Federated Search of Text-Based Digital Libraries in Hierarchical Peer-to-Peer Networks

Jie Lu, Jamie Callan

School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA
{jielu, callan}@cs.cmu.edu

Abstract. Peer-to-peer architectures are a potentially powerful model for developing large-scale networks of text-based digital libraries, but peer-to-peer networks have so far provided very limited support for text-based federated search of digital libraries using relevance-based ranking. This paper addresses the problems of resource representation, resource ranking and selection, and result merging for federated search of text-based digital libraries in hierarchical peer-to-peer networks. Existing approaches to text-based federated search are adapted and new methods are developed for resource representation and resource selection according to the unique characteristics of hierarchical peer-to-peer networks. Experimental results demonstrate that the proposed approaches offer a better combination of accuracy and efficiency than more common alternatives for federated search in peer-to-peer networks.

1 Introduction

Peer-to-peer (P2P) networks are an appealing approach to federated search over large networks of digital libraries. The activities involved for search in peer-to-peer networks include issuing requests ("queries"), routing requests ("query routing"), and responding to requests ("retrieval"). The nodes in peer-to-peer networks can participate as clients and/or servers. Client nodes issue queries to initiate search in peer-to-peer networks; server nodes provide information contents, respond to queries with documents that are likely to satisfy the requests, and/or route queries to other servers.

The first peer-to-peer networks were based on sharing popular music, videos, and software. These types of digital objects have relatively obvious or well-known naming conventions and descriptions, making it possible to represent them with just a few words from a name, title, or manual annotation. From a Library Science or Information Retrieval perspective, these systems were designed for *known-item* searches, in which the goal is to find a single instance of a known object (e.g., a particular song by a particular artist). In a known item search, the user is familiar with the object being requested, and any copy is as good as any other. Known-item search of popular music, video, and software file-sharing systems is a task for which simple solutions suffice. If P2P systems are to scale to more varied content and larger digital libraries, they must adopt more sophisticated solutions.

A very large number of text-based digital libraries were developed during the last decade. Nearly all of them use some form of relevance ranking, in which term frequency information is used to rank documents by how well they satisfy an unstructured text query. Many of them allow free search access to their contents via the Internet, but do not provide complete copies of their contents upon request. Many do not allow their contents to be crawled by Web search engines. How best to provide federated search across such independent digital libraries is an unsolved problem often referred to as the "Hidden Web" problem.

This paper addresses the problem of using peer-to-peer networks as a federated search layer for text-based digital libraries. We start by assuming the current state of the art; that is, we assume that each digital library is a text database running a reasonably good conventional search engine, and providing individual documents in response to full text queries. Furthermore, we assume that each digital library cooperatively provides accurate resource description of its content upon request. We present in this paper how resource descriptions of digital libraries are used for efficient query routing, and how results from different digital libraries are merged into a single, integrated ranked list in P2P networks. It is worth noting that the general framework described in this paper for text-based federated search of digital libraries in cooperative environments also applies to uncooperative environments, although different approaches are required for acquiring resource descriptions and result merging.

In the following section we give an overview of the prior research on federated search of text-based digital libraries and P2P networks. Section 3 describes our approaches to federated search of text-based digital libraries in P2P networks. Sections 4 and 5 discuss our data resources and evaluation methodologies. Experimental settings and results are presented in Section 6. Section 7 concludes.

2 Overview

Accurate and efficient federated search in P2P networks of text-based digital libraries requires both the appropriate P2P architecture and effective search methods developed for the chosen architecture. In this section we present an overview of the prior research on federated search of text-based digital libraries, P2P network architectures, and text-based search in P2P networks in order to set the stage for the descriptions of our approaches to text-based federated search in peer-to-peer networks.

2.1 Federated Search of Text-Based Digital Libraries

Prior research on federated search of text-based digital libraries (also called "distributed information retrieval") identifies three problems that must be addressed:

- Resource representation: Discovering the contents or content areas covered by each resource ("resource description");
- Resource ranking and selection: Deciding which resources are most appropriate for an information need based on their resource descriptions; and
- Result merging: Merging retrieval results from a set of selected resources.

A directory service is responsible for acquiring resource descriptions of the digital libraries it serves, selecting the appropriate resources (digital libraries) given the query, and merging the retrieval results from selected resources into a single, integrated ranked list. Distributed information retrieval offers solutions to all three problems for the case of a single directory service. We briefly review them below.

Resource Representation. Different techniques for acquiring resource descriptions require different degrees of cooperation from digital libraries. STARTS is a cooperative protocol that requires every digital library to provide an accurate resource description to the directory service upon request [5]. Query-based sampling is an alternative approach to acquiring resource descriptions without requiring explicit cooperation from digital libraries [2]. The resource description of a digital library is constructed by sampling its documents via the normal process of submitting queries and retrieving documents.

