The Design of PIRS, a Peer-to-Peer Information Retrieval System

Wai Gen Yee and Ophir Frieder

Database and Information Retrieval Laboratory
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616
yee@iit.edu, ophir@ir.iit.edu

Abstract. We describe the design of PIRS, a peer-to-peer information retrieval system. Unlike some other proposed approaches, PIRS does not require the centralization of data onto specially designated peers. It is therefore applicable to a larger environment. We explain our design decisions, analyzing its benefits and potential shortcomings. We then show that PIRS significantly improves over search performance found in todays P2P file sharing systems.

1 Introduction

Many of today's peer-to-peer (P2P) file sharing systems were initially conceived as successors to Napster, which was used primarily for the exchange of music. As such, they are designed to allow simple annotation of files, including the artist and song title.

As long as a file's metadata are well-known, searches are simple. A query matches a file if its terms are substrings of the metadata's terms. For example, consider two instances of the same song, annotated by the terms "Smashing Pumpkins 1979," and "Pumpkins Greatest Hit 1979," respectively. A query for a query "Smashing Pumpkins Greatest Hits" (which likely refers to the song) will not return either instance.

This problem illustrates a limitation of P2P search: It requires the user to know the exact metadata associated with an instance of the file to perform a successful search. This is problematic for the naive user, who is unaware of annotation conventions, or for a user not looking for a particular song, but for a particular type of music (e.g., "songs from bands from Chicago"). This problem is exacerbated by the fact that many song files are automatically annotated using Web databases, such as freedb.org. Such annotation results in the identical annotation of all copies of a particular song, and gives users disincentives to make their own annotations.

The goal of this paper is to describe how information retrieval (IR) can help alleviate this problem in a P2P environment. In our analysis, we spend some time describing the characteristics of P2P systems, and the degree to which existing

P2P systems possess them in Section 2. In Section 3, we informally describe the P2P file sharing model. We describe the design of PIRS, our **P**2P **IR S**ystem, in Section 4, and present some experimental results in Section 5. In Section 6, we discuss PIRS's compatibility with other work, as well as the future of P2P IR. We make concluding remarks and discuss future work Section 7.

2 Limits of Current Work in P2P Information Retrieval

2.1 Characteristics of P2P Systems

As described in [1], P2P systems are characterized by low maintenance overhead, improved scalability and reliability, synergistic performance, increased autonomy, privacy, and dynamism. P2P systems are inexpensive to maintain and have good scalability because they use the full resources of all participants, who function as both clients and servers. They are more reliable, because the failure of any one peer does not disable the entire system. They offer synergistic performance because peers can utilize resources (e.g., data, CPU cycles, disk space) that are underutilized by others. They are dynamic, allowing peers to autonomously join and leave the system, and thereby changing the types of resources offered. They offer privacy because P2P networks lack central authentication servers. P2P systems are therefore ideal in environments populated by many autonomous users with dynamic needs and surplus resources.

These characteristics distinguish P2P systems from previous technologies, such as distributed computing, and ad-hoc communication. Distributed computing refers to computing on a platform where resources are located on many physically distinct locales. Unlike P2P systems, these resources may be highly integrated and interdependent. Ad-hoc communication refers to a communication platform in which a client can automatically join an existing network. Ad-hoc networking deals with the lower-level communication infrastructure on which P2P systems can be built. P2P computing is therefore a unique paradigm that creates new possibilities for information technologies.

2.2 The Limits of Information Retrieval Using Web Search Engines

The Internet offers a medium by which everyone can easily gather, and share information. Today, the dominant paradigm for sharing information is via the Web. Organizations set up Web servers that host Web sites where they can publish information. Individuals also have a chance to publish information through personal Web pages or *blog* pages. To access this information, users simply type in the appropriate URL into a Web browser.

There is a gap, however, in bringing together information publishers and consumers–how exactly does one find the appropriate URL that points to desired information? Today, this gap is bridged by Web search engines, such as Google. A consumer enters some relevant terms into Google, which returns heuristically-defined best matches.

The problem with relying on Google to find published data is that publishers must wait for Google to *crawl* their Web pages before they appear in any search results. Because crawling occurs only periodically, material published today will not be found for some time. Another problem is that Google caches indexed content. Once this happens, publishers lose control over the dissemination of their material [2]. Furthermore, the results returned by search engines can be suspect. For example, rankings can be influenced by commercial interests [3]. Finally, centralizing query services limits the scalability and reliability of the search engine: A single server can only index so much content (Google claims to index slightly more than 4 billion Web pages, which is considered only a small fraction of those available), and also is a single point of failure. A more relevant example is Napster, whose centralized architecture made it easy prey for legal attacks.

