Conjunction Dysfunction: The Weakness of Conjunctive Queries in Peer-to-Peer File-sharing Systems

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

Abstract

Peers in peer-to-peer file-sharing systems use conjunctive queries as a way of controlling query cost in the absence of information about the behavior of other peers. Conjunctive queries are good at increasing the precision of query result sets, but may be overly selective, decreasing overall performance. We consider relaxing the conjunctive matching criterion and its impact on performance and cost. Experimental results indicate that significant performance improvements are possible at reasonable cost.

1. Introduction

Peer-to-peer (P2P) file-sharing systems are a leading application of P2P technology. In such systems, files are binary and must be described independently by users (publishers) who share them. To find a file, a user composes a query made up of relevant terms. If these terms "match" those used by a publisher to describe the file, then the file's descriptor and the publicsher's identity (e.g., a filename and IP address, respecttively) are returned to the client. This information is used by the client to identify and download the file.

Most P2P file-sharing systems support only conjunctive queries [1] – all query terms must appear in the file's descriptor for a match to occur. One reason for using conjunctive matching is that its performance is intuitive to a user: as the number of terms in a query increases, the result set should be more selective and more precisely fit the desires of the user.

The conjunctive nature of queries helps conserve bandwidth as well. Because peers respond to queries independently, it is reasonable for them, in the interest of efficiency, to be conservative, and return the smallest result set. Besides taking a load off the network, the user at the client has fewer results through which to search.

However, the conjunctive matching criterion has a disproportionately negative impact on the ability to find desired results. As queries grow, they become increasingly selective. In the worst case, all relevant results may fail to match long queries. This is particularly problematic in P2P file-sharing systems because descriptors are limited in length (a filename is typically limited to about 200 bytes).

We propose relaxing the matching criterion, thereby returning more results to the client as a result of the query. In particular, we propose using disjunctive queries instead of conjunctive ones: a query matches a descriptor if any query term is contained in the descriptor. We hypothesize that this will have several positive effects, two of which are:

- Increasing the likelihood that the desired file is returned as a result of a query. As mentioned above, a longer, overly selective query reduces the likelihood that any result will be contained in the result set.
- Increasing the amount of metadata available for identifying the desired result. Each descriptor (contained in a result) contributes to describing a file. Increasing the number of contributing descriptors makes the desired file more identifiable.

We claim that the improvement in query accuracy the – ability to identify the desired result - afforded by disjunctive matching outweighs its negative impact on efficiency. In addition, simple techniques can be applied to improve efficiency.

In this work, we assume that a user is searching for a particular file in a zero-knowledge P2P file-sharing environment. That is, the network and the set of shared data are dynamic as peers autonomously join and leave the system, and there are no centralized indices storing global statistics and no central controller using them.

This is a conservative assumption, but simplifies our model and is inclusive of most applications, such as the current generation of P2P file-sharing systems as well as mobile P2P applications. Finally, the user poses queries for the desired file using a set of terms s/he believes best describes it.

2. Related Work

Much of the work done in P2P file-sharing systems is in the design of structured network topologies based on distributed hash tables (DHTs), which maximize routing performance [8][9]. Search in basic DHTs is exact keyword matching, and, to support multi-keyword queries, multiple inverted lists must be maintained and their intersections must be computed [4]. Other techniques to reduce the cost of queries include the use of Bloom filters, the maintenance of additional statistics and inverted indices [15], and the use of composite instead of single keywords [20]. Search in structured P2P file-sharing systems is guaranteed to find the desired file if it exists. However, structured P2P networks are known to suffer when the environment is highly dynamic, characterized by transient data and peers.

Unstructured P2P systems, such as Gnutella, control neither the overlay topology, nor the placement of files. Peers are free to choose their neighbors, and files are freely replicated over the network [21]. Most research on unstructured P2P networks focuses on guiding the query routing process, by using statistical information to select the right peers to which to forward the query: routing indices are used in [10][16]; queries are forwarded only to a small number of powerful peers in [14] or to peers that have the similar interests [11]; content signatures are used to rank neighboring peers in [12][19]. Techniques to build semantic overlay networks have also developed in which peers find and make connections with others that have similar content [19]. Unstructured P2P systems are robust and scalable due to their simplicity. However, search in unstructured P2P file-sharing systems is not guaranteed to find the desired file, regardless of whether or not it exists. To cope with this, [18] proposes to use a quorum system to guarantee with high probability that the desired file can be found.