Resource Ranking and Selection. Resource selection aims at selecting a small set of resources that contain a lot of documents relevant to the information request. Resources are ranked by their likelihood to return relevant documents and top-ranked resources are selected to process the information request.

Resource selection algorithms such as CORI [2], gGlOSS [6], and the Kullback-Leibler (K-L) divergence-based algorithm [21] treat the resource description of a digital library as a document and use techniques adapted from document retrieval for resource ranking. Other resource selection algorithms including ReDDE [17] and the decision-theoretic framework for resource selection [12] rank resources by directly estimating the number of relevant documents from each resource for a given query.

Deciding how many top-ranked resources to search ("thresholding") is usually simplified to use of a heuristic value (e.g., 5 or 10).

Result Merging. Result merging algorithms can be divided into two categories: Approaches based on normalizing resource-specific document scores into resource-independent document scores, and approaches based on recalculating resource-independent document scores at the directory service.

The CORI and the Semi-Supervised Learning result merging algorithm belong to the first category. The CORI merging algorithm uses a heuristic linear combination of the digital library score and the document score to produce a resource-independent document score [2]. The Semi-Supervised Learning result merging algorithm uses the documents obtained by query-based sampling as training data to learn score normalizing functions [16].

Usually in order to recalculate document scores, the directory service needs to download all the documents in the retrieval results. Downloading documents is not necessary if all the statistics required for score recalculation can be obtained alternatively. Kirsch's algorithm [9] requires each resource to provide summary statistics for each of the retrieved documents. It allows very accurate normalized document scores to be determined without the high communication cost of downloading.

2.2 P2P Network Architectures

As mentioned in Section 1, the activities involved for search in peer-to-peer networks include issuing queries, query routing, and retrieval. Query routing is essentially a problem of resource selection and location. Resource location in first generation P2P networks is characterized by Napster, which used a single logical directory service, and Gnutella 0.4, which used undirected message flooding and a search horizon. The former proved easy to attack, and the latter didn't scale. They also explored very different solutions: Napster was centralized and required cooperation (sharing of accurate information); Gnutella 0.4 was decentralized and required little cooperation.

Recent research provides a variety of solutions to the flaws of the Napster and Gnutella 0.4 architectures, but perhaps the most influential are hierarchical and structured P2P architectures. Structured P2P networks associate each data item with a key and distribute keys among directory services using a Distributed Hash Table (DHT) [13, 15, 18, 19]. Hierarchical P2P networks [8, 10, 20] use the top-layer of directory services to serve regions of the bottom-layer of digital libraries, and directory services work collectively to cover the whole network. The common characteristic of both approaches is the construction of an overlay network to organize the nodes that provide directory services for efficient query routing. An important distinction is that structured P2P networks require the ability to map (via a distributed hash table) from an information need to the identity of the directory service that satisfies the need, whereas hierarchical P2P networks rely on message-passing to locate directory services. Structured P2P networks require digital libraries to cooperatively share descriptions of data items in order to generate keys and construct distributed hash tables. In contrast, hierarchical P2P networks enable directory services to automatically discover the contents of (possibly uncooperative) digital libraries, which is well-matched to networks that are dynamic, heterogeneous, or protective of intellectual property.

2.3 Text-Based Search in P2P Networks

Most of the prior research on search in peer-to-peer networks only supports simple keyword-based search. There has been some recent work on developing systems that adopt more sophisticated retrieval models to support text-based search (also called "content-based retrieval") in peer-to-peer networks. Examples are PlanetP using a completely decentralized P2P architecture [4], pSearch using a structured P2P architecture [19], and content-based retrieval in hierarchical P2P networks [11].

In PlanetP [4], a node uses a TF.IDF algorithm to decide which nodes to contact for information requests based on the compact summaries it collects about all other nodes' inverted indexes. Because no special resources are dedicated to support directory services in completely decentralized P2P architectures, it is somewhat inefficient for each node to collect and store information about the contents of all other nodes.

pSearch [19] uses the semantic vector (generated by Latent Semantic Indexing) of each document as the key to distribute document indices in a structured P2P network. To compute semantic vectors for documents and queries, global statistics such as the inverse document frequency and the basis of the semantic space need to be disseminated to each node in the network, which makes this approach difficult to be extended to uncooperative and heterogeneous environments.

Content-based resource selection and document retrieval algorithms for a single directory service are extended to multiple directory services in [11]. Experimental results demonstrate that content-based resource selection and document retrieval can provide more accurate and more efficient solutions to federated search in P2P networks of text-based digital libraries than the flooding and keyword-based approaches.

