Masked Queries for Search Accuracy in Peer-to-Peer File-Sharing Systems

Wai Gen Yee, Linh Thai Nguyen, Ophir Frieder
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616
yee@iit.edu, nguylin@iit.edu, ophir@ir.iit.edu

Abstract

Peer-to-peer file-sharing systems suffer from the over-specification of query results due to the fact that queries are conjunctive and the descriptions of shared files are sparse. Ultimately, longer queries, which should yield more accurate results, actually do the opposite. The judicious masking of query terms circumvents the shortcomings of conjunctive query processing, significantly improving query accuracy.

1. Introduction

Peer-to-peer (P2P) file-sharing is a popular Internet application, with millions of users sharing petabytes of data [21]. Due to this application's scale, it is vital that results presented to the user are accurately ranked.

Two characteristics of P2P file-sharing, however, make accurate ranking difficult: sparse description of shared files and conjunctive query processing [11]. Sparse description is a consequence of the fact that many (binary) files are described by their filenames, which are limited to about 200 bytes, and perhaps by a small amount of metadata embedded in the actual binary (e.g., ID3 data [7]). Benefits of conjunctive query processing include its simplicity and its conservative use of network bandwidth. Together, these characteristics conspire to decrease the accuracy of search with longer queries. This behavior is contrary to expected behavior of most search engines.

Result set *precision* (the percentage of the result set that is desired) increases with query length in the P2P file-sharing environment, as expected. However, at some point, the query becomes so constrained that no instances of the desired result are returned, reducing overall *accuracy* (described in more detail in Section 4). Indeed, recent measurement studies suggest that most P2P network traffic consists of far more queries than results [19]. One factor for this may be over-specific queries.

1-4244-0910-1/07/$20.00 ©2007 IEEE.

Consider a search for Mozart's *Clarinet concerto*, preferably in the key A major, by clarinetist Michele Zukovsky. We conducted a search for this song recently on the eDonkey file-sharing system with various combinations of query terms. This experiment revealed that increasing the number of query terms generally yields fewer but more precise results. However, a query containing all candidate query terms returned no results. (The number of results is denoted *nresp* in Table 1.)

Table 1. Number of results with various queries issued on the eDonkey P2P file-sharing system.

terms					nresp /
mozart	clarinet	A	major	zukovsky	query
X	X				80
X	X	X	X		54
X	X	X	X	X	0
X	X			X	2

It is only with an appropriate subset of terms, that we retrieve the desired result; the last combination in Table 1 contained only relevant results. That this combination yields the correct results also proves that the empty result set for the full query was not caused by the desired result's non-existence in the system but by query over-specification. Note that issuing the full query on Google resulted in better accuracy than did any sub-query.

We address the *query over specification* problem described above by having the client automatically *mask* out a subset of long queries before they are sent to servers. Shortening the query increases the size of the result set, thereby increasing the likelihood that it contains at least one instance of the desired result.

In our experiments, masking queries has a significant impact on accuracy, increasing it by 40%. To understand these results, two questions must be addressed:

1. How should candidate query terms be chosen for masking?
2. How many terms should be masked?

We discuss possible alternatives in answering these two questions in Sections 5 and 6. In Section 7, we differenti-

ate a masked long query and an initially short (base) query, namely by how the former yields more accurate results.

Masking comes at a cost; longer queries that are expected to return fewer results, conserving computing and network resources, no longer do so. In our experiments, the number of returned results increases by a factor of 3. Although some may argue that the increased accuracy is a worth the expense, we discuss possible cost-controlling measures in Section 8. Surprisingly, in some cases, it is possible to both increase accuracy and reduce cost.

2. Query Processing Specification

Each peer in the system shares an individually maintained local repository of binary files. Files may be replicated across peers, and each replica is identified by a user-tuned descriptor, which also contains an identifying key (e.g., an MD5 hash on the file's bits). All replicas of the same file naturally share the same key. A client's query is routed to all reachable servers until the query's time-to-live expires. Servers compare each query to their local descriptors; a query *matches* a replica if it is contained in the replica's descriptor. In this case, the server returns its system identifier and the matching replica's descriptor as a result. This information is necessary to allow the client to distinguish and download the associated file.

Formally, let O be the set of files, M be the set of terms, and P be the set of peers. Each file $o_1,o_2 \in O$ has a key associated with it, denoted k_{o1}, such that $k_{o1}=k_{o2}$ if and only if $o_1=o_2$.

Each file o has a set of terms, $T_o \subseteq M$, that *validly* describe it. Intuitively, T_o is the set of all terms an average person might use to describe o. Each term $t \in T_o$ has a strength of association with o, denoted $soa(t, o)$, where $0 \leq soa(t, o) \leq 1$ and $\sum_{t \in To} soa(t, o)=1$. The strength of association a term t has with a file o describes the relative likelihood that it is to be used to describe o, assuming all terms are independent. The distribution of *soa* values for a file o is called the *natural term distribution* of o.

A peer $p \in P$ is defined as a pair, (R_p, g^p), where R_p is the peer's set of replicas (i.e., its local repository) and g^p is its unique identifier (e.g., its IP address). Each replica $r^o{}_p \in R_p$ is a copy of file $o \in O$, maintained by p, and has an associated locally maintained descriptor, $d(r^o{}_p) \subseteq M$, which is a multiset of terms. Each descriptor $d(r^o{}_p)$ also contains k_o, the key of file o. The maximum number of terms that a descriptor can contain is fixed.

A query $Q^o \subseteq T_o$ for file o is also a multiset of terms. The terms in Q^o are expected to follow o's natural term distribution. When a query Q arrives at a server p, the server returns *result set* $U^Q{}_p=\{(d(r^o{}_p), g^p) \mid r^o{}_p \in R_p$ and $Q \subseteq d(r^o{}_p)$ and $Q \neq \emptyset\}$—membership in the result set requires that a result's descriptor contain all query terms, in accordance with the matching criterion.