The problem of conjunctive queries filtering out many relevant results is known as the *word mismatch* problem in the area of information retrieval; the searchers and the content providers use different words to describe the same content. This problem can be addressed by the semantic matching approach. To our knowledge, pSearch [15] is the first work on semantic searching in P2P networks. pSearch uses latent semantic indexing (LSI) as the indexing technique, and a DHT as the P2P routing infrastructure. The scalability of LSI, however, is questionable. In another work [13], the authors address the problem by expanding a query based on keyword relationships, which are discovered by a relevance-feedback-liked mechanism. This method requires maintaining a large base of statistics.

Our work is distinguished from the work above in the minimal amount of information it requires from the infrastructure and its focus on the application-level behavior of peers that share binary (i.e., non-self-describing) files.

3. Model

Peers collectively share (or publish) a set of (binary) files by maintaining local replicas of them. Each replica is represented by a descriptor, which also contains an identifying key (e.g., an MD5 cryptographic hash on file's bits). All replicas of the same file share the same key. A query issued by a client is routed to all reachable peers until the query's time-to-live expires. The query matches a replica if it is fully contained in the replica's descriptor. For each match, the server returns its system identifier and the matching replica's descriptor. The client uses this information to subsequently download the actual file.

Formally, let O be the set of files, M be the set of terms, and P be the set of peers. Each file $o \in O$ has a unique key, denoted k_o, such that $k_o = k_p$ if and only if o=p (i.e., the MD5 hash value mentioned above). Each file $o \in O$ has a set of terms that *validly* describe it, denoted as T_o, where $T_o \subseteq M$. Intuitively, T_o is the set of all terms that a 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 $\sum_{t \in T_o} soa(t, o) = 1$. 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 $s \in P$ is defined as a pair, (R_s, g^s), where R_s is the peer's set of shared replicas and g^s is the peer's unique system identifier. Each replica $r^o_s \in R_s$ is a copy of file o, maintained by peer s. Each r^o_s has an associated descriptor, $d(r^o_s) \subseteq M$, which is a multiset of terms that is maintained independently by s. Each descriptor $d(r^o_s)$ also contains k_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 $Q \neq \emptyset$ arrives at a server s,

the server returns *result set* $U^Q_s=\{(d(r^o_s), g^s) \mid r^o_s \in R_s, Q \subseteq d(r^o_s)\}$. In other words, in accordance with the (conjunctive) matching criterion, a result's descriptor must contain all query terms.

The client receives result set $U^Q=\cup_s U^Q_s$, $s \in P$, and groups individual results by key, forming $G=\{G_{O1}, G_{O2}, \ldots\}$, where $G_{Oi}=(d_i, k_{Oi}, l_i)$, $d_i=\{\oplus d(r^{Oi}_s) \mid (d(r^{Oi}_s), g^s) \in U^Q\}$ is the group's descriptor, k_{Oi} is the key of O_i, and $l_i=\{g^s \mid (d(r^{Oi}_s), g^s) \in U^Q\}$ is the list of servers that returned the results in G_{Oi}. In this definition, $\oplus$ is the multiset sum operation.

The client assigns a rank score to each group with function $F_i \in F$, defined as $F: 2^M \times 2^M \times \mathcal{Z} \times \mathcal{Z} \rightarrow \mathcal{R}^+$. If $F_i(d_j, Q, |G_j|, time_j) > F_i(d_k, Q, |G_k|, time_k)$, where G_j, G_k are groups, then we say that G_j is ranked higher than G_k with respect to query Q. In these definitions of F, $|G_j|$ is the number of results contained in a group, and $time_j$ is the creation time of the G_j (i.e., the time when the first result in G_j arrived).