3 Text-Based Federated Search in Hierarchical P2P Networks

The research described in this paper adopts a hierarchical P2P architecture because it provides a flexible framework to incorporate various solutions to resource selection and result merging. Following the terminology of prior research, we refer to text-based digital libraries as "leaf" nodes, and directory services as "hub" nodes. Each leaf node is a text database that provides functionality to process full text queries by running a document retrieval algorithm over its index of a local document collection and generating responses. Each hub acquires and maintains necessary information about its neighboring hubs and leaf nodes and uses it to provide resource selection and result merging services to a P2P network. In addition to leaf nodes and hubs, there are also "client" nodes representing users with information requests. Leaf nodes and client nodes only connect to hubs and hubs connect with each other.

Search in peer-to-peer networks relies on message-passing between nodes. A request message ("query") is generated by a client node and routed from a client node to a hub, from one hub to another, or from a hub to a leaf node. A response message ("queryhit") is generated by a leaf node and routed back along the query path in reverse direction. Each message in the network has a time-to-live (TTL) field that determines the maximum number of times it can be relayed in the network. The TTL is decreased by 1 each time the message is routed to a node. When the TTL reaches 0, the message is no longer routed. The initial value of TTL was 6 in our experiments.

When a client node has an information request, it sends a query message to each of its connecting hubs. A hub that receives the query message uses its resource selection algorithm to rank and select one or more neighboring leaf nodes as well as hubs. A leaf node that receives the query message uses its document retrieval algorithm (K-L divergence document retrieval algorithm in our experiments) to generate a relevance ranking of its documents and responds with a queryhit message to include a list of top-ranked documents. Each top-level hub (the hub that connects directly to the client node that issues the request) collects the queryhit messages and uses its result-merging algorithm to merge the documents retrieved from multiple leaf nodes into a single, integrated ranked list and returns it to the client node. If the client node issues the request to multiple hubs, it also needs to merge results returned by top-level hubs.

In this paper we assume a "static" network setting (i.e., fixed topology without node failure) so that we can focus on the solutions to resource representation, resource ranking and selection, and result merging for higher efficiency and accuracy in federated search of hierarchical P2P networks. Previous research has demonstrated that with a simple set of network reorganization protocols, content-based federated search in hierarchical P2P networks is robust in terms of unexpected hub failures [14].

3.1 Resource Representation

The description of a resource is a very compact summary of its content. Compared with the complete index of a collection of documents, resource description requires much less communication and storage costs but still provides useful information for resource selection algorithms to determine which resources are more likely to contain documents relevant to the query. The resource description used by most resource selection algorithms include a list of terms with corresponding term frequencies, and corpus statistics such as the total number of terms and documents provided or covered by the resource. The resource here could be a single leaf node, a hub that covers multiple leaf nodes, or a "neighborhood" that includes all the nodes reachable from a hub. Different methods are required to acquire different types of resources descriptions.

Resource Descriptions of Leaf Nodes. Resource descriptions of leaf nodes are used by hubs for query routing ("resource selection") among connecting leaf nodes. In this paper we focus on cooperative environments where leaf nodes provide accurate resource descriptions to connecting hubs (e.g., STARTS [5]).

Resource Descriptions of Hubs. The resource description of a hub is the aggregation of the resource descriptions of its connecting leaf nodes. Since hubs work collaboratively in hierarchical P2P networks, neighboring hubs can exchange with each other their aggregate resource descriptions. However, because the aggregate resource descriptions of hubs only have information for nodes within 1 hop, if they are directly used by a hub to decide which neighboring hubs to route query messages to, the routing would not be effective when the nodes with relevant documents sit beyond this "horizon". Thus for effective hub selection, a hub must have information about what contents can be reached if the query message it routes to a neighboring hub may further travel multiple hops. This kind of information is referred to as the resource description of a neighborhood and is introduced in the following subsection.

Resource Descriptions of Neighborhoods. A *neighborhood* of a hub H_i in the direction of its neighboring hub H_j is a set of hubs that can be reached by following the path from H_i to H_j. The resource description of a neighborhood provides information about the contents covered by all the hubs in this neighborhood. A hub uses resource descriptions of neighborhoods to route queries to its neighboring hubs.

Resource descriptions of neighborhoods provide similar functionality as routing indices [3]. An entry in a routing index records the number of documents that may be found along a path for a set of topics. The key difference between resource descriptions of neighborhoods and routing indices is that resource descriptions of neighborhoods represent contents with unigram language models (terms with their frequencies). Thus by using resource descriptions of neighborhoods, there is no need for hubs and leaf nodes to cluster their documents into a set of topics and it is not necessary to restrict queries to topic keywords.

Similar to exponentially aggregated routing indices [3], a hub calculates the resource description of a neighborhood by aggregating the resource descriptions of all the hubs in the neighborhood decayed exponentially according to the number of hops so that contents located nearer are weighted more highly. For example, in the resource description of a neighborhood $N_{i,j}$ (the neighborhood of H_i in the direction of H_j), a term t's exponentially aggregated frequency is:

$$\sum_{H_k \in N_{i,j}} \{tf(t, H_k) / F^{[numhops(H_i, H_k) - 1]}\} \ . \quad \textbf{(1)}$$

where $tf(t, H_k)$ is t's term frequency in the resource description of hub H_k, and F is the average number of hub neighbors each hub has in the network.

