

Modeling Search Engine Effectiveness for Federated Search

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

ABSTRACT

Federated search links multiple search engines into a single, virtual search system. Most prior research of federated search focused on selecting search engines that have the most relevant contents, but ignored the retrieval effectiveness of individual search engines. This omission can cause serious problems when federating search engines of different qualities.

This paper proposes a federated search technique that uses utility maximization to model the retrieval effectiveness of each search engine in a federated search environment. The new algorithm ranks the available resources by explicitly estimating the amount of relevant material that each resource can *return*, instead of the amount of relevant material that each resource *contains*. An extensive set of experiments demonstrates the effectiveness of the new algorithm.

Categories & Subject Descriptors:

H.3.3 [Information Search and Retrieval]:

General Terms: Algorithms

Keywords: Model Search Engine Effectiveness

1. INTRODUCTION

Federated Search, also known as *distributed information retrieval* [3,6,14] searches information that cannot be accessed by conventional search engines such as Google or AltaVista by linking search engines of resources that contain this type of information. Federate search includes three sub-problems: i) information about the contents of each resource must be acquired (*resource description*) [3,9,16]; ii) a set of resources must be selected for search (*resource selection*) [7,8,15,16,18]; and iii) after results have been returned from selected resources, the results must be merged into a single ranked list (*results merging*) [3,17].

The solution of federated search is highly influenced by different environment characteristics. In this paper, we focus on uncooperative environments such as Web or wide area network, which contain multiple types of independent, non-cooperative resources. Many applications fall into this type of environment such as the QProber [10] system which supports browsing and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGIR '05, August 15-19, 2005, Salvador, Brazil.

Copyright 2005 ACM 1-59593-034-5/05/0008...\$5.00.

searching of hidden Web resources, or the federated search solution for FedStats portal [5] which provides a single interface to multiple Web sites of government agencies.

One practical problem that should be addressed in federated search solution of uncooperative environment is that search engines of some resources may not be effective and return very few of their relevant documents. These search engines may use ineffective retrieval methods such as the Boolean retrieval algorithm [1]. They may choose good retrieval algorithms but still suffer from bad retrieval parameter setting such as bad smoothing parameter in the language model retrieval algorithm [19]. Even if they use effective retrieval algorithms and choose good parameters at the early stage of system development, these parameters may become worse and worse when more and more documents with different characteristics are added into the system. Ineffective search engines exist in real world applications like the Boolean retrieval engine of the PubMed¹ Web site. Some other examples are the ineffective search engines linked by FedStats portal [5] that return unranked/(randomly ranked) results, or return many documents that do not exist, as in the case of broken links.

In this paper we propose a new algorithm to incorporate the factor of retrieval effectiveness of search engines into federated search framework. Our new algorithm measures the effectiveness of search engines by: i) sending a small number of training queries to retrieve documents from available resources; and ii) investigating the consistency between the ranked lists of individual search engines and the lists generated by an effective centralized retrieval algorithm on the returned documents. The behavior of how each search engine ranks its documents can be learned from the above steps. Then, in the resource selection phase, resources are ranked by both considering how many relevant documents they may contain and how effectively each search engine has ranked its returned documents in the past. This is accomplished by formalizing the resource selection procedure as an optimization problem that maximizes the amount of relevant documents to return.

Empirical study was conducted on several testbeds and several search algorithms with different characteristics to show the advantage of the new research over prior research that did not consider retrieval effectiveness of search engines. It provides more accurate results than the state-of-the-art algorithms when some search engines are not effective, and is at least as effective as the prior solutions when all the search engines are effective. Detailed analysis shows that the advantage of the new research comes from the explicit consideration of the retrieval effectiveness of individual search engines.

¹ <http://www.ncbi.nlm.nih.gov/entrez/query.fcgi>

The next section discusses related work. Section 3 describes our new approach to model search engine retrieval effectiveness. Section 4 explains our experimental methodology. Section 5 presents the experimental results and the corresponding discussion. Section 6 concludes.

2. PRIOR RESEARCH

There has been considerable research on the subproblems of federated search. We only survey the most related work with the new research in this paper.

Our research interest is uncooperative environments such as Web where resources provide basic services (e.g., running queries and returning documents), but do not provide other type of cooperation. In uncooperative environments, query-based sampling (QBS) [3] has been shown to acquire accurate unigram resource descriptions using a small number of queries (e.g., 75) and a small number of documents (e.g., 300). At the same time, the acquired documents by query-based sampling can be used to build a centralized sample database, which plays an important factor in estimating resource sizes [16], making resource selection decision [15,16,18] and merging the results [17].

There is a large body of prior research on resource selection [3,7,8,15,16,18]. The CORI [3] and KL-divergence [20] resource selection algorithms have been the state-of-the-art for a long time. They model resources as big documents, and rank resources using the similarities between those big documents and user queries. Recent studies [16,18] have shown that the “big document” resource selection algorithms are not directly designed to select resources that contain the largest number of relevant documents, and they do not work well in the environments of skewed resource sizes.

