Server Ranking for Distributed Text Retrieval Systems on the Internet

Budi Yuwono

Department of Computer and Information Science
The Ohio State University
Columbus, Ohio, U.S.A.

yuwono-b@cis.ohio-state.edu

Dik L. Lee

Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong

dlee@cs.ust.hk

Abstract

Keyword-based search services have become necessary tools for finding information resources on the Internet today. The rapid growth of information on the Internet renders centralized keyword index services incapable of collecting comprehensive resource meta-data in a timely manner. We argue that delegating the task of meta-data collection to local index servers is a more scalable approach. We propose a mechanism for integrating distributed autonomous index servers into a cooperative resource discovery system. Focusing on the retrieval effectiveness of the system, we propose a set of methods, called CVV-based methods, for ranking and selecting index servers with respect to a query, and merging the results returned by the index servers. Through experiments, we evaluate the effectiveness of the CVV-based methods, and compare our server ranking method with methods proposed by other researchers.

Keywords information retrieval, internet databases.

1 Introduction

With the rapid growth of the amount of information on the Internet today, keyword-based search engines are gaining popularity among Internet users. Most online search engines use meta-information or index databases which map keywords to documents or, in a more general term, to information resources. We refer to such systems as index servers.

It is impractical for a single index server to maintain a comprehensive and uptodate index of the entire Internet. The scalability of such an approach is questionable considering the high volatility of data on the Internet. We believe that a distributed architecture consisting of cooperating autonomous index servers is a viable solution to this scalability problem. In this paper we discuss a framework for integrating distributed autonomous index servers into a large virtual index server. This scheme is known as the collection fusion problem [13]. Our scheme is designed to work in an environment where index servers are heterogeneous in terms of implementation and search algorithms employed. This work is part of our continuing research project called the Distributed WWW Index Servers and Search Engine (D-WISE) which is aimed at designing a scalable Internet resource discovery system. D-WISE is a logical extension to WISE [14], our stand-alone WWW index server.[1]

Section 2 discusses the issues involved in integrating the existing index services on the Internet into a cooperative system and our approaches to them. Section 3 provides an overview on the basics of text retrieval methods. Section 4 discusses our index server ranking and result merging methods. In Section 5 we provide a brief survey of relevant work and compare our method with their methods through experiments. Section 6 closes this paper with conclusions.

2 Distributed Index Servers

The major issues involved in building a cooperative distributed index servers are: (1) interoperability among servers, (2) scalability, and (3) effectiveness. In this paper, we focus mainly on the issue of effectiveness which is concerned with how well such a system identifies and locates resources carrying information relevant to user queries. As for the rest of the issues, our approach to the interoperability problem is to use data sets which can be provided by typical index servers. Also, we consider our general approach that relies on autonomous collection servers for maintaining the index data to be a positive step toward solving data scalability problem. Other important issues which are of concern in a commercial setting such as access control and charging are beyond the scope of this paper. In this paper, we assume that all

[1] WISE is accessible at: ⟨http://www.cs.ust.hk/IndexServer/⟩.

Proceedings of the Fifth International Conference on Database Systems for Advanced Applications, Melbourne, Australia, April 1–4, 1997.

networked resources are publicly accessible for free, or index servers carrying resources for restricted accesses and/or accessible for fee do not participate in the the system.

The basic architecture of our cooperative text retrieval system is a two-level architecture with broker servers on top of index servers. In order to alleviate performance bottleneck at the broker servers, broker servers can be replicated, created and removed as needed. In the rest of the paper, we refer to the index servers as *collection servers*, as each of them can be viewed as a database carrying a collection of *documents*, where the documents are descriptive texts representing networked information resources. The discussion on the communication protocol for meta-data exchange between collection servers and broker server is beyond the scope of this paper.

3 Text Retrieval Models

The most widely used text-based information retrieval models are the Boolean and the vector space retrieval models. The Boolean model employs Boolean logic constructs to specify the criterion for a hit. A document either satisfies the criterion (hit) or doesn't (miss).