The client that issued Q receives result set $U^Q=\cup_p U^Q{}_p$, $p \in P$, and groups individual results by key, forming $G=\{G_1, G_2, \ldots\}$, where $G_i=(d(G_i), i, l_i)$, $d(G_i)=\{\oplus d(r^j{}_p) \mid (d(r^j{}_p), g^p) \in U^Q$ and $k_i=i\}$ is the group's descriptor, i is the key of G_i, and $l_i=\{g^p \mid (d(r^j{}_p), g^p) \in U^Q$ and $k^j=i\}$ is the list of servers that returned the results in G_i. In this definition, $\oplus$ denotes the multiset sum operation.

To measure the relevance of query results to the user's desires, the client assigns a rank score to each group with function $F_i \in F$, defined as F: $2^M \times 2^M \times \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{R}^+$. If $F_i(d(G_j), Q, |G_j|, \text{time}_{Gj}) > F_i(d(G_k), Q, |G_k|, \text{time}_{Gk})$, where G_j, G_k are groups, then we say that G_j is ranked higher than G_k with respect to query Q and ranking function F_i. In these definitions, $|G_j|$ is the number of results contained in G_j and time_{Gj} is the creation time of the G_j (i.e., the time when the first result in G_j arrived at the client).

In commercial P2P file-sharing systems, such as various implementations of the Gnutella protocol or eDonkey, file keys are generated by the MD5 or SHA-1 cryptographic hash function, and results are grouped based on these keys. Ranking is based on *group size*, as a large group can better ensure a quick, successful download:

$$F_G(d(G), Q, |G|, \text{time}_G) = |G|.$$

Descriptors in these systems are generally implemented via filenames, but a small amount of descriptive information may be embedded in the actual binary of the replica, as mentioned in Section 1. Furthermore, when a file is downloaded, the descriptor of this new replica is initialized as a duplicate of one of the servers' in the result set.

To simplify our explication, we use the term "result" informally to describe either a group or an individual result, and clarify the usage if necessary. We refer to the collective set of terms contained in (individual result or group) descriptors as metadata.

3. The Information Tradeoff

If a user adds a unique term t to Q^o then s/he increases the amount of information Q^o contains about his/her interests by a unit. However, due to conjunctive matching, the addition of t to Q^o decreases $P(Q^o \subseteq d(r^p))$ – the probability that Q^o matches $d(r^p)$ – by a factor of at least $1-(1-soa(t,p))^{length(d(rp))} < 1$, where $length(d(r^p))$ is the number of terms in $d(r^p)$; as we gain information about user interests linearly, we lose information about results exponentially. This is a problem because we may be excluding the last instance of the desired result from the result set (as shown in the example in Table 1 of Section 1) or we may be excluding information from the result set, in the form of descriptor terms, that may help us identify irrelevant results.

Alternatively, although excluding t from Q^o may in-

clude r^p in the result set, a shorter Q^o means that the client has less information on user interests, thereby reducing the query's distinguishing power. This compromises the client's ability to effectively rank results. The *information tradeoff* is therefore between the user and the system: more user information leads to less result information, and less user information leads to more result information. Either way, query accuracy is compromised.

We address the information tradeoff caused by long queries by relaxing the conjunctive matching criterion through masking. In Section 8, we demonstrate how to exploit the additional information existing in longer queries to further improve query accuracy.

4. Experimental Setup

We motivate our discussion on masking with some experimental results. We simulate the performance of a P2P file-sharing system to test the large-scale performance of our methods. In accordance with the accepted model described in [14] and observations presented in [13], we include in our experimental model *interest categories*, a partitioning of O into sets $C_i \in C$, where $C_i \subseteq O$, and $\cup C_i = O$. Interest categories are used to model constraints on user interests.

Each category C_i has popularity b^i, which is skewed using a Zipf distribution, to model the fact that some interest categories are more popular than others. At initialization, each peer p is randomly assigned a number of interests $I_p \subseteq C$, based on b^i.

Each file o within each instance of an interest category varies in popularity, which is also skewed using a Zipf distribution. This popularity governs the likelihood that a peer who has interest in the category containing o is either initialized with a replica of o or decides to search for it. Each replica, $r^o{}_p$, allocated at initialization has a randomly initialized descriptor subject to o's natural term distributions. Peer p's interest categories also constrain its searches; p only searches files from $\cup L_i$, where $L_i \in I_p$.

We use Web data to simulate our term distributions and interest categories. Web data are a convenient choice because they constitute a grouping of terms into documents (we use terms' relative frequencies in documents to simulate natural term distributions for files) and a grouping of documents into domains (we use domains to simulate interest categories). The use of Web data to populate P2P simulations is common practice (e.g., [24]). Real data from P2P applications are preferable, but no standard sets are known [22]. (Recently, a data set for P2P text repositories has been designed [4], but we are considering the sharing of binary files. We are currently investigating the creation of an appropriate data set using a P2P network crawling tool we recently developed [25].)

Our data consist of an arbitrary set of 1,000 Web documents from the TREC 2GB Web track (WT2G). These documents come from 37 Web domains. Terms are stemmed, and markup and stop words are removed. The final data set contains approximately 800,000 terms, some 37,000 of which are unique. We also conducted experiments using other data sets with other data distributions, but, due to space constraints, we only present a representative subset of our results. The data used for all experiments can be found on our Web site [20]. The other experimental results are available on request.

Terms for a query are picked randomly based on the desired file's natural term distribution. The query length distribution was derived from observations of query logs we collected over several days in the Spring and Summer of 2006 using modified LimeWire file-sharing software (Table 2) [25]. The simulation parameters listed in Table 3 are based on observations of real-world P2P file-sharing systems and are comparable to the parameters used in the literature.

