Unified Utility Maximization Framework for Resource Selection

Luo Si
Language Technology Inst.
School of Compute Science
Carnegie Mellon University
Pittsburgh, PA 15213
lsi@cs.cmu.edu

Jamie Callan
Language Technology Inst.
School of Compute Science
Carnegie Mellon University
Pittsburgh, PA 15213
callan@cs.cmu.edu

ABSTRACT
This paper presents a unified utility framework for resource selection of distributed text information retrieval. This new framework shows an efficient and effective way to infer the probabilities of relevance of all the documents across the text databases. With the estimated relevance information, resource selection can be made by explicitly optimizing the goals of different applications. Specifically, when used for database recommendation, the selection is optimized for the goal of high-recall (include as many relevant documents as possible in the selected databases); when used for distributed document retrieval, the selection targets the high-precision goal (high precision in the final merged list of documents). This new model provides a more solid framework for distributed information retrieval. Empirical studies show that it is at least as effective as other state-of-the-art algorithms.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval]:**

General Terms
Algorithms

Keywords
distributed information retrieval, resource selection

1. INTRODUCTION
Conventional search engines such as Google or AltaVista use ad-hoc information retrieval solution by assuming all the searchable documents can be copied into a single centralized database for the purpose of indexing. *Distributed information retrieval*, also known as *federated search* [1,4,7,11,14,22] is different from ad-hoc information retrieval as it addresses the cases when documents cannot be acquired and stored in a single database. For example, "*Hidden Web*" contents (also called "invisible" or "deep" Web contents) are information on the Web that cannot be accessed by the conventional search engines.

CIKM '04, November 8--13, 2004, Washington, DC, USA.

Hidden web contents have been estimated to be 2-50 [19] times larger than the contents that can be searched by conventional search engines. Therefore, it is very important to search this type of valuable information.

The architecture of distributed search solution is highly influenced by different environmental characteristics. In a small local area network such as small company environments, the information providers may cooperate to provide corpus statistics or use the same type of search engines. Early distributed information retrieval research focused on this type of *cooperative* environments [1,8]. On the other side, in a wide area network such as very large corporate environments or on the Web there are many types of search engines and it is difficult to assume that all the information providers can cooperate as they are required. Even if they are willing to cooperate in these environments, it may be hard to enforce a single solution for all the information providers or to detect whether information sources provide the correct information as they are required. Many applications fall into the latter type of *uncooperative* environments such as the Mind project [16] which integrates non-cooperating digital libraries or the QProber system [9] which supports browsing and searching of uncooperative hidden Web databases. In this paper, we focus mainly on uncooperative environments that contain multiple types of independent search engines.

There are three important sub-problems in distributed information retrieval. First, information about the contents of each individual database must be acquired (*resource representation*) [1,8,21]. Second, given a query, a set of resources must be selected to do the search (*resource selection*) [5,7,21]. Third, the results retrieved from all the selected resources have to be merged into a single final list before it can be presented to the end user (*retrieval and results merging*) [1,5,20,22].

Many types of solutions exist for distributed information retrieval. Invisible-web.net[1] provides *guided browsing* of hidden Web databases by collecting the resource descriptions of these databases and building hierarchies of classes that group them by similar topics. A *database recommendation system* goes a step further than a browsing system like Invisible-web.net by recommending most relevant information sources to users' queries. It is composed of the resource description and the resource selection components. This solution is useful when the

[1] http://www.invisible-web.net

users want to browse the selected databases by themselves instead of asking the system to retrieve relevant documents automatically. *Distributed document retrieval* is a more sophisticated task. It selects relevant information sources for users' queries as the database recommendation system does. Furthermore, users' queries are forwarded to the corresponding selected databases and the returned individual ranked lists are merged into a single list to present to the users.

The goal of a database recommendation system is to select a small set of resources that contain as many relevant documents as possible, which we call a *high-recall* goal. On the other side, the effectiveness of distributed document retrieval is often measured by the Precision of the final merged document result list, which we call a *high-precision* goal. Prior research indicated that these two goals are related but not identical [4,21]. However, most previous solutions simply use effective resource selection algorithm of database recommendation system for distributed document retrieval system or solve the inconsistency with heuristic methods [1,4,21].

This paper presents a unified utility maximization framework to integrate the resource selection problem of both database recommendation and distributed document retrieval together by treating them as different optimization goals.

First, a *centralized sample database* is built by randomly sampling a small amount of documents from each database with query-based sampling [1]; database size statistics are also estimated [21]. A logistic transformation model is learned off line with a small amount of training queries to map the centralized document scores in the centralized sample database to the corresponding probabilities of relevance.

Second, after a new query is submitted, the query can be used to search the centralized sample database which produces a score for each sampled document. The probability of relevance for each document in the centralized sample database can be estimated by applying the logistic model to each document's score. Then, the probabilities of relevance of all the (mostly unseen) documents among the available databases can be estimated using the probabilities of relevance of the documents in the centralized sample database and the database size estimates.

For the task of resource selection for a database recommendation system, the databases can be ranked by the expected number of relevant documents to meet the high-recall goal. For resource selection for a distributed document retrieval system, databases containing a small number of documents with large probabilities of relevance are favored over databases containing many documents with small probabilities of relevance. This selection criterion meets the high-precision goal of distributed document retrieval application. Furthermore, the Semi-supervised learning (SSL) [20,22] algorithm is applied to merge the returned documents into a final ranked list.

The unified utility framework makes very few assumptions and works in uncooperative environments. Two key features make it a more solid model for distributed information retrieval: i) It formalizes the resource selection problems of different applications as various utility functions, and optimizes the utility functions to achieve the optimal results accordingly; and ii) It shows an effective and efficient way to estimate the probabilities of relevance of all documents across databases. Specifically, the framework builds logistic models on the centralized sample database to transform centralized retrieval scores to the corresponding probabilities of relevance and uses the centralized sample database as the bridge between individual databases and the logistic model. The human effort (relevance judgment) required to train the single centralized logistic model *does not* scale with the number of databases. This is a large advantage over previous research, which required the amount of human effort to be linear with the number of databases [7,15].

The unified utility framework is not only more theoretically solid but also very effective. Empirical studies show the new model to be at least as accurate as the state-of-the-art algorithms in a variety of configurations.

The next section discusses related work. Section 3 describes the new unified utility maximization model. Section 4 explains our experimental methodology. Sections 5 and 6 present our experimental results for resource selection and document retrieval. Section 7 concludes.