The decision-theoretic framework (DTF) [8,15] and unified utility maximization (UUM) algorithm [18] turn away from the “big document” approach by explicitly maximizing the number of relevant documents contained in the selected resources. DTF requires a large amount of training data to build separate models for each resource to estimate the number of relevant documents that each resource contains. UUM is more practical than DTF because it builds a single model on the centralized sample database to estimate the probabilities of relevance of documents, and makes the single model serve as a bridge to connect individual resources in order to estimate the amount of relevant documents that each resource contains.

Furthermore, the UUM algorithm shows that high Precision (i.e., high precision of top ranked documents in the final list) should be the goal of resource selection for federated document retrieval systems. This is because that federated document retrieval systems only retrieve part of relevant documents (high-Precision) from each selected resource and merge the results into a single final list. UUM algorithm assumes that each resource is effective to return the documents that are most probable to be relevant. Empirical study [18] has shown that UUM algorithm acquires more accurate results than several other algorithms in the case when all search engines of resources are effective to return their relevant documents.

The last step of federated search is to merge ranked lists returned by selected resources. The results merging task of federated search is different from that of Meta-search as very

little or even no overlap can be found among individual ranked lists. It is usually treated as a problem of transforming resource-specific document scores into comparable resource-independent document scores. The Semi-Supervised Learning (SSL) [17] results merging algorithm uses the documents acquired by query-based sampling as training data and uses linear regression to learn the resource-specific, query-specific merging model. Prior study has shown SSL to be effective to provide accurate merged list [17]. It is used as a part of the UUM framework [18] and also the new search in this paper.

Very little prior research addressed the issue of search engine retrieval effectiveness in federated search. Liu, et al., [13] showed how to estimate the parameters of similarity functions used by different search engines, but they assumed that similarity functions were of a known form (e.g., dot-product functions); many search engines don’t satisfy that requirement. Neither did their work address the issue of how to model search engine retrieval effectiveness in resource selection.

It might be possible to model search engine retrieval effectiveness within the decision-theoretic framework (DTF) [8,15], but no work has been conducted yet. Furthermore, the attempt in this framework may also be hindered by the possible excessive requirement of human judgment data to estimate the amount of relevant documents that each resource contains, which has been discussed before.

The research of unified utility maximization [18] framework made an unrealistic assumption for real world applications that individual search engines are always effective to return their relevant documents. When such an assumption is not valid, it claims that the semi-supervised learning results merging algorithm can detect irrelevant documents returned by ineffective search engines so that the those results can be discarded and do not hurt the final ranked list. However, the key point is that the failure to consider the retrieval effectiveness of search engines has already damaged the overall accuracy in the resource selection phrase. If better method has been utilized in resource selection, it is possible to choose resources that can return more relevant documents for better final ranked list.

3. RETURNED UTILITY MAXIMIZATION METHOD

Federated search can be viewed as trying to maximize the utility of selected resources. Most prior research defined utility as the amount of relevant material *contained* in a resource. However, the utility required by users is defined by the amount of relevant material *returned* by the resource. If retrieval effectiveness of search engines is not considered, the utility model does not match the user’s expectations. This motivates our new research: The returned utility maximization method.

This section first presents how to measure search engine effectiveness; then briefly describes how to estimate the probabilities of relevance of all documents in federated search environments; and finally shows the optimization framework of returned utility maximization method to select the resources.

3.1 Measuring the Retrieval Effectiveness of Search Engines

There is no direct way to measure the retrieval effectiveness of search engines in uncooperative federated search environments. One could estimate the parameters of ranking functions when these functions can be assumed to have special forms [13]. However, this assumption is rarely true in real world applications, so we do not consider it here.

In this work, a new method is proposed to model search engine effectiveness in federated search by investigating how consistent are the ranked lists returned from the search engines with the lists generated by an effective centralized retrieval algorithm on the same set of documents. The intuition is that if a search engine tends to generate ranked lists which are consistent with those of a centralized retrieval algorithm, this search engine is probable to be effective, and vice versa.

One may argue that it is more appropriate to measure the effectiveness of search engines by evaluating their returned results with human relevance judgment. However, this requires human efforts to provide relevance judgment data for the search engine of each resource, which is an excessive amount of work when there are many search engines in the federated search environment. Instead, the results of an effective centralized retrieval algorithm in the federated search environment can be a reasonably good surrogate for the human relevant judgment data. The comparison between the results from individual search engines and the results generated by centralized retrieval algorithm can be conducted automatically without requiring human efforts. Similar idea of taking advantage of a centralized retrieval algorithm in federated search has been successfully utilized for resource selection [18] and results merging [17].

