Mixture Model with Multiple Centralized Retrieval Algorithms for Result Merging in Federated Search

Dzung Hong
Department of Computer Science
Purdue University
250 N. University Street
West Lafayette, IN 47907, USA
dthong@cs.purdue.edu

Luo Si
Department of Computer Science
Purdue University
250 N. University Street
West Lafayette, IN 47907, USA
lsi@cs.purdue.edu

ABSTRACT

Result merging is an important research problem in federated search for merging documents retrieved from multiple ranked lists of selected information sources into a single list. The state-of-the-art result merging algorithms such as Semi-Supervised Learning (SSL) and Sample-Agglomerate Fitting Estimate (SAFE) try to map document scores retrieved from different sources to comparable scores according to a single centralized retrieval algorithm for ranking those documents. Both SSL and SAFE arbitrarily select a single centralized retrieval algorithm for generating comparable document scores, which is problematic in a heterogeneous federated search environment, since a single centralized algorithm is often suboptimal for different information sources.

Based on this observation, this paper proposes a novel approach for result merging by utilizing multiple centralized retrieval algorithms. One simple approach is to learn a set of combination weights for multiple centralized retrieval algorithms (e.g., logistic regression) to compute comparable document scores. The paper shows that this simple approach generates suboptimal results as it is not flexible enough to deal with heterogeneous information sources. A mixture probabilistic model is thus proposed to learn more appropriate combination weights with respect to different types of information sources with some training data. An extensive set of experiments on three datasets have proven the effectiveness of the proposed new approach.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Design, Performance

Keywords

Federated Search, Result Merging, Mixture Model

1. INTRODUCTION

Federated search (also known as distributed information retrieval) [17, 23, 29] is an important research area of information retrieval. Unlike traditional information search systems such as Google or Bing, which index webpages or documents that can be crawled and collected, federated search targets on information distributed in independent information providers. Many contents in this environment may not be arbitrarily crawled and searched by traditional search engines, due to various reasons such as copyright, security and data protection. Only the owners of those documents can provide a full searching service to their set of documents. We refer to a collection of documents with its own, customized search engine as an information source. The size of this type of information (i.e., hidden Web contents) has been estimated to be many times larger than Web contents searchable by traditional search engines [3].

Federated search offers a solution by building a bridge between users, who have little knowledge about which kind of information sources she is looking for, and the information sources that reveal limited information about their documents through source-specific search engines. To achieve the goal, federated search includes three main research problems: resource representation, resource selection and result merging. Resource representation learns important information about the sources such as their contents and their sizes. Resource selection selects a subset of information sources which are most useful for users' queries. Result merging combines documents retrieved from selected sources into a single ranked list before presenting the list to the end users.

Among the above main problems, result merging substantially suffers from the heterogeneity of information sources. Each information source may adopt a different, customized retrieval model. A query can also be processed in many ways. Even if different sources use similar retrieval algorithms, they may have different source statistics (e.g., different values of inverse document frequencies). All of this makes it difficult to compare documents of different sources. A simple solution that downloads all document contents and ranks them with a single method for each user query may yield good results, but it is also costly in an online setting. Other solutions such as downloading parts of the documents

[6] or incorporating scores from the resource selection component into source-specific document scores (e.g., CORI [4]) also suffer when information sources do not provide enough information or vary greatly in their scales of document ranking scores.

The state-of-the-art result merging algorithms merge documents by learning how to map document scores in ranked lists of multiple information sources to comparable document scores. The basic idea is to utilize a centralized sample database created with all sample documents obtained in resource representation. For each query, these algorithms rank documents in the centralized sample database with a centralized retrieval algorithm, and then build a mapping function between source-specific document scores/ranks and comparable document scores. By mapping document scores/ranks returned from all selected sources to a common ground, it is possible to construct the final ranked list. Algorithms of this class such as SSL [27], SAFE [24] and WCF [12] have shown promising results. However, despite using various learning algorithms, those methods still do not fully address the heterogeneity of retrieval algorithms in different information sources. The problem lies in the fact that all these existing methods arbitrarily select a single fixed centralized retrieval algorithm for learning the mapping, which is problematic in a heterogeneous federated search environment, as a single algorithm is often suboptimal for learning comparable scores for different sources.

In this paper, we propose a novel result merging algorithm that utilizes multiple centralized retrieval algorithms. This method can generate more accurate results in result merging due to the flexibility of using multiple types of centralized retrieval algorithms for estimating comparable document scores. In particular, the paper shows that it is not desirable to learn a fixed set of weights (e.g., with a logistic regression approach) for different centralized retrieval algorithms in estimating comparable document scores. A mixture probabilistic model is proposed to automatically learn the appropriate weights for different types of information sources with some training data. The mixture model approach is more flexible in calculating comparable document scores for a heterogeneous set of information sources. Empirical studies have been conducted with three federated search datasets to show the advantages of the proposed result merging algorithm. In particular, one new dataset is created from the Wikipedia collection of ClueWeb data.

The rest of the paper is organized as follows. Section 2 discusses some research work generally related with the work in this paper. Section 3 discusses two specific state-of-the-art results merging algorithms (SSL and SAFE) as they are directly related with the proposed research. Section 4 proposes the novel result merging algorithm with multiple centralized retrieval algorithms. Section 5 introduces experimental methodology. Section 6 presents the detailed experimental results and provides some discussions. Section 7 concludes and points out some future research directions.

2. RELATED WORK

Federated search includes three main research problems: resource representation, resource selection and result merging. There is a large volume of previous research work in all of those research problems. This section first briefly introduces most related prior research in resource representation and resource selection. Then it will provide more details about the literature of result merging.

Resource representation is to collect information about each information source. Such information usually includes sources' sizes, document frequencies, term frequencies, and other statistics. The START protocol [9] is one of the first attempt to standardize the communication between information sources and a broker (or centralized agent) in order to collect, search and merge documents from individual sources. However, this approach can only work in cooperative environments. In an uncooperative environment, it is more practical to collect source statistics with sampling algorithms. The query-based sampling method [4] is a popular algorithm for sampling documents from a set of information sources. In principal, query-based sampling sends randomly generated terms as queries to a source, and downloads the top documents as sample documents for each query. After this process finishes, the set of all sample documents can be collected in a *centralized sample database* to build a single index. The centralized sample database is often a good representative of the (imaginary) complete database of all documents in a federated search environment.