Recent P2P file sharing systems that focus on file transfer, such as BitTorrent [4] suffer from the same problems as Web search engines. BitTorrent is different from Gnutella in that the former focuses on download performance, whereas the latter focuses on search. BitTorrent allows a client to download a single file from multiple sources simultaneously, thereby improving performance, and distributing load. However, it relies on Web search engines to provide search capabilities for shared content, and therefore has all the problems discussed above.

P2P systems solve many of these problems. Because queries are sent directly to data sources, results are up-to-date and reflect the currently available data. Upon receipt of results, the peer can use custom algorithms to rank them. This eases their perusal as users have more trust in their rankings. Finally, there is no single point of failure with P2P systems. A query likely returns results even with the failures of multiple nodes [5].

2.3 Current Peer-to-peer Information Retrieval Systems and their Limits

Work on P2P information systems has focused on either bandwidth efficiency, or the development of unifying data models. The PeerSearch system [6] is built on top of the CAN routing infrastructure [7]. CAN places content in a P2P network based on its hash value. PeerSearch proposes to create a distributed index, which is partitioned and similarly placed on a network. This deterministic placement of content improves bandwidth efficiency by constraining the way a query is routed. (The original version of Gnutella, in contrast, floods queries over the network [8].) In [9], the authors take a similar approach. They assume a hybrid networking architecture where some the peers that are deemed more reliable and capable act as servers. These servers, besides routing queries, also store metadata statistics, such as term frequencies, that are used by traditional IR algorithms.

Other systems, such as Edutella [10] and PeerDB [11] propose data models that standardize the way data and services are published and queried.

While these systems have much potential, they are limited due to the constraints that they put on the infrastructure and applications. The PeerSearch system works best in an environment where peers are reliably connected to the Internet. This is necessary because shared content is centralized in certain peers based on their hash values. The loss of a peer results in the loss of all its associated content or the transfer of massive quantities of data. Furthermore, it takes control of data placement out of the users' hands. These characteristics violate the principles of P2P systems that are described in Section 2.1. A solution to this problem is to replicate content by applying multiple hash functions on content. This is problematic as well, because it increases both the amount of data every peer must maintain and network traffic as well. Notably, no work we know of has been done on P2P information retrieval in highly dynamic environments where peers frequently join and leave the network.

Edutella and PeerDB focus more on standards than on information retrieval. Standardization, however, tends to raise the bar for entry into a network because it forces users to do more work to publish content. This has the effect of limiting the amount of data that are published, thereby reducing the network's overall usefulness [12].

Note that it is not our goal to be purists in P2P system design. The popularity of Napster (in terms of market impact and user satisfaction) demonstrates that, under certain conditions, there is no need. At the same time, pure P2P systems were shown to have scalability problems [13], which can be alleviated by the use of a hybrid architecture [14]. However, the fact that a system works without being purely P2P does not mean that it might not work better if it were so.

3 Model

In a typical file sharing model, each peer (which we may refer to as a client or a server, depending on the context) maintains a set of shared files (or *data objects*). Each file is annotated with a set of metadata *terms*, contained in a *descriptor*. The particular terms contained in a descriptor of an *instance* of a file is user-tunable.

Users create *queries* to find files in the P2P system. A query is a metadata set that a user thinks best describe a file. These queries are routed to reachable peers. (Queries generally do not reach all peers due to network conditions or details of the routing protocol.) Returned are pointers to instances of files that match the query, and the file's metadata set. The *matching criterion* is for all the query terms to be substrings of some term of the file's metadata set.

Users study the returned metadata sets to decide on the file to download. Once the user makes her selection, she downloads the file by selecting its associated file pointer. The client follows the pointer, downloads the file, and then becomes a server for an instance of that file.

Note that although our discussion uses music file sharing as an application, it also applies to other applications. For example, an HTML document is also a file that can analogously be annotated with metadata in the form of `META` tags. The terms in the `META` tags can be tuned independent of the "content" of the HTML document.