In the vector space retrieval model, documents and queries are represented by term vectors in a multi-dimensional space. The relevance score of a document to a query is measured by the similarity between the respective vectors, which is computed as the inner product between the vectors. More formally, the similarity between query q and document doc_i:

$$S_{i,q} = \sum_{j=1}^{|V|} W_{q,j} \cdot W_{i,j} \qquad (1)$$

where V is the set of all keywords (vocabulary), $W_{q,j}$ and $W_{i,j}$ is the term weights of term q_j assigned to q and doc_i respectively. In TFxIDF, the most well known algorithm of the vector space model [10], a term weight is a function of the occurrence frequency of a term in the text, or *term frequency* (TF), and the inverse of the number of documents containing the term in the collection, or *inverse document frequency* (IDF). Such a weighting formula gives higher weights to terms which occur frequently in a small set of the documents. Among the most commonly used term weighting formula is the so-called *atc* which uses vector-length normalization to give all texts an equal chance of being retrieved regardless of their lengths. More formally, the weight of term q_j assigned to text (document or query) i is,

$$W_{i,j}(atc) = \frac{(0.5 + 0.5\frac{TF_{i,j}}{TF_{i,max}}) log(\frac{N}{DF_j})}{\sqrt{\sum_{k=1}^{|V|}(0.5 + 0.5\frac{TF_{i,j}}{TF_{i,max}})^2 log^2(\frac{N}{DF_j})}} \qquad (2)$$

where $TF_{i,j}$ is the term frequency of q_j in text i, $TF_{i,max}$ is the maximum term frequency in text i, N is the number of texts in the database, and DF_j is the number of texts containing q_j, or the document frequency of q_j, in the database. The term-frequency component $(0.5 + 0.5TF_{i,j}/TF_{i,max})$ of the above term weighting formula is known as the augmented normalized term frequency [9] which is normalized by the maximum TF in the text and further normalized to lie between 0.5 and 1.0.

According to [8], in comparing the similarity between short text excerpts (e.g., a few sentences long), better results can be obtained using the so called *atn* term weight:

$$W_{i,j}(atn) = (0.5 + 0.5\frac{TF_{i,j}}{TF_{i,max}}) \cdot log(\frac{N}{DF_j})$$

which equals to *atc* without the vector-length normalization component. For retrieval of Internet resource descriptors, the result of our experiment comparing the performance of the formula with vector-length normalization and the formula without the normalization supports this conclusion. This point is elaborated later in this chapter.

In our previous study [14], we observed that most queries submitted by users to index services on the Internet are short so that a query term appears in the query at most once. Therefore, we simplify the similarity formula by using a binary term vector to represent a query, i.e., if a term is present in the query then the corresponding vector component has a value of one, otherwise its value is zero. Modifying Eq. 1, the relevance score of document doc_i with respect to query q:

$$S_{i,q} = \sum_{q_j \in q}(0.5 + 0.5\frac{TF_{i,j}}{TF_{i,max}}) \cdot log(\frac{N}{DF_j}) \qquad (3)$$

The final step of TFxIDF algorithm is to retrieve the top H documents or to retrieve documents having relevance scores greater than or equal to a specified threshold value θ, from the ranking result. The values of H and θ are user-specified. *Top-H* or the *maximum number of hits* is more commonly used than thresholding because the user does not know the range of the document scores before hand. On the other hand, *top H* has a problem with deciding the cut-off point when there are more documents having the same relevance score as the H-th document such that including all of them would exceed the limit H. A reasonable compromise between the two is what we call the *top H+* criterion which accepts all documents having scores greater than or equal to the H-th document in the ranking result. In this paper, we use the *top H+* criterion exclusively.

4 Multiple Collection Search

Conceptually, a set of centrally administered document collections can be treated as a single large document collection, where a query can be broadcast to all collection servers and the results are then merged together. However, such a scheme is a waste of network bandwidth and processing time at the collection servers since not all of the servers contain information relevant to the query. The problem is how to optimize the resource utilization by selecting servers which potentially carry the relevant documents and forwarding the query only to those servers.

4.1 Collection Fusion

Let us now consider the situation where the cost of sending queries and results across the network is negligible. Employing the vector space model in Eq. 3, the only component of the term weight formula which needs to be computed at the global level is the *DF* component, i.e., the document frequency across all collections combined. Therefore, the *DF* data must be kept track of at a centralized server. This can be achieved by requiring every collection server to report any update on its *DF* data to the central server. In our architecture, this centralized server is the broker server.

Given a query, the broker server computes the *IDF* values of the query terms, and then broadcasts the query along with the *IDF*'s to the collection servers[2]. Upon receiving the query, every collection server then performs the TFxIDF ranking algorithm and sends the *top H+* documents, back to the broker server, where the value of *H* can be user-specified. If no documents in a collection server contains any of the query terms, an empty result message is returned. The results from the collection servers are then merged and sorted at the broker server, and the *top H+* documents of the merged result is presented to the user.

