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

Jie Lu
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
jielu@cs.cmu.edu

Jamie Callan
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
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 two 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 are both more accurate and more efficient than more common alternatives for text-based federated search in peer-to-peer networks.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Retrieval models, Search process, Selection process

General Terms
Algorithms, Design, Experimentation, Performance

Keywords
Peer-to-peer, Hierarchical, Federated Search, Text-Based, Retrieval, Digital Library

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, or even complete title lists for their contents, upon request. Many do not allow their contents to be crawled by Web search engines. They do not cooperate by conforming to a single method of text representation, query processing, or document retrieval; they don't even provide information about how these operations are done. We would argue that most of the recent research on peer-to-peer networks offers little useful guidance for providing federated search of current text-based digital libraries.

This paper addresses the problem of using peer-to-peer networks as a federated search layer for text-based digital libraries. We study federated search in two different types of environments: cooperative environments where each digital library provides accurate resource description of its content upon request, and uncooperative environments where resource descriptions must be obtained indirectly. 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, that it provides search access to its holdings, and that it provides individual documents in response to full text queries. We present in this paper how resource descriptions of digital libraries are obtained and used for efficient query routing, and how results from different digital libraries are merged into a single, integrated ranked list in peer-to-peer networks.

In the following section we give an overview of the prior research on federated search of text-based digital libraries and peer-to-peer networks. Section 3 describes our approaches to federated search of text-based digital libraries in peer-to-peer 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 peer-to-peer networks of text-based digital libraries requires both the appropriate peer-to-peer architecture and the 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, peer-to-peer network architectures, and text-based search in peer-to-peer 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" in the research literature) 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 ranked 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. Solutions to all these three problems for the case of a single directory service have been developed in distributed information retrieval. We briefly review them below.

2.1.1 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 [6]. STARTS is a good solution in environments where cooperation can be guaranteed. However, in some environments where digital libraries may not cooperate or may have an incentive to cheat, STARTS cannot be used to acquire accurate resource descriptions.

Query-based sampling is an alternative approach to acquiring resource descriptions without requiring explicit cooperation from digital libraries. The resource description of a digital library is constructed by sampling its documents via the normal process of submitting queries and retrieving documents. Query-based sampling has been shown to acquire fairly accurate resource descriptions using a small number of queries and documents in distributed information retrieval environments [1].

The total number of documents of a digital library is one of the most important corpus statistics required by many resource selection algorithms. Capture-Recapture [12] and Sample-Resample [20] are two methods of estimating the total number of documents of an uncooperative digital library. Experimental results show that in most scenarios, Sample-Resample is more accurate and has less communication costs than the Capture-Recapture method.

2.1.2 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 [1], gGlOSS [7], and Kullback-Leibler (K-L) divergence-based algorithms [24] use techniques adapted from document retrieval for resource ranking. The resource description of a digital library used by these algorithms includes a list of terms with corresponding collection term frequencies, and corpus statistics such as the total number of terms and documents in the collection. These algorithms have been shown to work well with resource descriptions provided by cooperative digital libraries or acquired using query-based sampling.

Other resource selection algorithms including ReDDE [20] and DTF (the decision-theoretic framework for resource selection) [16] rank resources by directly estimating the number of relevant documents from each resource for a given query. ReDDE relies on sampled documents obtained using query-based sampling for such estimation. DTF has three variants DTF-rp, DTF-sample and DTF-normal. DTF-rp estimates the number of relevant documents from a resource by assuming a linearly decreasing recall-precision function and calculating the expected precision and recall from the resource. DTF-sample uses sampled documents to estimate how relevant documents are distributed among the available resources. DTF-normal models the distribution of document scores from a resource with normal distribution and map document scores to probability of relevance using a function learned with user relevance feedback.

Deciding how many top-ranked resources to be selected ("thresholding") is a problem that is usually simplified. Most resource selection algorithms use heuristic values such as 10 and 20 for the number of selected resources.

2.1.3 Result Merging

Many result-merging algorithms have been proposed in distributed information retrieval. Various approaches can be divided into two categories: approaches based on normalizing resource-specific document scores into resource-independent document scores, and approaches based on recalculating document scores at the directory service.

The CORI merging algorithm uses a heuristic linear combination of digital library scores and document scores to normalize the scores of the documents from different digital libraries. The intuition is to favor documents from digital libraries with high scores and also to enable high-scoring documents from low-scoring digital libraries to be ranked highly. It is effective when used together with the CORI resource selection and INQUERY document retrieval algorithms in federated search using a single directory service [1].

There has been some work on using logistic regression to learn merging models to normalize document scores but relevance judgments are required for training [2].

The Semi-Supervised Learning result-merging algorithm uses the documents obtained by query-based sampling as training data to learn score normalizing functions on a query-by-query basis. It is shown to work well with a variety of resource selection and document retrieval algorithms and is the current state-of-the-art for result merging in distributed information retrieval [19].

Document scores can be recalculated at the directory service by downloading all the documents in the retrieval results from

selected resources, indexing them, and re-ranking them using a document retrieval algorithm.

Downloading documents is not necessary if all the statistics required for score recalculation can be obtained alternatively. Kirsch's algorithm [10] 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.

The corpus statistics required for recalculating document scores could also be substituted by a reference statistics database containing all the relevant statistics for some set of documents. This method is explored in [3] for federated search using a single directory service and shown to be effective compared with using the corpus statistics provided by cooperative digital libraries.