3.1. Model Specifics

In popular P2P file-sharing systems, such as various versions of Gnutella and eDonkey, result keys are generally generated by the MD5 cryptographic hash function and results are grouped based on these keys. Ranking is based on group size:

$$F_G(d, Q, s, t) = s.$$

Descriptors in these systems are generally implemented via filenames, although some descriptive information may be embedded in the binaries (e.g., ID3 data embedded in MP3 files [5]). When a client downloads a file, the descriptor of this new replica is initialized as a duplicate of one of the servers' in the result set. We also assume such behavior in our model.

For simplicity, we will use the term *result* to describe a group, an individual result, or a result's descriptor; and clarify the usage if necessary.

4. An Alternative to Conjunctive Queries

Conjunctive queries are problematic because they may be excessive in shrinking the result set, perhaps selecting away desired results. We now sketch the intuition behind this behavior. Based on our model, given a query Q^o with term, t, the probability that a descriptor $d(r^p)$ contains t is

$$1-[1-soa(t, p)]^{\|d(rp)\|},$$

where $\|d(r^p)\|$ is the number of (not necessarily unique) terms in $d(r^p)$. Let Q^o' be the set of unique terms of Q^o. The probability that $d(r^p)$ contains Q^o is

$$\prod_{ti \in Qo'}[1-[1-soa(t_i, p)]^{\|d(rp)\|-i}], \|d(r^p)\| \geq |Q^o'|.$$

As Q^o grows, the probability that it is contained by $d(r^p)$ decreases exponentially. The conjunctive matching criterion has a tendency of filter out results that are not relevant (i.e., have different natural term distributions). Such results likely do not contain all the terms that are the most strongly associated with file o, and therefore do not contain Q^o. However, the conjunctive matching criterion may also filter out desired results because their descriptors are too small or happen, by chance, to not contain all of Q^o.

To alleviate this problem, we propose relaxing the conjunctive matching criterion. There are many relaxation alternatives, so we do not attempt to enumerate all of them. Rather, our goal is to show that relaxing the conjunctive criterion is viable. We therefore propose the following alternative because of its simplicity:

$$\text{If } Q^o \cap d(r^p) \neq \emptyset, \text{ then } Q^o \text{ matches } d(r^p).$$

This disjunctive matching criterion has the following probability of matching Q^o' with $d(r^p)$:

$$\sum_{t \in Qo'} Pr(t \in d(r^p)) - \sum_{t1,t2 \in Qo'} Pr(t1, t2 \in d(r^p)) + \ldots$$

The expression above is a result of the *inclusion-exclusion* principle of probability theory [6]. In this expression, Pr(event) signifies the probability of an event. The point we are making is that additional terms in the query do not negatively impact its ability to yield an inclusive result set.

5. 5. 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 model described in [2] and observations presented in [3], we enhance our experimental model with interest categories, which model the fact that some users have stronger interests in some well-known subsets of data than other.

We partition the set of files, O, into sets C_i, where $C_i \subseteq O$, $C_i \cap C_j = \emptyset$ if $i \neq j$, and $\cup_i C_i = O$. Each category C_i has an assigned popularity, b_i, which describes how likely it is to be assigned to a peer. The values of b_i follow the Zipf distribution [2]. Within each interest category, each file varies in popularity, which is also skewed according to the Zipf distribution [2]. This popularity governs the likelihood that a peer who has its interest category is either initialized with a replica of the file or decides to search for it.

At initialization, each peer $s \in P$ is assigned some interests $I_s \subseteq C$, and is allocated a set of replicas R_s from

this interest set: $R_s=\{r^o_s \mid o \in \cup_i C_i$, where $C_i \in I_s\}$. For each replica, r^i_s, allocated at initialization, $d(r^i_s) \subseteq T_i$, where term allocation is governed by natural term distributions. Peer s's interest categories also constrain its searches; it only searches for files from $\cup_i C_i$, where $C_i \in I_s$.

Table 1. Query length distribution

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

Table 2. 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. files per peer at initialization	10-30
Num. trials per experiment	10

We use Web data to simulate 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 Web domains to simulate interest categories). Other researchers have also used Web data for P2P experimentation [23]. Real data from P2P applications would be preferable, but we know of none [22]. We are currently investigating the creation of such data.