4.2 Collection Ranking

In reality, the cost, including the user's time, of broadcasting queries to all collection servers over a wide area network such as the Internet is not negligible. Moreover, if the number of collection servers and the frequency of queries are high, network congestion could ensue, which would further degrade the system's response time. The selection of collection server or servers to which the query should be sent becomes important. Intuitively, the *DF* data available at the broker server as in the architecture mentioned earlier can provide a good indication as to whether a collection carries any documents containing a given query term or not.

[2] It is also possible to use the term weights specified by the user in place of the *IDF*'s.

We propose a method for ranking the collection servers based on their estimated suitability for answering a given query, called the *Cue-Validity Variance* or CVV ranking method. CVV method solely relies on *DF* data. Given a set of collections C, the CVV method assigns a goodness score $G_{i,q}$ to collection $c_i \in C$ with respect to query q as follows:

$$G_{i,q} = \sum_{j=1}^{M} CVV_j \cdot DF_{i,j}$$

where $DF_{i,j}$ is the DF_j of collection c_i, and CVV_j is the variance of CV_j, the *cue validity* of term j, across all collections.

The concept of *cue validity* is used in the same sense as in [4]. The cue validity of term j for c_i, $CV_{i,j}$, measures the degree to which term j distinguishes documents in collection c_i from those in the other collections, and is defined as follows.

$$CV_{i,j} = \frac{\frac{DF_{i,j}}{N_i}}{\frac{DF_{i,j}}{N_i} + \frac{\sum_{k \neq i}^{|C|} DF_{k,j}}{\sum_{k \neq i}^{|C|} N_k}}$$

where N_i is the number of documents in c_i, and $|C|$ is the number of collections in the system. The population variance CVV_j of $CV_{i,j}$ measures the skewness of the distribution of term j across the collections, which can be used to estimate the usefulness of term j for distinguishing one collection from another. The larger the variance is the more useful the term. CVV_j is computed as follows.

$$CVV_j = \frac{\sum_{i=1}^{|C|} (CV_{i,j} - \overline{CV_j})^2}{|C|}$$

where $\overline{CV_j}$ is the population mean of $CV_{i,j}$ over all collections, and is defined as follows.

$$\overline{CV_j} = \frac{\sum_{i=1}^{|C|} CV_{i,j}}{|C|}$$

The goodness score $G_{i,q}$ gives neither a definite indication of how many relevant documents that collection c_i contains, nor, if such documents exist, how relevant they are to query q. $G_{i,q}$ is only an indicator as to where, among the $|C|$ collections, the query terms are concentrated at.

4.3 Query Forwarding

Given the goodness scores of the collections with respect to a given query, the broker server then decides the collection servers to which the query should be sent. One of the following two schemes can be used, (1) single-cast the query to at most one collection server, i.e., the best server, and (2) multicast the query to at most σ collection servers where $\sigma > 1$ for some pre-determined value of σ.

The first scheme significantly simplifies the implementation of the system because it does not require the broker to perform result merging and sorting; in fact, the collection server can directly send the results to the user, bypassing the broker. The disadvantage of the single-cast scheme is that the user may miss some relevant documents at collections other than the selected one.

The second scheme, the multicast scheme, alleviates the above problem by selecting a number of collection servers whose goodness scores are above some threshold value or, simpler yet, by selecting the best σ servers, and forwarding the query to them. As a tradeoff, this scheme is obviously more resource intensive than the first scheme as it produces multiple folds more network traffic and consumes more computing power.

Finally, there are two main requirements for a collection server to join the cooperative system. The first requirement is that the server must be able to compute its own DF data. This requirement should be easy to meet by any keyword-based search engines. The worst case is only to count the number of documents containing each word in a given keyword set. The second requirement is that the server must be able to store the DF data and serve the data to the broker server. A server which runs on a WWW server (HTTPD), as many servers do, can take advantage of the GET, PUT and CGI scripting capabilities of HTTP for handling server-to-server data storing and fetching.

4.4 Result Merging

The multicast scheme requires a mechanism to combine the results returned by the selected servers. We propose a method for merging search results obtained from a set of semi-heterogeneous index servers. By semi-heterogeneous we mean that there is no requirement as to what search and ranking algorithm each of the servers must use except that (**i**) it has to be based on word occurrence, so that the CVV-based methods can be applied, and (**ii**) it has to assign a relevance rank to each document returned. In the case of Boolean search engines, the ordering of the returned documents can be used to imply the relevant ranks of the documents. Since any higher degree of homogeneity requirement is impossible to impose on, we have no choice but to assume that the document ranking algorithms are comparable with one another.