4 The Design of PIRS

4.1 Goals

Our goal is to design a P2P IR system that focuses on client behavior and is fully distributed. The system must make little or no assumptions about the underlying communication infrastructure and the behavior of servers (i.e., other peers). For example, CAN routing and PeerSearch (mentioned above in Section 2) tacitly assumes that the network is stable and servers are reliable. Consequently, although these systems have potentially excellent performance, violating either of these assumptions results in the loss of either queries or data. In this light, these systems tend to fall somewhat between the categories of distributed and P2P systems.

A system that does not make assumptions about the communication infrastructure and behavior of peers avoids these problems. The obvious questions to ask therefore are:

1. How well would such a P2P IR system work? For example, IR requires global statistics about the available data for effective ranking. In a highly dynamic environment, such statistics are hard to yield. Furthermore, even if such data were available, would it be possible to implement IR ranking functions in a P2P application? The question here is about performance in terms of query result quality as well as computational complexity.
2. Could such a system adapt to changes in system conditions? Making no assumptions in designing a P2P system may be too conservative an approach. In some cases, the network and peers are capable and reliable. Can the P2P system take advantage of this condition, if available? Gnutella's Ultrapeer architecture demonstrates adaptability; it conserves bandwidth given a stable environment, but also works (albeit less efficiently) in an unstable one [14].

Our goal is to answer these two questions. To do this, we describe the design of PIRS. In doing this, we highlight the complexities of applying IR techniques to a P2P environment.

4.2 Overview

PIRS is designed to combine the search capabilities of information retrieval systems with the dynamic network management of P2P systems. It works by managing metadata in such a way as to *gradually increase the variety* of queries that can be answered for a given file. This is done by adapting the annotation of a particular file to match query patterns. PIRS accomplishes its goals in three ways:

1. Metadata collection (Section 4.3) - Collect as much metadata as possible for a shared file, using various means. Increasing the amount of metadata increases the likelihood that a query will find matches. For simplicity, the size of the metadata set is generally limited in size, thus a decision must be made as to which metadata to maintain.

2. Metadata distribution (Section 4.4) - Heuristically replicate metadata from other peers when downloading a file. By sharing metadata from multiple peers, the variety of queries that can be matched for a given file increases. Again, the client must decide on a limited set of metadata to maintain.
3. Metadata use (Section 4.5) - Utilize IR techniques to rank results, disambiguating them, and thereby improving the likelihood of a correct download.

The processes of metadata collection, distribution, and use work together to improve the search capabilities of PIRS. Ostensibly, they can work independently to improve search, but with diminished benefits. For example, IR ranking functions alone can be incorporated into Gnutella, without PIRS's metadata distribution techniques.

By design, PIRS is simple to incorporate into many existing P2P protocols. This is a consequence of its functionality being concentrated on client behavior, and its independence from networking infrastructure. Many existing P2P protocols focus on aspects of query routing, which is independent of PIRS's functionality. Consequently, PIRS can be built on top of many of today's popular P2P file sharing applications, such as Gnutella and FastTrack. We discuss this in more detail in Section 6.

PIRS Versus Other P2P IR Systems The major difference between PIRS and other P2P IR systems is that PIRS treats metadata as a dynamic resource that should be managed collectively by all peers. Effective management of metadata improves query result quality. The inspiration of this work stems from the notion that, from a client's perspective, the P2P network is a repository of files, each of which is described collectively by a *body of metadata*. The better a file's body of metadata describes the file, the easier the file should be to find.

Current P2P IR systems do not have this perspective. They treat each download as an individual transaction, without regard to how it (the download) affects the file's body of metadata. The download of a file generally also results in the replication of that file's metadata from a particular server. The downloading client becomes an additional server for the file, but with marginal benefit, because the clients it serves are exactly those which the original server serves. The additional server's role in the network is largely redundant.

4.3 Metadata Collection

Metadata collection is the process by which a file is annotated with identifying terms. We now describe how metadata collection is typically done in commercial P2P systems. We also describe a unit of metadata that PIRS exploits for good performance.

Metadata terms are directly used for query matching. It is therefore important to build into PIRS effective means of annotating files. One of these means includes creating an easy to use user interface, which encourages users to add metadata. Other means include automatic annotation via metadata *foraging* and the use of *intrinsic* file characteristics.

Recent versions of P2P file sharing systems offer templates that help a user annotate certain types of files, such as audio files, using special application-specific fields [15]. These templates structure metadata, potentially increasing the query matching possibilities.