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 800,000 terms, 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 we used for all experiments can be found on our Web site [24]. The other experimental results are available on request.

Queries for files are generated using associated terms with a length distribution that is typical of that found in Web search engines [4] as shown in Table 1. Other simulation parameters shown in Table 2 are based on observations of real-world P2P file-sharing systems and are comparable to the parameters used in the literature.

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

Performance is measured using a standard metric called 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 file in query i's result set. If the file is not in the result set, then ranki=∞. MRR is an appropriate metric in applications where the user is looking for a single, particular result.

For reference, we also present precision and recall, which have slightly different definitions than they do in traditional IR, due to the fact that replicas exist in the P2P file-sharing environment, and assuming that queries are for particular files. Let A be the set of existing replicas of the desired file in the entire system, and R be the result set of the query. Precision and recall are defined as follows:

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

Precision measures the percentage of a result set that is relevant to the query and recall measures the percentage of the existing results retrieved by a query. When computing average precision, we only consider the cases when result sets are non-empty. Reported precisions and recalls are the averages of these measures over all queries in a trial.

Because recall and precision are commonly inversely related in information retrieval research, the F-score metric has been devised to combine their relative contributions in a single metric [7]:

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

These more traditional IR metrics are useful in roughly diagnosing the performance of query processing and in generalizing the presented performance to other domains.

Note that the results may exhibit some variance due to the experimental nature of the results. We have computed statistical significances of the results, but have left them out for brevity. The statistical significances of the presented results should be obvious, however.

Finally, note that we present some results over various query lengths. These results were yielded by keeping track of the length of each of the 10,000 experimental queries. We are not reporting results where query length was the independent variable.

5.1. Disjunctive Query Performance

In Figure 1, we compare the performances of queries using both conjunctive and disjunctive matching. Disjunctive matching outperforms conjunctive matching by more than 50%.

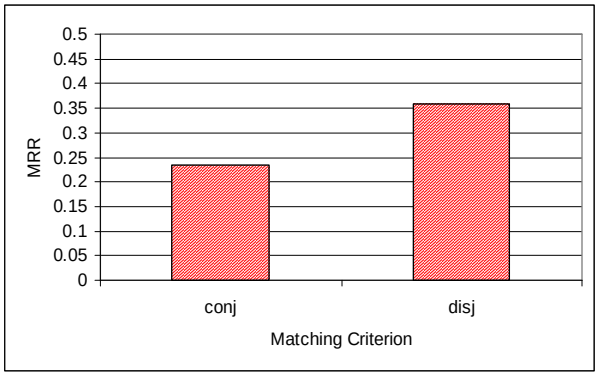

Figure 1. MRR of two matching criteria

As conjunctive queries get longer, they become more selective. Longer queries will likely return only correct results, if they return anything at all. As shown in Figure 2, the precision of conjunctive queries is about 90% higher than that of disjunctive queries. However, the recall of conjunctive queries is about 50% lower than that of disjunctive queries. That the F-score of disjunctive queries is about 30% greater than that of conjunctive queries suggests a net gain in the use of disjunctive queries.

Precision is an important metric, because, the likelihood of identifying a desired result increases as the percentage of that result set made of the desired result increases. The caveat of this claim, however, is that *the result set be non-empty*.

Conjunctive queries, however, prioritize precision over recall to the point that near-matches involving desired results are rejected. In the worst case, no desired results are returned at all. As shown in Figure 3, as queries get longer, the percentage of result sets containing the desired result decreases from 70% when queries contain only one term to less than 2% when queries contain 8 terms.

In contrast, with disjunctive queries, the percentage of result sets that contain the desired result is consistently about 70%. These figures are significant because only if the desired result is in the result set can it be identified by the ranking function and user. Intuitively, the *percentage-contained* metric acts as an upper bound to our main metric, MRR.