Resource selection is to select a subset of information sources most relevant to the user's query. Resource selection has been studied intensively during the last two decades. Many algorithms have been developed, such as GlOSS [10], CORI [4], ReDDE [26], CRCS [22], topic models [2], the classification-based model [1] and many others. The Relevant Document Distribution Estimation (ReDDE) resource selection algorithm and its variants have been shown to generate good and robust resource selection results in different types of federated search environments. ReDDE selects relevant sources by first ranking sample documents in the centralized sample database. Then, each document among the top of the list can contribute a score to its containing source. The magnitude of the score depends on both document's rank and the source's size. Finally, the relevance of a source is measured by the combined score of all of its sample documents.

Result merging is to collect the ranked list of documents from each selected source and combine them into a single ranked list to present to users. Result merging in federated search is similar to data fusion [32, 25], or merging process in multilingual information retrieval [30]. In data fusion, different retrieval models are applied to a single information source, and the problem is to get the best combination of retrieval algorithms. Whereas, in federated search, there are multiple information sources with different (often unknown) retrieval models. Similar to information fusion, multilingual information retrieval also assumes that the whole collection index is available to the merger during the process, which is not always the case in federated search.

One scenario is that the broker can download all returned documents from selected sources, and apply a centralized retrieval algorithm to produce the final ranked list. However, in practice, this method is rarely used since the high cost of communication and time may impair user experience. In another simple case, when all sources implement the same retrieval model, documents' scores (or ranks) returned by the source may be comparable with each other. Thus, merging their scores (or ranks) directly (also known as Raw Score Merging), or in a round-robin fashion may give good results with low cost. However, it is noticed that even if all sources share the same model, some statistics such as document frequency of a term are still different across different sources. It is generally not practical to assume that all independent sources share such a same set of collection statistics.

Some other algorithms in the early generation of federated search also relied on term statistics for making decision. Craswell et al. suggested that by partially downloading a part of the top returned documents, we can approximate term statistics to build the final rank list [6]. Xu and Croft requested that document statistics of query terms should be provided to the broker, in such a way that they can calculate the global inverse document frequencies [34]. However, these algorithms again require some type of collaboration from the independent sources, which is often unavailable.

CORI result merging algorithm [4] is a relatively simple, yet effective algorithm. The intuition is that comparable document scores should depend on two factors: (i) how good a document is compared to other documents from the same source; and (ii) how good the source containing a particular document is compared to other sources. CORI makes a linear combination of those two factors and gets the final score of a document as:

$$D' = \frac{D + 0.4 \times D \times C'}{1.4}$$

where D' is the global score, D is the original score within the source, and C' is the normalized source score from the resource selection step.

The merging algorithm proposed by Rasolofo et al. [19] also explores the combination between document scores and source weights. Unlike CORI, their source weights are not directly related with the sources' relevance scores. Rather, the weight of a source depends on the total number of documents that it returns. The algorithm assumes that a source containing more relevant documents may return a longer ranked list, which is not always true for information sources using different types of ranking algorithms.

The intuition of combining document and resource scores can also be seen in a variant of the PageRank algorithm in distributed environments [31]. In this work, Wang and DeWitt employed the source's ServerRank and the document's LocalRank to derive the global PageRank values.

Semi-Supervised Learning (SSL) [27] and the Sample Agglomerate Fitting Estimate (SAFE) [24] result merging algorithms offer a better trade-off in efficiency and effectiveness. Both methods try to map source-specific document ranks into comparable document scores generated by a single centralized retrieval algorithm. We will provide more detailed information about SSL and SAFE in the following section as they are directly related with the new research in this paper.

3. RESULT MERGING BY SEMI-SUPERVISED LEARNING & SAMPLE-AGGLOMERATE FITTING ESTIMATE

3.1 Semi-Supervised Learning Merging

Semi-Supervised Learning Merging (SSL) [27] uses curve fitting model to calculate comparable document scores from different sources for result merging. Specifically, given a user's query, SSL sends the query to the centralized sample database and retrieves the sample ranked list with relevance score of each document. Upon receiving documents from a selected information source, SSL checks for overlapping documents exist in the sample database. Those overlapping documents are characterized by two features: the relevance scores in the central sample database, and the relevance ranks in the specific source. The task is to estimate the relevance scores of all non-overlapping documents in the centralized complete database (the imaginary dataset of all documents of all sources). Assume that there is a linear mapping between centralized relevance scores and source-specific document ranks, then that mapping can be inferred by using a regression method on the overlapping documents. Having said that, let R_{ij} be the source-specific rank of document d_i in source C_j, and S_{ij} be the relevance score of document d_i in the centralized sample database, we can build a linear relationship.

$$S_{ij} = a_j \times R_{ij} + b_j$$

where a_j, b_j are two parameters depending on each pair of an information source and a query.

With enough overlapping documents for a source and a query, we can train a regression matrix

$$\begin{bmatrix} R_{1j} & 1 \\ R_{2j} & 1 \\ \cdots & 1 \\ R_{nj} & 1 \end{bmatrix} \times \begin{bmatrix} a_j \\ b_j \end{bmatrix} = \begin{bmatrix} S_{1j} \\ S_{2j} \\ \cdots \\ S_{nj} \end{bmatrix}$$

In the above equation, let us denote the first matrix by X, the second matrix by W, and the third matrix by Y. By minimizing the square loss error, we can derive the solution to the parameters W as

$$W = (X^T X)^{-1} X^T Y$$

One main problem of SSL is that if there is not enough overlapping documents (three requested in the original SSL work) for building a linear mapping, the model will back off to the CORI result merging formula, which is often much less effective.

3.2 Sample-Agglomerate Fitting Estimate Merging

Sample-Agglomerate Fitting Estimate (SAFE) [24] overcomes the SSL's problem of not having enough overlapping documents by estimating the ranks of unoverlapping documents in the centralized sample database. If we assume that the sampling process is uniform, then each sample document will represent the same number of unseen documents in the selected information source. Therefore, a sample document ranked at position i-th in the source-specific sample ranked list will have an approximate rank $i \times \frac{|C|}{|C_s|}$ in the source-specific full rank, where $|C|$ is the source's estimated size, and $|C_s|$ is the source's sample size. By using the estimated source-specific ranks together with true centralized ranks (of

Table 1: Transformation Functions

Name	$f(x)$	Model
LIN	$f(x) = x$	$S = a \times R + b$
SQRT	$f(x) = \sqrt{x}$	$S = a' \times \sqrt{R} + b'$
LOG	$f(x) = \log x$	$S = a'' \times \log R + b''$
POW	$f(x) = 1/x$	$S = a''' \times \frac{1}{R} + b'''$

overlapping documents), SAFE could apply regression with more information than SSL. A problem may occur when there are not enough sample documents of a selected information source in the centralized sample database. However, this is rarely the case, if ReDDE (or its variants) is used for selecting information sources, since this method usually selects a source if it has a significant number of documents in the centralized ranked list with respect to the query.