Our result merging method, which is an extension to the CVV collection ranking method, is basically a function that maps *local document ranks* obtained from a collection server into *global document scores* which can then be merged together with document scores from other collection servers. The local document ranks are the ranks of the documents at a collection server resulted from relevance scoring computed locally by the server. The global document scores are the new scores of the documents after being merged.

To better explain the method, suppose a set of collections $C = \{c_1, \ldots, c_{|C|}\}$ with goodness scores $G_{1,q}$, ..., $G_{|C|,q}$ has been selected for query q. Each of the collections returns a set of documents, called result set, ranked by their relevance scores with the document or documents having the highest score ranks first. $r_{i,j}$ denotes the rank of document j in collection c_i. In the final stage, H top documents are to be retrieved. Next, we make the following assumptions:

> **Assumption 1:** *The best document in collection c_i is equally relevant to query* q *(has the same global score) as the best document in collection c_k for any $k \neq i$ and $G_{i,q}, G_{k,q} > 0$.*

Assumption 1 is necessary because, in an environment involving different search algorithms, we can not always compare relevance scores computed at one server from another. This assumption allows a collection containing a few but highly relevant documents to contribute to the final result. To make sure that every collection contains at least one relevant document, only collections with high goodness scores, say, not less that half of the highest score are selected.

> **Assumption 2:** *The distance, in terms of absolute relevance score difference, between two consecutive document ranks in the result set of a collection is inversely proportional to the goodness score of the collection.*

Assumption 2 is an approximation of the distribution of document scores in each collection's result set. This assumption is based on the result of our previous experiment [15] which shows that the relative goodness score of a collection is roughly proportional to the number of documents contributed by the collection to the final result.

Based on the above assumptions, we define the following local document rank to global document score mapping.

$$s_{i,j} = 1 - (r_{i,j} - 1)D_i$$

where $s_{i,j}$ is the global relevance score of the j-th document in collection c_i. Note that the first rank document or documents in a collection has a global relevance score $s_{i,j} = 1$. D_i is the estimated relevance score distance between two consecutive document ranks in collection c_i's result set, and is defined as follows.

$$D_i = \frac{G_{min,q}}{H \cdot G_{i,q}}$$

where $G_{min,q}$ is the smallest goodness score among the $|C|$ collections. Notice that collection $c_k \in C$ whose goodness score is $G_{k,q} = G_{min,q}$ has the largest rank to rank distance, i.e., $D_k = 1/H$. Notice also that if there is no tied rank among documents within each of the collections then the number of documents contributed by collection c_i to the final result is $H \cdot G_{i,q}/\sum_{j=1}^{|C|} G_{j,q}$, i.e., proportional to its goodness score. The resulting global document scores are then sorted in a non-increasing score order, and the best H or *top* $H+$ documents are returned.

This document score mapping is somewhat similar to the document interleaving algorithm proposed in [13] where a rank position is filled by a document selected by rolling a $|C|$-faced dice biased by the number of documents still to be picked from each of the $|C|$ collections. The difference is that our algorithm is a deterministic process which guarantees that each of the selected collections contributes to the first few top ranked documents. Also, our algorithm takes into account the distribution of document ranks within each of the collections.

5 Comparison

5.1 Related Work

Research on keyword-based collection ranking is gaining some attention from the information retrieval community in the last few years. Some researchers have proposed the use of standard subject classification systems such as the U.S. Library of Congress subject numbering [2], Dewey Decimal Coding, and the ACM Computing Review Classification system, to categorize document collections. The main problem with this method is that it is not always easy to find which category or categories a user query falls into, unless a large and ever expanding online concept-categorization table is provided. Even if that is available, keyword distribution data would still be needed to rank the candidate collections. Another scheme, which shares the same problem with the above scheme, is one which uses manually-written short descriptions to represent collections such as in ALIWEB [7]. Still another scheme is one which requires every collection server to report on the first occurrence of a word to the broker server as in WHOIS++[3] [3]. While this scheme does not have the problem faced by the two earlier schemes, it does not provide enough information to select the best server or servers among those carrying a given set of search words.

Voorhees [13] proposed a collection fusion method which can also be used for collection ranking and selection (i.e., by excluding servers which are not likely to carry any relevant documents). Unlike other collection fusion methods presented in this paper, her method employs the so called *isolated merging strategies* where the broker has no access to meta-information on the individual collection servers. In this method, collections are scored based on their past responses to training queries which are the most similar to the current query. This method is very cost efficient in terms of resource utilization and implementation effort. On the other hand, it is not clear how to generate a set of training queries which can anticipate all possible queries for a large number of collections carrying a wide variety of topics. In addition, as more training queries are used, the cost of conducting the training process would increase dramatically because the process involves accesses to all of the collection servers in the system and requires relevance assessment for each query-collection pair.