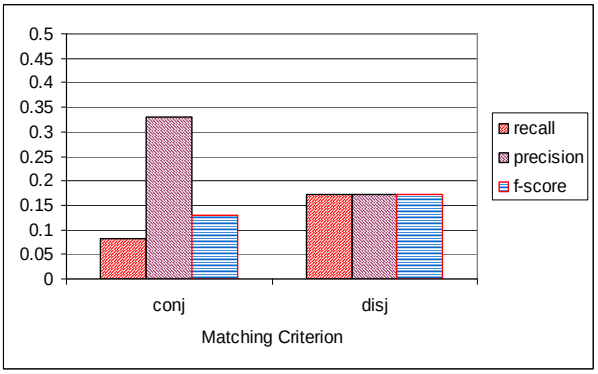

Figure 2. Recall, precision, and F-score of two matching criteria

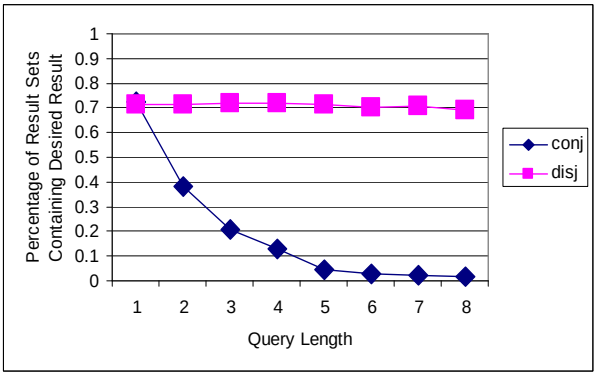

Figure 3. Percentage of result sets containing the desired result of two matching criteria over various query lengths

The relationship between MRR and percentage-contained is clear from the MRR results shown in Figure 4. As query length increases, the MRR of disjunctive queries persists at approximately 35%, whereas with conjunctive queries, there is a significant drop-off. We attribute the bumpiness exhibited in these graphs to random variance.

We tried the same experiments on different data sets with different data distributions, and, predictably, they yielded similar results: disjunctive queries out-performed conjunctive ones. In Figure 5, we show the performance improvement using disjunctive queries over data sets of different sizes (400K and 1.2M terms), different data sets with 800K terms (sets 2 and 3), and when using a uniform distribution of data popularity. These results indicate that our reported results are representative, so we will no longer consider other data sets in this work.

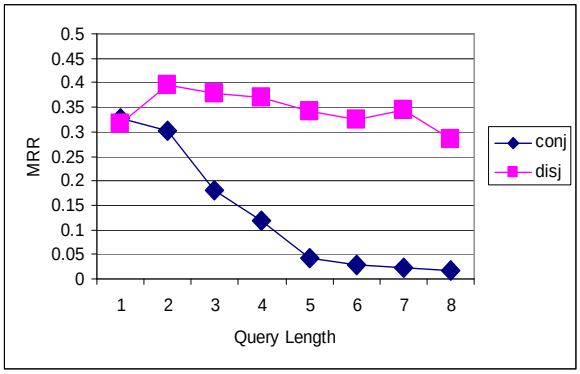

Figure 4. MRR of two matching criteria over various query lengths

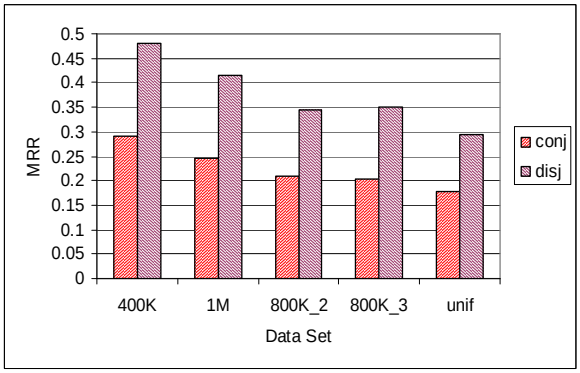

Figure 5. MRR of two matching criteria with different data sets and distributions

5.2. Cost Analysis

Using disjunctive queries increases MRR, but they come at a price: more results are returned putting more load on the network, the server, and the client. One might argue that the price of the increased MRR is worth it, as better-ranked results increase the likelihood that the desired result is found and decreases the result that:

1. Subsequent queries are issued to find the desired file, and
2. The user downloads an irrelevant file, which may be followed by another search for the original file.