The exponentially aggregated total number of documents in a neighborhood is:

$$\sum_{H_k \in N_{i,j}} \{numdocs(H_k) / F^{[numhops(H_i, H_k) - 1]}\} \ . \quad \textbf{(2)}$$

The creation of resource descriptions of neighborhoods requires several iterations at each hub and different hubs can run the creation process asynchronously. A hub H_i in each iteration calculates and sends to its hub neighbor H_j the resource description of neighborhood $N_{j,i}$ denoted by $ND_{j,i}$ by aggregating its hub description HD_i and the most recent resource descriptions of neighborhoods it receives from all of its neighboring hubs excluding H_j. $ND_{j,i}$ is calculated as:

$$ND_{j,i} = HD_i + \sum\nolimits_{H_k \in directneighbors(H_i) \backslash H_j} \{ND_{i,k} / F\} \ . \quad \textbf{(3)}$$

The stopping condition could be either the number of iterations reaching a predefined limit, or the difference in resource descriptions between adjacent iterations being small enough. The maximum number of iterations was 6 in our experiments.

The process of maintaining and updating resource descriptions of neighborhoods is identical to the process used for creating them. The resource descriptions of neighborhoods could be updated periodically, or when the difference between the old and the new value is significant.

For networks that have cycles, frequencies of some terms and the number of documents may be overcounted, which will affect the accuracies of resource descriptions. How to deal with cycles in peer-to-peer networks using routing indices is discussed in detail in [3]. We could use the same solutions described in [3] for cycle avoidance or cycle detection and recovery. For simplicity, in this paper, we take the "no-op" solution, which completely ignores cycles. Experimental results show that resource selection using resource descriptions of neighborhoods generated in networks with cycles is still quite efficient and accurate.

3.2 Resource Ranking and Selection

Query routing aims at an optimal cost-effective solution to directing the information request to those nodes that are most likely to contain relevant documents. In this paper, the cost of query routing is measured by the number of messages carrying the information request (query messages). The flooding technique guarantees to reach

nodes with relevant information contents but requires an exponential number of query messages. Randomly forwarding the request to a small subset of neighbors can significantly reduce the number of query messages but the reached nodes may not be relevant at all. To achieve both efficiency and accuracy, each hub needs to rank its neighboring leaf nodes by their likelihood to satisfy the information request, and neighboring hubs by their likelihood to reach nodes with relevant information contents, and only forwards the request to the top-ranked neighbors. Because the resource descriptions of leaf nodes and those of neighborhoods are not in the same magnitude in vocabulary size and term frequency, direct comparison between leaf nodes and neighborhoods would not be fair. Therefore, a hub handles separately the ranking and selection of its neighboring leaf nodes and hubs.
In this paper all the hubs are required to use the same resource ranking/selection methods. In future work, we are interested in studying the performance of federated search in P2P networks when this requirement is relaxed.

Leaf Node Ranking. Adapting language modeling approaches for ad-hoc information retrieval, we use the Kullback-Leibler (K-L) divergence-based method [21] for leaf node ranking, which calculates the conditional probability $P(L_i \mid Q)$ of predicting the collection of leaf node L_i given the query Q and uses it to rank different leaf nodes.

$$P(L_i \mid Q) = \frac{P(Q \mid L_i) \times P(L_i)}{P(Q)} \propto P(Q \mid L_i) \cdot \tag{4}$$

with uniform prior probability for leaf nodes;

$$P(Q \mid L_i) = \prod_{q \in Q} \frac{tf(q, L_i) + \mu \times P(q \mid G)}{numterms(L_i) + \mu} \cdot \tag{5}$$

where $tf(q, L_i)$ is the term frequency of query term q in leaf node L_i's resource description (collection language model), $P(q \mid G)$ is the background language model used for smoothing and μ is the smoothing parameter in Dirichlet smoothing.

Leaf Node Selection with Unsupervised Threshold Learning. After leaf nodes are ranked based on their $P(L_i \mid Q)$ values, the usual approach is to select the top-ranked leaf nodes up to a predetermined number. In hierarchical P2P networks, the number of leaf nodes served by individual hubs may be quite different, and different hubs may cover different content areas. In this case, it is not appropriate to use a static, hub-independent number as the threshold for a hub to decide how many leaf nodes to select for a given query. It is desirable that each hub has the ability to learn its own selection threshold automatically.