Collection Centroid

One method which is based on vector-space retrieval model that is often alluded to in many information retrieval literature is the use of centroid vectors to represent clusters of documents. A centroid vector is defined as a vector whose components are the average term weights across all documents belonging to a cluster. In other words, a cluster of documents or a collection is viewed as a large virtual document represented by its centroid vector. Employing the TFxIDF term weighting formula, the j-th component of centroid vector V_i of collection c_i corresponds to the average weight of query term q_j in the collection, which is defined as follows.

$$V_{i,j} = \frac{\sum_{k=1}^{N_i}(0.5 + 0.5\frac{TF_{k,j}}{TF_{k,max}}) \cdot log(\frac{\hat{N}}{\hat{DF_j}})}{N_i}$$

where $\hat{N}$ and $\hat{DF_j}$ are the system-wide total number of documents and the system-wide DF_j. N_i is the number of documents in c_i. $TF_{k,j}$ and $TF_{k,max}$ are as defined in Eq. 2. Using the vector space document scoring as an analogy, the goodness score of c_i with respect to query q, $G_{i,q}$, is computed as follows.

$$G_{i,q} = \sum_{j=1}^{M} V_{i,j}$$

where M is the number of query terms. This method works best when the documents within each collection are relatively homogeneous, i.e., discussing similar or closely related topics. It remains to be seen whether this method can also be used for collection ranking with arbitrary topic distributions.

CORI

One of the most recent work is the Collection Retrieval Inference Network [1] (CORI) which uses

[3] The term server *centroid* in WHOIS++ is not the same as the vector-based collection *centroid* used in this paper.

the TFxIDF document ranking method as an analogy for collection ranking. CORI modifies a variant of TFxIDF document scoring formula by replacing *TF* with *DF*, and *IDF* with *ICF* (*inverse collection frequency*), the inverse of *CF*. CF_j is defined as the number of collections carrying at least one document which contains query term q_j. The goodness score of collection c_i is computed as the combined belief or probability $P(q_j|c_i)$, that c_i contains the relevant documents due to observing terms q_j, for $j = 1, \ldots, M$. $P(q_j|c_i)$ is defined as follows.

$$P(q_j|c_i) = \prod_{j=1}^{M}(d_b + (1 - d_b)T_{i,j}\ I_{i,j})$$

$$T_{i,j} = d_t + (1 - d_t)\frac{log(DF_j + 0.5)}{log(DF_{i,max} + 1.0)}$$

$$I_{i,j} = \frac{log(\frac{N_i+0.5}{CF_j})}{log(N_i + 1.0)}$$

where N_i, DF_j and CF_j are as defined previously. $DF_{i,max}$ is the maximum DF of a term in collection c_i. d_t and d_b are the default values of the term frequency component and the belief component, respectively, when a term occurs in a collection [11]. Both values are set to 0.4 [1]. Finally, the goodness score of collection c_i, $G_{i,q}$, with respect to query q of *M* terms is obtained by combining $P(w_j|c_i)$ for $1 \leq j \leq M$. It is assumed that all of the query terms are of equal importance.

DFxICF

For the sake of completeness, we introduce a method, called the DFxICF method, which is based on the same TFxIDF analogy as CORI and is similar in form as CVV, i.e., taking the sum of *DF* multiplied by *ICF* (the inverse of *CF*) in place of *CV* variance. As with CVV_j in CVV method, ICF_j can be viewed as the collection-discriminating power of term q_j as IDF_j to documents in TFxIDF. In this method, the goodness score $G_{i,q}$ of collection i with respect to query q is computed as follows.

$$G_{i,q} = \sum_{j=1}^{M} DF_{i,j} \cdot log(\frac{|C|}{CF_j})$$

where $|C|$, $DF_{i,j}$ and CF_j are as defined previously.

gGlOSS

Another collection ranking method comparable to CVV is the one used in the generalized Glossary of Servers Server (gGlOSS) [5], a keyword-based distributed database broker system. One of the main differences between gGlOSS and CVV ranking method is that in addition to DF data, gGlOSS also relies on the weight-sum of every term in a collection. The main assumption of gGlOSS is that a term in a collection is distributed evenly among all documents containing the term in a collection. The general form of the gGlOSS collection scoring formula, i.e., the goodness score of collection c_i with respect to query q, is as follows.

$$G_{i,q} = \sum_{j=1}^{M} W_{i,j}$$

where $W_{i,j}$ is the sum of document weights contributed by term q_j in collection c_i. The above generalized formula is obtained by setting the value of the threshold l, which disqualifies term q_j if $W_{i,j}/DF_{i,j}$ (the average document weight contributed by q_j) falls below l [5], to zero. We opted to use the generalized formula because it is not clear how to obtain the optimal value of l which applies to all queries.

