an important step forward in the state-of-the-art.

## ACKNOWLEDGMENTS

This research was supported by NSF grants EIA-9983253 and IIS-0118767. Any opinions, findings, conclusions, or recommendations expressed in this paper are the authors', and do not necessarily reflect those of the sponsor.

## REFERENCES

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

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

[5] N. Fuhr. A decision-theoretic approach to database selection in networked IR. *ACM Transactions on Information Systems, 17(3):229--249*, 1999.

[6] 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.

[7] J. Xu and J. Callan. Effective Retrieval with Distributed Collections. In *Proc. of the 21st Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval,* 1998.

[8] B. Yuwono and D. Lee. Server Ranking for Distributed Text Retrieval Systems on Internet. *In Proc. of the Int. Conf. on Database Systems for Adv. Applications, pages 41-49,* 1997.

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

[10] C. L. Viles and J. C. French. Dissemination of Collection Wide Information in a Distributed Information Retrieval System. In *Proc. of the 18th Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval*, 1995.

[11] S. T. Kirsch. Document retrieval over networks wherein ranking and relevance scores are computed at the client for multiple database documents. U.S. Patent 5,659,732.

[12] 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

[13] 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.

[14] J. H. Lee. Analyses of multiple evidence combination. In *Proc. of the 20th Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval***,** 1997.

[15] R. Manmatha, T. Rath, and F. Feng. Modeling score distributions for combining the outputs of search engines. In *Proc. of the 24th Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval,* 2001.

[16] A. Le Calv , J. Savoy. Database Merging Strategy Based on Logistic Regression. *Information Processing & Management, 36(3),* 2000.

[17] A.L. Powell, J.C. French, J. Callan, M. Connell, and C.L. Viles, The impact of database selection on distributed searching. In *Proc. of the 23rd Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.

[18] J. Xu and W.B. Croft, 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.

[19] 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.

[20] 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.

[21] J. A. Aslam, M. Montague. Models for Metasearch. In *Proc. of the 23rd Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.

[22] P. Ogilvie, J. Callan. Experiments using the Lemur toolkit. *In Proc of 2001 Text REtrieval Conference (TREC 2001).* National Institute of Standards and Technology, special publication.

[23] Ellen Voorhees, Narendra K. Gupta, and Ben Johnson-Laird. Learning Collection Fusion Strategies. In *Proc. of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1995.