Another contribution of SAFE to SSL is that, instead of using the raw rank information of documents, SAFE applies different transformation functions to the rank, in order to find the best regression. More specifically, there are four different transformations as in Table 1. Each transformation function is applied to the source-specific ranked list to learn the set of parameters (a_{ij}, b_{ij}). Then, SAFE selects the best transformation by comparing the goodness of curve-fitting of all models based on their coefficient of determination R^2 values [11]. Specifically, for a linear regression equation $X \times w = Y$, the coefficient of determination is calculated as follows.

$$R^2 = \frac{||\hat{Y}||^2}{||Y||^2} = \frac{Y^T P Y}{Y^T Y}$$

where $P = X(X^T X)^{-1} X^T$

4. MIXTURE OF RETRIEVAL MODELS FOR RESULT MERGING

SSL and SAFE are state-of-the-art algorithms for result merging in federated search. However, because of their choosing of a single centralized retrieval algorithm for calculating comparable document scores, these algorithms still do not fully address the heterogeneity of different information sources in federated search environments . A single centralized retrieval algorithm may have good curve-fittings for some information sources, but may also give bad fits for some others. This paper proposes to use multiple centralized retrieval algorithms to retrieve a set of ranking scores for each document. Moreover, rather than assigning a fixed set of weights to combine the above scores, our model learns a more appropriate combination of weights with respect to different types of information sources. We assume that there is an underlying distribution (i.e., latent groups) of sources according to their adopted retrieval models. Learning the proposed model thus becomes learning the distribution of groups and the combination weights associated with each group. This model is called the Mixture of Retrieval Models (MoRM) for result merging. MoRM is related with SSL and SAFE in the way that it uses the centralized sample database to learn the comparable document scores by curve-fitting. However, unlike SSL and SAFE, MoRM employs multiple retrieval algorithms for the centralized sample database. Therefore it is more flexible to address the heterogeneity of information sources in federated search environments for improving the accuracy of result merging.

4.1 MoRM's Framework

In this section, we describe the general framework of MoRM for document merging. The following steps are applied when a query comes:

- A resource selection algorithm such as ReDDE [26] selects a subset of information sources that are most relevant to the user's query.
- The query is then forwarded to the selected information sources. Each source will return a ranked list of documents. Scores of the returned documents will help the model's performance but are not required. Document ranks are usually sufficient for the next step.
- MoRM also issues the query to the centralized sample database and retrieves documents using a set of predetermined algorithms. At the end of this step, it obtains a set of ranked lists of sample documents.
- For each ranked list, MoRM tries to learn a mapping between source-specific document ranks and the centralized document scores. Ranks of sample documents that are not in the source-specific ranked list are estimated in a similar way as in the SAFE algorithm. All transformation functions as listed in Table 1 are tested in order to find the best curve fitting parameters. The best transformation function is applied to predict the comparable scores of all returned documents.
- All comparable scores of a document are combined using a set of combination weights learned from a training dataset. These final scores are used to rank documents.

In the following sections, we will propose a simple logistic regression model (for learning a single set of combination weights) and then propose the mixture of retrieval models (for learning multiple sets of combination weights) for the task of estimating documents' comparable scores.

4.2 Logistic Regression for Learning Comparable Scores

A learning algorithm such as logistic regression may address the problem of combining different document scores seamlessly. We chose logistic regression to demonstrate the approach of learning a single set of combination weights for ranking documents. Logistic regression is a discriminative model that models the probability that a binary event happens using a sigmoid function.

$$P(y^q_{cd} = 1 | \vec{w}, \vec{x}^q_{cd}) = \frac{1}{1 + \exp(-\vec{w} \cdot \vec{x}^q_{cd})} = \sigma(\vec{w} \cdot \vec{x}^q_{cd}) \quad (1)$$

and

$$P(y^q_{cd} = -1 | \vec{w}, \vec{x}^q_{cd}) = 1 - P(y^q_{cd} = 1) = \sigma(-\vec{w} \cdot \vec{x}^q_{cd}) \quad (2)$$

where the superscript q denotes the features with respect to the query q; $y^q_{cd} = 1$ means document D_{cd} (i.e., d-th document from source c) is relevant to the user query q, otherwise $y^q_{cd} = -1$; $\vec{x}^q_{cd}$ is the feature vector of document

D_{cd} (the set of comparable scores of D_{cd} according to different centralized retrieval algorithms), and $\vec{w}$ is the set of weights associated with $\vec{x}^q_{cd}$. We also use $\sigma(z)$ for the sigmoid function.

$$\sigma(z) = \frac{1}{1+\exp(-z)}$$

Given $\mathcal{C}$, the number of sources; Q, the number of queries, and all the returned documents D_{cd} with respect to the training queries, we can write the likelihood function of the model as

$$\mathcal{L}(\vec{w}) = \prod_{q=1}^{|Q|}\prod_{c=1}^{\mathcal{C}}\prod_{d=1}^{D_c} P(y^q_{cd}|\vec{w},\vec{x}^q_{cd}) = \prod_{q=1}^{|Q|}\prod_{c=1}^{\mathcal{C}}\prod_{d=1}^{D_c}\sigma(y^q_{cd}\vec{w}\cdot\vec{x}^q_{cd})$$

where we have combined equations (1) and (2) above. Learning the combination weight $\vec{w}$ can be done by maximizing the log-likelihood function using the iterative re-weighted least squares method [8].