Much metadata are also automatically foraged from Internet sources. For example, when *wav* files are *ripped* from commercial compact disks, the ripping software automatically collects ID3 metadata (e.g., title, artist) [16] for it from Web sites such as freedb.org. Other metadata are *intrinsic* to the file. Such metadata include the size of the file, its filename, and the last time it was accessed. Making these metadata available for querying requires some simple programming.

Finally, some systems automatically derive useful metadata from the intrinsic characteristics of the file. BitTorrent, for example, generates a unique *hash key* (e.g., an SHA-1 hash [17]) for each file, which can simplify its search and be a means of validating the file's contents [4]. A hash key can also be used to group files that are returned by queries.

PIRS uses a file's hash key for validating and grouping files. Such use of the hash key has not been universally adopted. LimeWire's Gnutella groups by filename, file type, and file size [18]. BearShare and eDonkey only use hash keys to authenticate files.

One problem with requiring all files to be annotated with a hash key is its computational cost. This problem has been acknowledged by BearShare, which claims that computing keys in a background process takes only 25% of a CPU's cycles [19]. Hash keys can also be computed while a file is being downloaded, extracted (if it is compressed), or ripped. Piggybacking these processes amortizes the cost of computing the hash key.

Maintaining a hash key for files also does not hurt PIRS's compatibility with existing P2P file sharing systems. It would be treated as another generic unit of metadata by a peer that did not realize its significance.

4.4 Metadata Distribution

Metadata distribution is the process by which peers exchange metadata with each other in order to describe a file. In this process, each peer does just a little work to better collectively describe shared data. This process complements metadata collection in building an effective body of metadata for each shared file.

Metadata distribution is crucial for two reasons. First, if metadata are not distributed among multiple nodes, then the system may become vulnerable. If all metadata were concentrated on a single node (e.g., as with Napster), the system becomes unusable if that node becomes unreachable. This vulnerability violates a basic principle of P2P systems.

Second, data that are not distributed *properly* could leave correlations in term occurrences, which limit the degree of query matching. For example, assume there are two metadata ripping systems for song files: one extracts the album name (denoted t_1) and the song's track number (t_2), and the other extracts the album's label (t_3) and year (t_4). If files were only annotated using these two rippers, then a query, $\{t_1, t_3\}$ would not return any results due to the matching criterion.

PIRS distributes metadata in a way that avoids this problem. During a query, it groups all metadata for each unique file in the results. When a user selects a file, metadata are heuristically replicated from the file's group onto the client. Grouping of unique files is straightforward, as each result is assumed to contain the file's hash key. The heuristics we consider for metadata replication include:

– Server terms (**server**) - The client selects the metadata that exist on the single server from which it downloads. This is the solution that is commonly used in today's P2P file sharing systems. It is notable for its simplicity.
– Most frequent terms in the group (**mfreq**) - The client selects the terms that occur most frequently in the group. The justification for this approach stems from the assumption that, because these terms appear so much, they are strongly associated with the file, and therefore most likely to occur in queries.
– Least frequent terms (**lfreq**) - The client selects the terms that occur least frequently in the group. The usefulness of this approach is that these terms help distinguish this file from others. It also balances out the term distribution.
– Random terms (**rand**) - The client randomly selects terms from the group, maximizing the number of term combinations.
– Random terms based on freq (**wrand**) - The client randomly selects terms from the group weighting more frequently occurring terms proportionately higher. Like **rand**, this technique also increases the number of term combinations, but gives preference to more commonly occurring terms.

In the last four techniques, **mfreq**, **lfreq**, **rand**, and **wrand**, the client selects metadata terms until it reaches a system defined limit.

These metadata distribution techniques are an improvement over the current technique of replicating metadata from a single server. They increase the variety and sizes of metadata sets, and thereby should improve their ability to accurately describe a file. The effects they have on the states of bodies of metadata vary, however, and a goal of the PIRS project is to examine their effects of query results quality.

4.5 Metadata Use

IR style ranking in P2P systems is difficult, due to certain characteristics of P2P systems. For example queries are short and peers are unreliable. PIRS acts a testbed for both traditional and P2P-specific ranking functions. Specifically, we use PIRS to determine the dependence of ranking functions on metadata distribution techniques.

