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

JIE LU jielu@cs.cmu.edu
JAMIE CALLAN callan@cs.cmu.edu
Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Abstract. Peer-to-peer (P2P) networks integrate autonomous computing resources without requiring a central coordinating authority, which makes them a potentially robust and scalable model for providing federated search capability to large-scale networks of text-based digital libraries. However, peer-to-peer networks have so far provided very limited support for full-text federated search with relevance-based document ranking. This paper provides solutions to full-text federated search of text-based digital libraries in hierarchical peer-to-peer networks. Existing approaches to full-text search are adapted and new methods are developed for the problems of resource representation, resource selection, and result merging 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 of text-based digital libraries in peer-to-peer networks.

Keywords: full-text, federated search, text-based digital libraries, peer-to-peer networks

1 Introduction

A very large number of text-based digital libraries were developed during the last decade. Nearly all of them use some form of relevance-based 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 so that the contents provided by these digital libraries cannot be accessed by Web search engines such as Google and AltaVista that only conduct search on centralized repositories. How best to provide federated search across such independent digital libraries is an unsolved problem often referred to as the “Hidden Web” problem.

Many collections of text documents also reside in enterprise networks. Collecting and maintaining an internal centralized repository for search is not always practical for heterogeneous, multi-vendor, or lightly-managed enterprise networks. Federated search in these environments requires an effective, convenient and cost-efficient solution that is decentralized in nature.

Peer-to-peer (P2P) networks integrate autonomous computing resources without requiring a central authority, which makes them a good choice for providing federated search capability to a large number of digital libraries on the Internet and in enterprise networks. The decentralized nature of P2P networks also enables robustness and scalability, which are critical to federated search of large scales. To capitalize on the power and scaling properties of large distributed P2P systems, we were motivated to explore federated search of text-based digital libraries in P2P networks.

To date, P2P networks are primarily used for file-sharing of popular music, videos, and software, or for distributed storage of digital archives. These types of digital objects have relatively obvious or well-known naming conventions and descriptions, making it convenient to represent them with just a few words from a name, title, or manual annotation. Search in these systems is typically *known-item search*, in which the goal is to find a single instance of a known object (e.g., a particular song by a particular artist).[1] 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 digital objects with well-known naming conventions is a task for which simple solutions (e.g., Boolean keyword matching over document names and annotations) suffice. To use P2P networks as a federated search layer for text documents, more sophisticated solutions for search based on the full body of each document are required because i) text documents do not have well-known naming conventions and it is relatively difficult to represent the content of a text document by using just a few words over a small vocabulary, and ii) search is mostly no longer known-item search because the user usually only has a general description of an information need in his/her mind instead of the identity of any particular document. The principal goal of search becomes locating documents that contain relevant contents to satisfy the information need, not finding a copy of a specific document. We would argue that most of the recent research on P2P networks offers little useful guidance for providing full-text 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 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 in the form of relevance-based ranking ("*full-text ranked retrieval*"). The content information required for query routing ("*resource descriptions*") can be provided cooperatively by individual digital libraries upon request ("*cooperative*" environments), or created from the documents sampled from each digital library via the normal process of submitting queries and retrieving documents ("*uncooperative*" environments). 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 in both cooperative and uncooperative environments.

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 full-text federated search of text-based digital libraries in P2P networks. Sections 4 and 5 discuss our data resources and methodology for evaluation. Experimental settings and results are presented in Sections 6 and 7. Section 8 concludes.

[1] There has been some recent work on distributed collaborative filtering in P2P networks, in which the goal is to recommend other items to a user based on the items he/she previously downloaded (Wang et al. 2005). Although collaborative filtering is a different task from known-item search, it typically works with the same types of digital objects that are common for known-item search, which have well-known naming conventions and descriptions based on small or controlled vocabularies.

2 Overview

In this section we present an overview of the prior research on federated search of text-based digital libraries, P2P network architectures, and full-text search in P2P networks in order to set the stage for the descriptions of our approaches to full-text 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 digital library ("*resource description*");
- Resource selection: Deciding which digital libraries are most appropriate for an information need based on their resource descriptions; and
- Result merging: Merging ranked retrieval results from a set of selected digital libraries.

A single, centralized *directory service* is responsible for acquiring resource descriptions of the digital libraries it serves, selecting the appropriate digital libraries for a given query, and merging the retrieval results from selected digital libraries into a single, integrated ranked list. Solutions to all these three problems have been developed in distributed information retrieval. 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 (Gravano et al. 1997). Query-based sampling is an alternative approach to acquiring resource descriptions without requiring explicit cooperation from digital libraries (Callan 2000). The resource description of a digital library is constructed by sampling its documents via the normal process of submitting queries and retrieving documents.

Resource Selection. Resource selection aims to select a small set of resources that contain many documents relevant to the information request. Typically resource selection includes *resource ranking* which ranks resources by their likelihood of returning relevant documents, and *thresholding for resource selection* which selects the top-ranked resources to process the information request.

Resource selection algorithms such as CORI (Callan 2000), gGlOSS (Gravano and García-Molina 1995), and the Kullback-Leibler (K-L) divergence-based algorithm (Xu and Croft 1999) 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 (Si and Callan 2003b) and the decision-theoretic framework for resource selection (Nottelmann and Fuhr 2003) 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 for resource selection*") is usually simplified to use of a heuristic value (e.g., 5 or 10).

Result Merging. Result merging algorithms can be categorized into 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 algorithms 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 (Callan 2000). The Semi-Supervised Learning result merging algorithm uses the documents obtained by query-based sampling as training data to learn score normalizing functions (Si and Callan 2003a).

Usually in order to recalculate document scores, the directory service needs to download all of the documents in the retrieval results. Downloading documents is not necessary if the statistics required for score recalculation can be obtained in another way. Kirsch's algorithm (Kirsch 1997) requires each resource to provide summary statistics for each of the retrieved documents (e.g., document length and how often each query term matches). 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 (Craswell et al. 1999).

2.2 P2P Network Architectures

Each peer (an abstract notion of a participating entity) in P2P networks can participate as a client and/or a server. Clients are *information consumers* which issue queries to initiate search in P2P networks. Servers can be *information providers* and/or *directory services*. Information providers provide information contents and respond to queries with documents that are likely to satisfy the requests. Directory services route queries to other servers. The main activities involved for search in P2P networks thus include issuing requests ("*queries*"), routing requests ("*query routing*"), and responding to requests ("*retrieval*"). Compared with search in centralized environments, query routing is a problem that is unique to search in P2P networks. The goal of query routing is to locate resources with relevant contents ("*resource location*"). Various P2P network architectures can be distinguished by their different solutions to resource location.