4.3 Mixture of Retrieval Models for Learning Comparable Document Scores

We now describe the mixture model of retrieval algorithms (MoRM) for result merging. MoRM offers more prediction capability by automatically learning multiple sets of combination weights, each of them is associated with a "soft" information source cluster. The word "soft" means that we use probability to assign a source to its cluster, rather than fixing a hard assignment. Specifically, assuming that there are K of such clusters, and let π_{ck} be the probability that the source c belongs to group k, then the following constraints must be hold:

$$\textstyle\sum_{k=1}^{K}\pi_{ck} = 1 \quad \text{for } c = 1, 2, \cdots, \mathcal{C}$$

To make our notations uncluttered, in this section, we will first derive the formulations for only one query, and drop the superscript q of y_{cd} and $\vec{x}_{cd}$. At the end of this section, we will extend the formulations for the set of training queries. Furthermore, we will denote σ_{cdk} for $P(y_{cd}|\vec{w}_k,\vec{x}_{cd})$ (the probability that the document D_{cd} has relevance y_{cd} to the query in question, given that the collection c belongs to cluster k. In short, $\sigma_{cdk} = \sigma(y_{cd}\vec{w}_k\cdot\vec{x}_{cd})$).

Let $\vec{\mathbf{w}} = \{\vec{w}_k|k = 1,\cdots,K\}$, and $\boldsymbol{\pi} = \{\pi_{ck}|c = 1,\cdots,\mathcal{C}; k = 1,\cdots,K\}$. Given a query q, let $\vec{\boldsymbol{\theta}} = \{\vec{\mathbf{w}},\boldsymbol{\pi}\}$ denote the set of parameters, in which each combination weight $\vec{w}_k$ is associated with the k-th cluster. MoRM assumes the same combination weight for all sources of a cluster for building robust combination model with a limited amount of training data, hence we will set $\pi_k = \pi_{ck} = \pi_{c'k}$ for all sources c, c'.

The probability that a document D_{cd} has relevance y_{cd} given all parameters is calculated as follows.

$$P(y_{cd}|\vec{\boldsymbol{\theta}},\vec{x}_{cd}) = \sum_{k=1}^{K}\pi_k P(y_{cd}|\vec{w}_k,\vec{x}_{cd}) = \sum_{k=1}^{K}\pi_k\sigma_{cdk} \qquad (3)$$

The component π_k acts as the prior of the clusters' distribution, which adjusts the belief of relevance according to each cluster. This equation is also known as the mixture of logistic regression. Given that model, the likelihood function for the training dataset with respect to one query is as follows.

$$\mathcal{L}(\vec{\boldsymbol{\theta}}) = \prod_{c=1}^{\mathcal{C}}\prod_{d=1}^{D_c}\sum_{k}^{K}\pi_k\sigma_{cdk} \qquad (4)$$

where D_c is the number of documents returned by the source c.

It is difficult to optimize the above function directly, since taking its logarithm still presents the summation inside the log. Therefore, we will utilize the Expectation Maximization (EM) algorithm [7] to learn the parameters. The derivation of EM algorithm is discussed in the following section.

4.4 Learning MoRM using EM Algorithm

Let $\boldsymbol{z_c}\{c = 1,\cdots,\mathcal{C}\}$ be a K-dimensional random variable having a 1-of-K representation, i.e., only one element of $\boldsymbol{z_c}$ equal 1 and the other elements equal 0. Therefore, $\boldsymbol{z_c}$ must satisfy the following constraints.

$$\textstyle\sum_{k=1}^{K} z_{ck} = 1 \quad \text{for } c = 1, 2, \cdots, \mathcal{C}$$

where z_{ck} is the k-th element of $\boldsymbol{z_c}$.

We then use $\boldsymbol{z_c}$ as the indicator of the membership of source c. If c belongs to cluster k, then $z_{ck} = 1$, and the other elements of $\boldsymbol{z_c}$ equal 0. Given the definition of π_k as above, and note that $\pi_{ck} = \pi_k$ for all c, we can write the following equation.

$$P(z_{ck} = 1) = \pi_k \qquad (5)$$

Then the distribution of the vector $\boldsymbol{z_c}$ can be written as

$$P(\boldsymbol{z_c}) = \prod_{k=1}^{K}{\pi_k}^{z_{ck}} \qquad (6)$$

Define another random variable $\boldsymbol{Z} = \{\boldsymbol{z_c}|c = 1,\cdots,\mathcal{C}\}$ associated with all sources. Since each source is independent of each other, the prior of $\boldsymbol{Z}$ is just the multiplication over all sources.

$$P(\boldsymbol{Z}) = \prod_{c=1}^{\mathcal{C}}\prod_{k=1}^{K}{\pi_k}^{z_{ck}} \qquad (7)$$

Similarly, given the membership vector $\boldsymbol{z_c}$, the probability that document D_{cd} has the relevance y_{cd} is $\prod_{k=1}^{K}{\sigma_{cdk}}^{z_{ck}}$. The likelihood function of the model is obtained by multiplying the above term over all sources and documents.

$$P(\boldsymbol{X},\boldsymbol{Y}|\boldsymbol{Z},\vec{\boldsymbol{\theta}}) = \prod_{c=1}^{\mathcal{C}}\prod_{d=1}^{D_c}\Big(\prod_{k=1}^{K}{\sigma_{cdk}}^{z_{ck}}\Big) \qquad (8)$$

where $\boldsymbol{X}$ denotes all document feature vectors, and $\boldsymbol{Y}$ denotes the relevance vector of all documents. Multiplying the equations (7) and (8) above, one can calculate the complete likelihood function as

$$P(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}|\vec{\boldsymbol{\theta}}) = \prod_{c=1}^{\mathcal{C}}\prod_{k=1}^{K}\Big({\pi_k}^{z_{ck}}\prod_{d=1}^{D_c}{\sigma_{cdk}}^{z_{ck}}\Big) \qquad (9)$$

Therefore, taking the logarithm of the above function yields the complete log-likelihood as follows.

$$\log P(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}|\vec{\boldsymbol{\theta}}) = \sum_{c=1}^{\mathcal{C}}\sum_{k=1}^{K} z_{ck}\{\log\pi_k + \sum_{d=1}^{D_c}\log\sigma_{cdk}\} \quad (10)$$

The EM algorithm involves two steps. For the E-step, we need to calculate the posterior probability $P(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{Y},\vec{\boldsymbol{\theta}})$. Using (9), we can derive the following relation.

$$P(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{Y},\vec{\boldsymbol{\theta}}) = \frac{P(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}|\vec{\boldsymbol{\theta}})}{P(\boldsymbol{X},\boldsymbol{Y}|\vec{\boldsymbol{\theta}})} \propto \prod_{c=1}^{\mathcal{C}}\prod_{k=1}^{K}\Big(\pi_k\prod_{d=1}^{D_c}\sigma_{cdk}\Big)^{z_{ck}} \tag{11}$$

where we can use the proportional sign $\propto$ because the denominator $P(\boldsymbol{X},\boldsymbol{Y}|\vec{\boldsymbol{\theta}})$ does not depend on $\boldsymbol{Z}$.