The problem of learning the threshold to convert relevance ranking scores into a binary decision has mostly been studied in information filtering [22]. One approach to learn the threshold is to find an optimal threshold value that maximizes a given utility function based on the distributions of the scores of relevant and non-relevant documents. However, it requires user relevance feedback as training data, which is not easily available for federated search in peer-to-peer networks. Our goal is to develop a technique for each hub to learn the selection threshold without supervision based on the information and functionality it already has. In peer-to-peer networks, because each hub has the ability to merge the retrieval results from multiple leaf nodes into a single, integrated ranked list, as long as the result merging has reasonably good performance, we could assume that the top-ranked merged documents are relevant. If we further assume that when a hub merges the documents returned by all of its leaf nodes for a training query, a leaf node with at least n documents among the top-ranked merged documents at this hub is a relevant leaf node with respect to the query and non-relevant otherwise, then we can define a utility function based on the distributions of the normalized scores of relevant and non-relevant leaf nodes at this hub for a set of queries, and the leaf node selection threshold is one that maximizes the value of this utility function.

To be more specific, a linear utility function $U(\theta)$ is defined as below and the optimal value θ^* that maximizes $U(\theta)$ is used as the threshold for leaf node selection.

$$U(\theta) = N_{rel}(\theta) - N_{nonrel}(\theta) - w \times N(\theta) \ , \ \theta^* = \arg\max_{\theta} U(\theta) \ . \qquad \mathbf{(6)}$$

where $N_{rel}(\theta)$ and $N_{nonrel}(\theta)$ are the number of relevant and non-relevant leaf nodes respectively whose normalized scores are above threshold θ, $N(\theta)$ is the total number of leaf nodes with normalized scores above threshold θ, and w is the weight to control the tradeoff between accuracy and efficiency.

Because larger $N(\theta)$ leads to more selected leaf nodes and thus low efficiency, the term $w \times N(\theta)$ is included in $U(\theta)$ to penalize low efficiency since efficiency is equally important as accuracy for federated search in most peer-to-peer environments.

At a hub, the number of relevant and non-relevant leaf nodes with normalized scores above threshold θ can be calculated as:

$$N_{rel}(\theta) = \alpha \times \int_{\theta}^{1} P(s \wedge rel) ds = \alpha \times \int_{\theta}^{1} P(s \mid rel) \times P(rel) ds \ . \qquad \mathbf{(7)}$$

$$N_{nonrel}(\theta) = \alpha \times \int_{\theta}^{1} P(s \wedge nonrel) ds = \alpha \times \int_{\theta}^{1} P(s \mid nonrel) \times (1 - P(rel)) ds \ . \qquad \mathbf{(8)}$$

where $P(s \wedge rel)$ is the probability of a leaf node having score s and being relevant, $P(s \wedge nonrel)$ is the probability of a leaf node having score s and being non-relevant, $P(s \mid rel)$ is the probability of a relevant leaf node having score s, $P(s \mid nonrel)$ is the probability of a non-relevant leaf node having score s, $P(rel)$ is the probability of a leaf node being relevant, and α is the total number of leaf nodes in training data.

The relevancy of a leaf node depends on whether it has at least n (empirically chosen to be 5 in our experiments) documents among the top-ranked merged documents at a hub. $P(s \mid rel)$ and $P(s \mid nonrel)$ at a hub can be estimated from the scores of relevant and non-relevant leaf nodes for a set of training queries. In information filtering, the score distributions of relevant and non-relevant documents are fitted using a Gaussian distribution and an exponential distribution respectively [1]. However, in our experience the score distributions of relevant and non-relevant leaf nodes at a hub $P(s \mid rel)$ and $P(s \mid nonrel)$ cannot be fitted very well by distributions such as exponential and Gaussian. For this reason, instead of fitting continuous distributions to the training data, each hub uses the empirical discrete score distributions learned from a set of training queries. Experimental results not shown in this paper verified that using the discrete score distributions had better performance (and was simpler) than using the fitted exponential score distributions.

Usually $P(rel)$ is estimated with maximum likelihood estimation using training data. However, in our experiments, because the amounts of training data for relevant and non-relevant leaf nodes are very unbalanced, using maximum likelihood estimation for $P(rel)$ yielded poor performance. Therefore, here we assume that all the leaf nodes connecting to a hub have equal probability of being relevant and non-relevant, i.e., $P(rel)$ is 0.5. This is a reasonable assumption when each hub covers a specific content area so that all of its connecting leaf nodes have somewhat similar contents.

$N(\theta)$ as a function of θ can be fitted quite well by an exponential function whose parameters are learned from the leaf node ranking results of a set of training queries.