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 peer-to-peer 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; both systems demonstrated the importance of robust and reliable methods of locating information in peer-to-peer networks. 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 architecture associates each data item with a key and distributes keys among directory services using a Distributed Hash Table (DHT) [17, 18, 21, 22, 28]. Hierarchical P2P architecture [9, 11, 23] uses top-layer directory services to serve regions of bottom-layer 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 (also called "look up services" by DHT-based approaches) 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 support simple keyword-based search. Matches between query terms and keywords of documents are used to determine how to route queries and which documents to be retrieved. 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 PlantP using a completed decentralized P2P architecture [5], pSearch using a structured P2P architecture [22], and content-based retrieval in hierarchical P2P networks [13].

In PlantP [5], 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, especially in dynamic P2P networks.

pSearch [22] uses the semantic vector (generated by Latent Semantic Indexing) of each document as the key to distribute document index in a structured P2P network so that documents close in distance have similar contents. The relevance of a document to a query is determined by the similarity between their semantic vectors. 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. Because global statistics can only be obtained in completely cooperative environments where each digital library shares its document and corpus statistics, this approach cannot be easily extended to uncooperative and heterogeneous environments.

There has been some prior research on content-based resource selection and document retrieval in hierarchical P2P networks of digital libraries [13]. Viewing peer-to-peer networks as a particular type of distributed information retrieval environment, content-based resource selection is extended to the case of multiple directory services in peer-to-peer environments where digital libraries cooperatively provide resource descriptions to connecting directory services. Experimental results demonstrate that content-based resource selection and document retrieval can provide more accurate and more efficient solutions to federated search in peer-to-peer networks of text-based digital libraries compared with the flooding and keyword-based approaches.

The problem of result merging in hierarchical P2P networks of uncooperative and barely-cooperative text-based digital libraries has also been studied in [15]. The Semi-Supervised Learning (SSL) result-merging algorithm is modified and an algorithm Score Estimation with Sample Statistics (SESS) which extends Kirsch's approach to result merging is proposed. Experimental results show that modified SSL has satisfactory precision for top-ranked merged documents, and SESS is able to provide near optimal performance with a small amount of cooperation from digital libraries.

3. TEXT-BASED FEDERATED SEARCH IN HIERARCHICAL P2P NETWORKS

The research described in this paper adopts a hierarchical P2P architecture because it provides a more flexible framework to incorporate various solutions to resource selection and result merging in both cooperative and uncooperative environments. 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

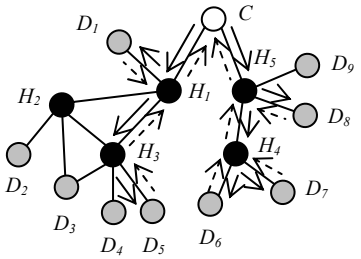

Figure 3.1 Federated search in hierarchical P2P networks.

retrieval algorithm over its index of local document collection and generate responses. Each hub acquires and maintains necessary information about its neighboring hub and leaf nodes and uses it to provide resource selection and result merging services to peer-to-peer networks. In addition to leaf nodes and hubs, there are also nodes representing users with information requests in peer-to-peer networks. They are referred to as "client" nodes. In a hierarchical P2P network, leaf nodes and client nodes can 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.

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 and routes the query to them if the message's TTL hasn't reached 0. A leaf node that receives the query message uses its document retrieval algorithm 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 more than one hub, then it also needs to merge results returned by multiple top-level hubs.

Figure 3.1 illustrates federated search of text-based digital libraries in hierarchical P2P networks. The *C* (white) node is the client node that issues the information request, the *H* (black) nodes are hubs, and the *D* (gray) nodes are leaf nodes (digital libraries). The edges between nodes represent connections. The arrows with solid lines indicate the directions to send query messages and the arrows with dashed lines indicate the directions to send queryhit messages.

In the following subsections, we present in more details the solutions to the problems of resource representation, resource ranking and selection, and result merging in both cooperative and uncooperative peer-to-peer environments.

3.1 Resource Representation

The description of a resource is a very compact summary of its content. Compared with a copy of 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. As mentioned in Section 2.1.2, the resource description used by most resource selection algorithms include a list of terms with corresponding term frequencies (collection language model), 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 neighboring leaf nodes, or a "neighborhood" that include all the nodes reachable from a hub. Although resource descriptions for different types of resources have the same format, different methods are required to acquire them, which we introduce below.

3.1.1 Resource Descriptions of Leaf Nodes

Resource descriptions of leaf nodes are used by hubs for query routing ("resource selection") among connecting leaf nodes. In cooperative environments, each leaf node provides accurate resource description to its connecting hubs upon request. In uncooperative environments, each hub conducts query-based sampling independently to obtain sampled documents from its connecting leaf nodes. Sampled documents from a leaf node are used to generate its collection language model. They are also used by the Sample-Resample method to estimate the total number of documents in this leaf node's collection.

3.1.2 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 has 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.

3.1.3 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. Figure 3.2 illustrates the concept of neighborhood. Hub H_1 has three neighboring hubs H_2, H_3 and H_4. Thus it has three neighborhoods marked by $N_{1,2}$, $N_{1,3}$ and $N_{1,4}$. 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 select and route queries to its neighboring hubs.