Table 2. Distribution of query lengths.

Length	1	2	3	4	5	6	7	8
Prob.	.28	.30	.18	.13	.05	.03	.02	.01

Table 3. Parameters Used in the Simulation.

Parameter	Value(s)
Num. peers	1000
Num. queries	10,000
Max. descriptor size (terms)	20
Num. terms in initial descriptors	3-10
Num. categories of interest per peer	2-5
Num. data objects per peer at initialization	10-30
Num. trials per experiment	10

Although other behavior is possible, we assume that the user identifies and downloads the correct file with a probability 1/*rank*, where *rank*≥1 is its position in the ranked set of results.

Main performance (i.e., accuracy) is measured using a standard metric known as *mean reciprocal rank score* (***MRR***), defined as

$$MRR = \frac{\sum_{i=1}^{N_q} \frac{1}{rank_i}}{N_q},$$

where N_q is the number of queries and $rank_i$ is the rank of the desired result in query i's result set. If the desired result is not in the result set, then $rank_i = \infty$. MRR is an appropriate metric in applications where the user is looking for a single, particular result, as is generally the case in P2P file-sharing systems.

The percentage of result sets containing the desired result (**percentage-contained**, for short and denoted *pctcont* in the figures) is also reported. Percentage-contained is the upper bound for MRR: only if a result set contains the desired result can it contribute to MRR.

For reference, we also use the metrics precision and recall in our analyses. They have slightly different semantics in the P2P environment than they do in traditional IR due to the existence of data replication. Let A be the set of replicas of the desired file existing in the system, and R be the result set of the query. Precision and recall are defined as

$$precision = \frac{|A \cap R|}{|R|}, \quad recall = \frac{|A \cap R|}{|A|}.$$

Because precision and recall are often inversely related, they are often replaced by their linear combination in the f-score metric:

$$f - score = \frac{2 \times precision \times recall}{precision + recall}$$

Finally, because network load and the work peers have to do to process a query is proportional to the number of query results, we use it as our basic cost metric. We report the number of query results over the 10,000 queries (denoted *nres*) in units of10^7 results.

5. Masking Technique

Masking a term from a query results in all instances of the term (if the term is repeated) being removed from the query. Given a query Q and a term t with a frequency $freq(t, Q)$ to mask from it, we define the following operation using multiset notation:

$$mask(Q, t) = Q - (t, freq(t, Q)).$$

When a client masks a query, it executes the mask operation on $N_M < |Q|$ unique terms from **base query** Q that are chosen by a **masking metric** S_M. (We refer to N_M as the **degree of masking**.) The result of the N_M mask operations is a **masked query**, Q^M, which is sent to servers.

MaskingTechnique(S_M, Q, N_M)

1. Set Q^M to Q.
2. Rank $t \in Q^M$ by masking metric S_M.
3. Mask the min(N_M, $|Q|$-1) top S_M-ranked terms from Q^M.

The results of Q^M are grouped and ranked by the client as described in Section 2.

To reiterate, masking is a technique designed to counteract the conjunctive nature of queries in P2P file-sharing systems. Its simplicity makes it immediately applicable to many P2P environments.

6. Masking Performance

The goal of masking is to increase the accuracy of query results by maximizing the query's recall – specifically, to increase its likelihood of yielding the desired result. The two basic questions that must be addressed are how many and which terms to mask.

To maximize recall, all but one query term should be masked. By the nature of conjunctive queries, the result sets of shorter queries are necessarily supersets of the result sets of longer queries. The term that is left in the query should be the one least likely to disqualify the desired result from being in the result set. Equivalently, the term(s) with the greatest strength(s) of association are kept in the masked query.

The problem with this technique is that the client does not know a priori the terms' strengths of association. We therefore use a term's frequency within a query as a means of approximating it, making the assumption that the frequency at which a term appears in the query is proportional to the expected frequency at which it appears in a descriptor. Referring back to the technique described in Section 5, the masking metric, S_M, is "lowest $freq(t, Q)$," where t is a term in query Q. We call this masking technique *min-qtf* for "minimum query term frequency." *Max-qtf* is analogously defined.

We report the performances of the following masking techniques, with masking degree N_M=7, in Figure 1:

- *nomask* – no masking,
- *min-qtf* – mask the least frequent terms from Q,
- *max-qtf* – mask the most frequent terms from Q,
- *min-soa* – mask the lowest strength of association terms from Q.

Max-qtf is included in the results to demonstrate how different masking techniques might influence the results. *Min-soa* is included how well masking would work with global knowledge.

MRR is higher using masking because of the increased recall. This result is directly correlated with the increase in f-score – the increased recall offsets the decreased precision. Percentage-contained is low without masking because queries are so selective. In Figure 2, we show explicitly the impact that masking has on percentage-contained. Without masking, it drops off with query length. With masking, it is sustained, and in fact, it slightly rises with query length by the design of *min-qtf*.

The 35% increase in *MRR* is not as great as the 100% increase in percentage-contained because there are many more undesired results to contend with using masking; precision decreases by nearly 50%. Undesired results reduce the expected contribution that each result set makes to *MRR*.

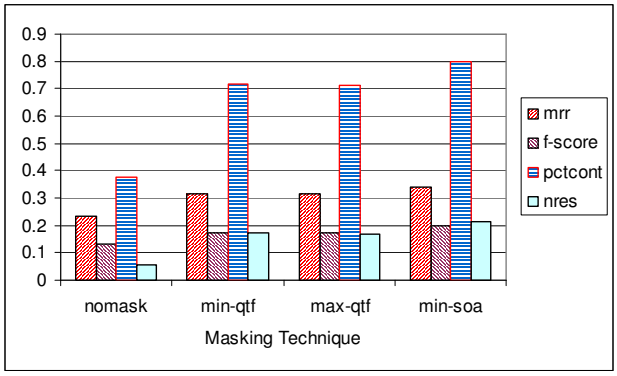

Figure 1. Performance as a function of masking technique.