The retrieval effectiveness profiles of search engines are built during the resource representation phase. The centralized sample database is created by all the sampled documents from query-based sampling. At the same time, an effective centralized retrieval algorithm (e.g., INQUERY [4]) is applied on the centralized sample database with a small number of training queries. The training queries are also sent to search available resources. For each resource, some documents in the individual ranked lists (e.g., every fifth documents in the top 250 documents) are downloaded. Note that only some sample documents (e.g., one sample document for the block size of 5) are downloaded to reduce the communication cost. These documents are ranked by the centralized retrieval algorithm with the corpus statistics on the centralized sample database. For a particular i th search engine, a mapping is learned for each j th training query, which transforms the document rank in the list from this search engine to the document rank in the list generated by the centralized retrieval algorithm. Formally as:

$$\phi_{ij} : d_{db_i}(r_1) \rightarrow d_c(r_2) \quad (1)$$

where ϕ_{ij} is the ranked list transformation of the j th training query for the i th resource, $d_{db_i}(r_1)$ represents the r_1 th document in the ranked list from the i th resource and $d_c(r_2)$ represents the r_2 th document in the ranked list generated by centralized retrieval algorithm. Equation (1) means that the r_1 th document in the ranked list of j th training query from the i th resource is mapped to the r_2 th document in the corresponding ranked list generated by the centralized retrieval algorithm. As a single sample document is used to represent a block of nearby documents in the ranked list, all the documents in this block are mapped to similar ranks by the centralized retrieval algorithm.

All the learned transformations of different training queries for each search engine are collected together to build the retrieval effectiveness profile for this search engine. Formally, for a particular search engine of the i th resource, the profile is: $\{\phi_j, 1 \leq j \leq J\}$, where there are altogether J transformations learned for the search engine. Those profiles will be utilized in the resource selection phase to estimate how many relevant documents in each resource are probable to be returned by the corresponding search engine.

3.2 Learning Probabilities of Relevance

As well as the retrieval effectiveness measure of search engines, the probabilities of relevance of all the documents across available resources should also be estimated to calculate the number of relevant documents that can be returned. A similar approach as that of unified utility maximization (UUM) [18] is used here. In resource representation, this method estimates a logistic model, which maps the centralized document scores (on centralized sample database) to the corresponding probabilities of relevance. In resource selection, for each particular resource this method first estimates the centralized document scores for all the documents in this resource, then maps the centralized document scores into the probabilities of relevance. We roughly describe this method; more detail can be referred to [18].

In resource representation, a logistic model is built to transform the normalized centralized document scores (document scores divided by the maximum centralized document score for each query) to the probabilities of relevance. This is accomplished with a small number of training queries (e.g., 50) that require small amount of human relevant judgment (e.g., 100 documents for each query). The model can be expressed formally as:

$$R(d) = P(\text{rel} | d) = \frac{\exp(a_c + b_c \tilde{S}_c(d))}{1 + \exp(a_c + b_c \tilde{S}_c(d))} \quad (2)$$

where $R(d)$ represents the probabilities of relevance for a particular document, $\tilde{S}_c(d)$ is the normalized centralized document score for this document, and a_c and b_c are the two parameters of the logistic model. Note that human judgment data is only required to build a single model on the centralized sample database, which is in contrast to the excessive amount of training data required by DTF to build separate models for different resources.

In resource selection, the centralized document scores for all documents in available resources are estimated. Specifically, for each resource, it is assumed that each of its sampled documents in the centralized sample database represents many unseen documents in the resource, which have similar centralized document scores. A non-parametric linear interpolation method is used to estimate the centralized document scores for all documents in the resource. Finally, by normalizing all the centralized document scores and applying the logistic model in Equation (2), a list of probabilities of relevance for all documents in this resource can be estimated as:

$\{R_i(d_c(r)), r \in [1, \hat{N}_{db_i}]\}$ (where \hat{N}_{db_i} is the estimated source size by Sample-Resample method [16] for the resource). Note that this list of probabilities is ranked in a decreasing manner of the centralized document scores. The documents, which are more probable to be relevant, are ranked at the top part. These steps

are conducted for all resources and the probabilities of relevance of all documents across resources are acquired.

3.3 Returned Utility Maximization Method

For federated document retrieval systems, selected resources are searched and individual ranked lists are merged into a final list. Users hope to acquire the largest number of relevant documents in the final list. Therefore, the utility for federated document retrieval systems should be the amount of relevant documents returned from selected resources. Let d_i denote the number of documents will be retrieved from the i th resource and $\vec{d} = \{d_1, d_2, \dots\}$ as the resource selection decision for all resources. Formally, the returned utility can be expressed by:

$$\sum_i I(d_i) \sum_{r=1}^{d_i} R(d_{db_i}(r)) \quad (3)$$

where the indicator function $I(d_i)$ represents whether the i th resource is selected or not; $R(d_{db_i}(r))$ is the probability of relevance of the top- r th document returned from the database. It can be seen from Equation (3) that in order to calculate the returned utility, the probabilities of relevance for documents in the top part of ranked lists of individual resources should be provided, which it is not known. Section 3.2 describes a method to estimate the probabilities of relevance for documents in the ranked list on the centralized sample database (i.e., $\{R_i(d_c(r)), r \in [1, \hat{N}_{db}]\}$), which is generated by the centralized retrieval algorithm. There is a gap between them.