Resource descriptions of neighborhoods provide similar functionality as routing indices [4]. 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

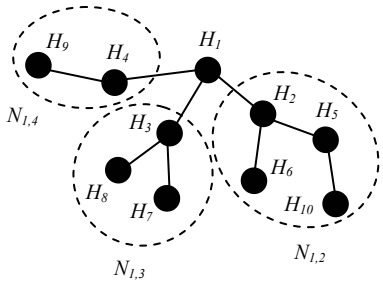

Figure 3.2 Neighborhoods in hierarchical P2P networks.

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 as exponentially aggregated routing indices [4], 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. 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 term frequency is calculated as:

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

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 calculated as:

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

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:

$$ND_{j,i} = HD_i + \sum_{H_k \in directneighbors(H_i) \setminus H_j} \{ND_{i,k} / F\} \quad (3)$$

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 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 when the difference between the old and the new value is significant, or periodically, or when a node disconnects from the network.

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 [4]. We could use the same solutions described in [4] 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

The goal of query routing is to direct the information request to those nodes that are most likely to contain relevant documents with minimum number of query messages. The flooding technique guarantees to reach nodes with relevant information contents but requires exponential number of query messages. Random 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 top-ranked neighbors. Because the resource descriptions of leaf nodes and those of neighborhoods are not in the same magnitude, a hub handles separately the ranking and selection of its neighboring leaf nodes and hubs.

3.2.1 Leaf Node Ranking

Adapting language modeling approaches for ad-hoc information retrieval, we use the Kullback-Leibler (K-L) divergence-based method [24] for leaf node ranking. In the language modeling framework, the K-L divergence resource selection algorithm calculates $P(L_i \mid Q)$, the conditional probability of predicting the collection of leaf node L_i given the query Q and uses it to rank different leaf nodes. $P(L_i \mid Q)$ is calculated as follows:

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

with uniform prior probability for leaf nodes;

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

where $tf(q \mid L_i)$ 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.

3.2.2 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, query-independent and hub-independent number as threshold for a hub to decide how many leaf nodes to select for a given query. It is desirable that hubs have the ability to learn hub-specific and query type-specific thresholds automatically.

The problem of learning threshold to convert relevance ranking scores into a binary decision has mostly been studied in information filtering [25, 26, 27]. However, the user relevance

feedback required as training data is not as easily available for federated search in peer-to-peer networks as for the task of information filtering. Our goal is to develop a technique for each hub to learn selection threshold without supervision based on the information and functionality it already has. 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 so, the distribution of the top-ranked merged documents over the leaf nodes should provide useful hints on the number of relevant documents each leaf node is likely to retrieve. This is analogous to query expansion with pseudo-relevance feedback which treats the top-ranked documents retrieved initially as relevant documents and uses them to improve the quality of the query. The key differences are i) our approach uses the information about which top-ranked merged documents are from which leaf nodes and ignores the actual contents of these documents, and ii) the direct goal here is not to improve immediately the retrieval quality for current query, but to learn resource selection thresholds that are specific to hubs and types of queries and improve the overall retrieval performance for a set of queries.

For leaf node selection, if a hub selects more leaf nodes than necessary, although the retrieval results will include a lot of irrelevant documents, as long as there are enough relevant documents, a reasonably good result merging algorithm can rank most relevant documents above irrelevant documents, yielding good precisions at top-ranked documents. In this case, it seems that a loose threshold will almost always give good performance. However, a loose threshold leads to low efficiency and high communication costs. Because for search in peer-to-peer networks, accuracy and efficiency are equally important, the resource selection threshold must be not too loose in order to guarantee efficiency, and not too tight as well so that enough relevant documents are returned (high recall). With the above criteria in mind, a hub uses the following procedure to decide the threshold of leaf node selection for a query:

1. Given a query, the hub uses K-L divergence resource selection algorithm to calculate leaf node scores and sorts them in descending order;
2. The hub selects up to 100 top-ranked leaf nodes and normalizes their scores using the formula:

$$S' = \frac{S - S_{\min}}{S_{\max} - S_{\min}} \quad (6)$$

   where $S_{\max}$ is the maximum score and $S_{\min}$ is the minimum score among these selected leaf nodes;
3. The hub forwards the query to selected leaf nodes and merges the retrieval results returned by these leaf nodes;
4. The hub calculates for each selected leaf node the number of documents that are ranked among top 50 in the merged result;
5. The hub goes down the list of leaf nodes sorted by their scores and stops at the leaf node which has the largest number of documents ranked among top 50 in the merged results (highest recall using pseudo-relevance feedback);
6. The hub regards the normalized score of this leaf node as the threshold of its leaf node selection for the given query.

Learning thresholds for individual queries is not useful unless the same queries appear again. Thus queries need to be classified into different types and thresholds for individual queries are used to compute thresholds for different query types. Queries can be classified based on their contents or statistical properties. When the number of queries for training is small (which is desired due to its low communication cost), classifying queries by contents often leads to sparse and skewed training data for various query types. Hence in our experiments we focused on classifying queries by their statistical properties and found the average probability of the query terms in a hub's resource description to be a good feature for query classification. Given a set of training queries that have average probabilities of query terms in different ranges, probability values ranging from 0 to the maximum term probability in a hub's resource description are divided into 10 non-overlapping bins so that all bins have roughly the same number of queries for training. A query type is associated with each bin, so there are 10 query types in total. A query is classified into one of these 10 types based on the average probability of its terms in the hub's resource description.