We wish to calculate the expectation of the variable $\boldsymbol{Z}$ under the above posterior distribution, since that term will be useful in the following M step. Given that all $\boldsymbol{z_c}$ are independent, and the right-hand side of equation (11) can be factorized over c, we can derive the expectation of each variable z_{ck} as

$$\begin{aligned}\mathbb{E}[z_{ck}] &= \frac{\sum_{z_{ck}} z_{ck}\big(\pi_k\prod_{d=1}^{D_c}\sigma_{cdk}\big)^{z_{ck}}}{\sum_{z_{cj}} z_{cj}\big(\pi_j\prod_{d=1}^{D_c}\sigma_{cdj}\big)^{z_{cj}}} \\ &= \frac{\pi_k\prod_{d=1}^{D_c}\sigma_{cdk}}{\sum_{j=1}^{K}\pi_j\prod_{d=1}^{D_c}\sigma_{cdj}} = \gamma(z_{ck}) \qquad (12)\end{aligned}$$

where we have defined a new variable $\gamma(z_{ck})$.

In the M-step, the updated parameters $\vec{\boldsymbol{\theta}}^{new}$ are calculated according to the following formula

$$\vec{\boldsymbol{\theta}}^{new} = \arg\max_{\vec{\boldsymbol{\theta}}} \mathcal{Q}(\vec{\boldsymbol{\theta}},\vec{\boldsymbol{\theta}}^{old}) \tag{13}$$

where

$$\mathcal{Q}(\vec{\boldsymbol{\theta}},\vec{\boldsymbol{\theta}}^{old}) = \mathbb{E}\big[\log P(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}|\vec{\boldsymbol{\theta}})\mid P(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{Y},\vec{\boldsymbol{\theta}})\big]$$

Taking the expectation of $\log P(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}|\vec{\boldsymbol{\theta}})$ (as derived in equation (10)) gives us the following objective function for the M-step.

$$\{\boldsymbol{\pi}^{new},\vec{\mathbf{w}}^{new}\} = \arg\max_{\boldsymbol{\pi},\vec{\mathbf{w}}}\sum_{c=1}^{\mathcal{C}}\sum_{k=1}^{K}\gamma(z_{ck})(\log\pi_k + \sum_{d=1}^{D_c}\log\sigma_{cdk})$$

To find the new value of π_k, we only need to maximize the first part of the above function

$$\boldsymbol{\pi}^{new} = \arg\max_{\boldsymbol{\pi}}\sum_{c=1}^{\mathcal{C}}\sum_{k=1}^{K}\gamma(z_{ck})\log\pi_k$$

subject to the constraint $\sum_{k=1}^{K}\pi_k = 1$

Using Lagrange multiplier and setting the gradient to 0, one can solve the optimal values of π_k as

$$\pi_k{}^{new} = \frac{\sum_{c=1}^{\mathcal{C}}\gamma(z_{ck})}{\sum_{k=1}^{K}\sum_{c=1}^{\mathcal{C}}\gamma(z_{ck})} \tag{14}$$

Searching for the value of $\vec{w}_k^{new}$ is a bit trickier, since we have to solve the following optimization problem.

$$\vec{\mathbf{w}}^{new} = \arg\max_{\vec{\mathbf{w}}}\sum_{c=1}^{\mathcal{C}}\sum_{d=1}^{D_c}\sum_{k=1}^{K}\gamma(z_{ck})\log\sigma_{cdk} \tag{15}$$

In fact, the gradient of the above objective function with respect to $\vec{w}_k$ is equal to:

$$\sum_{c=1}^{\mathcal{C}}\sum_{d=1}^{D_c}\gamma(z_{ck})(1-\sigma_{cdk})y_{cd}\vec{x}_{cd}$$

Therefore, one can apply a gradient descent algorithm to find the maximized value of $\vec{w}_k$.

In the implementation of the algorithm discussed so far, there is an issue about $\gamma(z_{ck})$. As equation (12) has shown, computing $\gamma(z_{ck})$ involves calculating the product $\prod_{d=1}^{D_c}\sigma_{cdk}$. This could lead to numerical underflow since σ_{cdk} is a probability smaller than 1. Therefore, we need to calculate $\gamma(z_{ck})$ under the log space. Let

$$\gamma(z_{ck}) \propto \pi_k\prod_{d=1}^{D_c}\sigma_{cdk} = \alpha(z_{ck})$$

and $\alpha_{max} = \max_{k=1}^{K}\alpha(z_{ck})$. Therefore

$$\gamma(z_{ck}) = \frac{\alpha(z_{ck})}{\sum_{j=1}^{K}\alpha(z_{cj})} = \frac{\exp\{\log\alpha(z_{ck}) - \log\alpha_{max}\}}{\sum_{j=1}^{K}\exp\{\log\alpha(z_{ck}) - \log\alpha_{max}\}}$$

Since each $\log\alpha(z_{ck})$ is computable, the above equation will avoid the underflow problem. Finally, we extend the formulations to the set of training queries. In this case, the E-step becomes:

$$\gamma(z_{ck}) = \frac{\pi_k\prod_{q=1}^{|Q|}\prod_{d=1}^{D_c}\sigma_{cdk}^{q}}{\sum_{j=1}^{K}\pi_j\prod_{q=1}^{|Q|}\prod_{d=1}^{D_c}\sigma_{cdj}^{q}}$$

In the M-step, the update formula of π_k remains the same (equation (14)), while the gradient of the function in equation (15) with respect to $\vec{w}_k$ becomes

$$\sum_{c=1}^{\mathcal{C}}\sum_{q=1}^{|Q|}\sum_{d=1}^{D_c}\gamma(z_{ck})(1-\sigma_{cdk}^{q})y_{cd}^{q}\vec{x}_{cd}^{q}$$

5. EXPERIMENTAL METHODOLOGY

In this section, we will describe the methodology and datasets of this work. The experiments were conducted on three datasets: two standard TREC datasets, and one Wikipedia dataset for federated search based on the ClueWeb.