We consider five ranking functions. Some of these techniques are classical IR techniques, and some are unique to P2P file sharing:

– Group size (**gsize**) - The number of results in a group. A large GS indicates that either a particular file has large support for satisfying a query, or that the file is generally popular, and is therefore something desirable anyway.
– Term frequency (**tf**) - Counting the number of times query terms appear in a file's metadata. Terms that occur frequently in metadata sets likely represent the contents of the file.
– Precision (**prec**) - Dividing TF by the total number of terms in the group. Precision adjusts for problems with TF caused by large metadata sets.
– Cosine similarity (**cos**) - Cosine similarity maps group descriptors and the query to vectors. It ranks highest the groups with the descriptor vectors that have the highest cosine similarity to the query vector.

We implemented other ranking functions, such as term frequency with inverse document frequency (**tf/idf** from [20]). **Tf/idf**, however, requires some modification, because global information on the number of documents in which each term appears, required by **tf/idf**, does not exist in P2P systems. We instead approximate *document frequency* by the number of query results in which a term appears. Since **tf/idf** is another variation of vector space model ranking, of which **cos** is a representative, and its performance is not much different, we do not further discuss it.

4.6 Implementation Issues

Due to the distributed nature of a P2P system, query results arrive at clients asynchronously, over a period of time. The client must be able to display these results and update their rankings in real-time.

PIRS groups each of the N results in $O(\log N)$ time using the hash key. It also updates rankings for all results within $O(N^2)$ time, depending on the ranking function. While this complexity is a current upper bound, it is within the $O(N^2 \log{(N)} KM)$ complexity of grouping of Limewire's Gnutella [18], where K and M are grouping similarity metrics. More details about the implementation of grouping and ranking in PIRS are posted on the authors' Web sites.

5 Experimental Results

We now demonstrate the effect that metadata distribution and ranking have on query result quality via simulation. We measure performance by the number *successful* queries (i.e., those that lead to the download of the desired data object) that the clients perform. We do not consider traditional IR metrics, such as *precision* and *recall*. Precision measures the percentage of correct results to a query, and is irrelevant because, in our model, the user requests a specific data object, and any replica of the desired data object will satisfy her. For the same reason, recall, the percentage of possible results returned, is also irrelevant in our model.

5.1 The Simulator

The design of our simulator is based on observations and analyses of P2P file sharing systems. In the event that relevant design parameters are unavailable, we borrow from work on done on Web information systems and IR.

The major objects in our simulator are terms, data objects, peers, and queries. The **universal set** of terms T that can describe a data object is finite, and each term is assigned a relative access probability based on the accepted Zipf distribution [21]. A random number of terms from T are assigned to each data object's (F_i) universal term **subset** (T_i) based on the initial Zipf distribution. The terms of each data object's universal term subset are then reassigned probabilities according to a Zipf distribution to diversify term usage, as described in [22]. For example, a term that is rarely associated with one data object need not be so for another. We call the set of probabilities that terms will be associated with a data object the data object's **natural (term) distribution**.

We also make the generally unrealistic assumption that terms are independent. For example, the occurrence of "Britney" in a descriptor is independent of the occurrence of "Spears". This is incorrect in general, but is common practice, as it simplifies the simulation environment without making it trivial. Note, however, that this term independence assumption is not unique to our work. Such an assumption is heavily relied upon in the probabilistic information retrieval model in IR.

Each data object is also associated with an access probability, according to a Zipf distribution. This conforms to the access patterns observed for Web objects that were described in [23]. Observations of data object frequency in a P2P system also suggest a high access skew [24].

Initially, a random number of copies of each data object are instantiated, each with a subset of its universal term subset in its descriptor. These copies are assigned to random clients.

There are a fixed number of peers and a fixed number of data objects in the system. At each iteration of the simulation, a random peer is chosen to download a random data object based on the data object' access probability distribution. To do this, the peer generates a query of random length containing a subset of the data object's universal term subset. We assume that length distributions follow those of Web search engines, and use the empirical distribution described in [25]. Personal observations of queries in LimeWire's query monitor window seems to corroborate this assumption. Each term in the query is randomly chosen based on the data object's natural term distribution.

The query is routed to a random subset of servers. We do not send the query to all servers because, in practice, only a subset of them is reachable at any time [24]. The servers return results that fulfill the matching criterion (Section 3) to the client.