During the learning phase, each hub in the network learns the thresholds for a set of training queries and the learned thresholds for queries of the same type are averaged to get the threshold for this query type at the hub. Given a new query, a hub determines the type of the query, ranks up to 100 leaf nodes, normalizes their scores, and uses the query type-specific threshold to select the leaf nodes that have normalized scores no less than the threshold.

3.2.3 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 node H_i represents is calculated as follows and used to rank neighboring hubs:

$$P(N_i \mid Q) = \frac{P(Q \mid N_i) \times P(N_i)}{P(Q)} \propto P(Q \mid N_i) \times numdocs(N_i) \quad (7)$$

$$P(Q \mid N_i) = \prod_{q \in Q} \frac{tf(q, N_i) + \mu \times P(q \mid G)}{numterms(N_i) + \mu} \quad (8)$$

where $tf(q \mid N_i)$ 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 used 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. In cooperative environments, Kirsch's algorithm [10] is

extended for result merging in peer-to-peer networks. In addition to a list of retrieved documents, each resource is required to provide summary statistics for each of the retrieved documents, for example, document length and how often each query term matched. The corpus statistics comes from the aggregation of the hub's resource description and the resource descriptions of neighborhoods for all its neighboring hubs.

The modified Semi-Supervised Learning algorithm (modified SSL) [15] is used for result merging in uncooperative environments. Each hub along the query path contributes to result merging by providing document statistics for "overlap" documents, which are documents that appear both in the sampled documents maintained at the hub for its leaf node neighbors and in the retrieval results sent to the hub by these neighbors. Top-level hubs use these document statistics provided by collaborative hubs to recalculate document scores for "overlap" documents and pair them with their original scores returned in the retrieval results to use as training data for learning score normalizing functions. The main difference between result merging in cooperative environments and that in uncooperative environments is that in cooperative environments leaf nodes provide document statistics for all the retrieved documents to top-level hubs, while in uncooperative environments, hubs provide document statistics for a subset of retrieved documents ("overlap" documents) to top-level hubs.

If the client node issues the request to more than one hub, then it also needs to merge results returned by multiple top-level hubs. Because client nodes don't maintain information about the contents of other nodes and corpus statistics as hubs do in hierarchical P2P networks, they cannot use advanced result-merging algorithms. Thus only simple, but probably less effective, merging methods can be applied at client nodes. For example, results can be merged based on the document scores returned by top-level hubs ("raw score merge") or in a round robin fashion.

4. TEST DATA

We used the P2P testbed [14] developed based on the TREC WT10g web test collection [8] to evaluate the performance of federated search in hierarchical P2P networks of text-based digital libraries. The P2P testbed consists of 2,500 collections obtained by dividing WT10g data into 11,485 collections based on document URLs and randomly selecting 2,500 of them. The total number of documents in these 2,500 collections is 1,421,088. Each collection defines a leaf node (digital library) in a hierarchical P2P network.

There are 25 hubs in total 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, associating each cluster with a hub, and connecting all the leaf nodes within a cluster to the associated hub.

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.

Table 4.1 summarizes some statistics for the testbed.

Experiments were run on two sets of queries. The first set of queries came from the title fields of TREC topics 451-550 used for TREC-8 and TREC-9 Web Tracks. The standard TREC relevance assessments supplied by the U. S. National Institute for Standards and Technology were used.

Table 4.1 Summary statistics for the testbed.

	min	avg	max
Number of documents for a leaf node	8	568	26,505
Number of leaf nodes for a hub	10	376	1,008
Number of hubs a leaf node connects to	1	4	12

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. Table 4.2 shows the distribution of query lengths among the selected 1,000 queries. Table 4.3 shows the distribution of term frequencies in WT10g for all the query terms in these 1,000 queries. Because it is expensive to obtain relevance judgments for these automatically generated queries, we used the ranked retrieval results from a single large collection as the baseline ("single collection" baseline), and measured how well federated search in the hierarchical P2P network could reproduce this baseline. The single large collection was the subset of the WT10g used to define the contents of the 2,500 leaf nodes in the peer-to-peer network, and the 50 top-ranked documents retrieved using this single large collection (WT10g-subset) were treated as the relevant documents for each query.

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

5. EVALUATION METHODOLOGY

A simulator was used to evaluate the performance of text-based federated search in hierarchical P2P networks. Both retrieval accuracy and query routing efficiency are used as performance measures.

5.1 Measuring Retrieval Accuracy

Retrieval accuracy was measured by both set-based and rank-based Recall and Precision. Set-based Recall and Precision are defined as follows:

$$\text{Recall} = |r| / |A| \quad (9)$$

$$\text{Precision} = |r| / |R| \quad (10)$$

where R is the set of the documents returned by retrieval in the P2P network, A is the set of relevant documents for a query among the 100 TREC queries, or the set of (up to 50) top-ranked documents returned by retrieval using the single WT10g-subset collection for a query among the 1,000 WT10g queries, and r is the intersection of R and A. $|\bullet|$ denotes the size of the set.

The quality of document rankings was measured using precisions

Table 4.2 Distribution of query length for 1,000 queries.

Length	1	2	3	4	5	6
Distribution	33%	33%	19%	7%	4%	4%

Table 4.3 Distribution of term frequency for 1,000 queries.