Hub Ranking and Selection. The K-L divergence resource selection algorithm used for leaf ranking is also used for hub ranking. The resource descriptions of neighborhoods are used to calculate the collection language models needed by the resource selection algorithm. For hub selection, because selecting a neighboring hub is essentially selecting a neighborhood, using a prior distribution that favors larger neighborhood could lead to better search performance, which was indeed the case in our experiments. Thus the prior probability of a neighborhood is set to be proportional to the exponentially aggregated total number of documents in the neighborhood. Given the query Q, the probability of predicting the neighborhood N_i that a neighboring hub H_i represents is calculated as follows and used for hub ranking:

$$P(N_i \mid Q) = \frac{P(Q \mid N_i) \times P(N_i)}{P(Q)} \propto P(Q \mid N_i) \times numdocs(N_i) \; . \tag{9}$$

$$P(Q \mid N_i) = \prod_{q \in Q} \frac{tf(q, N_i) + \mu \times P(q \mid G)}{numterms(N_i) + \mu} \; . \tag{10}$$

where $tf(q, N_i)$ is the term frequency of query term q in the resource description of neighborhood N_i (collection language model), $P(q \mid G)$ is the background language model for smoothing and μ is the smoothing parameter in Dirichlet smoothing.

A fixed number of top-ranked neighboring hubs are selected. It remains to be future work to apply unsupervised threshold learning to hub selection.

3.3 Result Merging

As described earlier, result merging takes place at each top-level hub. Kirsch's algorithm [9] is extended for result merging in P2P networks which requires each resource to provide summary statistics for each of the retrieved documents, for example, document length and how often each query term matched. The corpus statistics come from the aggregation of the hub's resource description and the resource descriptions of neighborhoods for its neighboring hubs. Documents are merged according to the document scores recalculated by a K-L divergence retrieval algorithm using the above document and corpus statistics.

If the client node issues the request to multiple hubs, it also needs to merge the results returned by these hubs. Because client nodes don't maintain information about the contents of other nodes and corpus statistics as hubs do, they can only use simple, but probably less effective, merging methods. In this paper, client nodes directly use the document scores returned by top-level hubs to merge results.

4 Test Data

We used the P2P testbed [11] based on the TREC WT10g web test collection to evaluate the performance of federated search in hierarchical P2P networks of text-based digital libraries. The P2P testbed consists of 2,500 collections based on document URLs. The total number of documents in these 2,500 collections is 1,421,088. Each collection defines a leaf node in a hierarchical P2P network.

There are 25 hubs in the P2P testbed, each of which covers a specific type of content. The connections between leaf nodes and hubs were determined by clustering leaf nodes into 25 clusters using a similarity-based soft clustering algorithm, and connecting all the leaf nodes within a cluster to the hub associated with this cluster.

The connections between hubs were generated randomly. Each hub has no less than 1 and no more than 7 hub neighbors. A hub has on average 4 hub neighbors.

Experiments were run on two sets of queries. The first set of queries came from the title fields of TREC topics 451-550. The standard TREC relevance assessments supplied by the U. S. National Institute for Standards and Technology were used.

The second set of queries was a set of 1,000 queries selected from the queries defined in the P2P testbed. Queries in the P2P testbed were automatically generated from WT10g data by extracting key terms from the documents in the collection. Because it is expensive to obtain relevance judgments for these automatically generated queries, we treated 50 top-ranked documents retrieved using a single large collection for each query as the "relevant" documents ("single collection" baseline), and measured how well federated search in the hierarchical P2P network could locate and rank these documents. The single large collection was the subset of the WT10g used to define the contents of the 2,500 leaf nodes (WT10g-subset).

For each query, a leaf node was randomly chosen to act as a client node temporarily to issue the query and collect the merged retrieval results for evaluation.

5 Evaluation Methodology

A version of the JavaSim network simulator [7] was developed to evaluate the performance of text-based federated search in hierarchical P2P networks.

Both retrieval accuracy and query routing efficiency were used as performance measures. Precisions at document ranks 5, 10, 15, 20, 30, and 100 were used to measure retrieval accuracy. The efficiency of query routing was measured by the average number of query messages (messages to carry the information requests) routed for each query in the network.

6 Experiments and Results

A series of experiments was conducted to study text-based federated search in cooperative P2P environments. Three hub selection methods were compared: the flooding method (a hub broadcasting query messages to all of its hub neighbors), random hub selection (a hub randomly selecting some of its hub neighbors for query routing) and hub selection based on resource descriptions of neighborhoods. Two leaf node selection methods were compared: selecting a fixed percentage of top-ranked leaf nodes and selecting using the learned thresholds. Unsupervised threshold learning required a set of training queries. For each experiment that used threshold-based leaf node selection to run the 100 TREC queries, two runs were conducted with each using half of the 100 TREC queries for training and half for testing. The results from two runs were averaged to get the final results. For the experiments that used threshold-based leaf node selection to run the 1,000 WT10g queries, the 100 TREC queries were used as training data. Unsupervised threshold learning only used queries and the retrieved documents for training. The NIST relevance judgments for the 100 TREC queries were not used to learn thresholds. The weight *w* for unsupervised threshold learning was adjusted so as to yield similar overall query routing efficiency as selecting a fixed percentage (1%) of top-ranked leaf nodes. The number of top-ranked merged documents used for unsupervised threshold learning was 50.