Client Behavior If more than one group forms in response to a query, then the client ranks the groups. The highest ranked group is selected for download. Although, in general, the user may be equally likely to select any one out of the first few highest ranked groups, all else being equal, we can generally assume that she will select the one that is highest ranked.

We say that the query is **successful** if the desired data object is ranked first and downloaded. In an unsuccessful query, either the incorrect group is ranked first and downloaded, or there are no results.

Once the data object is downloaded, the user has a probability of manually annotating the data object with some personally chosen terms. These terms are randomly chosen from the data object's universal term subset, based on the natural term distribution. This is the only way that the variety of terms that exists in the system for a data object can increase beyond what exists at initialization. If the user downloads the incorrect data object, then she may mis-annotate it in this step, leading to incorrect metadata for the data object.

After this is done, the client heuristically copies some of the chosen group's metadata into the replica's descriptor, with the constraint that only a limited number of terms may be copied. The data object is then available for other peers to find in subsequent iterations of the simulation.

We do not model freeriders or malicious users. Freeriders are users who download, but do not upload data objects. Since they do not contribute any metadata to the system, they do not affect the results. Malicious users are those who may contribute misleading metadata for data objects to the system. These users may affect the rankings, but only marginally. Rankings are based on the aggregate metadata of a group of users, not on the metadata of an individual.

Parameter	Value or Range
Number of peers	1000
Number of data objects	1000
Number of terms in universal set	10000
Number of terms in the universal set of a data object	100-150
Maximum descriptor size for a data object on a peer (terms)	20
Number of terms in initial descriptors	3-10
Number of replicas of each data object at initialization	3
Probability that a peer is reachable	0.5
Probability of client adding metadata	0.05
Number of Terms Added by client	1-5
Query length	1-8, dist from [25]
Number of queries	10000
Number of trials	50

Table 1. Parameters Used in the Simulation.

The parameters we use in the experiments are shown in Table 1. The size of the simulation is scaled down to reveal any convergences in the results more quickly. More significant than the scale of the simulation are the relative values of each parameter, such as the total number of possible terms for a data object, versus the number of terms with which each data object is initially annotated. These numbers are based on observations from other studies [26, 24], as well as personal observations. For example, song data objects that appear on Gnutella networks typically have about three or more types of information associated with them from ID3 data: artist, song name, album name, track number, etc. This is reflected in the *Number of terms in initial descriptors* parameter.

We performed fifty trials with each set of parameters and report the average results. The 95% confidence intervals generally were well within 4% of the reported mean–the results are statistically significant. However, to simplify the presentation of the main results, we do not present them.

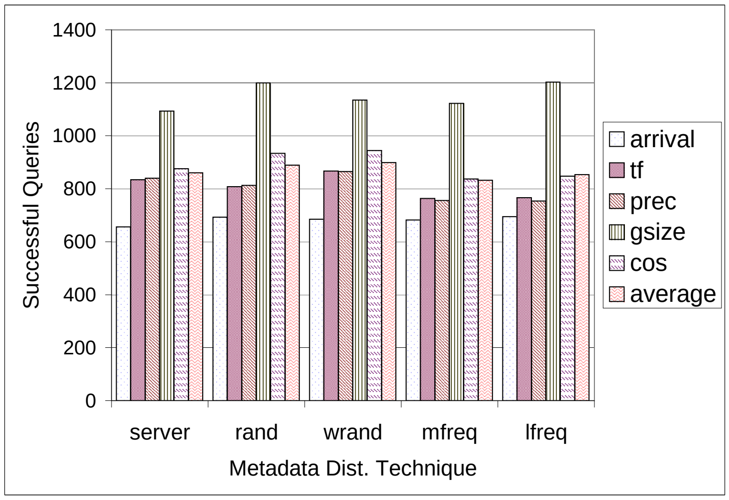

Fig. 1. Number of Successful Queries vs. Metadata Distribution Technique for Multiple Ranking Functions.

5.2 Results for Various Combinations of Metadata Distribution and Ranking

We see that **gsize** is the best ranking function regardless of metadata distribution technique in 1. This is somewhat surprising, considering its simplicity. **Gsize** works relatively well because only the correct data object will likely contain *all* query terms, and thereby satisfy the matching criterion. Other data objects' descriptors may be near-misses. **Cos** does a distant second best. It does poorly because the matching criterion does not return an unbiased sample of results; all results contain all query terms. It therefore cannot discriminate between relevant and irrelevant very well. Finally, **prec** and **tf** are subject to the same problems they have in traditional IR; they are highly influenced by large metadata sets or by noise.