Frequency Scale	10^0	10^1	10^2	10^3	10^4	10^5	10^6
Distribution	1.7 %	5.5 %	10.6 %	25.5 %	31.8 %	22.5 %	2.4 %

Table 6.1 Choices of algorithms in the experiments.

	Algorithm
Leaf descriptions	Provided by leaf nodes in cooperative environments, OR Generated by hubs using documents sampled from leaf nodes by query-based sampling in uncooperative environments
Hub descriptions	Generated by hubs by aggregating leaf descriptions
Neighborhood descriptions	Generated by hubs by aggregating hub descriptions and exponentially decayed neighborhood descriptions over several iterations
Leaf node ranking	K-L divergence resource selection algorithm using leaf descriptions
Leaf node selection	1% of top-ranked leaf nodes, OR Fixed number of top-ranked leaf nodes, OR Top-ranked leaf nodes with normalized scores no less than the learned threshold (Section 3.2.2)
Hub ranking	K-L divergence resource selection algorithm using neighborhood descriptions
Hub selection	All neighboring hubs (flooding), OR 1 randomly selected neighboring hubs, OR Top-ranked neighboring hub
Document retrieval	K-L divergence document retrieval algorithm
Result merging at top-level hubs	Extended Kirsch's algorithm in cooperative environments, OR Modified Semi-Supervised Learning in uncooperative environments (Section 3.3)
Result merging at client node	Raw score merge (Section 3.3)

at document ranks 5, 10, 15, 20, 30, and 100.

Set-based Recall and Precision focus attention on how well text-based federated search in hierarchical P2P networks returns the "right" documents for a query, while rank-based metrics measure directly the performance of document ranking and result merging.

5.2 Measuring Query Routing Efficiency

The efficiency of query routing was measured by the average number of query messages routed for each query in the network. The average number of query messages routed from hubs to leaf nodes ("Hub-Leaf Messages") for each query was also used to measure the efficiency of leaf node selection in some experiments.

6. EXPERIMENTS AND RESULTS

A series of experiments was conducted to study resource selection and result merging in both cooperative ("COOP") and uncooperative ("UNCOOP") P2P environments. The choices of the algorithms used for resource representation, resource ranking and selection, document retrieval and result merging are shown in Table 6.1. Table 6.2 shows the values of some parameters used in our experiments.

Unsupervised threshold learning required a set of queries for training. For each experiment that used leaf node selection with unsupervised threshold learning to run the 100 TREC queries, two runs were conducted. The first run used the first half of the 100 TREC queries for training and the second half for testing. The second run worked the other way around. The results from two runs were averaged to get the final results. For the experiments that used leaf node selection with unsupervised threshold learning to run the 1,000 WT10g queries, the 100 TREC queries were used as training data. Unsupervised threshold learning only used queries and retrieved documents for training. The relevance judgments provided by NIST for the 100 TREC queries were not used to learn thresholds for leaf node selection.

Tables 6.3a and 6.3b 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. Both cooperative and uncooperative environments were studied. The "single collection" baseline which returned 50 top-ranked documents for each query by retrieval using the single WT10g-subset collection is also shown in Table 6.3a for the 100 TREC queries.

The following subsections present the analysis of the results from different perspectives.

6.1 Set-Based Recall/Precision vs. Precisions at Top Document Ranks

The set-based Precision figures (column 4) are much lower than one might expect because the number of relevant documents was very small (50 on average for the 100 TREC queries using relevance judgments and 50 maximum for the 1,000 WT10g queries using the "single collection" baseline), but the total number of retrieved documents was at least ten times larger for most queries in the hierarchical P2P network. This demonstrates a limitation of set-based Recall and Precision metrics for this task since generally users only care about the retrieval accuracy of top-ranked documents, but we include them as another way of comparing resource ranking and selection methods.

Compared with set-based Precision, the differences between precisions at top document ranks for federated search in the hierarchical P2P network and for search using a centralized index are smaller. This implies that both result merging algorithms for cooperative and uncooperative environments performed quite well by ranking most irrelevant documents lower than relevant documents in spite of low set-based Precision.

6.2 TREC Queries vs. WT10g Queries

In contrast to real queries and manual relevance judgments, the

Table 6.2 Parameter values used in the experiments.

Parameters	Values
Initial TTL for messages	6
Number of documents sampled from each leaf node	Up to 300
Number of resample queries used for Sample-Resample to estimate total number of documents	5
Number of iterations to create neighborhood descriptions	6
F (Average number of hub neighbors each hub has)	4
μ (Dirichlet smoothing parameter in K-L divergence resource selection)	1000
Number of documents retrieved from each leaf node	Up to 50

Table 6.3a Search performance evaluated on the 100 TREC queries using relevance judgments provided by NIST.