2. PRIOR RESEARCH

There has been considerable research on all the sub-problems of distributed information retrieval. We survey the most related work in this section.

The first problem of distributed information retrieval is resource representation. The STARTS protocol is one solution for acquiring resource descriptions in cooperative environments [8]. However, in uncooperative environments, even the databases are willing to share their information, it is not easy to judge whether the information they provide is accurate or not. Furthermore, it is not easy to coordinate the databases to provide resource representations that are compatible with each other. Thus, in uncooperative environments, one common choice is query-based sampling, which randomly generates and sends queries to individual search engines and retrieves some documents to build the descriptions. As the sampled documents are selected by random queries, query-based sampling is not easily fooled by any adversarial spammer that is interested to attract more traffic. Experiments have shown that rather accurate resource descriptions can be built by sending about 80 queries and downloading about 300 documents [1].

Many resource selection algorithms such as gGlOSS/vGlOSS [8] and CORI [1] have been proposed in the last decade. The CORI algorithm represents each database by its terms, the document frequencies and a small number of corpus statistics (details in [1]). As prior research on different datasets has shown the CORI algorithm to be the most stable and effective of the three algorithms [1,17,18], we use it as a baseline algorithm in this work. The relevant document distribution estimation (ReDDE [21]) resource selection algorithm is a recent algorithm that tries to estimate the distribution of relevant documents across the available databases and ranks the databases accordingly. Although the ReDDE algorithm has been shown to be effective, it relies on heuristic constants that are set empirically [21].

The last step of the document retrieval sub-problem is results merging, which is the process of transforming database-specific document scores into comparable database-independent

document scores. The semi supervised learning (SSL) [20,22] result merging algorithm uses the documents acquired by query-based sampling as training data and linear regression to learn the database-specific, query-specific merging models. These linear models are used to convert the database-specific document scores into the approximated centralized document scores. The SSL algorithm has been shown to be effective [22]. It serves as an important component of our unified utility maximization framework (Section 3).

In order to achieve accurate document retrieval results, many previous methods simply use resource selection algorithms that are effective of database recommendation system. But as pointed out above, a good resource selection algorithm optimized for high-recall may not work well for document retrieval, which targets the high-precision goal. This type of inconsistency has been observed in previous research [4,21]. The research in [21] tried to solve the problem with a heuristic method.

The research most similar to what we propose here is the decision-theoretic framework (DTF) [7,15]. This framework computes a selection that minimizes the overall costs (e.g., retrieval quality, time) of document retrieval system and several methods [15] have been proposed to estimate the retrieval quality. However, two points distinguish our research from the DTF model. First, the DTF is a framework designed specifically for document retrieval, but our new model integrates two distinct applications with different requirements (database recommendation and distributed document retrieval) into the same unified framework. Second, the DTF builds a model for each database to calculate the probabilities of relevance. This requires human relevance judgments for the results retrieved from each database. In contrast, our approach only builds one logistic model for the centralized sample database. The centralized sample database can serve as a bridge to connect the individual databases with the centralized logistic model, thus the probabilities of relevance of documents in different databases can be estimated. This strategy can save large amount of human judgment effort and is a big advantage of the unified utility maximization framework over the DTF especially when there are a large number of databases.

3. UNIFIED UTILITY MAXIMIZATION FRAMEWORK

The Unified Utility Maximization (UUM) framework is based on estimating the probabilities of relevance of the (mostly unseen) documents available in the distributed search environment. In this section we describe how the probabilities of relevance are estimated and how they are used by the Unified Utility Maximization model. We also describe how the model can be optimized for the high-recall goal of a database recommendation system and the high-precision goal of a distributed document retrieval system.

3.1 Estimating Probabilities of Relevance

As pointed out above, the purpose of resource selection is high-recall and the purpose of document retrieval is high-precision. In order to meet these diverse goals, the key issue is to estimate the probabilities of relevance of the documents in various databases. This is a difficult problem because we can only observe a sample of the contents of each database using query-based sampling. Our strategy is to make full use of all the available information to calculate the probability estimates.

3.1.1 Learning Probabilities of Relevance

In the resource description step, the centralized sample database is built by query-based sampling and the database sizes are estimated using the sample-resample method [21]. At the same time, an effective retrieval algorithm (Inquery [2]) is applied on the centralized sample database with a small number (e.g., 50) of training queries. For each training query, the CORI resource selection algorithm [1] is applied to select some number (e.g., 10) of databases and retrieve 50 document ids from each database. The SSL results merging algorithm [20,22] is used to merge the results. Then, we can download the top 50 documents in the final merged list and calculate their corresponding centralized scores using Inquery and the corpus statistics of the centralized sample database. The centralized scores are further normalized (divided by the maximum centralized score for each query), as this method has been suggested to improve estimation accuracy in previous research [15]. Human judgment is acquired for those documents and a logistic model is built to transform the normalized centralized document scores to probabilities of relevance as follows:

$$R(d) = P(rel \mid d) = \frac{\exp(a_c + b_c \bar{S}_c(d))}{1 + \exp(a_c + b_c \bar{S}_c(d))} \quad (1)$$

where $\bar{S}_c(d)$ is the normalized centralized document score and a_c and b_c are the two parameters of the logistic model. These two parameters are estimated by maximizing the probabilities of relevance of the training queries. The logistic model provides us the tool to calculate the probabilities of relevance from centralized document scores.

3.1.2 Estimating Centralized Document Scores

When the user submits a new query, the centralized document scores of the documents in the centralized sample database are calculated. However, in order to calculate the probabilities of relevance, we need to estimate centralized document scores for *all* documents across the databases instead of only the sampled documents. This goal is accomplished using: the centralized scores of the documents in the centralized sample database, and the database size statistics.