To bridge the gap, we take advantage of the retrieval effectiveness profile built for each search engine, which is described in Section 3.1. Specifically, in resource selection, it is assumed for the i th resource, the current user query has a probability $P_i(j)$ to have the same rank pattern from the individual search engine and the centralized retrieval algorithm as the j th training query. Then, the top ranked documents from the individual search engines can be mapped to their corresponding ranks from the centralized retrieval algorithm. This step is conducted for all resources with all training queries, and the returned utility can be estimated as:

$$\sum_i I(d_i) \sum_{j=1}^J P_i(j) \sum_{r=1}^{d_i} R_i(\phi_{ij}(d_{db_i}(r))) \quad (4)$$

This can be calculated with the lists of probabilities of relevance for all documents derived in Section 3.2. It is assumed that the rank behavior of each training query is equally probable. Two constraints are added that only a small number (N_{sdb}) of resources should be selected to retrieve a fixed number (N_{rdoc}) of documents. Then, the resource selection optimization problem can be formally expressed as:

$$\begin{aligned} \vec{d}^* = \arg \max_{\vec{d}} & \sum_i I(d_i) \sum_{j=1}^J \frac{1}{J} \sum_{r=1}^{d_i} R_i(\phi_{ij}(d_{db_i}(r))) \\ \text{subject to :} & \sum_i I(d_i) = N_{sdb} \\ & d_i = N_{rdoc}, \text{ if } d_i \neq 0 \end{aligned} \quad (5)$$

This problem can be easily solved by calculating the estimated returned utility from top ranked documents of each resource as:

$$\hat{RU}_i = \sum_{j=1}^J \frac{1}{J} \sum_{r=1}^{d_i} R_i(\phi_{ij}(d_{db_i}(r))) \quad (6)$$

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
WT10g	7.8	300	1169	26505	0.3	8.4	161

Finally, the selection decision is made by selecting a few resources that contribute the largest amount of returned utility, and their returned results are merged with the SSL method [17].

4. EXPERIMENTAL METHODOLOGY

It is most desirable to evaluate federated search algorithms with real world applications. However, real world applications are usually short of relevance judgment data and it is difficult to obtain full control of different components of the systems (e.g., varying the retrieval algorithm of each resource). These difficulties prevent us to do thorough study with real world applications. Instead, an extensive set of experiments was designed in research environments to closely simulate real world applications such as [5].

4.1 Testbed

Two types of data were used in this paper: TREC web collections and TREC news/government collections. Both of them have advantages and disadvantages to simulate real world federated search applications.

TREC Web collection WT10g [6,12] provides a way to partition documents by different Web servers, which is consistent with that of Web resources. A relatively large number of resources (i.e., about 1,000) can be created, which makes this testbed a good candidate to simulate large-scale federated search applications. However, two weakness of this testbed are: i) many resources contain small number of documents (about two thirds of resources contain less than 1,000 documents); ii) the contents of WT10g are built by arbitrarily crawling Web data. The characteristics are not consistent with those observed from the FedLemur (i.e., FedStats) project [5], where most resources contain reasonable amount of documents (more than 1,000) with valuable topic-oriented contents (government agency statistics).

Another collection is the TREC news/government data [3,15,17,18]. TREC news/government data are much more similar to the contents provided by valuable topic-oriented resources than an arbitrary web page. By average, a resource in this collection contains thousands of documents, which is more realistic than that of the TREC Web collection. These characteristics make the TREC news/government data a better candidate to simulate the environment of domain-specific hidden Web or enterprise search. However, the TREC news/government collection is usually split into relatively small number (i.e., about 100) of resources by publication source and date, which does not well simulate federated search environments with large number of resources.

Specifically, three testbeds were used in this work:

Trec123-100col-bysource (Trec123): 100 collections were created from TREC CDs 1, 2 and 3. They are organized by source and publication date, and are somewhat heterogeneous. The sizes of the databases are not skewed.

Table 2. Precision on the trec123-100col-bysource testbed when 5 resources were selected. All search engines in the testbed were assigned a single type of retrieval algorithm as INQUERY, Language Model (LM) with good smoothing parameter, SMART with ltc weighting, INQUERY with random noise, Language Model with bad smoothing parameter or Extended Boolean algorithm.

Precision at Doc Rank	INQUERY	LM	SMART	INQUERY RAND	LM Bad Parameter	Extended Boolean
5 docs	0.4460	0.4920	0.4400	0.3200	0.3000	0.2600
10 docs	0.4340	0.4520	0.4080	0.2900	0.2720	0.2440
15 docs	0.4053	0.4040	0.3733	0.2680	0.2560	0.2320
20 docs	0.3750	0.3800	0.3590	0.2480	0.2540	0.2170
30 docs	0.3400	0.3527	0.3333	0.2240	0.2373	0.1920

Table 3. Precision on the WT10g testbed when 20 resources were selected. The same retrieval algorithms described in Table 2 were used.