Resource location in first generation P2P networks is characterized by the original Napster[2] and similar P2P file-sharing applications (Yaga) (MusicNet) (Intel 2003), which use a single logical directory service ("*brokered*" P2P architectures), and Gnutella 0.4, which uses undirected message flooding and a search horizon ("*completely decentralized*" P2P architecture). The former proves easy to attack, and the latter doesn't scale. They also explore very different solutions: The original Napster was centralized and re-

[2] The original Napster refers to the Napster peer-to-peer music-sharing service that was started in 1999 and shut down in 2001.

quired cooperation (sharing of accurate information); Gnutella 0.4 is decentralized and requires 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 architectures associate each data item with a key and distribute keys among directory services using a distributed hash table (DHT) so that each directory service is responsible for a certain range of keys in the multi-dimensional key space (Ratnasamy et al. 2001) (Rowstron and Druschel 2001) (Stoica et al. 2001) (IRIS) (Maymounkov and Mazières 2002) (eDonkey) (eMule) (RevConnect) (Tang et al. 2003) (Manku et al. 2003) (Zhao et al. 2004). The resource responsible for a queried key is located by distributed hash table lookup. Hierarchical P2P architectures typically use a two-level hierarchy of peers including an upper-level of directory services and a lower-level of digital libraries or clients (BearShare) (Edutella) (Gnucleus) (Gnutella2) (JXTA) (KaZaA) (Limewire) (Morpheus) (Shareaza) (Swapper.NET). Each directory service conducts resource location for a region of digital libraries and clients, and directory services work collectively to cover the whole network. The common characteristic of structured and hierarchical P2P architectures is the construction of an *overlay* to organize peers for efficient query routing. An important distinction is that a structured P2P architecture enforces a strict overlay structure of peer connections based on its distributed hash table architecture, whereas the connections between peers in a hierarchical P2P architecture are not determined by hash mapping. Structured P2P architectures require digital libraries to cooperatively share descriptions of data items in order to generate keys and construct distributed hash tables. In contrast, hierarchical P2P architectures 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. Another advantage of hierarchical P2P architectures over structured P2P architectures is that they more easily support sophisticated search techniques that are not constrained to controlled or small vocabularies, which is more appropriate for full-text search. However, hierarchical P2P architectures typically have higher communication costs and are more complex than structured P2P architectures.

2.3 Full-Text Search in P2P Networks

Most of the prior research on search in peer-to-peer networks only supports simple exact or Boolean keyword matching over document metadata. There has been some recent work on developing systems that adopt more sophisticated retrieval models to support full-text search in peer-to-peer networks. Examples are PlanetP using a completely decentralized P2P architecture (Cuenca-Acuna and Nguyen 2002), pSearch using a structured P2P architecture (Tang et al. 2003), and full-text search in hierarchical P2P networks (Lu and Callan 2003).

In PlanetP (Cuenca-Acuna and Nguyen 2002), a peer uses a TF.IDF algorithm to decide which peers to contact for information requests based on the compact summaries it collects about all other peers' inverted indexes. Because no special resources are dedicated to support directory services in completely decentralized P2P architectures, it is somewhat inefficient for each peer to collect and store information about the contents of all other peers, limiting its scalability.

pSearch (Tang et al. 2003) uses the semantic vector (generated by Latent Semantic Indexing) of each document as the key to distribute document indices in a structured P2P network. This is a way of mapping a large, uncontrolled vocabulary onto a small, controlled vocabulary. 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 peer in the network, which makes this approach difficult to extend to uncooperative and heterogeneous environments.

Full-text resource selection and document retrieval algorithms for a single directory service are extended to multiple directory services in (Lu and Callan 2003). Experimental results demonstrate that full-text 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 flooding or random selection with Boolean keyword matching over document names.

3 Full-Text Federated Search in Hierarchical P2P Networks

The research described in this paper adopts a hierarchical P2P architecture due to several reasons. First, a hierarchical P2P architecture uses multiple regional directory services to work collectively to cover the network without relying on a central authority. Therefore, it is more robust and scalable than a brokered P2P architecture with a single, centralized directory service. Second, because full-text search requires a full-text resource representation, which is more expensive in terms of storage and communication costs, peers need more processing power and connection bandwidth to perform the duties of directory services. Hierarchical P2P architectures relieve peers with limited computing and network resources of the burden of conducting directory services, which makes them a better choice than completely decentralized P2P architectures. Third, it is difficult to use full-text resource representations in structured P2P architectures due to their complete reliance on distributed hash tables with limited-dimensional key spaces for content placement and resource location. In contrast, existing techniques developed for full-text search can be adapted to hierarchical P2P architectures in a straightforward manner.

As described in Section 2.2, a hierarchical P2P architecture consists of a two-level hierarchy of two types of peers: a lower level of leaf peers and an upper level of hub peers. A peer located at the leaf level can be an information provider (digital library), or an information consumer (user, client). A peer located at the hub level is a directory service. For full-text federated search of text-based digital libraries, each provider peer is a text database that processes full-text queries by running a document retrieval algorithm over a local document collection. Each consumer peer represents a user with information requests. Each hub acquires and maintains necessary information about its neighboring hubs and providers, which it uses to provide resource selection and result merging services to the network. Leaf peers only connect to hubs. Hubs connect with leaves and other hubs. Figure 1 illustrates a hierarchical P2P network.

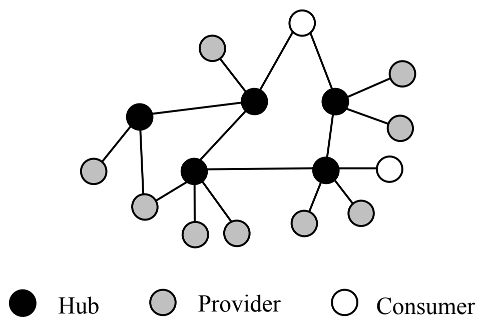

Figure 1. Illustration of a hierarchical P2P network.

For full-text federated search, when a consumer peer has an information request, it sends a query message to its neighboring hubs. A hub that receives the query message uses its resource selection algorithm to rank and select one or more neighboring providers as well as hubs for query routing. A provider that receives the query message uses its document retrieval algorithm to generate a relevance-based ranking of its documents and responds with a queryhit message that contains a list of the top-ranked documents. A hub is also responsible for collecting the queryhit messages generated by multiple neighboring providers, using its result merging algorithm to merge multiple ranked lists of documents into a single, integrated ranked list, and returning it to the consumer. Finally, a consumer may need to merge results returned by multiple hubs. 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 peer. When the TTL reaches 0, the message is no longer routed. Each peer discards duplicate messages it receives.