We define the *database scale factor* for the i^{th} database as the ratio of the estimated database size and the number of documents sampled from this database as follows:

$$SF_{db_i} = \frac{\hat{N}_{db_i}}{N_{db_i\_samp}} \quad (2)$$

where $\hat{N}_{db_i}$ is the estimated database size and $N_{db_i\_samp}$ is the number of documents from the i^{th} database in the centralized sample database. The intuition behind the database scale factor is that, for a database whose scale factor is 50, if one document from this database in the centralized sample database has a centralized document score of 0.5, we may guess that there are about 50 documents in that database which have scores of about 0.5. Actually, we can apply a finer non-parametric linear interpolation method to estimate the centralized document score curve for each database. Formally, we rank all the sampled documents from the i^{th} database by their centralized document

scores to get the sampled centralized document score list $\{S_c(ds_{i1}), S_c(ds_{i2}), S_c(ds_{i3}),.....\}$ for the i^{th} database; we assume that if we could calculate the centralized document scores for all the documents in this database and get the complete centralized document score list, the top document in the sampled list would have rank $SF_{dbi}/2$, the second document in the sampled list would rank $SF_{dbi}3/2$, and so on. Therefore, the data points of sampled documents in the complete list are: $\{(SF_{dbi}/2, S_c(ds_{i1})), (SF_{dbi}3/2, S_c(ds_{i2})), (SF_{dbi}5/2, S_c(ds_{i3})),.....\}$. Piecewise linear interpolation is applied to estimate the centralized document score curve, as illustrated in Figure 1. The complete centralized document score list can be estimated by calculating the values of different ranks on the centralized document curve as: $\hat{S}_c(\hat{d}_{ij}),\ j \in [1, \hat{N}_{db_i}]$.

It can be seen from Figure 1 that more sample data points produce more accurate estimates of the centralized document score curves. However, for databases with large database scale ratios, this kind of linear interpolation may be rather inaccurate, especially for the top ranked (e.g., $[1, SF_{dbi}/2]$) documents. Therefore, an alternative solution is proposed to estimate the centralized document scores of the top ranked documents for databases with large scale ratios (e.g., larger than 100). Specifically, a logistic model is built for each of these databases. The logistic model is used to estimate the centralized document score of the top 1 document in the corresponding database by using the two sampled documents from that database with highest centralized scores.

$$\hat{S}_c(\hat{d}_{i1}) = \frac{\exp(\alpha_{i0} + \alpha_{i1} S_c(ds_{i1}) + \alpha_{i2} S_c(ds_{i2}))}{1 + \exp(\alpha_{i0} + \alpha_{i1} S_c(ds_{i1}) + \alpha_{i2} S_c(ds_{i2}))} \quad (3)$$

α_{i0}, α_{i1} and α_{i2} are the parameters of the logistic model. For each training query, the top retrieved document of each database is downloaded and the corresponding centralized document score is calculated. Together with the scores of the top two sampled documents, these parameters can be estimated.

After the centralized score of the top document is estimated, an exponential function is fitted for the top part ($[1, SF_{dbi}/2]$) of the centralized document score curve as:

$$\hat{S}_c(\hat{d}_{ij}) = \exp(\beta_{i0} + \beta_{i1} * j) \qquad j \in [1, SF_{db_i}/2] \quad (4)$$

$$\beta_{i0} = \log(\hat{S}_c(\hat{d}_{i1})) - \beta_{i1} \quad (5)$$

$$\beta_{i1} = \frac{(\log(S_c(ds_{i1}) - \log(\hat{S}_c(\hat{d}_{i1}))}{(SF_{db_i}/2 - 1)} \quad (6)$$

The two parameters β_{i0} and β_{i1} are fitted to make sure the exponential function passes through the two points $(1, \hat{S}_c(\hat{d}_{i1}))$ and $(SF_{dbi}/2, S_c(ds_{i1}))$. The exponential function is only used to adjust the top part of the centralized document score curve and the lower part of the curve is still fitted with the linear interpolation method described above. The adjustment by fitting exponential function of the top ranked documents has been shown empirically to produce more accurate results.

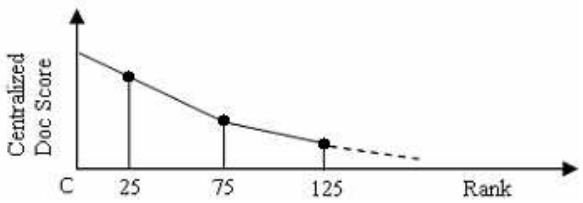

Figure 1. Linear interpolation construction of the complete centralized document score list (database scale factor is 50).

From the centralized document score curves, we can estimate the complete centralized document score lists accordingly for all the available databases. After the estimated centralized document scores are normalized, the complete lists of probabilities of relevance can be constructed out of the complete centralized document score lists by Equation 1. Formally for the i^{th} database, the complete list of probabilities of relevance is: $\hat{R}(\hat{d}_{ij}),\ j \in [1, \hat{N}_{db_i}]$.

3.2 The Unified Utility Maximization Model

In this section, we formally define the new unified utility maximization model, which optimizes the resource selection problems for two goals of high-recall (database recommendation) and high-precision (distributed document retrieval) in the same framework.

In the task of database recommendation, the system needs to decide how to rank databases. In the task of document retrieval, the system not only needs to select the databases but also needs to decide how many documents to retrieve from each selected database. We generalize the database recommendation selection process, which implicitly recommends all documents in every selected database, as a special case of the selection decision for the document retrieval task. Formally, we denote d_i as the number of documents we would like to retrieve from the i^{th} database and $\vec{d} = \{d_1, d_2, ...\}$ as a selection action for all the databases.

The database selection decision is made based on the complete lists of probabilities of relevance for all the databases. The complete lists of probabilities of relevance are inferred from all the available information specifically $\vec{R_s}$, which stands for the resource descriptions acquired by query-based sampling and the database size estimates acquired by sample-resample; $\vec{S_c}$ stands for the centralized document scores of the documents in the centralized sample database.

If the method of estimating centralized document scores and probabilities of relevance in Section 3.1 is acceptable, then the most probable complete lists of probabilities of relevance can be derived and we denote them as $\theta^* = \{(\hat{R}(\hat{d}_{1j}), j \in [1, \hat{N}_{db_1}]), (\hat{R}(\hat{d}_{2j}), j \in [1, \hat{N}_{db_2}]), ...\}$. Random vector θ denotes an arbitrary set of complete lists of probabilities of relevance and $P(\theta \mid \vec{R_s}, \vec{S_c})$ as the probability of generating this set of lists. Finally, to each selection action $\vec{d}$ and a set of complete lists of

probabilities of relevance θ, we associate a utility function $U(\theta, \vec{d})$ which indicates the benefit from making the $\vec{d}$ selection when the true complete lists of probabilities of relevance are θ.