Despite the hypothetical cost savings, the overhead of a disjunctive query should be addressed. The number of results returned by the query is directly related to the work peers have to perform to serve, transmit, and rank them, so we use it as our cost metric.

We reduce the cost of disjunctive queries by performing Bernoulli sampling on the result set generated by each server. The disjunctive matching criterion with sampling becomes:

If $Q^o \cap d(r^p) \neq \emptyset \rightarrow Q^o$ matches $d(r^p)$ with probability P^m,

where P^m is a user-tuned parameter. By sampling in this way, we yield an unbiased sample of the original result. The size of the result set that arrives at the client should be reduced by a factor of P^m.

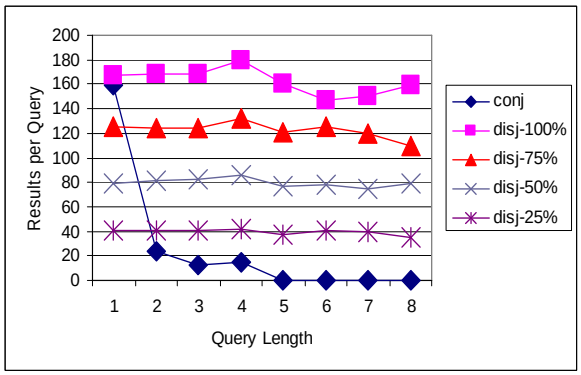

Figure 6. Number of results per query over various sampling rates and query lengths

Figure 6 shows the number of results per query with various sampling rates and query lengths. With conjunctive queries, we see that the number of results decreases as query length increases. With disjunctive queries, however, the number of results per query is constant with query length, but decreases predictably with sampling rate.

Figure 7 shows the average number of results per query with various sampling rates and random query lengths. Note that when the sampling rate of disjunctive queries is 25%, cost is about 28% lower than when using conjunctive queries without sampling. Sampling, predictably, has a negative impact on MRR. As argued above, the reason that disjunctive queries improve performance is because they result in increased recall and the percentage of query result sets that contain the desired result. Sampling counteracts these effects.

Figure 8 shows the impact that sampling has on the percentage of result sets that contain the desired result. Although percentage-contained of disjunctive queries are still, for the most part, higher than with conjunctive queries, the decrease in this metric is not in proportion to the decrease in cost. For example, with 75% sampling, the drop in percentage-contained is less than the expected 25%. This is due to the precision of the average result set. In our experiments, when using disjunctive queries, precision is 17%. This means that, on average, as long as there are $0.17^{-1} \approx 6$ results in the result set, then one of them is expected to be the

desired result, assuming that results are uniformly distributed.

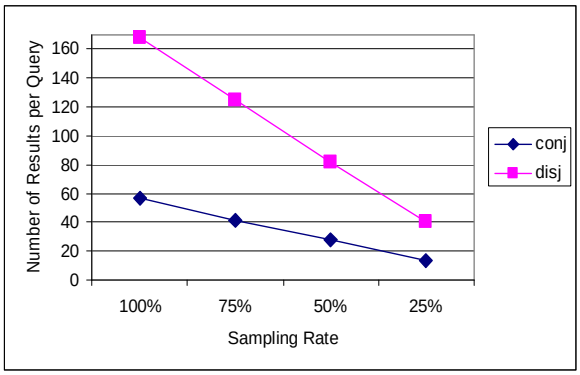

Figure 7. Average number of results per query with different sampling rates

In other words, the probability that a randomly selected result in a result set, R, corresponds to the desired file, o, is the precision of R, denoted precision(R, o). The likelihood that there are no instances of the desired result in R after sampling is

$$\Pr(\text{desired result} \notin R) = (1\text{-}precision(R, o))^{|R|},$$

where |R| denotes the number of results in R. Thus, removing one random result from R increases the likelihood that the desired result is not contained in R by a factor of

$$\frac{1}{1 - precision(R, o)}.$$

As the size of a result set shrinks, the likelihood that it contains the desired result decreases exponentially, albeit slowly. In particular, if |R| is shrunk by a factor of $0 \leq P^m \leq 1$, then the likelihood that there are no instances of the desired result in R increases by a factor of $(1\text{-precision}(R,o))^{Pm||R||-||R||} = (1\text{-precision}(R,o))^{||R||(Pm-1)|}$. This explains the percentage-contained trend shown in Figure 8 with different sampling rates.