The main problem with gGlOSS method is that the document weight $W_{i,j}$ may be computed differently from one collection to another, unless all of the participating collection servers employ exactly the same document scoring formula with global parameters such as a system-wide DF data set. In comparing gGlOSS with other methods, we assume that there is a centralized mechanism which enables the servers to share a global DF data set and the total number of documents. $W_{i,j}$ is computed as the sum of the weights obtained using the modified *atn* formula as follows.

$$W_{i,j} = \sum_{doc_k \in c_i} (0.5 + 0.5\frac{TF_{k,j}}{TF_{k,max}}) \cdot log(\frac{\hat{N}}{\hat{DF}_j})$$

where $\hat{N}$ and $\hat{DF}_j$ are the global N and the global DF_j, respectively. $TF_{k,j}$ and $TF_{k,max}$ are the TF of q_j in document doc_k in c_i and the maximum TF in document doc_k in c_i, respectively.

GlOSS

Considering the current state of the technology of the existing index servers on the Internet today, it is not uncommon to find many search engines which use simple Boolean search methods. Later in section 5.3, we empirically show that our collection ranking method also works well for Boolean retrievals, or at least is comparable to the method used in GlOSS [6]. In GlOSS [6], the goodness score of collection c_i with respect to Boolean query q, $G_{i,q}$, is measured as the probability of finding a document containing all of the query terms in the document. More formally, given query q of M terms, $G_{i,q}$ is defined as:

$$G_{i,q} = N_i \frac{\prod_{j=1}^{M} DF_{i,j}}{N_i^M}$$

5.2 Evaluation

As in [13], the effectiveness of a collection fusion method is typically measured by comparing its result with the result of a single collection run (i.e., retrieval using all collections combined into a single collection). In this paper, we measure the accuracy of a collection ranking method by comparing the collection goodness scores estimated using the method with the actual goodness scores with respect to the same query. We use a vector γ_q of $|C|$ components to represent the actual goodness score of $|C|$ collections with respect to query q, where each component $\gamma_{i,q}$ represents the goodness score of collection c_i and is computed as follows.

First, we identify the *top H+* documents using the TFxIDF algorithm with the document scoring formula as defined by Eq. 3 on single collection runs. It is worth noting that the retrieval recall/precision of the algorithm is irrelevant to this evaluation. We simply treat the relevance scores assigned by the algorithm with respect to a query as the actual relevance scores. Among the $H+$ documents, we then take the sum of the scores of documents belonging to collection c_i as the value of $\gamma_{i,q}$. The accuracy of a collection ranking method is measured as the cosine of the angle between γ_q and G_q, where G_q is the estimated goodness vector of the collections. Each of G_q's components, $G_{i,q}$, represents the goodness score of collection c_i for $1 \leq i \leq |C|$. More formally, the ranking accuracy is defined as:

$$accuracy = \frac{\sum_{i=1}^{|C|} G_{i,q}\ \gamma_{i,q}}{\sqrt{\sum_{i=1}^{|C|} G_{i,q}^2 \sum_{i=1}^{|C|} \gamma_{i,q}^2}}$$

where the value of *accuracy* ranges from 0 to 1.

For Boolean retrieval models, since there are no document scores, the number of documents in collection c_i which satisfy the Boolean query q is taken to be the value of $\gamma_{i,q}$.

To evaluate the effectiveness of our result merging method, we use an effectiveness metric which is the ratio between the sum of scores of the *top H+* documents resulted from using the merging method and the sum of scores of the *top H+* documents resulted from a single collection run. All document scores used in this metric are absolute scores computed using TFxIDF algorithm on the combined collection. Basically, this metric measures the percentage of total relevance score obtained/lost due to the similarity/difference between the document ranks resulted from using the merging algorithm and the ideal document ranks.

5.3 Experiments

We conducted experiments comparing the average accuracies of CVV collection ranking method and five other methods, namely the centroid vector methods, gGlOSS, CORI, DFxICF, and GlOSS. We included the original GlOSS method to see how well it performs in vector-space retrieval. In the experiments, we use the text collections that come with the Smart System,[4] a text retrieval system developed at Cornell University. Four collections, known by the acronyms of their sources, namely, CACM, CISI, CRAN and MED, were used. The queries that come with each collection were used as the test queries. In total, there are 7097 documents and 431 test queries. The standard word-stemming and stop-word removal algorithms, similar to those provided in the Smart system, were applied to the documents and the queries.