Therefore, the selection decision defined by the Bayesian framework is:

$$\vec{d}^{\,*} = \arg\max_{\vec{d}} \int_{\theta} U(\vec{d}, \theta)\, P(\theta \mid \vec{R_s}, \vec{S_c})\, d\theta \tag{7}$$

One common approach to simplify the computation in the Bayesian framework is to only calculate the utility function at the most probable parameter values instead of calculating the whole expectation. In other words, we only need to calculate $U(\theta^*, \vec{d})$ and Equation 7 is simplified as follows:

$$\vec{d}^{\,*} = \arg\max_{\vec{d}} U(\vec{d}, \theta^*) \tag{8}$$

This equation serves as the basic model for both the database recommendation system and the document retrieval system.

3.3 Resource Selection for High-Recall

High-recall is the goal of the resource selection algorithm in federated search tasks such as database recommendation. The goal is to select a small set of resources (e.g., less than N_{sdb} databases) that contain as many relevant documents as possible, which can be formally defined as:

$$U(\vec{d}, \theta^*) = \sum_i I(d_i) \sum_{j=1}^{\hat{N}_{db_i}} \mathrm{R}(\hat{d}_{ij}) \tag{9}$$

$I(d_i)$ is the indicator function, which is 1 when the i^{th} database is selected and 0 otherwise. Plug this equation into the basic model in Equation 8 and associate the selected database number constraint to obtain the following:

$$\begin{aligned} &\vec{d}^{\,*} = \arg\max_{\vec{d}} \sum_i I(d_i) \sum_{j=1}^{\hat{N}_{db_i}} \mathrm{R}(\hat{d}_{ij}) \\ &Subject\ \ to: \sum_i I(d_i) = N_{sdb} \end{aligned} \tag{10}$$

The solution of this optimization problem is very simple. We can calculate the expected number of relevant documents for each database as follows:

$$\hat{N}_{Rd_i} = \sum_{j=1}^{\hat{N}_{db_i}} \mathrm{R}(\hat{d}_{ij}) \tag{11}$$

The N_{sdb} databases with the largest expected number of relevant documents can be selected to meet the high-recall goal. We call this the UUM/HR algorithm (Unified Utility Maximization for High-Recall).

3.4 Resource Selection for High-Precision

High-Precision is the goal of resource selection algorithm in federated search tasks such as distributed document retrieval. It is measured by the Precision at the top part of the final merged document list. This high-precision criterion is realized by the following utility function, which measures the Precision of retrieved documents from the selected databases.

$$U(\vec{d}, \theta^*) = \sum_i I(d_i) \sum_{j=1}^{d_i} \mathrm{R}(\hat{d}_{ij}) \tag{12}$$

Note that the key difference between Equation 12 and Equation 9 is that Equation 9 sums up the probabilities of relevance of all the documents in a database, while Equation 12 only considers a much smaller part of the ranking. Specifically, we can calculate the optimal selection decision by:

$$\vec{d}^{\,*} = \arg\max_{\vec{d}} \sum_i I(d_i) \sum_{j=1}^{d_i} \mathrm{R}(\hat{d}_{ij}) \tag{13}$$

Different kinds of constraints caused by different characteristics of the document retrieval tasks can be associated with the above optimization problem. The most common one is to select a fixed number (N_{sdb}) of databases and retrieve a fixed number (N_{rdoc}) of documents from each selected database, formally defined as:

$$\begin{aligned} &\vec{d}^{\,*} = \arg\max_{\vec{d}} \sum_i I(d_i) \sum_{j=1}^{d_i} \mathrm{R}(\hat{d}_{ij}) \\ &Subject\ \ to: \sum_i I(d_i) = N_{sdb} \\ &\qquad d_i = N_{rdoc}, if\ d_i \neq 0 \end{aligned} \tag{14}$$

This optimization problem can be solved easily by calculating the number of expected relevant documents in the top part of the each database's complete list of probabilities of relevance:

$$\hat{N}_{Top\_Rd_i} = \sum_{j=1}^{N_{rdoc}} \mathrm{R}(\hat{d}_{ij}) \tag{15}$$

Then the databases can be ranked by these values and selected. We call this the UUM/HP-FL algorithm (Unified Utility Maximization for High-Precision with Fixed Length document rankings from each selected database).

A more complex situation is to vary the number of retrieved documents from each selected database. More specifically, we allow different selected databases to return different numbers of documents. For simplification, the result list lengths are required to be multiples of a baseline number 10. (This value can also be varied, but for simplification it is set to 10 in this paper.) This restriction is set to simulate the behavior of commercial search engines on the Web. (Search engines such as Google and AltaVista return only 10 or 20 document ids for every result page.) This procedure saves the computation time of calculating optimal database selection by allowing the step of dynamic programming to be 10 instead of 1 (more detail is discussed latterly). For further simplification, we restrict to select at most 100 documents from each database (d_i<=100) Then, the selection optimization problem is formalized as follows:

$$\begin{aligned} &\vec{d}^{\,*} = \arg\max_{\vec{d}} \sum_i I(d_i) \sum_{j=1}^{d_i} \mathrm{R}(\hat{d}_{ij}) \\ &Subject\ \ to: \sum_i I(d_i) = N_{sdb} \\ &\qquad \sum_i d_i = N_{Total\_rdoc} \\ &\qquad d_i = 10 * k, \quad k \in [0, 1, 2, .., 10] \end{aligned} \tag{16}$$

N_{Total_rdoc} is the total number of documents to be retrieved.

Unfortunately, there is no simple solution for this optimization problem as there are for Equations 10 and 14. However, a

Input: Complete lists of probabilities of relevance for all the |DB| databases.
Output: Optimal selection solution for Equation 16.

i) Create the three-dimensional array:
Sel (1..|DB|, 1..$N_{Total\_rdoc/10}$, 1..N_{sdb})
Each Sel (x, y, z) is associated with a selection decision $\vec{d}_{xyz}$, which represents the best selection decision in the condition: only databases from number 1 to number x are considered for selection; totally y*10 documents will be retrieved; only z databases are selected out of the x database candidates. And Sel (x, y, z) is the corresponding utility value by choosing the best selection.

ii) Initialize Sel (1, 1..N_{Total_rdoc}/10, 1..N_{sdb}) with only the estimated relevance information of the 1st database.

iii) Iterate the current database candidate i from 2 to |DB|
For each entry Sel (i, y, z):
Find k such that:

$$k^* = \arg\max_k \left(Sel(i-1, y-k, z-1) + \sum_{j \le 10*k} \hat{R}(d_{ij})\right)$$

$$subject\ to: 1 \le k \le \min(y, 10)$$

$$If\ \left(Sel(i-1, y-k^*, z-1) + \sum_{j \le 10*k^*} \hat{R}(d_{ij})\right) > Sel(i-1, y, z)$$

This means that we should retrieve $10 * k^*$ documents from the i[th] database, otherwise we should not select this database and the previous best solution Sel (i-1, y, z) should be kept.
Then set the value of $\vec{d}_{iyz}$ and Sel (i, y, z) accordingly.

iv) The best selection solution is given by $\vec{d}_{|DB| N_{Total\_rdoc/10} N_{sdb}}$ and the corresponding utility value is Sel (|DB|, $N_{Total\_rdoc/10}$, N_{sdb}).