We also see that the metadata distribution techniques that randomize the metadata (**rand** and **wrand**) do best on average. Furthermore, the combination of **rand** and **gsize** do the best. However, no ranking function in combination with **wrand** outperforms **gsize** with **lfreq**. The reason that **lfreq** does better than **wrand** in this case is that **lfreq** is better at increasing the total number of terms in the body of metadata for a data object. **Wrand** replicates common terms, introducing a skew. Descriptor space is therefore occupied by repetitions of common terms. **Lfreq**, in general, replicates terms that do not occur frequently. It therefore has the effect of replicating every term in the long run. This larger body of metadata is in this sense more descriptive, resulting in more relevant results for a query. Besides **gsize**, the best performing ranking function using **wrand** is **cos**. This is expected, because **cos** requires term frequencies to be skewed in order to work correctly. **Lfreq**, by comparison, does worse with **cos**, because it results in uniform distributions of terms in bodies of metadata.

Server and **mfreq** do poorly because they fail to mix the metadata during distribution. **Server** simply replicates a single server's descriptor. **Mfreq** replicates terms that have already been frequently replicated. This does little to increase the variety of descriptors that describe a data object.

6 Discussion

6.1 Compatibility with Existing Technologies

A feature of PIRS is that it is compatible with Gnutella. A PIRS peer interacts with others via message passing. These messages are in standard Gnutella format, and no special messages are required. Furthermore, no special architectures are required by PIRS. PIRS simply allows peers to create and respond to queries in a way that is transparent to standard Gnutella peers. A PIRS peer can therefore readily integrate itself into an established P2P file-sharing system.

In a similar vein, PIRS can also take advantage of optimizations designed for Gnutella-like P2P systems. Routing optimizations, such as *shortcuts* [27] and Ultrapeers [14] are available. Search optimizations, such as specially designated index nodes are also possible [9]. In the latter case, although special index nodes improve the quality of search results, they do not obviate the need for metadata distribution and client-side ranking of results.

6.2 The Outlook for Peer-to-Peer Information Retrieval

P2P file sharing system vendors have been actively pursuing new markets. Kazaa, for example, has adapted its networks for content distribution for media companies, online dating with MatchNet, and voice-over-IP telephony with Snype [28]. These new applications will surely bring new users into the area.

Other industry trends seem to indicate that P2P information retrieval will be a strategic technology in the near future. Google is currently working on Puffin, a desktop search tool that helps users find information stored on their desktops [29]. Whether this is a counterattack to Microsoft's Longhorn [30] strategy or not, it signals a new focus on harnessing the information stored on desktops.

P2P file sharing has been a consistently active Internet activity for the last several years. This condition shows no sign of weakening, despite recent legal actions by the recording industries [31]. As the user base and variety of P2P applications grows, PIRS and other P2P search tools will only gain in significance.

7 Conclusion

P2P file sharing is a popular activity among Internet users, and shows little signs of slowing down. As volumes of data grow, so does the need for good IR to sort through the results. PIRS, our P2P IR system is one solution. It is compatible with current P2P file sharing systems, but more powerful.

PIRS is flexible, as P2P systems should be. Unlike other work in P2P IR, it allows for unpredictable user behavior, and makes no assumptions about the underlying network. As a dynamic system, it also escapes some of the problems that exist when using centralized systems, such as Web search engines; data can be made available much quicker.

PIRS is unique in that it allows users to tune the ways in which a client distributes metadata. It treats the metadata that exist in all instances of a data object in the system as a collective description of the data object. With improved descriptiveness, query results improve in quality.

PIRS also includes various ranking functions. Our simulation results show that the effectiveness of ranking somewhat depends on the metadata distribution, and that the correct combination can improve performance from 15% up to 90%.

We are currently considering the relationship between the matching criterion and ranking functions. The current matching criterion is based on conjunctive queries. Although this economizes on bandwidth consumption, it may reduce the quality of queries results. We are considering the effect of alternatives, such as disjunction. We will focus on server-side responses to queries.