Because the number of results per disjunctive query ranges from 40 to 160, theoretically, with a 17% precision, we can sample out from 40-6=34 to 160-6 =154 results (i.e., from 85% to 96%) from a result set and still expect it to contain the desired result. Specifically, we should be able to decrease cost more quickly than we decrease search accuracy.

Figure 9 compares the MRR and percentage-contained of conjunctive and disjunctive queries over various sampling rates. As predicted by the results shown in Figure 8, MRR decreases with sampling rate. Nevertheless, with *all* sampling rates, disjunctive queries outperform conjunctive queries without sampling, by at least 15%. Considering these results and the cost results shown in Figure 7, we see that it is possible to improve on *both* the cost and the accuracy of conjunctive queries (i.e., using disjunctive queries with a sampling rate of 25%).

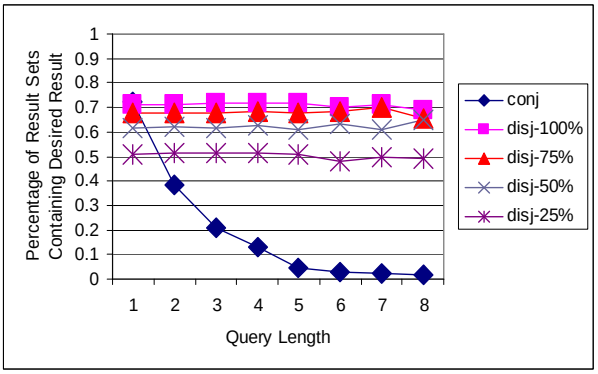

Figure 8. Percentage of result sets containing the desired result over various query lengths with various sampling rates

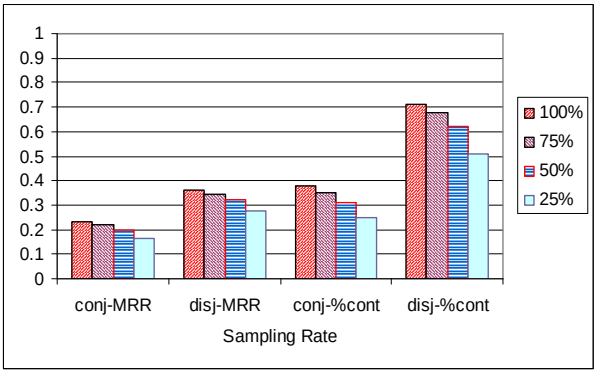

Figure 9. MRR and percentage of results containing the desired result over various sampling rates

The benefit of disjunctive matching, as stated in Section 4, is that it reduces the selectivity of queries, *increasing* the size of the result set. At the same time, sampling acts to *decrease* the size of the result set. The overall effect of disjunctive matching and sampling, however, is higher MRR. The reason for this is the relaxation of the conjunctive matching criterion has a tendency of selecting *for* relevant results over irrelevant ones, whereas sampling is indiscriminate in what it removes. All the terms in the query are expected to be the most strongly associated with the desired result. Allowing any of them to be the basis for a match is therefore expected to yield a result set that contains a large proportion of desired result – this is suggested by the F-score results shown in Figure 2.

5.3. Other Matching Criteria

Besides the disjunctive matching criterion presented above, we also tried various other matching criteria. For example, one criterion computed the *cosine similarity* [7] between the query and the descriptor and used it as a probability of returning the result. These alternative criteria have slightly different performance-cost profiles, but all essentially exhibited the similar performance-cost tradeoff.

6. Conclusion

P2P file-sharing systems are designed to handle queries conjunctively, ostensibly to be more efficient with network resources. This form of query matching tends to increase the proportion of desired results in query results sets. However, this criterion may be too strict; in some cases, all desired results are selected away. By using disjunctive matching, we could increase the likelihood that the desired result is contained in the result set, increasing search accuracy (MRR) by over 50%, but at triple the cost.