In this paper we assume a "static" network setting (i.e., fixed topology without peer arrivals, departures or failures) so that we can focus on the solutions to resource representation, resource selection, and result merging for higher efficiency and accuracy in federated search of hierarchical P2P networks. The problems of dynamic topology evolution due to peer arrivals and departures, fault tolerance and load balancing, although important, are outside the scope of this paper.

3.1 Resource Representation

A resource description is a very compact summary of its contents. Compared with the complete index of a collection of documents, a resource description requires much lower communication and storage costs, but still provides enough information for resource selection algorithms to determine which resources are more likely to contain documents relevant to queries. We adopt the format of a resource description used by previous resource selection algorithms (Callan 2000) (Gravano et al. 1994) (Gravano and García-Molina 1995) (Xu and Croft 1999) in distributed information retrieval, which includes 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 could be a single provider (digital library), a hub that covers multiple neighboring providers, or a "neighborhood" that includes all the peers reachable

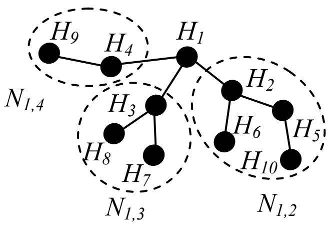

Figure 2. 3 neighborhoods that can be reached from H_1.

from a hub. Although resource descriptions for different types of resources have the same format, different methods are required to acquire them.

Resource Descriptions of Providers. Resource descriptions of providers are used by hubs for query routing ("*resource selection*") among adjacent providers. In cooperative environments, each provider provides an accurate resource description to its neighboring hubs upon request (i.e., using STARTS). In uncooperative environments, each hub conducts query-based sampling independently to obtain documents that can be used to generate resource descriptions for its neighbors (Callan 2000). Sample documents can also be used by the Sample-Resample algorithm to estimate the total number of documents in a provider's collection (Si and Callan 2003b).

Resource Descriptions of Hubs. A hub's resource description is the aggregation of the resource descriptions of its neighboring providers. It describes the content area covered at the hub. Since hubs work collaboratively in hierarchical P2P networks, neighboring hubs can exchange with each other their aggregate resource descriptions. However, because hubs' resource descriptions only have information for peers within one hop (providers directly connecting to them), if they are used by a hub to decide how to route query messages, the routing would not be effective when peers 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 travels several hops beyond each neighbor. This kind of information is referred to as the *neighborhood*'s resource description, 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 the set of hubs that a query message can reach by following the path from H_i to H_j and further traveling a number of hops. Each hub has its own view of neighborhoods near it, and each of its neighborhoods corresponds to one of its hub neighbors. Figure 2 illustrates the concept of neighborhood. Hub H_1 has three neighboring hubs H_2, H_3 and H_4. Thus it has three adjacent neighborhoods, labeled $N_{1,2}$, $N_{1,3}$ and $N_{1,4}$. A neighborhood's resource description 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 (Crespo and García-Molina 2002b). 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 neighborhoods' resource descriptions and routing indices is that neighborhoods' resource descriptions represent contents with unigram language models (terms with their frequencies). Thus by using resource descriptions of neighborhoods, there is no need for hubs and providers 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 (Crespo and García-Molina 2002b), 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:

$$\sum_{H_k \in N_{i,j}} \frac{tf(t, H_k)}{F^{numhops(H_i, H_k)-1}} \qquad \textbf{(1)}$$

where $tf(t, H_k)$ is t's term frequency in the resource description of hub H_k, and F is a factor for exponential decay, which can be the number of hub neighbors each hub has in the network.

The exponentially aggregated total number of documents in a neighborhood is calculated as below.

$$\sum_{H_k \in N_{i,j}} \frac{numdocs(H_k)}{F^{numhops(H_i, H_k)-1}} \qquad \textbf{(2)}$$

The creation of resource descriptions of neighborhoods requires several iterations at each hub. 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 previously received from all of its neighboring hubs excluding H_j. The calculation of $ND_{j,i}$ is provided by Equation 3.

$$ND_{j,i} = HD_i + \sum\nolimits_{H_k \in directneighbors(H_i) \setminus H_j} \frac{ND_{i,k}}{F} \qquad \textbf{(3)}$$

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. Each hub can run the creation process asynchronously.

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, the frequencies of some terms and the number of documents may be overcounted, which will affect the accuracy of resource descriptions. How to deal with cycles in peer-to-peer networks using routing indices is discussed in detail in (Crespo and García-Molina 2002b). We could use the same solutions described in (Crespo and García-Molina 2002b) 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 neighborhoods' resource descriptions in networks with cycles is still quite efficient and accurate.

3.2 Resource Selection

Query routing aims at an optimal cost-effective solution to directing queries to those peers 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 query. The flooding technique guarantees to reach peers with relevant information 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 peers may not be relevant. To achieve both efficiency and accuracy, each hub needs to rank its neighboring providers by their likelihood of satisfying the information request, and neighboring hubs by their likelihood of providing a path to peers with relevant information ("*resource ranking*"), and only forward the request to its top-ranked neighbors. The information each hub utilizes for resource selection is the content information about its neighboring providers as well as neighborhoods represented by resource descriptions. Because resource descriptions of providers and those of neighborhoods are not of the same magnitude in vocabulary size and term frequency, direct comparison between providers and neighborhoods is difficult since it requires better size normalization. For this reason, a hub handles separately the selection of its neighboring provider peers and hubs.

In this paper all of the hubs are required to use the same resource selection method, as is common in most operational and research P2P systems.[3] Because the Kullback-Leibler (K-L) divergence-based method has been shown to be one of the most effective resource ranking algorithms tested on various testbeds in distributed information retrieval (Si and Callan, 2004), we use it for ranking of both neighboring providers and neighboring hubs at each hub.