References

1. Milojicic, D.S., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B.: Peer-to-peer computing. Technical Report HPL-2002-57, Hewlett-Packard Laboratories, Palo Alto (2002)
2. Noguchi, Y.: Online search engines help lift cover of privacy. Washington Post (2004) Feb. 9, 2004.
3. Hansell, S.: Yahoo to charge for guaranteeing a spot on its index. New York Times (2004) Mar. 2, 2004.
4. Cohen, B.: Bittorrent home page. (Web Document) bitconjurer.org/BitTorrent.
5. Watts, D.J., Strogatz, S.H.: Collective dynamics of small-world networks. Nature **393** (1998)
6. Tang, C., Xu, Z., Dwarkadas, S.: Peer-to-peer information retrieval using self-organizing semantic overlay networks. In: Proc. ACM SIGCOMM. (2003)
7. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: Proc. ACM SIGCOMM. (2001)

8. LimeWire, LLC: Gnutella protocol 0.4. Web Document (2004) www9.limewire.com/developer/gnutella_protocol_0.4.pdf.
9. Lu, J., Callan, J.: Content-based retrieval in hybrid peer-to-peer networks. In: Proc. ACM Conf. on Information and Knowledge Mgt. (CIKM). (2003) 199–206
10. Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmr, M., Risch, T.: Edutella: A p2p networking infrastructure based on rdf. In: Proc. World Wide Web Conf. (2002)
11. Ng, W., Ooi, B.C., Tan, K., Zhou, A.: Peerdb: A p2p-based system for distributed data sharing. In: Proc. IEEE Intl. Conf. on Data Eng. (ICDE). (2003)
12. Google, I.: Simplicity and enterprise search. Technical report, Google, Inc. (2003)
13. Ritter, J.: Why gnutella can't scale. no, really. Web Document (2001) www.darkridge.com/∼jpr5/doc/gnutella.html.
14. Singla, A., Rohrs, C.: Ultrapeers: Another step towards gnutella scalability. Technical report, Limewire, LLC (2002) rfc-gnutella.sourceforge.net/src/Ultrapeers_1.0.html.
15. Thadani, S.: Meta information searches on the gnutella network. (Web document) www.bearguru.com/kb/articles/metainfo_searches.htm.
16. Nilsson, M.: Id3v2 web site. Web Document (2004) www.id3.org.
17. of Standards, N.I., Technology: Sha1 version 1.0. Web Document (1995) www.itl.nist.gov/fipspubs/fip180-1.htm.
18. Rohrs, C.: Search result grouping {in gnutella}. Technical report, LimeWire (2001) www.limewire.org/project/www/result_grouping.htm.
19. Free Peers, Inc.: Bearshare technical faq. Web document (2004) www.bearshare.com/help/faqtechnical.htm.
20. Grossman, D., Frieder, O.: Information Retrieval: Algorithms and Heuristics. Number ISBN 0-7923-8271-4. Kluwer Academic Publishers (1998)
21. Knuth, D.E.: The Art Of Computer Programming. Second edn. Volume 3:Sorting and Searching. Addison-Wesley Publishing Company (1975)
22. Schlosser, M.T., Condie, T.E., Kamvar, S.D.: Simulating a file-sharing p2p network. In: Proc. Wkshp. Semantics in Peer-to-Peer and Grid Comp. (2003)
23. Crovella, M., Bestavros, A.: Self-similarity in world wide web traffic: evidence and possible causes. IEEE/ACM Trans. Networking **5** (1997) 835–846
24. Saroiu, S., Gummadi, P.K., Gribble, S.D.: A measurement study of peer-to-peer file sharing systems. In: Proc. Multimedia Computing and Networking (MMCN). (2002)
25. Reynolds, P., Vahdat, A.: Efficient peer-to-peer keyword searching. In: Proc. ACM Conf. Middlware. (2003)
26. Ripeanu, M., Foster, I.: Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. In: Intl. Wkshp. on P2P Sys. (IPTPS). Number 2429 in LNCS (2002)
27. Sripanidkulchai, K., Maggs, B., Zhang, H.: Efficient content location using interest-based locality in peer-to-peer systems. In: Proc. IEEE INFOCOM. (2003)
28. CBC: The future. CBC/Radio-Canada (2004) www.cbc.ca/disclosure/archives/040309_swap/future.html.
29. Markoff, J.: Google moves toward clash with microsoft. New York Times (2004) May 19.
30. Microsoft, Inc.: Longhorn development center. Web Document (2004) msdn.microsoft.com/longhorn/.
31. Reardon, M.: Oops! they're swapping again. CNET News (2004)