Min-qtf slightly outperforms *max-qtf* because it is better able to retrieve the desired result, as indicated by the increases in f-score and percentage-contained. *Min-soa*, as expected, outperforms *min-qtf* by 8% in terms of *MRR* because the remaining query terms after *min-soa* masking are better able to match desired results.

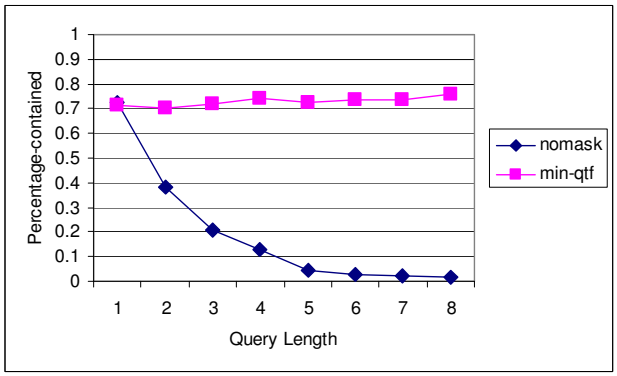

Figure 2. Percentage of result sets containing the desired result as a function of query length and masking technique.

Both *min-qtf* and *max-qtf* have similar performances because their differences are only manifested with long queries; short queries are unlikely to repeat terms. Because most queries are short (see Table 2), their performance differences are obscured. By definition, in fact, *min-qtf* and *max-qtf* are equivalent for queries containing fewer than three terms. The results in Figure 3 demostrate this phenomenon: *min-qtf*'s performance is equivalent to that of *max-qtf* for shorter queries, but superior for longer ones. Non-masking's *MRR* decreases with query size and *min-soa* performs the best, as expected.

In spite of the similarities between *min-qtf* and *max-qtf* in these experiments, we use *min-qtf* as our basic masking technique for two reasons: it performs better for longer queries, and yields a slightly higher f-score and percentage-contained. The benefit of these characteristics will become clear in later sections.

6.1. Using Local Statistics for Tie-Breaking

The query term frequency-based masking techniques described above have undefined behavior when there are no repeated terms, which is likely the case with shorter queries. We propose using a characteristic of P2P systems to address this problem: the existence of locally shared data. We assume that the data shared by a user reflects his/her interests. In the event of a tie in query term frequency-based masking, we can refer to statistics kept on locally shared data for a tie-breaker.

A term's document frequency (df) is the count of the number of documents in which it occurs. It is a common information retrieval metric indicating how strongly associated a term is to a document collection. (In our application, a document is a descriptor and a collection is the set of descriptors in a peer's local repository.) If a term t has a high df(t, R_s) over repository R_s of peer s, then t is highly relevant to s's interests.

However, descriptor frequency is a measure that is secondary in importance to query term frequency. Because queries are short and focus on particular data, all of its terms are strongly associated with the desired result. This is why we use local statistics as a tie-breaker instead of a primary masking metric. Only in the event of a tie in *min-qtf*, therefore, do we invoke descriptor frequency.
In this case, we mask the term that has a higher descriptor frequency, which we denote as *max-ldf* for maximum local descriptor frequency tie-breaking:

min-qtf/max-ldf masking: Let *t1*, *t2* be candidates for masking from query Q by peer s. If freq(t_1, Q) = freq(t_2, Q), then if df(t_1, R^s) > df(t_2, R^s), mask t_1, else mask t_2.

Figure 4 shows the performance of using tie-breaking masking techniques. For reference, minimum local descriptor frequency tie-breaking (*min-ldf*) is also included. *Max-ldf* tie-breaking increases *MRR* by about 6%. This is due to the increased selectivity of the query: recall decreases by 8%, but precision increases 6% and the number of results decreases by 15%. In this case, *max-ldf* tie-breaking increases accuracy and decreases cost over *min-qtf* alone.

Min-ldf tie-breaking has the opposite effect. It is less selective, resulting in higher recall, lower precision, and higher cost. On balance, *min-ldf* tie-breaking decreases *MRR* and f-score even though it increases percentage-contained.

The use of *max-ldf* instead of *min-ldf* as a tie-breaking technique may seem counter-intuitive. *Max-ldf* masking

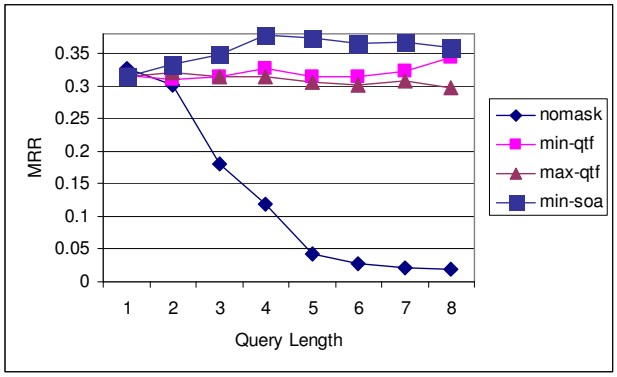

Figure 3. MRR as a function of query length and masking technique.

indicates that we remove from a query Q the terms that are more highly associated with the local repository. The justification for using *max-ldf* is that terms that are strongly associated with the local repository (those with a high descriptor frequency) are a strong indication of what the repository already contains. Leaving these terms in the query increases the likelihood that the client retrieves results it already possesses, so removing them from the query increases the likelihood of yielding new content. Relevant results are retrieved as we assume that *all* query terms are strongly associated with the desired result. *Max-ldf* merely makes the retrieved results distinct from local ones.