- **TREC123-100col-bysource (TREC123):** 100 collections (information sources) were created from TREC CDs 1,2 and 3 [4]. They are organized by publication source and publication date. This testbed comes with 100 queries (TREC topics 51-150) with judgments.
- **TREC4-100col-Kmeans (TREC4-Kmeans):** 100 collections were created from the TREC 4 data. A two-pass K-means clustering algorithm is used to organize the dataset by topic [33]. This testbed comes with 50 queries (TREC topics 201-250) with judgments.
- **Wikipedia-100col-Kmeans (ClueWeb-Wiki):** 100 collections were created from the Wikipedia dataset of the ClueWeb [13]. Similar to TREC4-Kmeans, we applied clustering algorithm [33] to divide the dataset into 100 collections. This testbed comes with 106 queries with judgments[1].

[1]The partition assignments are available at http://www.cs.purdue.edu/homes/dthong/clueweb

Table 2: Statistics of Three Testbeds

Testbed	Size (GB)	# of inf. sources	# of documents (x1000) Min	Avg	Max	# of queries	# of relevant docs/query Min	Avg	Max
TREC123	3.2	100	0.7	10.8	39.7	100	37	483.7	1,994
TREC4-Kmeans	2.0	100	0.3	5.7	82.7	50	0	127.2	416
ClueWeb-Wiki	252	100	4.4	58.6	434.5	106	1	20.1	93

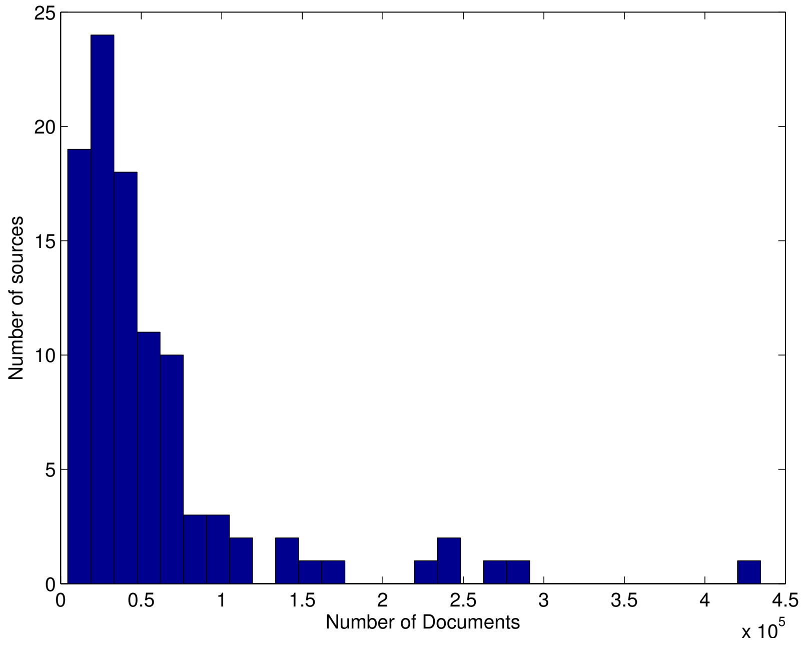


Figure 1: Histograms of the Number of Documents per Information Source in ClueWeb-Wiki. The number of bins is 30, the number of documents ranges from 4,400 to 434,525.

5.1 ClueWeb Wikipedia Dataset for Federated Search

Table 2 provides more statistics of the above datasets, including sizes, number of information sources; the max, min and average of the number of documents of each source. We also provide the query statistics of each dataset, including the number of queries; the max, min and average of the number of relevant documents per query.

The ClueWeb 09 is a large-scale collection of web documents that was collected in January and February 2009. The entire dataset consists of about one billion web pages in ten languages. For its tremendous size, the ClueWeb has been used in several tracks of the Text REtrieval Conference (TREC), most notably in the Web track. For distributed environment (in a different problem setting), ClueWeb has been used in [15]. It is desired to construct a new dataset based on ClueWeb for experiments in federated search.

Three Web tracks of TREC (from 2009 to 2011) have been using the ClueWeb so far. Each track has provided 50 queries based on which we build the new testbed. Within the full ClueWeb dataset, Wikipedia is the main contributor of relevant documents for Web track queries. The total size of Wikipedia is about 6 million documents, which is reasonable for creating a separate testbed. We extract all Wiki documents, and apply the same K-means algorithm that was used for creating the TREC4-Kmeans. We also select only 106 queries which contain at least one relevant Wikipedia document (out of the 150 provided queries) for training and testing. In the end, we constructed 100 information sources for the testbed ClueWeb-Wiki, with statistics provided in Table 2. The distribution of source sizes is also shown in Figure 1. Most of the sources have less 70,000 documents, and there are 12 sources of more than 100,000 documents.

5.2 Experiment Configuration

Given a set of information sources, we assign each source a retrieval algorithm chosen from a set of: vector space TF.IDF with "ltc" weighting [21], a unigram statistical language model with linear smoothing (with smooth parameter as 0.5) [16] and Okapi [20] in a round robin manner. This choice is based on the fact that those models are common and widely used in information retrieval. Each information source is set to return at most 200 documents for each query. At the centralized sample database, we utilize five models: the three above models, the Inquery [5] and the Indri [28] algorithm. All retrieval algorithm implementations use the Lemur Toolkit [14]. We randomly select a set of queries for training, and used the other set for testing. For the TREC123, there are 50 training queries out of 100; those numbers of TREC4-Kmeans and ClueWeb-Wiki are 25 out of 50 and 50 out of 106.

We choose K, the number of latent groups, to be 3 in our main results. Some experimental results with different K values are also presented. For each information source, we sample at most 300 documents for creating the centralized sample database. For each query, we use ReDDE to select the top 5 sources for TREC123, TREC4-Kmeans and ClueWeb-Wiki.

Our metrics for the performance is the high precision at document level, which is the percentage of the number of relevant documents in the final merged ranked list. Given that list, we measure the precision at top 5, 10, 15, 20 and 30 respectively. In next section, we will present our experimental results on all datasets.

6. EXPERIMENTAL RESULTS

6.1 High-precision Results

We now present the high-precision results on the above three testbeds. Tables 3-5 show the high-precision results on TREC123, TREC4-Kmeans and ClueWeb-Wiki respectively. The first column is our baseline using SAFE algorithm with Indri [18] as the single centralized retrieval algorithm. SAFE has been demonstrated to generate accurate and robust results compared with SSL and other results merging algorithms. We denote this method as SFI. The LR column presents the results using the logistic regression model to learn the combination weights of all centralized retrieval methods. The last column MoRM presents the results using the proposed mixture of retrieval algorithms. All precision results of LR and MoRM are compared with the baseline SFI using paired t-tests at level $p < 0.05$.