Figure 2. The dynamic programming optimization procedure for Equation 16.

dynamic programming algorithm can be applied to calculate the optimal solution. The basic steps of this dynamic programming method are described in Figure 2. As this algorithm allows retrieving result lists of varying lengths from each selected database, it is called UUM/HP-VL algorithm.

After the selection decisions are made, the selected databases are searched and the corresponding document ids are retrieved from each database. The final step of document retrieval is to merge the returned results into a single ranked list with the semi-supervised learning algorithm. It was pointed out before that the SSL algorithm maps the database-specific scores into the centralized document scores and builds the final ranked list accordingly, which is consistent with all our selection procedures where documents with higher probabilities of relevance (thus higher centralized document scores) are selected.

4. EXPERIMENTAL METHODOLOGY

4.1 Testbeds

It is desirable to evaluate distributed information retrieval algorithms with testbeds that closely simulate the real world applications.

The TREC Web collections WT2g or WT10g [4,13] provide a way to partition documents by different Web servers. In this way, a large number (O(1000)) of databases with rather diverse contents could be created, which may make this testbed a good candidate to simulate the operational environments such as open domain hidden Web. However, two weakness of this testbed are: i) Each database contains only a small amount of document (259 documents by average for WT2g) [4]; and ii) The contents of WT2g or WT10g are arbitrarily crawled from the Web. It is not likely for a hidden Web database to provide personal homepages or web pages indicating that the pages are under construction and there is no useful information at all. These types of web pages are contained in the WT2g/WT10g datasets. Therefore, the noisy Web data is not similar with that of high-quality hidden Web database contents, which are usually organized by domain experts.

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

Table2: Query set statistics.

Name	TREC Topic Set	TREC Topic Field	Average Length (Words)
Trec123	51-150	Title	3.1

Another choice is the TREC news/government data [1,15,17, 18,21]. TREC news/government data is concentrated on relatively narrow topics. Compared with TREC Web data: i) The news/government documents are much more similar to the contents provided by a topic-oriented database than an arbitrary web page, ii) A database in this testbed is larger than that of TREC Web data. By average a database contains thousands of documents, which is more realistic than a database of TREC Web data with about 250 documents. As the contents and sizes of the databases in the TREC news/government testbed are more similar with that of a topic-oriented database, it is a good candidate to simulate the distributed information retrieval environments of large organizations (companies) or domain-specific hidden Web sites, such as West that provides access to legal, financial and news text databases [3]. As most current distributed information retrieval systems are developed for the environments of large organizations (companies) or domain-specific hidden Web other than open domain hidden Web, TREC news/government testbed was chosen in this work.

Trec123-100col-bysource testbed is one of the most used TREC news/government testbed [1,15,17,21]. It was chosen in this work. Three testbeds in [21] with skewed database size distributions and different types of relevant document distributions were also used to give more thorough simulation for real environments.

Trec123-100col-bysource: 100 databases were created from TREC CDs 1, 2 and 3. They were organized by source and publication date [1]. The sizes of the databases are not skewed. Details are in Table 1.

Three testbeds built in [21] were based on the trec123-100col-bysource testbed. Each testbed contains many “small” databases and two large databases created by merging about 10-20 small databases together.

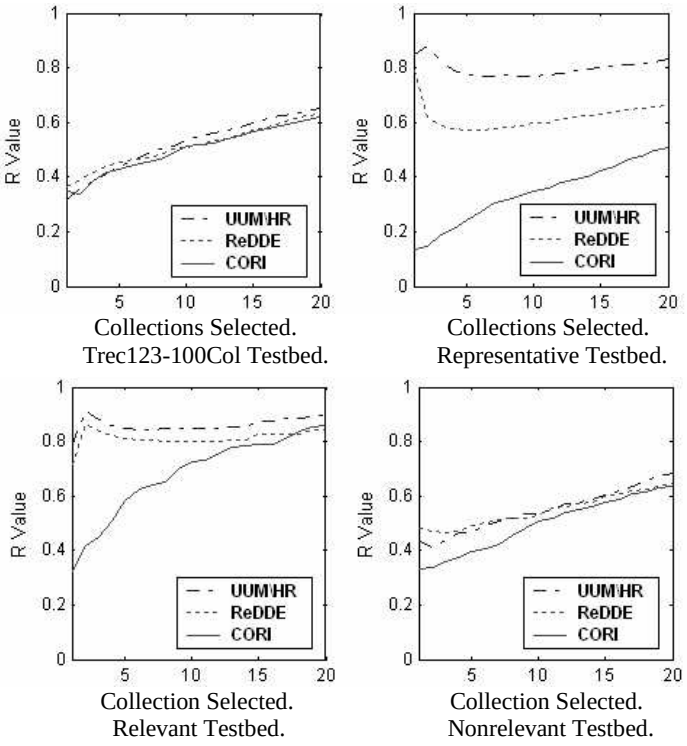

Figure 3. Resource selection experiments on the four testbeds.

Trec123-2ldb-60col ("representative"): The databases in the trec123-100col-bysource were sorted with alphabetical order. Two large databases were created by merging 20 small databases with the round-robin method. Thus, the two large databases have more relevant documents due to their large sizes, even though the densities of relevant documents are roughly the same as the small databases.

Trec123-AP-WSJ-60col ("relevant"): The 24 Associated Press collections and the 16 Wall Street Journal collections in the trec123-100col-bysource testbed were collapsed into two large databases APall and WSJall. The other 60 collections were left unchanged. The APall and WSJall databases have higher densities of documents relevant to TREC queries than the small databases. Thus, the two large databases have many more relevant documents than the small databases.