Resource Ranking of Providers. Each hub uses the K-L divergence resource ranking algorithm to calculate P(P_i | Q), the conditional probability of predicting the collection of provider P_i given the query Q and uses it to rank different providers. P(P_i | Q) is calculated as:

$$\mathrm{P}(P_i \mid Q) = \frac{\mathrm{P}(Q \mid P_i) \times \mathrm{P}(P_i)}{\mathrm{P}(Q)} \propto \mathrm{P}(Q \mid P_i) \qquad \textbf{(4)}$$

$$\mathrm{P}(Q \mid P_i) = \prod_{q \in Q} \frac{tf(q, P_i) + \mu \times \mathrm{P}(q \mid G)}{numterms(P_i) + \mu} \qquad \textbf{(5)}$$

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

Resource Ranking of Hubs. For ranking of hubs, the resource descriptions of neighborhoods are used to calculate the collection language models needed by the K-L divergence resource ranking algorithm. Because selecting a neighboring hub is essentially selecting a neighborhood, using a prior distribution that favors larger neighborhoods could lead to better search performance, which was indeed the case in our ex-

[3] Note that we do not assume that all providers (digital libraries) use the same document retrieval algorithm.

periments. 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:

$$\mathrm{P}(N_i \mid Q) = \frac{\mathrm{P}(Q \mid N_i) \times \mathrm{P}(N_i)}{\mathrm{P}(Q)} \propto \mathrm{P}(Q \mid N_i) \times numdocs(N_i) \qquad \textbf{(6)}$$

$$\mathrm{P}(Q \mid N_i) = \prod_{q \in Q} \frac{tf(q, N_i) + \mu \times \mathrm{P}(q \mid G)}{numterms(N_i) + \mu} \qquad \textbf{(7)}$$

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

Resource Selection of Providers with Unsupervised Threshold Learning. After providers are ranked based on their $\mathrm{P}(P_i \mid Q)$ values, the usual approach is to select the top-ranked providers up to a predetermined number, which is tuned empirically to optimize system performance. In a hierarchical P2P network with multiple hubs, different hubs may have different "optimal" threshold values. In addition, due to the dynamic nature of a P2P network, the "optimal" threshold value of each individual hub may change over time. Therefore, it is not appropriate to use a static, predetermined threshold for each hub to decide how many providers to select for a given query. It is desirable that hubs have the ability to learn their own selection thresholds automatically and autonomously.

The problem of learning the threshold to convert relevance-based ranking scores into a binary decision has mostly been studied in information filtering and text categorization (Arampatzis et al. 2001) (Zhang and Callan 2001). One approach to learning 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, the user relevance feedback required as training data may not be easily available for federated search in peer-to-peer networks. Therefore, it is preferable that threshold learning in P2P networks be conducted in an unsupervised manner. Our goal is to develop a technique for each hub to learn resource selection thresholds without supervision based on the information and functionality it already has. Because each hub has the ability to merge the retrieval results from multiple providers 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". A provider with at least n highly ranked documents in the merged result for a query is considered to be "relevant" with respect to the query and "non-relevant" otherwise. If we further assume that a hub is permitted to flood its neighbors with a small number of queries, it can use the results of these queries as training data to learn the distributions of the normalized ranking scores of relevant and non-relevant neighboring providers. Thus we can define a utility function based on these distributions, and the selection threshold is the 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 selection of providers:

$$U(\theta) = N_{rel}(\theta) - N_{nonrel}(\theta) - w \times N(\theta) \ , \ \theta^* = \arg\max_{\theta} U(\theta) \qquad \textbf{(8)}$$

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

Larger $N(\theta)$ leads to more selected providers and thus low efficiency. Since efficiency is also important for federated search in peer-to-peer environments, the term $w \times N(\theta)$ is included in $U(\theta)$ to penalize low efficiency.

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

$$N_{rel}(\theta) = \alpha \times \int_{\theta}^{1} \mathrm{P}(s \wedge rel)ds = \alpha \times \int_{\theta}^{1} \mathrm{P}(s \mid rel) \times \mathrm{P}(rel)ds \qquad \textbf{(9)}$$

$$N_{nonrel}(\theta) = \alpha \times \int_{\theta}^{1} \mathrm{P}(s \wedge nonrel)ds = \alpha \times \int_{\theta}^{1} \mathrm{P}(s \mid nonrel) \times (1 - \mathrm{P}(rel))ds \qquad \textbf{(10)}$$

where $\mathrm{P}(s \wedge rel)$ is the probability of a provider having score s and being relevant, $\mathrm{P}(s \wedge nonrel)$ is the probability of a provider having score s and being non-relevant, $\mathrm{P}(s \mid rel)$ is the probability of a relevant provider having score s, $\mathrm{P}(s \mid nonrel)$ is the probability of a non-relevant provider having score s, $\mathrm{P}(rel)$ is the probability of a provider being relevant, and α is the total number of providers in the training data.

The relevancy of a provider for a query 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. $\mathrm{P}(s \mid rel)$ and $\mathrm{P}(s \mid nonrel)$ at a hub can be estimated from the scores of relevant and non-relevant providers 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 (Arampatzis et al. 2001). However, in our experience the score distributions of relevant and non-relevant providers at a hub $\mathrm{P}(s \mid rel)$ and $\mathrm{P}(s \mid nonrel)$ cannot be fitted very well by the shapes of exponential or Gaussian distributions. 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. 10 equal sized bins are used for normalized scores ranging from 0 to 1. Experimental results not shown in this paper verified that using the discrete score distributions had better performance (and was simpler) than using the fitted continuous score distributions.

Usually $\mathrm{P}(rel)$ is estimated by maximum likelihood estimation using training data. However, because using the top-ranked merged documents as the set of "relevant" documents for each query yields very unbalanced amounts of training data for relevant and non-relevant neighbors at the hub (very few relevant neighbors but a lot of non-relevant neighbors), maximum likelihood estimated $\mathrm{P}(rel)$ using training data is not likely to be a good estimation of $\mathrm{P}(rel)$ for future queries. Therefore, we assume that all the providers connecting to a hub have equal probability of being relevant and non-relevant, i.e., $\mathrm{P}(rel)$ has a uniform distribution. This is a reasonable assumption when each hub covers a specific content area so that all of its connecting providers 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 ranking results of providers using a set of training queries.

Resource Selection of Hubs with Unsupervised Threshold Learning. The unsupervised threshold learning method described above requires each hub to estimate the distributions of the ranking scores of relevant and non-relevant neighbors using the merged retrieval results of a set of training queries. In a hierarchical P2P network, the number of hubs is usually much smaller than the number of providers. In this case, the number of hub neighbors each hub has may be too small for it to make a reliable estimate of these distributions. Therefore, we introduce another method for a hub to learn the selection threshold for its hub neighbors.

In contrast to learning a single threshold from the retrieval results of a set of training queries as a whole ("*set-based threshold learning*"), the threshold learning method for resource selection of hubs computes a threshold based on the separate thresholds learned for individual training queries ("*individual-based threshold learning*"). For each training query, after a hub merges the documents returned by its hub neighbors (including the retrieval results of all downstream peers), it treats the top-ranked merged documents as "relevant" documents and calculates how many "relevant" documents are returned by each hub neighbor. Given this information, the hub goes down the list of hub neighbors sorted by their normalized ranking scores until a sufficiently large percentage of "relevant" documents have been returned, i.e., a sufficiently high value of recall is obtained, and uses the last ranking score before stopping as the threshold for this training query. The individually learned thresholds for a set of training queries are averaged to get a single threshold at the hub.