Tables 6.1 and 6.2 show respectively the results of running the 100 TREC queries and the 1,000 WT10g queries for text-based federated search in a hierarchical P2P network using different methods. The column marked by "Msgs" shows the average number of query messages routed for each query. Precisions at different document ranks are shown in columns 4-9. The "single collection" baseline (Section 4) is also shown in Table 6.1 for the 100 TREC queries.

Because the "single collection" baseline was used as relevance judgment for WT10g queries, the retrieval performance on this set of queries directly measured the ability of federated search in the hierarchical P2P network to match the results from search in a centralized environment. The high precisions at top document ranks in Table 6.2 demonstrates that federated search in the hierarchical P2P network was able to locate most documents that were considered relevant by the centralized approach, which is an encouraging sign for federated search in peer-to-peer networks considering that only around 1% of the 2,500 digital libraries were actually searched.

If we compare the figures in Table 6.1 with those in Table 6.2, we can see that although the absolute values were quite different, the relative performance difference of different algorithms for the 1,000 WT10g queries was similar to that for the 100 TREC queries. Therefore the same conclusions drawn from the results of the 100 TREC queries could be drawn from the results of the 1,000 WT10g queries regarding the relative effectiveness of various algorithms, which indicates that the automatically generated queries and the "single collection" baseline are useful resources in studying federated search in peer-to-peer networks.

The results in Tables 6.1 and 6.2 demonstrate that compared with using the flooding technique for hub selection, hub selection based on resource descriptions of neighborhoods required around one third of the number of query messages with only a minor drop in search performance, irrespective of how hubs ranked and selected leaf nodes. Hub selection based on resource descriptions of neighborhoods and random hub selection gave similar query routing efficiency but the retrieval accuracy of the former was consistently higher than the latter.

The power of the peer-to-peer system using the learned thresholds for leaf node selection lies in its ability to adapt the thresholds automatically to different hubs in order to obtain better performance. As shown in Tables 6.1 and 6.2, with the same hub selection method, using leaf node selection with the learned thresholds in general gave better performance for text-based federated search in the hierarchical P2P network than selecting a fixed percentage of top-ranked leaf nodes for each hub.

Table 6.1. Search performance evaluated on the 100 TREC queries

Hub Selection	**Leaf Selection**	**Msgs**	**P@5**	**P@10**	**P@15**	**P@20**	**P@30**	**P@100**
Centralized	Centralized	N/A	0.324	0.287	0.255	0.241	0.208	0.175
Flooding	Top 1%	177	0.263	0.205	0.179	0.168	0.147	0.084
Flooding	Threshold	180	0.274	0.223	0.196	0.179	0.159	0.094
Random 1	Top 1%	63	0.240	0.191	0.170	0.154	0.130	0.066
Random 1	Threshold	65	0.252	0.205	0.177	0.159	0.137	0.067
Top 1	Top 1%	59	0.259	0.196	0.176	0.163	0.139	0.080
Top 1	Threshold	55	0.280	0.218	0.192	0.179	0.153	0.086

Table 6.2. Search performance evaluated on the 1,000 WT10g queries

Hub Selec-	**Leaf Selec-**	**Msgs**	**P@5**	**P@1**	**P@1**	**P@2**	**P@3**	**P@1**
tion	**tion**			**0**	**5**	**0**	**0**	**00**
Flooding	Top 1%	174	0.970	0.942	0.915	0.875	0.792	0.281
Flooding	Threshold	171	0.990	0.967	0.942	0.913	0.835	0.289
Random 1	Top 1%	60	0.874	0.809	0.753	0.698	0.595	0.198
Random 1	Threshold	55	0.891	0.831	0.773	0.718	0.616	0.195
Top 1	Top 1%	54	0.949	0.904	0.857	0.804	0.701	0.237
Top 1	Threshold	43	0.964	0.912	0.863	0.810	0.707	0.226

7 Conclusions and Future Work

This paper studies federated search of text-based digital libraries in hierarchical P2P networks. Although some existing approaches to resource representation, resource ranking and selection, and result merging for text-based federated search can be adapted to P2P environments in a straightforward manner, new development is still required to suit the solutions to the unique characteristics of hierarchical P2P networks. For example, in hierarchical P2P networks, hub ranking and selection should be based on not only the hub's likelihood to provide relevant documents with its own leaf nodes, but also its potential to reach other hubs that are likely to satisfy the information request. Thus new methods are needed to represent the contents covered by the available resources in the networks. In this paper, we define the concept of neighborhood and describe a method to create and use resource descriptions of neighborhoods for hub ranking and selection. Experimental results demonstrate that hub ranking and selection based on resource descriptions of neighborhoods offers a better combination of accuracy and efficiency than the alternative flooding and random selection.