Trec123-FR-DOE-81col ("nonrelevant"): The 13 Federal Register collections and the 6 Department of Energy collections in the trec123-100col-bysource testbed were collapsed into two large databases FRall and DOEall. The other 80 collections were left unchanged. The FRall and DOEall databases have lower densities of documents relevant to TREC queries than the small databases, even though they are much larger.

100 queries were created from the title fields of TREC topics 51-150. The queries 101-150 were used as training queries and the queries 51-100 were used as test queries (details in Table 2).

4.2 Search Engines

In the uncooperative distributed information retrieval environments of large organizations (companies) or domain-specific hidden Web, different databases may use different types of search engine. To simulate the multiple type-engine environment, three different types of search engines were used in the experiments: INQUERY [2], a unigram statistical language model with linear smoothing [12,20] and a TFIDF retrieval algorithm with "ltc" weight [12,20]. All these algorithms were implemented with the Lemur toolkit [12]. These three kinds of search engines were assigned to the databases among the four testbeds in a round-robin manner.

5. RESULTS: RESOURCE SELECTION OF DATABASE RECOMMENDATION

All four testbeds described in Section 4 were used in the experiments to evaluate the resource selection effectiveness of the database recommendation system.

The resource descriptions were created using query-based sampling. About 80 queries were sent to each database to download 300 unique documents. The database size statistics were estimated by the sample-resample method [21]. Fifty queries (101-150) were used as training queries to build the relevant logistic model and to fit the exponential functions of the centralized document score curves for large ratio databases (details in Section 3.1). Another 50 queries (51-100) were used as test data.

Resource selection algorithms of database recommendation systems are typically compared using the recall metric R_n [1,17,18,21]. Let B denote 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 i^{th} ranked database of B or E. Then R_n is defined as follows:

$$R_k = \frac{\sum_{i=1}^{k} E_i}{\sum_{i=1}^{k} B_i} \qquad (17)$$

Usually the goal is to search only a few databases, so our figures only show results for selecting up to 20 databases.

The experiments summarized in Figure 3 compared the effectiveness of the three resource selection algorithms, namely the CORI, ReDDE and UUM/HR. The UUM/HR algorithm is described in Section 3.3. It can be seen from Figure 3 that the ReDDE and UUM/HR algorithms are more effective (on the representative, relevant and nonrelevant testbeds) or as good as (on the Trec123-100Col testbed) the CORI resource selection algorithm. The UUM/HR algorithm is more effective than the ReDDE algorithm on the representative and relevant testbeds and is about the same as the ReDDE algorithm on the Trec123-100Col and the nonrelevant testbeds. This suggests that the UUM/HR algorithm is more robust than the ReDDE algorithm. It can be noted that when selecting only a few databases on the Trec123-100Col or the nonrelevant testbeds, the ReDEE algorithm has a small advantage over the UUM/HR algorithm. We attribute this to two causes: i) The ReDDE algorithm was tuned on the Trec123-100Col testbed; and ii) Although the difference is small, this may suggest that our logistic model of estimating probabilities of relevance is not accurate enough. More training data or a more sophisticated model may help to solve this minor puzzle.

Table 3. Precision on the trec123-100col-bysource testbed when 3 databases were selected. (The first baseline is CORI; the second baseline for UUM/HP methods is UUM/HR.)

Precision at Doc Rank	CORI	ReDDE	UUM/HR	UUM/HP-FL	UUM/HP-VL
5 docs	0.3640	0.3480 (-4.4%)	0.3960 (+8.8%)	0.4680 (+28.6%)(+18.1%)	0.4640 (+27.5%)(+17.2%)
10 docs	0.3360	0.3200 (-4.8%)	0.3520 (+4.8%)	0.4240 (+26.2%)(+20.5%)	0.4220 (+25.6%)(+19.9%)
15 docs	0.3253	0.3187 (-2.0%)	0.3347 (+2.9%)	0.3973 (+22.2%)(+15.7%)	0.3920 (+20.5%)(+17.1%)
20 docs	0.3140	0.2980 (-5.1%)	0.3270 (+4.1%)	0.3720 (+18.5%)(+13.8%)	0.3700 (+17.8%)(+13.2%)
30 docs	0.2780	0.2660 (-4.3%)	0.2973 (+6.9%)	0.3413 (+22.8%)(+14.8%)	0.3400 (+22.3%)(+14.4%)

Table 4. Precision on the trec123-100col-bysource testbed when 5 databases were selected. (The first baseline is CORI; the second baseline for UUM/HP methods is UUM/HR.)

Precision at Doc Rank	CORI	ReDDE	UUM/HR	UUM/HP-FL	UUM/HP-VL
5 docs	0.4000	0.3920 (-2.0%)	0.4280 (+7.0%)	0.4680 (+17.0%)(+9.4%)	0.4600 (+15.0%)(+7.5%)
10 docs	0.3800	0.3760 (-1.1%)	0.3800 (+0.0%)	0.4180 (+10.0%)(+10.0%)	0.4320 (+13.7%)(+13.7%)
15 docs	0.3560	0.3560 (+0.0%)	0.3720 (+4.5%)	0.3920 (+10.1%)(+5.4%)	0.4080 (+14.6%)(+9.7%)
20 docs	0.3430	0.3390 (-1.2%)	0.3550 (+3.5%)	0.3710 (+8.2%)(+4.5%)	0.3830 (+11.7%)(+7.9%)
30 docs	0.3240	0.3140 (-3.1%)	0.3313 (+2.3%)	0.3500 (+8.0%)(+5.6%)	0.3487 (+7.6%)(+5.3%)

Table 5. Precision on the representative testbed when 3 databases were selected. (The first baseline is CORI; the second baseline for UUM/HP methods is UUM/HR.)

Precision at Doc Rank	CORI	ReDDE	UUM/HR	UUM/HP-FL	UUM/HP-VL
5 docs	0.3720	0.4080 (+9.7%)	0.4640 (+24.7%)	0.4600 (+23.7%)(-0.9%)	0.5000 (+34.4%)(+7.8%)
10 docs	0.3400	0.4060 (+19.4%)	0.4600 (+35.3%)	0.4540 (+33.5%)(-1.3%)	0.4640 (+36.5%)(+0.9%)
15 docs	0.3120	0.3880 (+24.4%)	0.4320 (+38.5%)	0.4240 (+35.9%)(-1.9%)	0.4413 (+41.4%)(+2.2)
20 docs	0.3000	0.3750 (+25.0%)	0.4080 (+36.0%)	0.4040 (+34.7%)(-1.0%)	0.4240 (+41.3%)(+4.0%)
30 docs	0.2533	0.3440 (+35.8%)	0.3847 (+51.9%)	0.3747 (+47.9%)(-2.6%)	0.3887 (+53.5%)(+1.0%)

Table 6. Precision on the representative testbed when 5 databases were selected. (The first baseline is CORI; the second baseline for UUM/HP methods is UUM/HR.)