Precision at Doc Rank	INQUERY	LM	SMART	INQUERY RAND	LM Bad Parameter	Extended Boolean
5 docs	0.2247	0.2144	0.1918	0.1299	0.1155	0.1196
10 docs	0.1938	0.1814	0.1825	0.1206	0.1247	0.1093
15 docs	0.1718	0.1684	0.1636	0.1031	0.1134	0.0942
20 docs	0.1536	0.1572	0.1546	0.0918	0.1036	0.0871
30 docs	0.1271	0.1278	0.1316	0.0746	0.0849	0.0780

Trec123-2ldb-62col (“representative”): In order to reflect the skewed source size distribution of some real world applications such as FedLemur Projects [5], two large resources were created by merging 20 small resources with round-robin method. Therefore, these two large resources tend to contain more relevant documents than the small resources.

These two testbeds were created from TREC news/government data. The detailed statistics of Trec123-100Col are shown in Table 1. In order to make comparison with previous research [3,15,16,18], fifty TREC title queries (101-150) were used as training queries and another 50 queries (51-100) were used as test data. Specifically, in order to build the retrieval effectiveness profile for each search engine, training queries were sent to resources and the ranked lists of up to 10 queries (queries that return more than 250 documents) were utilized. This means that sample documents (every fifth) were downloaded and the centralized retrieval algorithm was applied on them to compare with the resource ranked lists and generate the retrieval effectiveness profiles as described in Section 3.2. The retrieval effectiveness profiles were used in resource selection, then the selected resources were searched and each selected resource returned 50 documents.

WT10g-934col (“WT10g”): 934 resources were obtained by dividing WT10g data into 11,485 collections by their URLs and selecting those that contain more than 300 documents. Detail is in Table 1. The WT10g testbed was created from TREC Web data. TREC Web queries 451-550 were used on this testbed. The first set of fifty queries and the second set of fifty queries were used as training data and test data alternatively, and the final results were averaged by the two fold evaluation. As most of the resources on WT10g return short ranked lists because of their small sizes, the training queries that were used to create retrieval effectiveness profiles were only required to retrieve 50 documents. During testing, all selected resources returned 30 documents.

4.2 Search Engines

Multiple types of search engines were used in the experiments to reflect the characteristic of uncooperative environments. Three types of effective search engines used were: INQUERY [4], a

unigram statistical language mode with linear smoothing (the smooth parameter is set to be 0.5) [19] and a TFIDF retrieval algorithm [2] with “ltc” weighting [2,11].

Three types of ineffective retrieval algorithms were introduced in the experiments. Specifically, an extended Boolean retrieval algorithm [1], which adds up the term frequencies of matched query terms without considering idf factor; a unigram language model with bad linear smoothing parameter [19], which is set to be 0.99 that bias towards the corpus language model; and an INQUERY retrieval algorithm [4] with added random noise to the original retrieval scores, where the random noise ranges from 0 to 0.3 and the original scores range from 0.4 to 1.

To give more insight of how effective are these retrieval algorithms, two sets of federated search experiments were conducted on the Trec123 and WT10g testbeds. All the search engines were assigned a single type of retrieval algorithm. 5 resources and 20 resources were selected by UUM algorithm. The results are shown in Tables 2 and 3. It can be seen that the three effective retrieval algorithms acquire much more accurate results than the three ineffective algorithms on both two testbeds. The extended Boolean retrieval algorithm seems to be even less effective than the other two ineffective algorithms.

5. EXPERIMENT RESULTS

An extensive set of experiments were conducted to address the following three questions:

- 1) How does the new returned utility maximization (RUM) method work in the federated search environment that contains ineffective search engines? Experiments were conducted to compare the RUM method with the state-of-the-art CORI and unified utility maximization (UUM) methods that do not consider retrieval effectiveness of search engines in the case when one third of search engines are ineffective.
- 2) What is the behavior of the RUM method in federated search environments with different proportions of ineffective search engines? Experiments were conducted to show the behavior of the RUM method in the case when all search engines are effective and in the case when two third of search engines are ineffective.

Table 4. Precision on the trec123-100col-bysource testbed when 5 resources were selected. One third of the search engines among all resources were assigned with ineffective retrieval algorithms.

Precision at Doc	5 Resources Selected		
	UUM	CORI	RUM
5 docs	0.3800	0.3400 (-10.5%)	0.4400 (+15.8%)
10 docs	0.3400	0.3200 (-5.9%)	0.3860 (+13.5%)
15 docs	0.3173	0.3067 (-3.3%)	0.3680 (+16.0%)
20 docs	0.3050	0.2990 (-2.0%)	0.3520 (+15.4%)
30 docs	0.2840	0.2733 (-3.8%)	0.3240 (+14.1%)

Table 5. Precision on the representative testbed when 5 resources were selected. One third of the search engines among all resources were assigned with ineffective retrieval algorithms.