Because the disjunctive selection criterion biases the additional results to the desired one, random sampling is able to reduce cost with a lesser impact on MRR. In fact, at a 25% sampling rate, MRR when using disjunctive queries is 15% better than with conjunctive queries without sampling, and cost is 28% lower.

One may use this performance characteristic to tune the sampling rate based on current network traffic: when bandwidth is plentiful, maximize MRR, otherwise, minimize cost. A particular heuristic, however, is outside of the scope of this work.

Our work in this area is ongoing. We are currently exploring other ways of relaxing the matching criterion and how other ranking functions may be used to further refine MRR.

7. References

[1] C. Rohrs, "Keyword Matching [in Gnutella]", LimeWire Technical Document, Dec., 2000, www.limewire.org/techdocs/KeywordMatching.htm.

[2] M. T. Schlosser, T. E. Condie and S. D. Kamvar, "Simulating a File-Sharing P2P Network", *Proc. Wkshp. Semantics in Peer-to-Peer and Grid Comp*., May, 2003.

[3] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems", *Proc. Multimedia Computing and Networking* (MMCN), Jan., 2002.

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

[5] M. Nilsson, ID3v2 Web Site, Web Document, 2006, www.id3.org

[6] S. M. Ross, *Introduction to Probability Models*, 6th ed., Academic Press, New York, 1997.

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

[8] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network", *Proc. ACM SIGCOMM*, 2001.

[9] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", *Proc. ACM SIGCOMM*, 2001.

[10] A. Singla and C. Rohrs, "Ultrapeers: Another Step Towards Gnutella Scalability", Whitepaper, Dec 2001.

[11] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems", *Proc. IEEE INFOCOM*, 2003.

[12] F. M. Cuenca-Acuna and T. D. Nguyen, "Text-based content search and retrieval in ad hoc p2p communities", *Proc. Wkshp P2P Comp.*, May 2002.

[13] K. Nakauchi, Y. Ishikawa, H. Morikawa and T. Aoyama, "Peer to Peer Keyword Search Using Keyword Relationship", *Proc. of IEEE/ACM Intl. Symposium on Cluster Computing and the Grid (CCGRID)*, 2003

[14] Y. Shao and R. Wang, "Buddynet:History-Based Peer-to-Peer Search", *ECIR-05*.

[15] C. Tang, Z. Xu, and S. Dwarkadas, "Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks", *Proc. of ACM SIGCOMM'03*.

[16] A. Crespo and H. Garcia-Monila, "Routing Indices for Peer-to-Peer Systems", *ICDCS'02*.

[17] J. Lu and J. Callan, "Content-based retrieval in hybrid peer-to-peer networks", *Proc. of the 12th Intl. Conf. on Information and Knowledge Management*

[18] R. Ferreira et. al, "Search with Probabilistic Guarantees in Unstructured Peer-to-peer Networks", *Proc. of the Fifth IEEE Intl. Conf. on Peer-to-Peer Computing, 2005*.

[19] Y. Zhu, X. Yang, and Y. Hu, "Making Search Efficient on Gnutella-like P2P Systems", *Proc. of the 19th IEEE Intl. Parallel and Distributed Processing Symposium*, 2005.

[20] O. Gnawali, "A Keyword-set Search System for Peer-to-Peer Networks", *MS Thesis, MIT, 2002.*

[21] T. Klingberg and R. Manfredi, Gnutella Protocol 0.6, Web Document, 2002, rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html.

[22] H. Nottelmann, K. Aberer, J. Callan, W. Nejdl, CIKM 2005 P2PIR Workshop Report, 2005, http://p2pir.is.informatik.uni-duisburg.de/2005/report.pdf.

[23] J. Lu and J. Callan, "Federated Search of Text-Based Digital Libraries in Hierarchical Peer-to-Peer Networks", *Proc. of the Wrkshp on Peer-to-Peer Information Retrieval*, 2004.

[24] PIRS: A Peer-to-Peer Information Retrieval System Project Home Page, http://ir.iit.edu/~waigen/proj/pirs