For TREC123, the performance of MoRM is significantly better than that of SFI. MoRM is also consistently better than the performance of logistics regression model. In general, both learning methods show improvements over the baseline method of one single feature. For TREC4-Kmeans, MoRM also outperforms SFI, although the differences are not significant as in TREC123. This can be explained as in TREC4-Kmeans, we only train on 25 queries, whereas in TREC123, we trained on 50 queries. These above results have shown the advantage of using multiple centralized retrieval algorithms for learning comparable document scores, over the previous model that uses only one single centralized retrieval algorithm. It also demonstrates the advantage of using the mixture model of multiple sets of weights over the logistic regression model that uses only one single set of combination weights.

For the Wikipedia dataset based on ClueWeb, the proposed model also consistently outperforms SFI and LR. The differences however are not significant. Such a significance may be harder to achieve, since on average, this dataset contains less relevant documents per query than the other datasets, as shown in Table 2.

6.2 Experiments with Different Number of Latent Variables

In this section, we discuss the experimental results when the number of latent variable K changes. We only report TREC123 and ClueWeb-Wiki for these experiments, and try different configuration of $K = \{1, 3, 5, 10\}$. Similar pattern can be observed on the TREC4-Kmeans. $K = 1$ is actually equivalent to the logistic regression model. Figure 2 shows the results of this experiment. It can be seen that the mixture of retrieval algorithms model is quite consistent with a small range of K values. For ClueWeb-Wiki, the performance of the mixture model with $K > 1$ is at least equal or higher than that of the logistic regression. For TREC123, the performances with different values of K are also stable for most of the test levels.

Table 3: High-precision result on TREC123 with 300 sample documents and top 5 information sources selected for each query. A * denotes a significant difference at level $p < 0.05$ compared to the original SAFE algorithm using Indri as the only centralized retrieval algorithm.

Doc	TREC123				
Rank	SFI	LR		MoRM	
@5	0.268	0.332	(+ 23.88 %)	**0.344**	(+ 28.36 %) *
@10	0.246	0.272	(+ 10.57 %)	**0.304**	(+ 23.58 %) *
@15	0.229	0.267	(+ 16.31 %)	**0.279**	(+ 21.54 %) *
@20	0.208	0.251	(+ 20.67 %) *	**0.261**	(+ 25.48 %) *
@30	0.208	0.229	(+ 9.95 %)	**0.232**	(+ 11.54 %)

Table 4: High-precision result on TREC4-Kmeans with 300 sample documents and top 5 information sources selected for each query. A * denotes a significant difference at level $p < 0.05$ compared to the original SAFE algorithm using Indri as the only centralized retrieval algorithm.

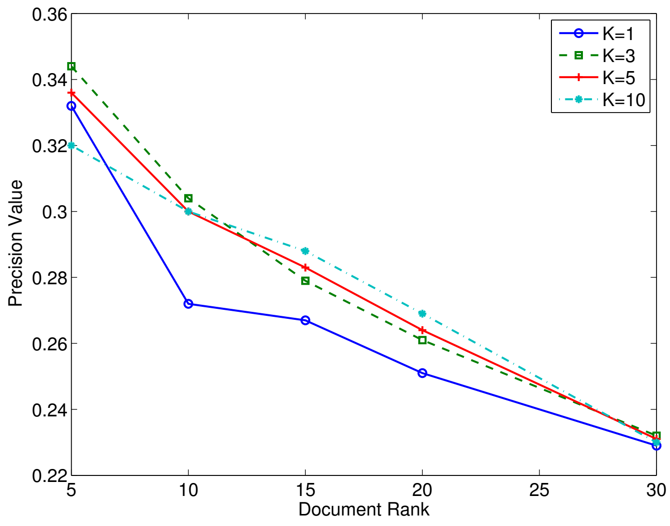
Doc	TREC4-Kmeans				
Rank	SFI	LR		MoRM	
@5	0.272	0.280	(+ 2.94 %)	**0.296**	(+ 8.82 %)
@10	0.244	0.252	(+ 3.28 %)	**0.256**	(+ 4.92 %)
@15	0.211	**0.227**	(+ 7.59 %)	**0.227**	(+ 7.59 %)
@20	0.192	0.212	(+ 10.42 %)	**0.216**	(+ 12.50 %)
@30	0.177	**0.193**	(+ 9.02 %)	0.192	(+ 8.29 %)

Table 5: High-precision result on ClueWeb-Wiki with 300 sample documents and top 5 information sources selected for each query. A * denotes a significant difference at level $p < 0.05$ compared to the original SAFE algorithm using Indri as the only centralized retrieval algorithm.

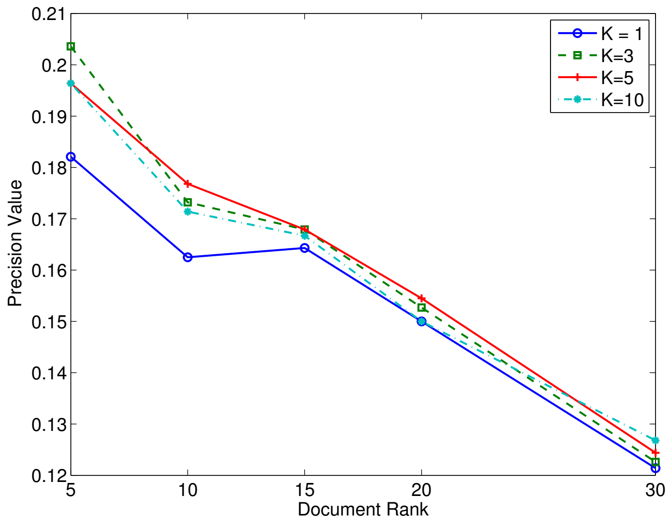
Doc	ClueWeb-Wiki				
Rank	SFI	LR		MoRM	
@5	0.168	0.182	(+ 8.46 %)	**0.204**	(+ 21.26 %)
@10	0.146	0.163	(+ 11.00 %)	**0.173**	(+ 18.31 %)
@15	0.139	0.164	(+ 17.95 %)	**0.168**	(+ 20.53 %)
@20	0.132	0.150	(+ 13.55 %)	**0.153**	(+ 15.59 %)
@30	0.111	0.121	(+ 9.67 %)	**0.123**	(+ 10.75 %)