Environment	Hub Selects Hub	Hub Selects Leaf	Set-based Recall/Precision	# Query Messages	Precision @ 5	Precision @ 10	Precision @ 15	Precision @ 20	Precision @ 30	Precision @ 100
Centralized	N/A	N/A	26.58 / 17.54	N/A	0.324	0.287	0.255	0.241	0.208	0.175
COOP	Flooding	Top 1%	29.74 / 1.41	177	0.263	0.205	0.179	0.168	0.147	0.084
COOP	Random 1	Top 1%	21.76 / 1.41	63	0.240	0.191	0.170	0.154	0.130	0.066
COOP	Top 1	Top 1%	25.51 / 1.69	59	0.259	0.196	0.176	0.163	0.139	0.080
COOP	Flooding	Threshold	37.39 / 1.22	212	0.295	0.236	0.202	0.187	0.164	0.099
COOP	Random 1	Threshold	23.67 / 1.23	77	0.263	0.212	0.180	0.159	0.137	0.070
COOP	Top 1	Threshold	26.59 / 1.74	58	0.263	0.214	0.187	0.169	0.148	0.082
UNCOOP	Flooding	Top 1%	29.17 / 1.32	178	0.257	0.209	0.182	0.172	0.148	0.077
UNCOOP	Random 1	Top 1%	20.23 / 1.27	65	0.223	0.174	0.159	0.140	0.112	0.057
UNCOOP	Top 1	Top 1%	24.87 / 1.60	59	0.246	0.196	0.168	0.157	0.131	0.066
UNCOOP	Flooding	Threshold	39.56 / 1.11	224	0.275	0.230	0.199	0.185	0.164	0.094
UNCOOP	Random 1	Threshold	24.81 / 1.12	84	0.235	0.198	0.171	0.152	0.126	0.069
UNCOOP	Top 1	Threshold	30.94 / 1.50	70	0.261	0.218	0.188	0.168	0.146	0.081

Table 6.3b Search performance evaluated on the 1,000 WT10g queries using the "single collection" baseline.

Environment	Hub Selects Hub	Hub Selects Leaf	Set-based Recall/Precision	# Query Messages	Precision @ 5	Precision @ 10	Precision @ 15	Precision @ 20	Precision @ 30	Precision @ 100
COOP	Flooding	Top 1%	69.92 / 12.88	174	0.970	0.942	0.915	0.875	0.792	0.281
COOP	Random 1	Top 1%	50.55 / 12.50	60	0.874	0.809	0.753	0.698	0.595	0.198
COOP	Top 1	Top 1%	60.63 / 14.10	54	0.949	0.904	0.857	0.804	0.701	0.237
COOP	Flooding	Threshold	72.82 / 12.47	177	0.989	0.967	0.945	0.915	0.840	0.296
COOP	Random 1	Threshold	51.11 / 13.12	58	0.890	0.830	0.768	0.716	0.615	0.199
COOP	Top 1	Threshold	60.43 / 15.42	47	0.967	0.918	0.868	0.818	0.717	0.235
UNCOOP	Flooding	Top 1%	66.82 / 12.31	173	0.924	0.877	0.835	0.786	0.694	0.265
UNCOOP	Random 1	Top 1%	47.61 / 11.83	59	0.812	0.738	0.671	0.612	0.516	0.181
UNCOOP	Top 1	Top 1%	52.44 / 12.74	50	0.850	0.775	0.711	0.654	0.556	0.200
UNCOOP	Flooding	Threshold	69.61 / 11.85	186	0.942	0.900	0.857	0.811	0.724	0.277
UNCOOP	Random 1	Threshold	48.36 / 12.46	61	0.834	0.758	0.694	0.632	0.530	0.184
UNCOOP	Top 1	Threshold	52.49 / 13.81	47	0.862	0.789	0.723	0.662	0.565	0.203

1,000 WT10g queries were generated automatically by extracting key terms from documents and the top-ranked documents retrieved using a single centralized index were used for relevance judgments. When this set of queries was used to evaluate the performance of text-based federated search in hierarchical P2P networks, it directly measured the ability of federated search in hierarchical P2P networks to match the results from search in a centralized environment. The strong performance indicated by high precisions at top document ranks in Table 6.3b demonstrates that federated search in the hierarchical P2P network mostly agreed with the centralized approach on which documents were most relevant. Additional evaluations on the 100 TREC queries by treating the documents in the "single collection" baseline as relevant documents (the same evaluation methodology as we used for the 1,000 WT10g queries) gave very similar results (not shown in this paper due to space reason) as those in Table 6.3b. This is an encouraging sign for federated search in peer-to-peer networks because although distributed retrieval systems are not yet better than the "single collection" baseline, our results show that their performance can be pretty close at top-ranked documents.

However, we note that Table 6.3b gives slightly overly optimistic view of federated search quality, because in cases where federated search in the hierarchical P2P network disagreed with search using a centralized index, federated search was more likely to give high rank to an irrelevant document which was ranked lowly by centralized search. Therefore, the performance difference between federated search in the hierarchical P2P network and search using a centralized index is expected to be slightly larger if we evaluate them using real relevance judgments, as shown in Table 6.3a.

In order to claim that a peer-to-peer system being able to reproduce the "single collection" baseline quite well is an effective system for federated search, we need to rely on the assumption that search using a centralized index is effective in satisfying user's information needs, which is not necessarily the case. Due to this reason, we were concerned with whether automatically generated queries would behave similarly as real queries and whether the conclusions drawn using the "single collection" baseline for evaluation would still be valid with real relevance judgments. If we compare the figures in Table 6.3a with those in Table 6.3b, 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.

6.3 Cooperative vs. Uncooperative

The results in Tables 6.3a and 6.3b show that the search performance in uncooperative environments was comparable to that in cooperative environments, despite that in uncooperative environments hubs only obtained partial information about the content of each resource and used the score normalizing approach to result merging which was less accurate than score recalculation. This indicates that query-based sampling and Semi-Supervised Learning for result merging are effective techniques for federated search of text-based digital libraries in uncooperative peer-to-peer networks.