Precision at Doc	5 Resources Selected		
	UUM	CORI	RUM
5 docs	0.3880	0.3400 (-12.4%)	0.4280 (+10.3%)
10 docs	0.3580	0.3280 (-8.4%)	0.3880 (+7.4%)
15 docs	0.3373	0.3240 (-3.9%)	0.3600 (+6.7%)
20 docs	0.3150	0.3030 (-3.8%)	0.3280 (+4.3%)
30 docs	0.2867	0.2827 (-1.4%)	0.2867 (+0.0%)

Table 6. Precision on the WT10g testbed when 20 resources were selected. One third of the search engines among all resources were assigned with ineffective retrieval algorithms.

Precision at Doc	20 Resources Selected		
	UUM	CORI	RUM
5 docs	0.1753	0.1649 (-5.9%)	0.1918 (+9.3%)
10 docs	0.1598	0.1371 (-14.2%)	0.1784 (+11.6%)
15 docs	0.1402	0.1127 (-19.6%)	0.1505 (+7.4%)
20 docs	0.1211	0.0974 (-19.6%)	0.1320 (+9.0%)
30 docs	0.1041	0.0804 (-22.8%)	0.1117 (+7.2%)

3) Does the RUM method really favor the effective search engines and disfavor ineffective search engines? Experiments were conducted to study how often the resources with different types of retrieval algorithm are selected.

5.1 Retrieval Results with One Third Ineffective Search Engines

One third of the resources on the three testbeds of Trec123, representative and WT10g were assigned ineffective search engines with a round-robin manner. Particularly, one large source of representative testbed was assigned the extended Boolean retrieval algorithm, while the other large source was assigned SMART search engine. The results on these three testbeds are shown in Tables 4, 5 and 6 respectively. Results of RUM method in bold typeface are significantly better than corresponding UUM method (paired Wilcoxon test at $p=0.05$).

It can be seen from Table 4 that the RUM method acquired a substantial improvement than the UUM method on the Trec123 testbed. For the representative testbed, the results in Table 5 show that RUM method is still better than UUM. However, the advantage of the RUM method on the representative testbed is smaller than that on the Trec123 testbed. More detailed analysis shows that as the two large resources on the representative testbed contain more relevant documents than small resources, RUM method still favors the two large resources than other small resources as the same behavior with UUM. Therefore, when relatively more resources (e.g., 5) were selected, the overlap in the resources selected by RUM and UUM was getting larger and larger and these two methods tend to generate similar

results. When fewer resources (e.g., 3) were selected on the representative testbed, which was not shown due to space limit, the advantage of RUM over UUM is larger (about 10%).

The results on WT10g testbed are shown in Table 6. The results of the RUM method are better than those of the UUM method, which is satisfactory. One observation that can be seen by comparing Tables 4 and 6 is that by average the advantage of the RUM method over UUM is lower on the WT10g testbed than that on the Trec123 testbed. This can be attributed to the different characteristics of the two testbeds. WT10g contains much more small size resources (totally 625 out of 934 resources contain less than 1,000 documents) than Trec123 (3 out of 100 resources contain less than 1,000 documents). Small resources tend to return shorter ranked lists than large resources.

In RUM, resources are ranked by their estimated returned utilities of Equation (6). To calculate the returned utility of a top ranked document in each resource, its estimated centralized ranks are first obtained from the rank patterns of training queries; then the utility of this document is calculated by averaging the probabilities of relevance of documents with the estimated centralized ranks. For most training queries, small size resources return short ranked lists and the rank transformations always map top ranked documents in the resources to high centralized ranks. Therefore, the rank transformation may be less effective in estimating the retrieval effectiveness of search engines for small resources than large resources. However, in real world applications, most resources can be expected to contain reasonable amount of documents (e.g., several thousands) as observed in [5]. So more notable improvement by RUM over UUM can be expected like that on the Trec123 testbed.

5.2 Retrieval Results with Different Amount of Ineffective Search Engines

To further investigate the behavior of the RUM method and other algorithms, experiments were conducted to compare the algorithms in the federated search environments with different amount of ineffective search engines.

The first set of experiments was conducted when all the search engines are the three types of effective retrieval algorithms (INQUERY, LM and SMART). The three search engines were assigned to the resources on different testbeds with a round-robin manner. This is exactly the same setting as that in prior research [16,18]. The results are shown in Tables 7-9. It can be seen that the CORI algorithm was not as effective as the UUM algorithm, which is consistent with previous research [18]. The advantage of the UUM algorithm over the CORI algorithm is larger on this set of experiments than those when some search engines are not effective (Tables 4-6, Tables 10-12), which is also consistent with our expectation that the assumption of the UUM algorithm is correct in this case as the search engines are relatively effective to return their relevant documents. It can be seen that the RUM method is at least as good as UUM in all cases of this set of experiments. This demonstrates the effectiveness of the RUM method when all search engines in federated search environment are relatively effective.

The second set of experiments was designed to make two thirds of search engines choose ineffective retrieval algorithms. All the search engines were assigned ineffective retrieval algorithms

Table 7. Precision on the trec123-100col-bysource testbed when 5 resources were selected. All the search engines were assigned with relatively effective retrieval algorithms.