Consider the query "Mozart clarinet" by a user who is a fan of Mozart in a P2P file-sharing network composed of other Mozart fans. "Mozart" likely has a high local descriptor frequency, but leaving it in the query instead of "clarinet" (Mozart composed few pieces for the clarinet) will return many irrelevant results. Although the results may conform to the general interests of the user, they are the goal of the current query.

Our use of descriptor frequency to identify unique results follows standard practice in the area of information retrieval. For example, the weighting of terms by their tf-idf values (term frequency multiplied by *inverse* document frequency) [1] is based on the premise that, although a term's frequency signifies its relevance to the desired result, its degree of distribution over a document collection signifies its lack of distinguishing power. Also relevant are recent attempts to yield query result sets that represents many diverse topics in the hope that one of them is desired [29].

Because the use of *max-ldf* tie-breaking increases query accuracy and reduces cost, we use it as our tie-breaker in all subsequent results.

7. Combining Masking with Content-based Ranking

So far, we have focused on improving *MRR* by increasing the recall of long queries. Masking addresses the problems associated with query over-selectivity. In fact, masking works better with longer queries as shown in Figure 3. In this spirit, we now consider how long queries can be further exploited to improve accuracy.

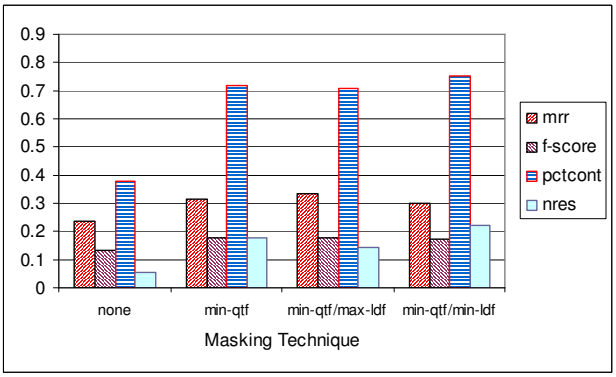

Figure 4. Performance as a function of tie-breaking masking technique.

Masking a *long* query results in a situation that is unlikely to occur without masking: the co-occurrence of a large amount of information about user interests and a large amount of data from the servers. This "high information" situation is ideal for the application of content-based ranking functions, which we use to compare the contents of queries to the contents of descriptors.

In the results presented so far, group size was used, as indicated in Section 2. Although group size considers neither the terms in the query, nor the terms in each result's descriptor, it performs surprisingly well, even compared with content-based ranking functions.

In Figure 5 (originally presented in [18]) we compare the performance of group size with that of term frequency ranking (ranking by the number of query terms that are in a result's descriptor, denoted tf) and time of arrival (denoted arrive) over various query lengths *without masking*. Term frequency and time of arrival are defined as:

$$F_{tf}(d(G), Q, |G|, time_G) = \sum_{t \in Q} freq(t, d(G))$$

$$F_{arrive}(d(G), Q, |G|, time_G) = time_G .$$

These experiments are meant to show how content-based ranking in general performs compared with non-content-based ranking. Although other content-based ranking functions are available (e.g., Jaccard's coefficient, cosine similarity [1]), we use term frequency ranking for its simplicity. Our goal is to show the applicability of content-based ranking, not to introduce new ranking functions. (The results using other content-based ranking functions are available, but the ones presented here are

representative.) Time of arrival is meant to indicate a lower bound (i.e., the non-ranking case) for ranking accuracy.

Figure 5 gives a sense of the magnitude of accuracy improvement yielded by applying ranking functions to order query results. Group size and term frequency ranking outperform order-of-arrival ranking significantly for shorter queries, but have less of an impact for longer queries.

As originally explained in [18], group size works well in conjunction with the matching criterion: only files that are strongly associated with the query terms are likely to be returned as results. The stronger the association, the greater the representation in the result set and the higher the group size score. Furthermore, group size is also a measure of a file's popularity, which many queries, by definition, are seeking. As shown in Figure 5, group size outperforms term frequency over all query lengths.

Term frequency does not work as well because of the matching criterion: all results contain all query terms, obscuring the results. Furthermore, given short queries, term frequency is highly vulnerable to skewed term distributions in the result set – single term frequencies can vary substantially from descriptor to descriptor and are related to files' replication degrees as well. Longer queries are more effective as they put additional requirements on results to achieve high ranks. Indeed, term frequency performs better with queries with two terms than with one.

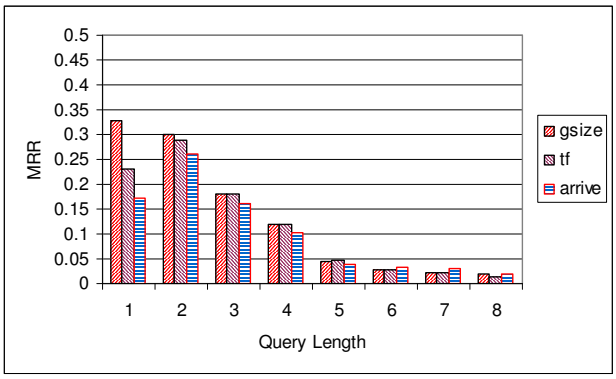

Figure 5. MRR as a function of query length and ranking function without masking.

As query lengths increase, however, all ranking functions perform poorly, at near order-of-arrival levels. This happens due to the selectivity of the longer queries, explained above. Query accuracy is dominated by the matching criterion.