We tested each of the ranking methods in 5 different collection setups each of which uses a different *document affinity probability* [12]. Document affinity probability, P_a, is defined as the probability that a document is stored in a collection, called the *home collection*, assigned for documents similar to or related with the document. If P_a is zero then documents are randomly distributed. If P_a is one then the documents in each collection are homogeneous. If $0 < P_a < 1$ then the probability that a document is stored in its home collection is $P_a + \frac{1}{|C|}(1 - P_a)$, where $|C|$ is the number of collections. To simplify the experiments, we assume that all documents belonging to the same Smart collection to be topically related with one another, where the documents which are relevant to a query taken from the collection's query set are all in that collection. Of course, this assumption is not entirely true as there are many cases where some documents not in the collection are also relevant to queries designed for that collection. Nonetheless, the assumption is reasonable enough for systematically creating widely varying test environments or collection setups in which the ranking methods are to evaluate. Based on the assumption, we designated one collection as the home collection for CACM documents, another for CISI documents, and so forth. The document affinity probabilities used in the setups were 1, 0.75, 0.5, 0.25 and 0. Five values of H for the *top H+* selection criterion, namely, 20, 40, 60, 80 and 100, were used. We took the average ranking accuracies among these five values of H.

Fig. 1 shows the average accuracies of each ranking method among the 431 test queries plotted against document affinity probability. The figure shows that the CVV method out performed the other collection ranking methods. As was expected, the original GlOSS did not perform well for vector-space retrieval. The CORI and DFxICF methods, which are based on the same TFxIDF analogy ap-

[4] Smart system is available at: ⟨ftp://ftp.cs.cornell.edu/pub/smart/⟩.

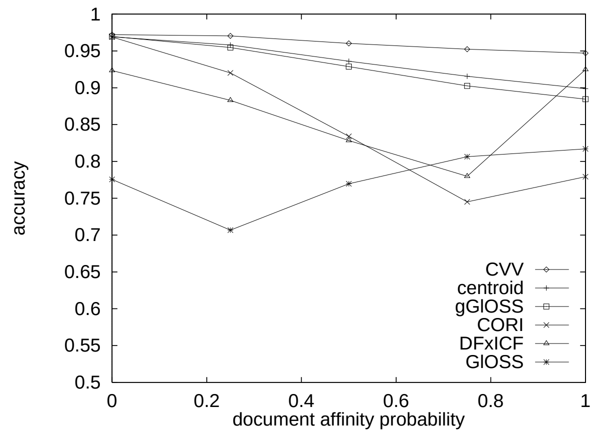

Figure 1: The average accuracies of the collection ranking methods against document affinity probability for the vector-space retrieval experiments.

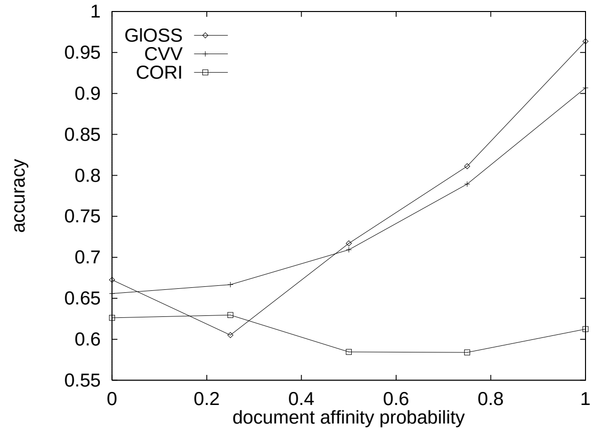

Figure 2: The average accuracies of the collection ranking methods against document affinity probability for the Boolean retrieval experiments.

proach, showed approximately similar average accuracies. All of the ranking methods showed good accuracies at lower document affinity probabilities, i.e., where the relevant documents are distributed evenly across the collections such that, for most of the queries, any collection is as good as another.

Next, we conducted experiments to compare the accuracy of the CVV method with those of the GlOSS and the CORI methods for Boolean retrievals. We included the CORI method because, as with CVV and GlOSS methods, it uses only DF data, which is the only data obtainable from a typical Boolean retrieval system. We used the same collections, test queries, and document affinity probabilities as in the previous experiments. Queries with zero hits are excluded from the computation of the accuracies. Fig. 2 shows the average accuracies of CVV, GlOSS and CORI methods plotted against document affinity probability. As shown in the figure, the performance of CVV method closely followed that of GlOSS.