Precision at Doc	5 Resources Selected		
	UUM	CORI	RUM
5 docs	0.4720	0.4000 (-15.3%)	0.4720 (+0.0%)
10 docs	0.4260	0.3800 (-10.8%)	0.4300 (+0.9%)
15 docs	0.3960	0.3560 (-10.1%)	0.4120 (+4.0%)
20 docs	0.3740	0.3430 (-8.3%)	0.3890 (+4.0%)
30 docs	0.3520	0.3227 (-8.3%)	0.3647 (+3.6%)

Table 8. Precision on the representative testbed when 5 resources were selected. All the search engines were assigned with relatively effective retrieval algorithms.

Precision at Doc	5 Resources Selected		
	UUM	CORI	RUM
5 docs	0.4280	0.3960 (-7.0%)	0.4400 (+2.8%)
10 docs	0.4460	0.3900 (-12.6%)	0.4520 (+1.3%)
15 docs	0.4440	0.3533 (-20.4%)	0.4440 (+0.0%)
20 docs	0.4310	0.3340 (-22.5%)	0.4280 (-0.7%)
30 docs	0.3993	0.2967 (-25.7%)	0.3973 (-0.5%)

Table 9. Precision on the WT10g testbed when 20 resources were selected. All the search engines were assigned with relatively effective retrieval algorithms.

Precision at Doc	20 Resources Selected		
	UUM	CORI	RUM
5 docs	0.2000	0.1812 (-9.0%)	0.2201 (+10.1%)
10 docs	0.1763	0.1427 (-19.1%)	0.1901 (+7.8%)
15 docs	0.1546	0.1264 (-18.2%)	0.1601 (+4.3%)
20 docs	0.1428	0.1104 (-22.7%)	0.1475 (+3.3%)
30 docs	0.1223	0.0913 (-25.4%)	0.1286 (+5.2%)

with a round-robin manner. The results are shown in Tables 10-12. It can be observed that the RUM method outperformed the UUM method substantially on all the three testbeds. The advantage of the RUM method over the UUM method is larger than that when one third search engines are ineffective, which is consistent with our imagination as the power of RUM to detect ineffective search engines is more substantial when there exist more and more ineffective search engines.

5.3 Study the Search Engines Selected by RUM Method

One interesting set of experiment, which provides more insight of the behavior of RUM method, is to study the percentage of selected resources with different retrieval algorithms.

Two sets of experiments were conducted on TREC 123 and WT10g, where one third of the search engines were assigned ineffective retrieval algorithms and 5 or 20 resources were selected respectively. The results are shown in Tables 13 and 14. As the effective search engines were assigned to twice as many resources as ineffective search engines, the UUM algorithm selects effective search engines roughly twice as frequent as ineffective search engines (not exactly due to data variance). It can be observed from the results that the RUM method really favors the effective search engines and disfavors ineffective search engines than the UUM method, which is exactly the same as what we have expected. At the same time, it can be noted that the UUM method is more powerful to detect ineffective search engines on the Trec123 testbed than on the WT10g testbed. The analysis of this behavior is discussed in Section 5.1.

Table 10. Precision on the trec123-100col-bysource testbed when 5 resources were selected. Two third of the search engines among all resources were assigned with ineffective retrieval algorithms

Precision at Doc	5 Resources Selected		
	UUM	CORI	RUM
5 docs	0.3400	0.3120 (-8.2%)	0.4240 (+24.7%)
10 docs	0.3040	0.3120 (+2.6%)	0.3740 (+23.0%)
15 docs	0.2893	0.2880 (-0.5%)	0.3413 (+18.0%)
20 docs	0.2680	0.2680 (+0.0%)	0.3170 (+18.2%)
30 docs	0.2413	0.2467 (+2.2%)	0.2780 (+15.2%)

Table 11. Precision on the representative testbed when 5 resources were selected. Two third of the search engines among all resources were assigned with ineffective retrieval algorithms.

Precision at Doc	5 Resources Selected		
	UUM	CORI	RUM
5 docs	0.3720	0.3240 (-12.9%)	0.4160 (+11.8%)
10 docs	0.3360	0.3040 (-9.5%)	0.3680 (+9.5%)
15 docs	0.3053	0.2867 (-6.1%)	0.3360 (+10.0%)
20 docs	0.2690	0.2600 (-3.4%)	0.3100 (+15.2%)
30 docs	0.2320	0.2300 (-0.9%)	0.2727 (+17.5%)

Table 12. Precision on the WT10g testbed when 20 resources were selected. Two third of the search engines among all resources were assigned with ineffective retrieval algorithms.