7. CONCLUSION & FUTURE WORK

This paper proposes a novel method of mixture model with multiple centralized retrieval algorithms for result merging in federated search. Existing result merging algorithms do not fully address the issue of heterogeneity of information sources in federated search. Their arbitrary choices of a single centralized retrieval algorithm suffer from the fact that information sources are inherently different in source



(a) TREC123



(b) ClueWeb-Wiki

Figure 2: High-precision Results of TREC123 and ClueWeb-Wiki with Different Number of Latent Variables

statistics, query processing techniques, and/or document retrieval algorithms. The proposed model attempts to combine various evidence from multiple centralized retrieval algorithms in a mixture model framework, in order to map source-specific document ranks to comparable scores for result merging. We have shown that a single set of combination weights of the evidence do not offer enough flexibility in dealing with such a heterogeneous environment. A mixture model that learns multiple sets of combination weights according to the clusters of sources proves to be a better choice. A set of experiments has been conducted with two traditional TREC datasets and a new dataset based on the ClueWeb. The empirical results in three datasets have demonstrated the effectiveness of the proposed mixture model with multiple centralized retrieval algorithms.

This model could be extended in many ways. For instance, we could add more flexibility to the model by customizing the prior distribution π_k independently for each source, which means a source will be associated with a set of combination weights independent of the others. However, this model could require a larger training dataset to learn the parameters. A hybrid model where a cluster of similar sources independently uses multiple sets of weights is more feasible. The similarity between sources will play an important factor in creating those clusters. Another direction is to build a mixture model based on cluster of queries instead of cluster of sources, in which each query will trigger a different set of combination weights of all features. Furthermore, we can combine both of the above methods. It is also interesting to explore other types of evidence, such as the links between documents from different sources and incorporate them into the learning model.

8. ACKNOWLEDGMENTS

This work is partially supported by NSF research grants IIS-0746830, CNS- 1012208 and IIS-1017837. This work also partially supported by the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370.

9. REFERENCES

[1] J. Arguello, J. Callan, and F. Diaz. Classification-based resource selection. *Proceeding of the 18th ACM conference on Information and knowledge management*, pages 1277–1286, 2009.

[2] M. Baillie and M. Carman. A multi-collection latent topic model for federated search. *Information Retrieval*, 14(4):390–412, Aug. 2011.

[3] M. Bergman. The deep web: surfacing the hidden value. Technical report, 2001.

[4] J. Callan. Distributed information retrieval. *Advances in Information Retrieval*, pages 127–150, 2000.

[5] J. Callan, W. B. Croft, and S. M. Harding. The inquery retrieval system. In *Proceedings of the Third International Conference on Database and Expert Systems Applications*, 1992.

[6] N. Craswell, D. Hawking, and P. Thistlewaite. Merging results from isolated search engines. In *Proceedings of the 10th Austrlasian Database Conference*, 1999.

[7] A. P. Dempster, N. M. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(B):1–38, 1977.

[8] R. Fletcher. *Practical methods of optimization, volume 1.* Wiley, 1987.

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

[10] L. Gravano, H. Garcia-Molina, and A. Tomasic. Gloss: text-source discovery over the internet. *ACM Transactions on Database Systems (TODS)*, 24(2):229–264, 1999.

[11] J. Gross. *Linear regression*, volume 175. Springer Verlag, 2003.

[12] C. He, D. Hong, and L. Si. A weighted curve fitting method for result merging in federated search. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 1177–1178, New York, NY, USA, 2011. ACM.

[13] http://lemurproject.org/clueweb09/. The clueweb09 dataset.

[14] http://www.lemurproject.org/. The lemur toolkit.

[15] A. Kulkarni and J. Callan. Document allocation policies for selective searching of distributed indexes. *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 449–458, 2010.

[16] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 111–119, 2001.

[17] W. Meng and C. Yu. Advanced metasearch engine technology. *Synthesis Lectures on Data Management*, 2(1):1–129, 2010.

[18] D. Metzler and W. B. Croft. Combining the language model and inference network approaches to retrieval. *Information Processing and Management*, 40(5):735–750, 2004.

[19] Y. Rasolofo, F. Abbaci, and J. Savoy. Approaches to collection selection and results merging for distributed information retrieval. *Proceedings of the tenth international conference on Information and knowledge management*, pages 191–198, 2001.

[20] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. *NIST SPECIAL PUBLICATION SP*, pages 109–109, 1995.

[21] G. Salton, E. Fox, and H. Wu. Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, 1983.

[22] M. Shokouhi. Central-rank-based collection selection in uncooperative distributed information retrieval. *Advances in Information Retrieval*, 2007.

[23] M. Shokouhi and L. Si. Federated search. 2011.

[24] M. Shokouhi and J. Zobel. Robust result merging using sample-based score estimates. *ACM Transactions on Information Systems (TOIS)*, 27(3):1–29, 2009.

[25] X. M. Shou and M. Sanderson. Experiments on data fusion using headline information. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 413–414, New York, NY, USA, 2002. ACM.

[26] L. Si and J. Callan. Relevant document distribution estimation method for resource selection. *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 298–305, 2003.

[27] L. Si and J. Callan. A semisupervised learning method to merge search engine results. *ACM Transactions on Information Systems (TOIS)*, 21(4):457–491, 2003.

[28] T. Strohman, D. Metzler, H. Turtle, and C. W. B. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligence Analysis*, 2004.

[29] P. Thomas. Server selection in distributed information retrieval: a survey. *To appear in: Journal of Information Retrieval*, 2012.

[30] M. Tsai, H. Chen, and Y. Wang. Learning a merge model for multilingual information retrieval. *Information Processing & Management*, 47(5):635–646, 2011.

[31] Y. Wang and D. J. DeWitt. Computing pagerank in a distributed internet search system. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 420–431. VLDB Endowment, 2004.

[32] S. Wu, Y. Bi, and X. Zeng. The linear combination data fusion method in information retrieval. In *Database and Expert Systems Applications*, pages 219–233. Springer, 2011.

[33] J. Xu and J. Callan. Effective retrieval with distributed collections. *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 112–120, 1998.

[34] J. Xu and W. B. Croft. Cluster-based language models for distributed retrieval. *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 254–261, 1999.