Precision at Doc Rank	CORI	ReDDE	UUM/HR	UUM/HP-FL	UUM/HP-VL
5 docs	0.3960	0.4080 (+3.0%)	0.4560 (+15.2%)	0.4280 (+8.1%)(-6.1%)	0.4520 (+14.1%)(-0.9%)
10 docs	0.3880	0.4060 (+4.6%)	0.4280 (+10.3%)	0.4460 (+15.0%)(+4.2%)	0.4560 (+17.5%)(+6.5%)
15 docs	0.3533	0.3987 (+12.9%)	0.4227 (+19.6%)	0.4440 (+25.7%)(+5.0%)	0.4453 (+26.0%)(+5.4%)
20 docs	0.3330	0.3960 (+18.9%)	0.4140 (+24.3%)	0.4290 (+28.8%)(+3.6%)	0.4350 (+30.6%)(+5.1%)
30 docs	0.2967	0.3740 (+26.1%)	0.4013 (+35.3%)	0.3987 (+34.4%)(-0.7%)	0.4060 (+36.8%)(+1.2%)

6. RESULTS: DOCUMENT RETRIEVAL EFFECTIVENESS

For document retrieval, the selected databases are searched and the returned results are merged into a single final list. In all of the experiments discussed in this section the results retrieved from individual databases were combined by the semi-supervised learning results merging algorithm. This version of the SSL algorithm [22] is allowed to download a small number of returned document texts "on the fly" to create additional training data in the process of learning the linear models which map database-specific document scores into estimated centralized document scores. It has been shown to be very effective in environments where only short result-lists are retrieved from each selected database [22]. This is a common scenario in operational environments and was the case for our experiments.

Document retrieval effectiveness was measured by Precision at the top part of the final document list. The experiments in this section were conducted to study the document retrieval effectiveness of five selection algorithms, namely the CORI, ReDDE, UUM/HR, UUM/HP-FL and UUM/HP-VL algorithms. The last three algorithms were proposed in Section 3. All the first four algorithms selected 3 or 5 databases, and 50 documents were retrieved from each selected database. The UUM/HP-FL algorithm also selected 3 or 5 databases, but it was allowed to adjust the number of documents to retrieve from each selected database; the number retrieved was constrained to be from 10 to 100, and a multiple of 10.

The Trec123-100Col and representative testbeds were selected for document retrieval as they represent two extreme cases of resource selection effectiveness; in one case the CORI algorithm is as good as the other algorithms and in the other case it is quite

Table 7. Precision on the trec123-100col-bysource testbed when 3 databases were selected (The first baseline is CORI; the second baseline for UUM/HP methods is UUM/HR.) (Search engines do not return document scores)

Precision at Doc Rank	CORI	ReDDE	UUM/HR	UUM/HP-FL	UUM/HP-VL
5 docs	0.3520	0.3240 (-8.0%)	0.3680 (+4.6%)	0.4520 (+28.4%)(+22.8%)	0.4520 (+28.4%)(+22.8)
10 docs	0.3320	0.3140 (-5.4%)	0.3340 (+0.6%)	0.4120 (+24.1%)(+23.4%)	0.4020 (+21.1%)(+20.4%)
15 docs	0.3227	0.2987 (-7.4%)	0.3280 (+1.6%)	0.3920 (+21.5%)(+19.5%)	0.3733 (+15.7%)(+13.8%)
20 docs	0.3030	0.2860 (-5.6%)	0.3130 (+3.3%)	0.3670 (+21.2%)(+17.3%)	0.3590 (+18.5%)(+14.7%)
30 docs	0.2727	0.2640 (-3.2%)	0.2900 (+6.3%)	0.3273 (+20.0%)(+12.9%)	0.3273 (+20.0%)(+12.9%)

a lot worse than the other algorithms. Tables 3 and 4 show the results on the Trec123-100Col testbed, and Tables 5 and 6 show the results on the representative testbed.

On the Trec123-100Col testbed, the document retrieval effectiveness of the CORI selection algorithm is roughly the same or a little bit better than the ReDDE algorithm but both of them are worse than the other three algorithms (Tables 3 and 4). The UUM/HR algorithm has a small advantage over the CORI and ReDDE algorithms. One main difference between the UUM/HR algorithm and the ReDDE algorithm was pointed out before: The UUM/HR uses training data and linear interpolation to estimate the centralized document score curves, while the ReDDE algorithm [21] uses a heuristic method, assumes the centralized document score curves are step functions and makes no distinction among the top part of the curves. This difference makes UUM/HR better than the ReDDE algorithm at distinguishing documents with high probabilities of relevance from low probabilities of relevance. Therefore, the UUM/HR reflects the high-precision retrieval goal better than the ReDDE algorithm and thus is more effective for document retrieval.

The UUM/HR algorithm does not explicitly optimize the selection decision with respect to the high-precision goal as the UUM/HP-FL and UUM/HP-VL algorithms are designed to do. It can be seen that on this testbed, the UUM/HP-FL and UUM/HP-VL algorithms are much more effective than all the other algorithms. This indicates that their power comes from explicitly optimizing the high-precision goal of document retrieval in Equations 14 and 16.

On the representative testbed, CORI is much less effective than other algorithms for distributed document retrieval (Tables 5 and 6). The document retrieval results of the ReDDE algorithm are better than that of the CORI algorithm but still worse than the results of the UUM/HR algorithm. On this testbed the three UUM algorithms are about equally effective. Detailed analysis shows that the overlap of the selected databases between the UUM/HR, UUM/HP-FL and UUM/HP-VL algorithms is much larger than the experiments on the Trec123-100Col testbed, since all of them tend to select the two large databases. This explains why they are about equally effective for document retrieval.