With masking, however, content-based ranking should perform better. Longer queries contain more information on user interests, and the result sets of masked query results are large enough to contain:

1. At least one instance of the desired result, and
2. Information necessary to identify it.

We reiterate that group size ranking considers the content of neither the query, nor the results' descriptors, and should therefore have relatively worse performance with longer queries than does term frequency ranking.

In Figure 6, we compare the *MRR*s of group size and term frequency ranking over various query lengths. As expected with the results shown in Figure 3, group size's *MRR* is mostly unaffected by query length. Its slight trend upward is due to the effect of min-qtf/max-ldf masking given longer queries.

Term frequency ranking, on the other hand, performs poorly initially, but steadily increases in *MRR* with query length. As argued above, term frequency ranking, and all content-based ranking functions in general, work better with longer queries.

The fact that one graph is flat and the other graph trends upwards with query length indicates the importance of content-based ranking. High information states can be exploited for better search accuracy.

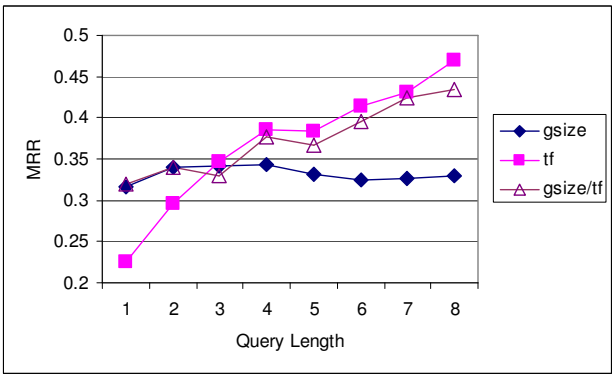

Figure 6. MRR as a function of query length and ranking function with masking.

Term frequency works so well in an environment with high information that it exceeds the performance of group size when the query length is three or greater. We utilize this bit of information by having the client dynamically switch between a low information (non-content-based) and a high information (content-based) ranking function based on query length:

If $length(Q) \geq T^Q$, then rank with F^h, else rank with F^l.

If the query length is greater than or equal to threshold T^Q=3, then rank the results with the high information ranking function, $F^h=F_{tf}$. Otherwise, rank the results with the low information ranking function, $F^l=F_G$. The performance of the dynamic ranking technique is shown in Figure 7, denoted gsize/tf. Dynamic ranking increases

overall *MRR* by approximately 3%. Overall accuracy improvement is tempered by the fact that most queries are short (i.e., of the low information variety).

8. Controlling Cost

Up to this point, our goal has been to maximize *MRR*. We have been successful, increasing it by over 40%. However, the cost of masking, as shown in Figure 4 (as well as the example in Table 1), is quite high: the number of results returned to the client increases by a factor of three. We consider two ways of reacting to the cost increase: reducing the masking degree and retrieving a random subset of results from the server.

Note, however, that the cost of masking is, in some respects, overstated. It can be argued that a user who puts in the effort to add more terms to a query is entitled to higher accuracy and the same system load as a user who issues a short query. The true benefits of accurate initial query results are not reflected in our experimental results: users who yield accurate query results likely issue fewer subsequent queries for the same file and likely download fewer files "on impulse."

Besides being more accurate, long masked queries are similar, and in some cases lower, in cost compared with short, unmasked queries. As shown in Figure 7, the cost of a masked query of length one varies depending on the masking technique. *Min-qtf* cost is greater because its goal is to maximize recall. However, by adding the selectivity-improving tie-breaker (*min-qtf/max-ldf* and *gsize/tf*), the cost per single-term query is actually lower than in the non-masking case by 10%.

8.1. Varying Masking Degree

Query cost is directly related to masking degree as shown in Figure 8. The percentage increase in cost, however, is much greater than the percentage increase in *MRR* with query length. A reasonable way to reduce cost, therefore, is to reduce masking degree.

One way of manipulating masking degree is based on network load. During the day, for instance, the network load is likely high, and during the night, it is likely low [15]. Correspondingly, the degree of masking can be low during the day and high during the evening to reduce network contention. Because of the random connectivity of P2P file-sharing overlay networks, however, neighboring peers may be in different time zones or different parts of the world. This makes time-based masking degree unreliable. Another way of measuring network load is by recording the number of incoming queries. A high volume of incoming queries indicates high load and should trigger a low masking degree. Techniques for varying masking degree to control cost are the subject of future work.

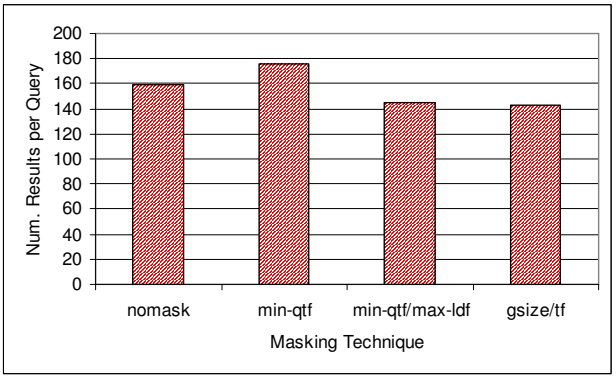

Figure 7. Number of results per query as a function of masking technique. Nomask refers to a single term base query.

8.2. Sampling Results from Servers

Another way of addressing query cost is by having the server return a random subset of matching results [23]. We propose Bernoulli sampling as a means of reducing this cost: for each query Q that arrives at server s, and for each replica $r \in R_s$,

If $Q \subseteq d(r^s)$, return $d(r)$ with probability P^m.

Sampling should reduce the size of the result set by a factor P^m, yet preserve the overall distribution of query results (e.g., precision). The question is how sampling affects accuracy.