6.4 Hub Selection

The results in Tables 6.3a and 6.3b 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 minor drop in search performance, irrespective of whether leaf nodes were cooperative and 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.

If we focus on the set-based Recall and Precision for three methods of hub selection, it is clear that random hub selection led to great loss in Recall with almost no change in Precision compared with the flooding technique, while hub selection based on resource descriptions of neighborhoods had consistent improvement in Precision over the flooding technique. This indicates that hub selection based on resource descriptions of neighborhoods was very effective at selecting hubs that could reach the nodes most likely to satisfy the user's information need and hence there were less irrelevant documents returned.

6.5 Leaf Node Selection

The power of the peer-to-peer system using learned thresholds for leaf node selection lies in its ability to adapt the thresholds automatically to different hubs and types of queries in order to obtain better performance. There is no need to decide and tune manually the threshold values each time the system is put into a new environment. As shown in Tables 6.3a and 6.3b, with the same hub selection method under the same environment, using leaf node selection with learned thresholds in general gave better performance for text-based federated search in the hierarchical P2P network than selecting a fixed percentage (1%) of top-ranked leaf nodes for each hub.

Leaf node selection with learned thresholds produced better retrieval accuracy, but it also required more query messages. It is unclear from this set of experiments whether the higher accuracy is due to a better method of selecting leaf nodes (i.e., learned thresholds), or more thorough search (i.e., more messages). We ran additional experiments to further compare the performance of leaf node selection using learned thresholds with leaf node selection using fixed number. The results are shown in Tables 6.4a and 6.4b. To make the comparison more clear, the number of query messages sent from hubs to leaf nodes were extracted from the total number of query messages and averaged over queries to get "Hub-Leaf Messages". In Tables 6.4a and 6.4b, for each combination of environment type and hub selection method, the fixed number (*n* in "Top *n*") for leaf node selection was chosen to yield the smallest number of "Hub-Leaf Messages" that was larger than or equal to the number of "Hub-Leaf Messages" given by leaf node selection with learned thresholds (i.e., "Top *n*" yielded larger or equal number of "Hub-Leaf Messages" but "Top *n-1*" yielded smaller number of "Hub-Leaf Messages" than "Threshold" in the corresponding entries of the tables). Leaf node selection with learned thresholds worked consistently better for precisions at top-ranked documents with higher or equal efficiency for routing queries from hubs to leaf nodes. Therefore, with similar or smaller number of query messages, leaf node selection with learned thresholds still outperformed the simple solution of selecting a fixed number of top-ranked leaf nodes.

7. CONCLUSIONS AND FUTURE WORK

This paper studies federated search of text-based digital libraries in hierarchical peer-to-peer networks. Although some existing approaches to resource representation, resource ranking and selection, and result merging for text-based federated search can be adapted to peer-to-peer environments in a straightforward manner, new development is still in demand to suit the solutions to the unique characteristics of hierarchical peer-to-peer networks. For example, in hierarchical peer-to-peer 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 method is needed to represent the contents or content areas covered by the available resources in the networks. In this paper, we define the concept of neighborhood and describe the 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 is both more accurate and more efficient than the alternative flooding and random selection.

Another unique character of hierarchical peer-to-peer networks is that there are multiple hubs and each hub must make local decisions on selecting from the set of 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 and query type-specific thresholds in the networks is much desired. This motivated us to develop a new approach for each hub to learn its own thresholds for various types of queries 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 and more efficient than the typical method of selecting a fixed number of top-ranked leaf nodes. However, there is still much to be explored on how to effectively make use of the information obtained from resource selection and result merging by running a set of training queries and we believe that the search performance can be further improved.

Table 6.4a Comparison of leaf node selection methods tested on the 100 TREC queries. The best results in ranked-based retrieval accuracy for cooperative and uncooperative environments are shown in bold.

Environment	Hub Selects Hub	Hub Selects Leaf	Set-based Recall/Precision	# Hub-Leaf Messages	Precision @ 5	Precision @ 10	Precision @ 15	Precision @ 20	Precision @ 30	Precision @ 100
COOP	Flooding	Top 6	36.91 / 1.20	141	0.278	0.220	0.190	0.174	0.158	0.095
COOP	Flooding	Threshold	37.39 / 1.22	137	**0.295**	**0.236**	**0.202**	**0.187**	**0.164**	**0.099**
COOP	Random 1	Top7	26.01 / 1.22	70	0.255	0.194	0.166	0.153	0.134	0.074
COOP	Random 1	Threshold	23.67 / 1.23	63	0.263	0.212	0.180	0.159	0.137	0.070
COOP	Top 1	Top6	28.21 / 1.64	46	0.278	0.209	0.180	0.169	0.149	0.083
COOP	Top 1	Threshold	26.59 / 1.74	46	0.263	0.214	0.187	0.169	0.148	0.082
UNCOOP	Flooding	Top 7	37.53 / 1.09	163	**0.282**	0.225	0.191	0.177	0.159	0.091
UNCOOP	Flooding	Threshold	39.56 / 1.11	148	0.275	**0.230**	**0.199**	**0.185**	**0.164**	**0.094**
UNCOOP	Random 1	Top 7	25.62 / 1.09	70	0.221	0.185	0.164	0.147	0.127	0.066
UNCOOP	Random 1	Threshold	24.81 / 1.12	69	0.235	0.198	0.171	0.152	0.126	0.069
UNCOOP	Top 1	Top 8	30.41 / 1.42	61	0.253	0.198	0.180	0.163	0.139	0.077
UNCOOP	Top 1	Threshold	30.99 / 1.50	58	0.263	0.216	0.187	0.168	0.147	0.081

Table 6.4b Comparison of leaf node selection methods tested on the 1,000 WT10g queries. The best results in rank-based retrieval accuracy for cooperative and uncooperative environments are shown in bold.