In real operational environments, databases may return no document scores and report only ranked lists of results. As the unified utility maximization model only utilizes retrieval scores of sampled documents with a centralized retrieval algorithm to calculate the probabilities of relevance, it makes database selection decisions without referring to the document scores from individual databases and can be easily generalized to this case of rank lists without document scores. The only adjustment is that the SSL algorithm merges ranked lists without document scores by assigning the documents with pseudo-document scores normalized for their ranks (In a ranked list of 50 documents, the first one has a score of 1, the second has a score of 0.98 etc) ,which has been studied in [22]. The experiment results on trec123-100Col-bysource testbed with 3 selected databases are shown in Table 7. The experiment setting was the same as before except that the document scores were eliminated intentionally and the selected databases only return ranked lists of document ids. It can be seen from the results that the UUM/HP-FL and UUM/HP-VL work well with databases returning no document scores and are still more effective than other alternatives. Other experiments with databases that return no document scores are not reported but they show similar results to prove the effectiveness of UUM/HP-FL and UUM/HP-VL algorithms.

The above experiments suggest that it is very important to optimize the high-precision goal explicitly in document retrieval. The new algorithms based on this principle achieve better or at least as good results as the prior state-of-the-art algorithms in several environments.

7. CONCLUSION

Distributed information retrieval solves the problem of finding information that is scattered among many text databases on local area networks and Internets. Most previous research use effective resource selection algorithm of database recommendation system for distributed document retrieval application. We argue that the high-recall resource selection goal of database recommendation and high-precision goal of document retrieval are related but not identical. This kind of inconsistency has also been observed in previous work, but the prior solutions either used heuristic methods or assumed cooperation by individual databases (e.g., all the databases used the same kind of search engines), which is frequently not true in the uncooperative environment.

In this work we propose a unified utility maximization model to integrate the resource selection of database recommendation and document retrieval tasks into a single unified framework. In this framework, the selection decisions are obtained by optimizing different objective functions. As far as we know, this is the first work that tries to view and theoretically model the distributed information retrieval task in an integrated manner.

The new framework continues a recent research trend studying the use of query-based sampling and a centralized sample database. A single logistic model was trained on the centralized

sample database to estimate the probabilities of relevance of documents by their centralized retrieval scores, while the centralized sample database serves as a bridge to connect the individual databases with the centralized logistic model. Therefore, the probabilities of relevance for all the documents across the databases can be estimated with very small amount of human relevance judgment, which is much more efficient than previous methods that build a separate model for each database. This framework is not only more theoretically solid but also very effective. One algorithm for resource selection (UUM/HR) and two algorithms for document retrieval (UUM/HP-FL and UUM/HP-VL) are derived from this framework. Empirical studies have been conducted on testbeds to simulate the distributed search solutions of large organizations (companies) or domain-specific hidden Web. Furthermore, the UUM/HP-FL and UUM/HP-VL resource selection algorithms are extended with a variant of SSL results merging algorithm to address the distributed document retrieval task when selected databases do not return document scores. Experiments have shown that these algorithms achieve results that are at least as good as the prior state-of-the-art, and sometimes considerably better. Detailed analysis indicates that the advantage of these algorithms comes from explicitly optimizing the goals of the specific tasks.

The unified utility maximization framework is open for different extensions. When cost is associated with searching the online databases, the utility framework can be adjusted to automatically estimate the best number of databases to search so that a large amount of relevant documents can be retrieved with relatively small costs. Another extension of the framework is to consider the retrieval effectiveness of the online databases, which is an important issue in the operational environments. All of these are the directions of future research.

ACKNOWLEGEMENT

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. (2000). Distributed information retrieval. In W.B. Croft, editor, *Advances in Information Retrieval*. Kluwer Academic Publishers. (pp. 127-150).

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

[3] J. G. Conrad, X. S. Guo, P. Jackson and M. Meziou. (2002). Database selection using actual physical and acquired logical collection resources in a massive domain-specific operational environment. Distributed search over the hidden web: Hierarchical database sampling and selection. In *Proceedings of the 28th International Conference on Very Large Databases (VLDB).*

[4] N. Craswell. (2000). Methods for distributed information retrieval. Ph. D. thesis, The Australian Nation University.

[5] N. Craswell, D. Hawking, and P. Thistlewaite. (1999). Merging results from isolated search engines. In *Proceedings of 10th Australasian Database Conference.*

[6] D. D'Souza, J. Thom, and J. Zobel. (2000). A comparison of techniques for selecting text collections. In *Proceedings of the 11th Australasian Database Conference.*

[7] N. Fuhr. (1999). A Decision-Theoretic approach to database selection in networked IR. *ACM Transactions on Information Systems*, 17(3). (pp. 229-249).

[8] L. Gravano, C. Chang, H. Garcia-Molina, and A. Paepcke. (1997). STARTS: Stanford proposal for internet meta-searching. In *Proceedings of the 20th ACM-SIGMOD International Conference on Management of Data*.

[9] L. Gravano, P. Ipeirotis and M. Sahami. (2003). QProber: A System for Automatic Classification of Hidden-Web Databases. *ACM Transactions on Information Systems*, 21(1).

[10] P. Ipeirotis and L. Gravano. (2002). Distributed search over the hidden web: Hierarchical database sampling and selection. In *Proceedings of the 28th International Conference on Very Large Databases (VLDB).*

[11] InvisibleWeb.com. http://www.invisibleweb.com

[12] The lemur toolkit. http://www.cs.cmu.edu/~lemur

[13] J. Lu and J. Callan. (2003). Content-based information retrieval in peer-to-peer networks. In *Proceedings of the 12th International Conference on Information and Knowledge Management*.

[14] W. Meng, C.T. Yu and K.L. Liu. (2002) Building efficient and effective metasearch engines. *ACM Comput. Surv.* 34(1).

[15] H. Nottelmann and N. Fuhr. (2003). Evaluating different method of estimating retrieval quality for resource selection. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[16] H., Nottelmann and N., Fuhr. (2003). The MIND architecture for heterogeneous multimedia federated digital libraries. *ACM SIGIR 2003 Workshop on Distributed Information Retrieval.*

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

[18] A.L. Powell and J.C. French. (2003). Comparing the performance of database selection algorithms. *ACM Transactions on Information Systems*, 21(4). (pp. 412-456).

[19] C. Sherman (2001). Search for the invisible web. Guardian Unlimited.

[20] L. Si and J. Callan. (2002). Using sampled data and regression to merge search engine results. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[21] L. Si and J. Callan. (2003). Relevant document distribution estimation method for resource selection. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[22] L. Si and J. Callan. (2003). A Semi-Supervised learning method to merge search engine results. *ACM Transactions on Information Systems*, 21(4). (pp. 457-491).