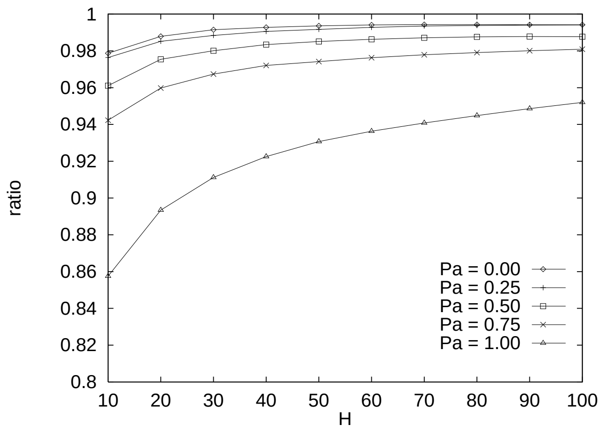

Figure 3: The average ratio between the sum of scores of the top $H+$ documents obtained using the merging method and the sum of scores of $H+$ documents for a single collection run, for the five values of document affinity probability P_a.

To evaluate the effectiveness of our result merging method, we tested the method for each of the five different collection setups mentioned previously. For each collection setup, we used 10 different values of H for the *top* $H+$ document selection criterion, namely, 10, 20, ..., 100. Taking the average ratio between the sum of absolute scores of the *top* $H+$ documents resulted from using the merging algorithm and the sum of absolute scores of the *top* $H+$ documents resulted from a single collection run, we obtained results as shown in Fig. 3. As can be seen in the figure, the merging algorithm is highly effective across different document affinity probabilities.

The results show that the effectiveness ratio decreases as P_a increases. This is because the goodness score differences among collections widen as P_a increases such that collections with very low goodness scores have only irrelevant documents to contribute to the final result. As mentioned in Section 4.4, the merging method is based on the assumption that the first rank document in a collection is as relevant as the first rank document in another collection, granted that the goodness scores of the two collections are close to one another. In this experiment, we included all collections regardless of their goodness scores.

6 Conclusion

We presented a framework for integrating distributed, autonomous, and heterogeneous text retrieval systems into a large index server. Our framework is designed with interoperability, scalability, and effectiveness in mind. Through experiments we showed that our collection ranking and merging methods, the CVV-based methods, worked well for virtually arbitrary document-topics distributions, for both vector space and Boolean retrieval models. Overall, the CVV-based methods require a small amount of meta-data interchange between broker servers and collection servers, simple computations, and a low storage requirement. We believe that these characteristics make the framework relatively easy to incorporate into existing text retrieval systems on the Internet.

References

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

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

[3] P. Deutsch, P. Faltstrom, R. Schoultz and C. Weider. Architecture of the WHOIS++ service. Internet Proposed Standard RFC–1835, 1995.

[4] J. Goldberg. CDM: an approach to learning in text categorization. In *Proceedings of the 7th IEEE International Conference on Tools with Artificial Intelligence*, 1995.

[5] L. Gravano and H. Garcia-Molina. Generalizing GlOSS to vector-space databases and broker hierarchies. Technical Report STAN-CS-TN-95-010, Stanford University, 1995.

[6] L. Gravano, H. Garcia-Molina and A. Tomasic. The effectiveness of GlOSS for the text database discovery problem. In *Proceedings of the 1994 ACM SIGMOD Conference*, 1994.

[7] M. Koster. ALIWEB: archie-like indexing in the Web. In *Proceedings of the First International Conference on the World Wide Web*, 1994.

[8] G. Salton and C. Buckley. Flexible text matching in information retrieval. Technical Report 90–1158, Dept. of Computer Science, Cornell University, 1990.

[9] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, Volume 24, Number 5, pages 513-523, 1988.

[10] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.

[11] H. Turtle and W. Croft. Efficient probabilistic inference for text retrieval. In *Proceedings of RIAO '91: A Conference on Intelligent Text and Image Handling*, pages 644-661, 1991.

[12] C. Viles and J. French. Dissemination of collection wide information in a distributed information retrieval system. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 12-20, 1995.

[13] E. Voorhees. Siemens TREC-4 report: further experiments with database merging. In *Proceedings of the Fourth Text Retrieval Conference (TREC-4)*. National Institute of Standards and Technology Special Publication, 1996.

[14] B. Yuwono, S. Lam, J. Ying and D. Lee. A world wide web resource discovery system. In *Proceedings of the Fourth International World Wide Web Conference*, 1995.

[15] B. Yuwono. A distributed cooperative Internet resource discovery system. Ph.D. Dissertation. In preparation.