3.3 Result Merging

In cooperative environments, result merging at each hub uses an extended version of Kirsch's algorithm (Kirsch 1997). Kirsch's algorithm requires each provider to provide summary statistics (e.g., document length and how often each query term matches) for each of the retrieved documents for a hub to recalculate document scores. It also requires global corpus statistics in score recalculation. To avoid the cost of acquiring and maintaining global corpus statistics at each hub, we extend the original Kirsch's algorithm to use the aggregation of the hub's resource description and the neighborhoods' resource descriptions for its neighboring hubs to substitute for the corpus statistics. Documents are merged according to the document scores recalculated by a K-L divergence retrieval algorithm (Ogilvie and Callan 2001) using the above document and corpus statistics.

In uncooperative environments where no summary statistics for the retrieved documents are available, the most effective approach to result merging at a hub is to adapt the Semi-Supervised Learning (SSL) result merging algorithm since SSL has been shown to be the most effective result merging algorithm in distributed information retrieval of uncooperative digital libraries using a single directory service (Si and Callan 2003a). For each query, SSL uses the documents sampled from a provider to generate unsupervised training data to learn a score normalizing function for that provider. To apply SSL in hierarchical P2P networks, given a query, documents that appear both in the retrieval result from the database of sample documents ("*centralized sample database*") at a hub and in the retrieval result from one of this hub's neighbors are identified ("*overlapping documents*"). Each overlapping document has a pair of scores — the score calculated at the hub and the score returned by the neighbor, which defines a point in two-dimensional space. The set of score pairs for overlapping documents can serve as training points to learn the score normalizing function for this neighbor with respect to the query. The normalizing function learned on a query-by-query basis transforms scores specific to the neighbor to scores independent to the neighbor but specific to the hub.

Typically, SSL uses linear regression to learn score normalizing functions, which is a choice that has not been explicitly justified. In addition, no evidence has been shown that using linear regression is more effective than nonlinear regression models. To make SSL work effectively in P2P networks where training data are often limited and biased, we modify SSL by combining linear regression with the more general locally-weighted linear regression and making other corresponding changes as follows (Lu and Callan 2004):

1. Locally-weighted linear regression (Atkeson et al. 1997) is applied to learn the score transformation when the number of training points in the neighborhood of data point x is at least N and the operation is interpolation instead of extrapolation; otherwise, globally non-weighted linear regression is applied. The weight of the i^{th} training point x_i with regard to x (resource-specific document score) is computed using Equation 11 and the parameters $a(x)$ and $b(x)$ of the normalizing function with respect to x are estimated from Equation 12 and used in Equation 13 to calculate the projected value y (normalized, resource-independent document score).

$$w_i(x) = \exp(-\frac{(x - x_i)^2}{k^2}) \qquad \textbf{(11)}$$

$$y_i = a(x) + b(x) w_i(x) x_i \qquad \textbf{(12)}$$

$$y = a(x) + b(x) x \qquad \textbf{(13)}$$

2. As is described in (Si and Callan 2003a), downloading more documents to serve as additional training data whenever needed could improve the performance of SSL. Thus when the number of overlapping documents is less than T for a neighboring peer, up to D documents in the retrieval result are downloaded from this peer and their original scores together with the scores recalculated at the hub are added to the training data to learn the score normalizing function.
3. When the score normalizing function learned using globally non-weighted linear regression has negative slope (which is counterintuitive and thus indicates insufficient training data), the documents in the retrieval result are downloaded one at a time and their original scores together with the scores recalculated at the hub are used as additional training data for linear regression until the slope is positive.

Parameters of the modified SSL algorithm N, T, and D are usually set with small values. The kernel width parameter k is set to include enough training points (depending on the values of N and T) in the neighborhood.

A consumer needs to merge the results returned by multiple hubs. Because consumers don't maintain information about the contents of other peers and corpus statistics, they can only use simple, but probably less effective, merging methods. In this paper, consumers directly use document scores returned by hubs to merge results.

4 Test Data

We used a P2P testbed based on the TREC WT10g web test collection (Lu and Callan 2003) to evaluate the performance of federated search in hierarchical P2P networks of text-based digital libraries. The WT10g dataset contains documents that the Internet Archive crawled from 11,485 Web sites. The set of documents crawled from one Web site defines a collection. We randomly selected (with a bias towards larger size) 2,500 collections to define 2,500 providers in a hierarchical P2P network. Collectively the 2,500 providers contain a total of 1,421,088 documents (an average of about 568 documents per provider).

There are 25 hubs in the P2P testbed. Because previous research has shown that clustering peers into content-based clusters and establishing connections based on cluster membership enables higher search accuracy and efficiency in P2P networks (Crespo and García-Molina 2002a), we required all of the providers that connect to a hub to form a content-based cluster so that each hub covers a specific type of content. The connections between providers and hubs were determined by clustering providers into 25 clusters using a similarity-based soft clustering algorithm, and connecting all of the providers 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 according to the query length distribution described in (Jansen et al. 2000). The P2P testbed queries ("*WT10g queries*") were automatically generated by extracting key terms from the documents in WT10g (Lu and Callan 2003). Because it is expensive to obtain relevance judgments for these automatically-generated queries, we treated the 50 top-ranked documents retrieved for each query from a single large collection as the "relevant" documents ("*single collection*" *baseline*), and measured how well federated search in the hierarchical P2P network could locate and rank these documents.[4] The single large collection was the subset of the WT10g used to define the contents of the 2,500 providers ("*WT10g-subset*").

For each query, a provider was randomly chosen to temporarily act as a consumer peer to issue the query and collect the merged retrieval results for evaluation.

5 Evaluation Methodology

Both search accuracy and query routing efficiency were used as performance measures. For the 100 TREC queries, precisions at document ranks 5, 10, 15, 20, and 30 were used to measure search accuracy. For the 1,000 automatically-generated WT10g queries, the percentages of overlap between the documents returned by search in the P2P network and the 50 top-ranked documents returned by centralized search ("*single collection*" *baseline*) were calculated at document ranks 5, 10, 15, 20, and 30 of federated search results and used to measure search accuracy. We refer to these percentage values as "*overlap precisions*" to distinguish them from precisions based on human relevance judgments.

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 Experimental Settings

A series of experiments was conducted to study full-text federated search in cooperative and uncooperative P2P environments. Four methods for resource selection of hubs were compared: the flooding method (a hub broadcasting query messages to all of its hub neighbors), random selection (a hub randomly selecting one of its hub neighbors for query routing), full-text selection using a fixed threshold (one top-ranked neighboring hub), and full-text selection using the learned thresholds. Two methods for resource selection of providers were compared: full-text selection of a fixed percentage (1%) of the top-ranked providers and full-text selection using the learned thresholds. Resource selection using the chosen fixed threshold values yielded comparable overall query routing efficiency to resource selection using the learned thresholds.