Environment	Hub Selects Hub	Hub Selects Leaf	Set-based Recall/Precision	# Hub-Leaf Messages	Precision @ 5	Precision @ 10	Precision @ 15	Precision @ 20	Precision @ 30	Precision @ 100
COOP	Flooding	Top 5	74.21 / 11.83	111	0.973	0.949	0.930	0.899	0.828	**0.302**
COOP	Flooding	Threshold	72.82 / 12.47	102	**0.989**	**0.967**	**0.945**	**0.915**	**0.840**	0.296
COOP	Random 1	Top5	51.87 / 11.76	46	0.875	0.809	0.752	0.700	0.606	0.204
COOP	Random 1	Threshold	51.11 / 13.12	45	0.890	0.830	0.768	0.716	0.615	0.199
COOP	Top 1	Top5	61.17 / 14.15	36	0.949	0.903	0.859	0.803	0.703	0.240
COOP	Top 1	Threshold	60.43 / 15.42	36	0.967	0.918	0.868	0.818	0.717	0.235
UNCOOP	Flooding	Top 6	71.14 / 11.03	131	0.928	0.887	0.851	0.809	**0.727**	**0.285**
UNCOOP	Flooding	Threshold	69.61 / 11.85	110	**0.942**	**0.900**	**0.857**	**0.811**	0.724	0.277
UNCOOP	Random 1	Top 6	49.20 / 10.99	54	0.822	0.752	0.689	0.630	0.531	0.193
UNCOOP	Random 1	Threshold	48.36 / 12.46	47	0.834	0.758	0.694	0.632	0.530	0.184
UNCOOP	Top 1	Top 6	53.40 / 12.20	39	0.852	0.776	0.712	0.656	0.561	0.207
UNCOOP	Top 1	Threshold	52.49 / 13.81	37	0.862	0.789	0.723	0.662	0.565	0.203

The results shown in this paper also provide additional support for using automatically generated queries and "single collection" baseline to evaluate the search performance in peer-to-peer networks. The same conclusions on the relative effectiveness of various algorithms for federated search in peer-to-peer 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) gives us the opportunity to study in the future how the network can learn from past queries and evolve in order 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 each individual component as well as the interaction between different components. Peer-to-peer networks add further complexity to the problem due to factors such as dynamic topology, uncertainty in locating relevant information, and concern in efficiency. How the data is distributed over the networks and how different nodes interact and communicate with each other also affect the use of different algorithms because all algorithms are developed based on either explicit or implicit assumptions of the environments. Our next step is to further understand the unique characteristics of peer-to-peer networks and to develop practical algorithms that are more appropriate for search in dynamic and heterogeneous peer-to-peer networks.

ACKNOWLEDGMENTS

This material is based on work supported by NSF grant IIS-0118767 and 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] J. Callan. Distributed information retrieval. W. B. Croft, editor, *Advances in information retrieval*, chapter 5, pages 127-150. Kluwer Academic Publishers, 2000.

[2] A. Le Calv and J. Savoy. Database merging strategy based on logistic regression. *Information Processing and Management*, 36(3), pages 341-359.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] D. Hawking. Overview of the TREC-9 web track. In *Proc. of the 9th Text Retrieval Conference (TREC-9)*, 2000.

[9] KaZaA. http://www.kazaa.com.

[10] 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.

[11] Limewire. http://www.limewire.com.

[12] K. Liu, C. Yu, W. Meng, A. Santos and C. Zhang. Discovering the representative of a search engine. In *Proc. of the 10th International Conference on Information Knowledge Management (CIKM 2001)*, 2001.

[13] J. Lu and J. Callan. Content-cased retrieval in hybrid peer-to-peer networks. In *Proc. of the 12nd International Conference on Information Knowledge Management (CIKM 2003)*, 2003.

[14] J. Lu and J. Callan. Peer-to-peer testbed definitions: trecwt10g-2500-bysource-v1 and trecwt10g-query-bydoc-v1. http://www.cs.cmu.edu/callan/Data, 2003.

[15] J. Lu and J. Callan. Merging retrieval results in hierarchical peer-to-peer networks (poster description). In *Proc. of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2004.

[16] 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.

[17] 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.

[18] 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, November 2001.

[19] 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.

[20] 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.

[21] 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.

[22] 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*, August 2003.

[23] S. Waterhouse. JXTA Search: Distributed search for distributed networks. Technical report, Sun Microsystems Inc., 2001.

[24] 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.

[25] C. Zhai, P. Jansen, E. Stoica, N. Grot and D. Evans. Threshold Calibration in CLARIT adaptive filtering. In *Proc. of the 7th Text Retrieval Conference (TREC-7)*, 1998.

[26] C. Zhai, P. Jansen and D. Evans. Exploration of a heuristic approach to threshold learning in adaptive filtering. In *Proc. of 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.

[27] 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.

[28] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCS/CSD-01-1141, Computer Science Division, University of California, Berkeley. 2001.