Another unique character of hierarchical P2P networks is that there are multiple hubs and each hub must make local decisions on selecting from the leaf nodes it covers to satisfy the information request. Because hubs are different in the number of leaf nodes and the content areas they cover, which could also change dynamically as nodes come and leave or change connections, the ability for hubs to learn automatically hub-specific resource selection thresholds in the networks is much desired. This motivated us to develop an approach for each hub to learn its own threshold in an unsupervised manner based on the retrieval results of a set of training queries. In our experiments the proposed approach was consistently more accurate than the typical method of selecting a fixed number of top-ranked leaf nodes with similar efficiency.

The results in this paper also provide additional support for using the automatically generated queries and the "single collection" baseline to evaluate the search performance in P2P networks. The same conclusions on the relative effectiveness of various algorithms for federated search in P2P networks can be drawn from the results of the 1,000 WT10g queries and from the results of the 100 TREC queries. This is encouraging because the large number of queries automatically generated from WT10g (in the magnitude of 10^6) give us the opportunity to study in the future how the network can learn from past queries and evolve to improve the search performance over time.

Federated search in distributed environments is complicated, the main components of which include resource representation, resource selection, document retrieval and result merging. The overall search performance is affected by the performance of individual components as well as the interaction between them. P2P networks add further complexity to the problem due to factors such as dynamic topology, uncertainty in locating relevant information, and efficiency concerns. How the data are distributed over the network and how different nodes interact with each other also affect the use of different algorithms. Our next step is to further understand the unique characteristics of P2P networks and to develop practical algorithms that are more appropriate for search in dynamic and heterogeneous P2P networks.

Acknowledgements

This material is based on work supported by NSF grant IIS-0240334. Any opinions, findings, conclusions or recommendations expressed in this material are the authors', and do not necessarily reflect those of the sponsor.

References

1. A. Arampatzis, J. Beney, C. Koster and T. van der Weide. KUN on the TREC-9 Filtering Track: Incrementality, decay, and threshold optimization for adaptive filtering systems. In *Proc. of the 9th Text REtrieval Conference*, 2001.
2. J. Callan. Distributed information retrieval. W. B. Croft, editor, *Advances in information retrieval*, chapter 5, pages 127-150. Kluwer Academic Publishers, 2000.
3. A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *Proc. of the International Conference on Distributed Computing Systems (ICDCS)*, July 2002.

4. F. Cuenca-Acuna and T. Nguyen. Text-based content search and retrieval in ad hoc p2p communities. Technical Report DCS-TR-483, Rutgers University, 2002.
5. L. Gravano, C. Chang, H. Garcia-Molina and A. Paepcke. STARTS: Stanford proposal for internet meta-searching. In *Proc. of the ACM-SIGMOD International Conference on Management of Data*, 1997.
6. L. Gravano and H. Garcia-Molina. Generalizing GlOSS to vector-space databases and broker hierarchies. In *Proc. of 21th International Conference on Very Large Data Bases (VLDB'95)*, pages 78-89, 1995.
7. Javasim. http://javasim.ncl.ac.uk/.
8. KaZaA. http://www.kazaa.com.
9. S. T. Kirsch. Document retrieval over networks wherein ranking and relevance scores are computed at the client for multiple database documents. U.S. Patent 5,659,732.
10. Limewire. http://www.limewire.com.
11. J. Lu and J. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *Proc. of the 12nd International Conference on Information Knowledge Management*, 2003.
12. H. Nottelmann and N. Fuhr. Evaluation different methods of estimating retrieval quality for resource selection. In *Proc. of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2003.
13. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of the ACM SIGCOMM'01 Conference*, August 2001.
14. M. Elena Renda and J. Callan. The robustness of content-based search in hierarchical peer-to-peer networks. In *Proc. of the 13rd International Conference on Information Knowledge Management*, 2004.
15. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329-350, 2001.
16. L. Si and J. Callan. A semi-supervised learning method to merge search engine results. *ACM Transactions on Information Systems*, 24(4), pages 457-491. ACM.
17. L. Si and J. Callan. Relevant document distribution estimation method for resource selection. In *Proc. of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2003.
18. I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of the ACM SIGCOMM'01 Conference*, August 2001.
19. C. Tang, Z. Xu and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *Proc. of the ACM SIGCOMM'03 Conference*, 2003.
20. S. Waterhouse. JXTA Search: Distributed search for distributed networks. Technical report, Sun Microsystems Inc., 2001.
21. J. Xu and W. B. Croft. Cluster-based language models for distributed retrieval. In *Proc. of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.
22. Y. Zhang and J. Callan. Maximum likelihood estimation for filtering thresholds. In *Proc. of 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.