The initial TTL (time-to-live) value for each query message was 6, which was not very large for full-text resource selection considering that on average each hub only forwarded the message to one hub neighbor.

For resource representation, in the cooperative environment, each provider provided an accurate resource description to each of its neighboring hubs. In the uncooperative environment, each hub conducted query-based sampling independently to obtain sample documents from its neighboring providers in order to create resource descriptions. Sample documents from a provider were also used by the Sample-Resample method (Si and Callan 2003b) to estimate the total number of documents in this provider's collection. A set of 500 two-term queries were randomly chosen from the queries in the P2P testbed and each hub sub-

[4] Although this use of a single collection baseline is not ideal, we believe that it is useful when relevance judgments are not available. We are not aware of any published research showing a federated search solution that consistently beats a single collection baseline; consistently matching a single collection baseline is considered success by most federated search researchers.

mitted a query randomly selected from this set to all of its neighboring providers, examined the 4 top-ranked documents from the results returned by each provider, and repeated the process until 300 unique documents were sampled from each provider or all the queries in the predefined set had been used. The parameter values of 4 and 300 were chosen to be consistent with prior research that used query-based sampling for federated search of uncooperative digital libraries (Callan 2000) (Si and Callan 2003a). Examining fewer top-ranked documents per query enabled a slightly less biased collection language model but required more queries and thus higher communication costs. Obtaining more unique sample documents did not dramatically improve the accuracy of the estimated collection language model.

For resource ranking, the smoothing parameter μ in Dirichlet smoothing was set to be 1000, a value which has been shown to work well for ad-hoc retrieval over various TREC test collections (Zhai and Lafferty 2001). It is also shown in (Zhai and Lafferty 2001) that retrieval using Dirichlet smoothing is quite robust when the value of μ is chosen from a wide range (500-10000).

Unsupervised threshold learning required a set of training queries. For each experiment that used learned thresholds for resource 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 learned thresholds 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 of unsupervised threshold learning in Equation 8 was adjusted so as to yield similar overall query routing efficiency as selecting a fixed percentage (1%) of the top-ranked providers. The number of the top-ranked merged documents used for unsupervised threshold learning was 50. The parameter n which determined the criterion for a provider to be considered relevant with respect to a query was set to 5. The sufficient recall level to decide the threshold for resource selection of hubs was 50%. These parameter values worked effectively for hubs that had different numbers of neighbors and covered different content areas.

For the modified SSL algorithm, the minimum number of training points N in the neighborhood for locally-weighted linear regression to apply was 3, the minimum number of overlapping documents T for globally non-weighted linear regression to apply was 2, and up to $D = 3$ documents in the retrieval result could be downloaded from a peer to serve as additional training data when there were not enough overlapping documents. Some experimental results not shown in this paper indicate that modified SSL gave very similar performance when N and T were chosen among the values of 2 to 4, and D was chosen among the values of 3 to 10. The kernel width parameter k was initially set to be one eighth of the difference between the maximum and minimum document scores that were used as training data for learning a score normalizing function. When there were not enough training points in the neighborhood, the kernel width was doubled until enough training points were included in the neighborhood.

Table 1. Search performance evaluated on the 100 TREC queries in the hierarchical P2P network of cooperative digital libraries.

Hub Selection	Provider Selection	Msgs	P@5	P@10	P@15	P@20	P@30
A centralized collection	N/A	N/A	0.324	0.287	0.255	0.241	0.208
A single, centralized hub	Top 1%	N/A	0.268	0.208	0.178	0.163	0.144
Flooding	Top 1%	177	0.263	0.205	0.179	0.168	0.147
Flooding	Threshold	180	0.274	0.223	0.196	0.179	0.159
Random 1	Top 1%	63	0.240	0.191	0.170	0.154	0.130
Random 1	Threshold	65	0.252	0.205	0.177	0.159	0.137
Full-text+Top 1	Top 1%	59	0.259	0.196	0.176	0.163	0.139
Full-text+Top 1	Threshold	55	0.280	0.218	0.192	0.179	0.153
Full-text+Threshold	Top 1%	65	0.261	0.200	0.177	0.165	0.142
Full-text+Threshold	Threshold	63	0.281	0.220	0.194	0.179	0.156

Table 2. Search performance evaluated on the 100 TREC queries in the hierarchical P2P network of uncooperative digital libraries.

Hub Selection	Provider Selection	Msgs	P@5	P@10	P@15	P@20	P@30
A single, centralized hub	Top 1%	N/A	0.247	0.196	0.172	0.158	0.136
Flooding	Top 1%	178	0.257	0.209	0.182	0.172	0.148
Flooding	Threshold	178	0.272	0.220	0.195	0.180	0.158
Random 1	Top 1%	65	0.223	0.174	0.159	0.140	0.120
Random 1	Threshold	66	0.234	0.186	0.167	0.146	0.128
Full-text +Top 1	Top 1%	59	0.246	0.197	0.169	0.157	0.132
Full-text+Top 1	Threshold	58	0.263	0.216	0.187	0.168	0.147
Full-text+Threshold	Top 1%	63	0.252	0.200	0.174	0.160	0.139
Full-text+Threshold	Threshold	64	0.269	0.218	0.191	0.172	0.153

7 Experimental Results

Tables 1 and 2 show respectively the results of running the 100 TREC queries for federated search using different methods in the hierarchical P2P network of cooperative digital libraries and the hierarchical P2P network of uncooperative digital libraries. 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 4-8. The results of federated search in a traditional distributed information retrieval setup (i.e., a single, centralized directory service for resource selection and result merging) are included in each table as a baseline. For each query, the single, centralized directory service selected the top 25 digital libraries (1% of the

Table 3. Search performance evaluated on the 1,000 WT10g queries in the hierarchical P2P network of cooperative digital libraries.

Hub Selection	Provider Selection	Msgs	OP@5	OP@10	OP@15	OP@20	OP@30
Flooding	Top 1%	174	0.970	0.942	0.915	0.875	0.792
Flooding	Threshold	171	0.990	0.967	0.942	0.913	0.835
Random 1	Top 1%	60	0.874	0.809	0.753	0.698	0.595
Random 1	Threshold	55	0.891	0.831	0.773	0.718	0.616
Full-text+Top 1	Top 1%	54	0.949	0.904	0.857	0.804	0.701
Full-text+Top 1	Threshold	43	0.964	0.912	0.863	0.810	0.707
Full-text+Threshold	Top 1%	59	0.955	0.920	0.871	0.819	0.715
Full-text+Threshold	Threshold	48	0.972	0.933	0.880	0.827	0.722

Table 4. Search performance evaluated on the 1,000 WT10g queries in the hierarchical P2P network of uncooperative digital libraries.