Precision at Doc	20 Resources Selected		
	UUM	CORI	RUM
5 docs	0.1464	0.1320 (-9.8%)	0.1629 (+11.3%)
10 docs	0.1216	0.1082 (-11.0%)	0.1361 (+11.9%)
15 docs	0.1107	0.1003 (-9.4%)	0.1223 (+10.5%)
20 docs	0.0990	0.0871 (-12.0%)	0.1129 (+14.0%)
30 docs	0.0883	0.0739 (-16.3%)	0.0959 (+8.6%)

6. CONCLUSION

Federated search has been a hot research topic for a decade. As more and more real world applications appear, federated search is turning from a cool research topic to a practical tool. One important problem exists in operational federated search applications is the existence of ineffective search engines in federated search environments. This problem is very serious as the selected resources with ineffective search engines usually return much fewer relevant documents than what have been expected. This problem has been observed in previous research [5,13,18]. However, only partial solutions have been discussed to detect retrieval effectiveness of search engines of limited types [13] or to alleviate the damage of selecting resources with ineffective search engines by disfavoring their returned documents in results merging [18].

This paper proposes a new algorithm to detect the retrieval effectiveness of search engines and incorporate these statistics into a resource selection optimization framework to maximize the amount of returned utility. This is different fundamentally from all previous federated research which tried to maximize the utility that is contained in selected resources. As far as we know, this is the first work to incorporate the factor of retrieval effectiveness of search engines into the task of federated search.

Specifically, this paper proposes to measure the effectiveness of search engines by investigating how consistent are the ranked lists returned from the search engines with those lists generated by an effective centralized retrieval algorithm. Thus, the retrieval effectiveness profiles of search engines can be built

Table 13. The Percentage of selected resources with different retrieval algorithms on Trec123 testbed. 5 resources were selected while one third of the search engines were assigned ineffective retrieval algorithms. The retrieval algorithms have been described in Table 2.

Algorithms	INQUERY	LM	SMART	INQUERY RAND	LM Bad Parameter	Extended Boolean
UUM	0.176	0.249	0.233	0.102	0.102	0.139
RUM	0.249	0.359	0.261	0.057	0.049	0.024

Table 14. The Percentage of selected resources with different retrieval algorithms on WT10g testbed. 20 resources were selected while one third of the search engines were assigned ineffective retrieval algorithms. The retrieval algorithms have been described in Table 2.

Algorithms	INQUERY	LM	SMART	INQUERY RAND	LM Bad Parameter	Extended Boolean
UUM	0.194	0.239	0.211	0.138	0.117	0.103
RUM	0.225	0.276	0.230	0.105	0.090	0.075

automatically without requiring large amount of human efforts. Then, the probabilities of relevance of all documents across resources can be estimated and the returned utility maximization method is used to select the desired set of resources that can return the largest amount of relevant documents.

Extensive sets of empirical study have been conducted on several testbeds that closely simulate real world applications. The results have shown the new RUM method is always better than the well-known CORI method. It provides more accurate results than the unified utility maximization method when some search engines are not effective, and is at least as good as UUM when all the search engines are effective. Careful analysis shows that the power of the RUM method comes from favoring resources with effective search engines and disfavoring resources with ineffective search engines.

The research problem of studying retrieval effectiveness of search engines in federated search is a very practical topic. The work reported in this paper detects retrieval effectiveness of search engines with automatic and efficient method. It incorporates retrieval effectiveness of search engines into a formalized returned utility maximization framework and can acquire good empirical results. This work represents the first step in this direction, which is very important in order to apply federated search research into real world applications.

REFERENCES

- [1] R. Baeza-Yates, and B. Ribeiro-Neto. (1999). Modern Information Retrieval. ACM Press / Addison Wesley.
- [2] C. Buckley, A. Singhal, M. Mitra, and G. Salton. (1995). New retrieval approaches using SMART. In *Proceedings of 1995 Text REtrieval Conference (TREC-3)*. National Institute of Standards and Technology, special publication.
- [3] J. Callan. (2000). Distributed information retrieval. In W.B. Croft, editor, *Advances in Information Retrieval*. Kluwer Academic Publishers. (pp. 127-150).
- [4] J. Callan, W.B. Croft, and J. Broglio. (1995). TREC and TIPSTER experiments with INQUERY. *Information Processing and Management*, 31(3). (pp. 327-343).
- [5] T. T. Avraami, L. Yau, L. Si, and J. Callan. The FedLemur project: Federated search in the real world. *Journal of the American Society for Information Science and Technology*. (In Press).
- [6] N. Craswell. (2000). Methods for distributed information retrieval. Ph. D. thesis, The Australian Nation University.
- [7] 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*.
- [8] N. Fuhr. (1999). A Decision-Theoretic approach to database selection in networked IR. *ACM Transactions on Information Systems*, 17(3). (pp. 229-249).
- [9] 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*.
- [10] 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).
- [11] The lemur toolkit. <http://www.cs.cmu.edu/~lemur>
- [12] 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*.
- [13] K.L. Liu, W. Meng, C. Yu. (2001). Discovery of similarity computations of search engines, *Proceedings of the ninth 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] 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*.
- [17] 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).
- [18] L. Si and J. Callan. (2004). Unified Utility Maximization Framework for Resource Selection. In *Proceedings of the ninth international conference on Information and knowledge management*.
- [19] C.X. Zhai and J. Lafferty. (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- [20] J. Xu and J. Callan. (1998). Effective retrieval with distributed collections. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.