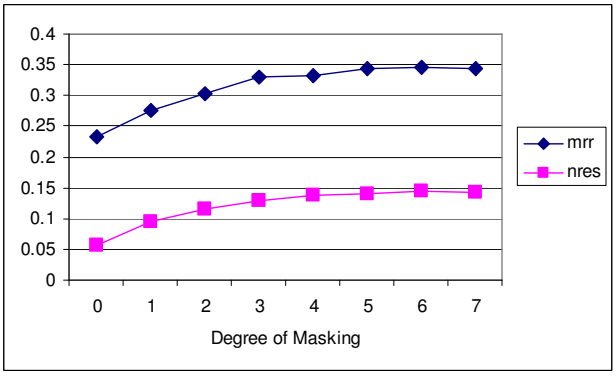

Figure 8. MRR and total number of results as a function of masking degree.

Figure 9 shows the changes in performance with decreasing sampling rates. Predictably, cost decreases by 75% with a 25% sampling rate, but *MRR* decreases by only less than 20%, which makes it still 20% more accurate than the non-masking case. Moreover, cost decreases by more than 35% compared with the non-masking case. By combining masking with result sampling, therefore, we are able to *both* increase accuracy and decrease cost.

To match the *MRR* of the non-masking case, we set the sampling rate to P^m=0.1. At this rate, the number of results decreases by 90% making its cost over 75% lower than in the non-masking case.

The reason for the lower cost is obvious; it is a direct consequence of sampling. The reason that *MRR* is preserved, however, is due to the fact that more query result sets contain more instances of the desired result when using masking as suggested in Figure 2. The sampling rate can decrease with an increase in the number of occurrences of the desired result in the average result set.

For example, assume a result set contains N_d and N_u desired and undesired results, respectively. If a fraction, $(N_d - 1)/N_d$, of them is randomly sampling away, then:

1. By expectation, the result set still contains one instance of the desired result,
2. The size of the result set is reduced by a factor $1-(N_d - 1)/N_d$.

Moreover, combining conditions 1 and 2 means that the worst-case rank of the desired result, which is N_u+1, improves because N_u decreases. This has a positive effect on the contribution this result set has on *MRR*. This benefit is particularly pronounced with longer queries as empty result sets contribute nothing to *MRR*.

For example, in our case, with masking, the average number of query results is approximately 140. We can approximate N_d as the product of average precision and

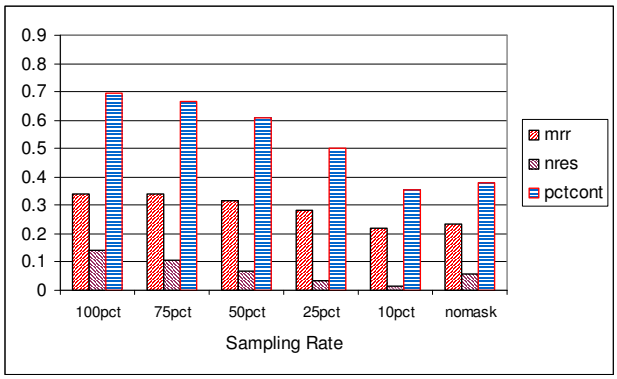

Figure 9. Performance as a function of sampling rate.

size of the result set. Because precision is 19% on average (not shown), N_d=(0.19)(140)≈27 and N_u=113 in an average result set. With a sampling rate of P^m=0.25, there should still be N_dP^m≈7 instances of the desired result in the sampled result set even though 75% of the results have been removed. The minimum contribution of this result set to *MRR* increases from 1/114 (N_u=113 plus one unique desired result) to 1/29 (N_u=28 with sampling plus one unique desired result). In theory, P^m could as low as $1-(N_d - 1)/N_d$≈0.04 and still contain an instance of the desired result. However, due to expected variances in result set sizes, such a P^m is not advisable.

9. Related Work

Much of today's work in P2P information retrieval (IR) research focuses on identifying highly reliable peers and giving them specialized roles in statistics maintenance, indexing, and routing [3][17][24][28]. The performances of such systems are impressive; however, the application domain is different than the one we consider. We make no assumptions about the relative capabilities of the peers. Specifically, other solutions are architectural in nature, designating some peers are designated for specialized roles. Our work, in contrast, makes no distinction among the functionality of the peers. One benefit of our model is that our work is more applicable to ad-hoc environments. Secondly, many of these works perform retrieval on text documents. We assume much sparser descriptors.

Our work also bears many similarities to that of meta-search engines [5]. The problems related to such systems include source selection, merging of results from independent sources, and query dispatching (the process of translating a query for each server's particular interface). Solutions include source sampling to determine content and the use of ontologies for result ranking [2]. Meta-search engines, however, operate in a highly stable and centralized environment that is not typical of P2P file-sharing systems.

Some P2P systems use past results to bias peer behavior in the network. Positive feedback from a peer increases the priority of its future results [15]. These systems require that peers maintain statistics on neighbors, which may not be scalable in a large system and impractical, given system unreliability.

Some systems employ distributed hash tables, or more recently, trees, to reduce search cost in distributed environments [16][27]. Because these search methods are based on exact key matching, multi-term queries are difficult to implement (e.g., semi-join-like techniques over multiple inverted lists have been proposed [10]). Masking facilitates the use of the proposed search structures because it can reduce a query to a single term, making key indices immediately applicable. In a similar way, masking can also relieve the "word-mismatch" problem in information retrieval – when two people independently use different word sets to describe the same data [26].

One alternative to masking is to use of query expansion to improve query performance in P2P file-sharing systems [6]. This work attempts to build a distributed semantic network of terms, revealing generalizations and synonyms, which can be used to increase the recall of a query. Besides the difficulty of maintaining the semantic network, a problem of this technique is that it may lead to a

"drift" in the original semantics of the query. As we know of no experimental validation of this technique, it difficult to accurately gauge its performance.