Hub Selection	Provider Selection	Msgs	OP@5	OP@10	OP@15	OP@20	OP@30
Flooding	Top 1%	173	0.924	0.877	0.835	0.786	0.694
Flooding	Threshold	170	0.942	0.900	0.857	0.811	0.724
Random 1	Top 1%	59	0.812	0.738	0.671	0.612	0.516
Random 1	Threshold	60	0.834	0.758	0.694	0.632	0.530
Full-text+Top 1	Top 1%	50	0.850	0.775	0.711	0.654	0.556
Full-text+Top 1	Threshold	47	0.869	0.794	0.727	0.666	0.566
Full-text+Threshold	Top 1%	55	0.871	0.787	0.725	0.670	0.564
Full-text+Threshold	Threshold	54	0.895	0.804	0.738	0.683	0.580

2,500 digital libraries) based on the resource descriptions acquired using the STARTS protocol (cooperative digital libraries) or query-based sampling (uncooperative digital libraries), and merged results using the extended Kirsch's algorithm (cooperative digital libraries) or the modified Semi-Supervised Learning algorithm (uncooperative digital libraries). In addition, the results of search using a single large collection (centralized search) are also shown in the first row of Table 1. Tables 3 and 4 show the results of running the 1,000 WT10g queries. "Overlap precisions" (Section 5) at different document ranks are shown in the columns marked by "OP@*n*" where *n* varies among 5, 10, 15, 20, and 30.

We focus first on the results in Tables 1 and 2 for the 100 TREC queries with human relevance judgments. Although federated search in the hierarchical P2P network had 20%-30% relative degradations in search accuracy compared with centralized search, it was able to locate and rank highly 60%-80% of the relevant documents that were ranked highly by centralized search. Considering that federated search only selected about 1% of the 2,500 digital libraries and each selected digital library only returned up to 50 documents, the results are an encouraging sign of the effectiveness of federated search in P2P networks. The performance of federated search in the hierarchical P2P network of multiple directory services was comparable to that of distributed information retrieval with a single, centralized directory service when a similar number of digital libraries were selected. This indicates that using multiple, regional directory services collaboratively could increase the robustness and scalability of federated search without any adverse effect on search accuracy.

The results in Tables 1 and 2 also demonstrate that compared with using the flooding technique for resource selection of hubs, full-text selection based on resource descriptions of neighborhoods required around one third of the number of query messages without a significant drop in search accuracy. The average relative degradations were less than 3% in the cooperative environment and 7% in the uncooperative environment. Full-text selection and random selection gave similar query routing efficiency but the search accuracy of the former was consistently higher than the latter. Using learned thresholds for resource selection of hubs yielded a few more query messages than using a fixed threshold but had better search accuracy, indicating the ability of unsupervised threshold learning to automatically determine thresholds for better performance. Resource selection of providers with the learned thresholds in general gave better performance than selection based on a fixed percentage of the top-ranked providers, again verifying the effectiveness of unsupervised threshold learning.

For the results in Tables 3 and 4, because the "single collection" baseline was used as relevance judgments for WT10g queries, they directly measured the ability of federated search in the hierarchical P2P network to match the results from search in a centralized environment. The high "overlap precisions" at top document ranks in Tables 3 and 4 demonstrate that federated search in the hierarchical P2P network was able to locate and rank highly most of the documents that were considered relevant by the centralized approach, which illustrates the effectiveness of resource selection and result merging algorithms in the hierarchical P2P network.

The effectiveness of full-text federated search in P2P networks can be further manifested by comparing the results evaluated using human relevance judgments (Tables 1 and 2) and those using the "single collection" baseline (Tables 3 and 4). The performance difference between various algorithms for the 100 TREC queries was due to differences in the number of relevant documents among the top-ranked documents. In contrast, with the "single collection" baseline, the performance difference between various algorithms for the 1,000 WT10g queries was due to the difference in the degree of overlap between the results of federated search and those of centralized search, which might include both relevant and non-relevant documents. The values of relative performance difference between flooding and full-text hub selection for the 100 TREC queries were 3% and 7% in the cooperative and uncooperative environments respectively, which were less than half of the 9% and 15% relative degradations for the 1,000 WT10g queries. This indicates that among the top-ranked documents returned by centralized search, full-text federated search missed more non-relevant documents than relevant ones by selecting only a subset of hubs, which can be explained by both the use of content-based clusters in constructing the network topology (so that relevant documents are likely to be concentrated at a few hubs), and the ability of full-text resource selection to take advantage of this content-based locality for effective hub selection. Despite the difference in the evaluation results of the 100 TREC queries and the 1,000 WT10g queries, the same conclusions drawn from the results of the 100 TREC queries could also 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 also show that the search performance in the uncooperative environment had on average less than 8% degradation for TREC queries and less than 16% degradation for WT10g queries compared with the search performance in the cooperative environment. Considering that in the uncooperative environment hubs only obtained partial information about the content of each resource, the search performance in the uncooperative environment was satisfactory. This indicates that query-based sampling, the Sample-Resample method, and the modified Semi-Supervised Learning result merging algorithm are effective techniques for full-text federated search of text-based digital libraries in P2P networks of uncooperative digital libraries.

8 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 selection, and result merging for full-text federated search can be adapted to P2P environments in a straightforward manner, new development is still required to fit the solutions to the unique characteristics of hierarchical P2P networks. For example, in hierarchical P2P networks, query routing among hubs should be based on not only the hub's likelihood of providing relevant documents with its own providers, but also its potential to provide a path to other peers 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 methods to create and use resource descriptions of neighborhoods for hub-hub query routing with full-text resource selection of hubs. Experimental results demonstrate that full-text selection offers a better combination of accuracy and efficiency than 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 neighboring peers for query routing. Because hubs are different in the neighbors they connect to, which could also change dynamically as peers come and leave or change connections, the ability of hubs to learn automatically hub-specific resource selection thresholds in the network 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 unsupervised training queries. In our experiments the proposed approach was consistently more accurate than the typical method of selecting a fixed number or percentage of the top-ranked neighboring peers with similar efficiency.

In addition to new developments in resource representation and resource selection, the current state-of-the-art Semi-Supervised Learning result merging algorithm is modified to combine linear regression with the more general locally-weighted linear regression and corresponding changes are made to make the algorithm work more effectively with the limited and biased training data available in hierarchical P2P networks.

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.

The evaluation using a P2P testbed of 2,500 digital libraries and 25 directory services (hubs) shows that our approaches to full-text federated search enable comparable performance to that of traditional distributed information retrieval using a single, centralized directory service. However, one might argue that the conclusion only holds for P2P networks of small to medium sizes with regulated network structures and organized content distributions. Although we believe that our approaches to full-text federated search can work effectively and efficiently in larger P2P networks, the size of the P2P testbed is not large enough to enable solid analysis on the network aspect of federated search, particularly query routing at the hub level. To really see the problems that P2P networks bring to federated search and how they affect search performance, we need to study and evaluate federated search in networks of larger scales, which remains a challenge for the entire research community. As to the dependency of federated search performance on the appropriate network structure and content distribution, we view it as the starting point of our future exploration rather than a limitation of our approaches. Much of our future effort will be shifted from further refinement of particular search algorithms to developing distributed algorithms for dynamic evolution of P2P network topologies so that desired content distribution and network navigability can be obtained automatically and autonomously to support high performance full-text federated search in P2P networks.

Acknowledgements

We thank the anonymous reviewers for their helpful comments and suggestions. This material is based on work supported by NSF grants 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

Arampatzis A, Beney J, Koster C and van der Weide T (2000) KUN on the TREC-9 Filtering Track: Incrementality, decay, and threshold optimization for adaptive filtering systems. In: Proceedings of the 9th Text Retrieval Conference, pp. 87–109.

Atkeson C, Moore A and Schaal S (1997) Locally weighted learning. Artificial Intelligence Review, 11(1-5): 11–73.

BearShare, http://www.bearshare.com (visited June 24th, 2005).

Callan J (2000) Distributed information retrieval. In: Croft W B, ed. Advances in Information Retrieval. Kluwer Academic Publishers, 2000, pp. 127–150.

Craswell N, Hawking D and Thistlewaite P (1999) Merging results from isolated search engines. In: Proceedings the 10th Australasian Database Conference, pp. 189–200.

Crespo A and García-Molina H (2002a) Semantic overlay networks for P2P systems. In: Technical report, Computer Science Department, Stanford University.

Crespo A and García-Molina H (2002b) Routing indices for peer-to-peer systems. In: Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS), pp. 23–32.

Cuenca-Acuna F and Nguyen T (2002) Text-based content search and retrieval in ad hoc p2p communities. Technical Report DCS-TR-483, Rutgers University, 2002.

Edutella, http://edutella.jxta.org (visited June 24th, 2005).

eDonkey, http://www.edonkey2000.com (visited June 24th, 2005).

eMule, http://www.emule-project.net (visited June 24th, 2005).

Gnucleus, http://www.gnucleus.com (visited June 24th, 2005).

Gnutella2, http://www.gnutella2.com (visited June 24th, 2005).

Gravano L, Chang C, García-Molina H and Paepcke A (1997) STARTS: Stanford proposal for Internet meta-searching. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 207–218.

Gravano L and García-Molina H (1995) Generalizing GlOSS to vector-space databases and broker hierarchies. In: Proceedings of 21st International Conference on Very Large Data Bases, pp. 78–89.

Intel (2003) Peer-to-peer content distribution: Using client PC resources to store and distribute content in the enterprise. White paper, Intel Information Technology.

IRIS, http://www.project-iris.net/ (visited June 24th, 2005).

Jansen M, Spink A and Saracevic T (2000) Real life, real users, and real needs: A study and analysis of user queries on the web. *Information Processing and Management*, 36(2): 207–227.

JXTA, http://www.jxta.org (visited June 24th, 2005).

KaZaA, http://www.kazaa.com (visited June 24th, 2005).

Kirsch S (1997) Document retrieval over networks wherein ranking and relevance scores are computed at the consumer for multiple database documents. U.S. Patent 5,659,732.

Limewire, http://www.limewire.com (visited June 24th, 2005).

Lu J and Callan J (2003) Full-Text retrieval in hybrid peer-to-peer networks. In: Proceedings of the 12nd International Conference on Information Knowledge Management, pp. 199–206.

Lu J and Callan J (2004) Merging retrieval results in hierarchical peer-to-peer networks. In: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 472–473.

Manku G, Bawa M and Raghavan P (2003) Symphony: Distributed hashing in a small world. In: Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS), pp. 127–140.

Maymounkov P and Mazières D (2002) Kademlia: A peer-to-peer information system based on the XOR metric. In: Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS 2002), pp. 53–65.

Morpheus, http://www.morpheus.com (visited June 24th, 2005).

MusicNet, http://www.musicnet.com (visited June 24th, 2005).

Nottelmann H and Fuhr N (2003) Evaluation different methods of estimating retrieval quality for resource selection. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 290–297.

Ogilvie P and Callan J (2001) Experiments using the Lemur toolkit. In: Proceedings of the 10th Text REtrieval Conference, pp. 103–108.

Ratnasamy S, Francis P, Handley M, Karp R and Shenker S (2001) A scalable content-addressable network. In: Proceedings the ACM SIGCOMM Conference, pp. 161–172.

RevConnect, http://www.revconnect.com (visited June 24th, 2005).

Rowstron A and Druschel P (2001) Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms, pp. 329–350.

Shareaza, http://www.shareaza.com (visited June 24th, 2005).

Si L and Callan J (2003a) A semi-supervised learning method to merge search engine results. *ACM Transactions on Information Systems*, 24(4): 457–491.

Si L and Callan J (2003b) Relevant document distribution estimation method for resource selection. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 298–305.

Stoica I, Morris R, Karger D, Kaashoek M and Balakrishnan H (2001) Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of the ACM SIGCOMM Conference, pp. 149–160.

Swapper.NET, http://www.revolutionarystuff.com/swapper/ (visited June 24th, 2005).

Tang C, Xu Z and Dwarkadas S (2003) Peer-to-peer information retrieval using self-organizing semantic overlay networks. In: Proceedings of the ACM SIGCOMM Conference, pp. 175–186.

Wang J, Reinders M, Lagendijk R and Pouwelse J (2005) Self-organizing distributed collaborative filtering. In: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 659–660.

Xu J and Croft W B (1999) Cluster-based language models for distributed retrieval. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 254–261.

Yaga, http://www.yaga.com (visited June 24th, 2005).

Zhai Z and Lafferty J (2001) A study of smoothing methods for language models applied to ad hoc information retrieval. In: Research and Development in Information Retrieval, pp. 334–342.

Zhang Y and Callan J (2001) Maximum likelihood estimation for filtering thresholds. In: Proceedings of 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 294–302.

Zhao B, Huang L, Stribling J, Rhea S, Joseph A and Kubiatowicz J (2004) Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1): 41–53.