Finally, masking is similar to the practice of reconciling "failed" database queries – queries that return empty results [8]. Reconciling failed queries requires changing of eliminating selection conditions, changing its semantics. Masking, in contrast, aims to retrieve the originally desired result.

10. Conclusions and Future Work

Masking works by increasing the recall of queries. By keeping only the most relevant terms in the masked query, accuracy is improved by over 40%.

Masking cost is manageable by tuning the masking degree or randomly sampling the result sets returned by servers. Ideally, $100N_d/(N_d\text{-}1)\%$ of the results can be sampled away. By sampling, we were able to reduce cost by over 35% compared with the non-masking case, while preserving an improvement in accuracy of 20%.

The masking techniques presented here are but a first step of a general process and are subject to many optimizations. One area of optimization we are working on is server-side masking. As servers contain a different view of the system, it is possible that they can give better hints on improving masking performance in terms of cost and benefit. Second, the use of server-side masking could avoid the conservative technique of maximizing recall by masking out all but one term at the client. Third, server-side masking could make time-zone based masking degree (see Section 8.1) simpler to implement. Another area we are considering is the collection of statistics in terms of network load and replication degrees [9] to determine a good dynamic ranking threshold value and sampling rate.

References

[1] D. Grossman and O. Frieder. *Information Retrieval: Algorithms and Heuristics*. Springer, 2nd ed., 2004.

[2] P. G. Ipeirotis and L. Gravano. Distributed search over the hidden web: Hierarchical database sampling and selection. In *Proc. VLDB*, pages 394—405, 2002.

[3] B. T. Loo, J. M. Hellerstein, R. Huebsch, S. Shenker, and I. Stoica. Enhancing p2p file-sharing with an internet-scale query processor. In *Proc. VLDB*, Toronto, 2004.

[4] T. Neumann, M. Bender, S. Michel, G. Weikum. A Reproducible Benchmark for P2P Retrieval. In *Proc. ACM Wkshp. Exp. DB*. 2006.

[5] W. Meng, C. Yu, and K.-L. Liu. Building efficient and effective metasearch engines. *ACM Comp. Surveys*, 34(1):48—84, Mar. 2002.

[6] K. Nakauchi, Y. Ishikawa, H. Morikawa, and T. Aoyama. Peer-to-peer keyword search using keyword relationship. In *Proc. Wkshp. Global and Peer-to-Peer Comp. Large Scale Dist. Sys* (GP2PC), 2003.

[7] M. Nilsson. Id3v2 web site. www.id3.org. 2006.

[8] I. Muslea and T. J. Lee. Online Query Relaxation via Bayesian Causal Structures Discovery. In *Proc. AAAI*, 2005.

[9] N. Ntarmos, P. Triantafillou, G. Weikum. Counting at Large: Efficient Cardinality Estimation in Internet Scale Data Networks. In *Proc. IEEE ICDE*, 2006.

[10] P. Reynolds and A. Vahdat. Efficient peer-to-peer keyword searching. In *Proc. ACM Middleware*, 2003.

[11] C. Rohrs. Keyword matching [in gnutella]. Technical report, LimeWire, Dec. 2000. www.limewire.org/techdocs/KeywordMatching.htm.

[12] C. Rohrs. Search result grouping [in gnutella]. Technical report, LimeWire, Aug. 2001. www.limewire.org/project/www/result_grouping.htm.

[13] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proc. Multimed Comp. and Netw.* (MMCN), 2002.

[14] M. T. Schlosser, T. E. Condie, and S. D. Kamvar. Simulating a file-sharing p2p network. In *Proc. Wkshp. Semantics in Peer-to-Peer and Grid Comp*., 2003.

[15] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *Proc. IEEE INFOCOM*, 2003.

[16] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. ACM SIGCOMM*, 2001.

[17] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *Proc. ACM SIGCOMM*, Aug. 2003.

[18] W.G. Yee and O.Frieder. On search in peer-to-peer file sharing systems. In *Proc. ACM SAC*, 2005.

[19] S. Ciraci, I. Korpeoglu, O. Ulusoy. Characterizing Gnutella Network Properties for Peer-to-Peer Network Simulation. In *Proc. ISCIS*, 2005.

[20] IIT P2P Information Retrieval Research Group Web Site. www.ir.iit.edu/~waigen/proj/pirs.

[21] Slyck.com P2P File-sharing Statistics. http://slyck.com/stats.php.

[22] H. Nottelmann, K. Aberer, J. Callan, and W. Nejdl, *CIKM 2005 P2PIR Workshop Report*, 2005.

[23] W. G. Yee, L. T. Nguyen, O. Frieder. Conjunction Dysfunction: The Weakness of Conjunctive Queries in Peer-to-Peer File-Sharing Systems. In *Proc IEEE P2P Conf*, 2006.

[24] J. Lu and J. Callan. User modeling for full-text federated search in peer-to-peer networks. In *Proc. ACM SIGIR*, 2006.

[25] S. Sharma, L. T. Nguyen, D. Jia. IR-Wire: A Research Tool for P2P Information Retrieval. In *Proc. ACM Wkshp. Open Source Inf. Retr.*, 2006.

[26] J. Xu and W. B. Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM Trans. Info. Sys.* 18(1), Jan., 2000.

[27] H.V. Jagadish, B. C. Ooi, K.-L. Tan, Q. H. Vu, R. Zhang. Speeding up search in peer-to-peer networks with a multi-way tree structure. In *Proc. ACM SIGMOD*, 2006.

[28] W.-T. Balke, W. Nejdl, W. Siberski, U. Thaden. Progressive Distributed Top k Retrieval in Peer-to-Peer Networks. In *Proc. ICDE*, 2005.

[29] H. Chen and D. R. Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In *Proc. SIGIR*, 2006.