Alternatives to Conjunctive Query Processing in Peer-to-Peer File-sharing Systems

Wai Gen Yee, Linh Thai Nguyen and Ophir Frieder
Illinois Institute of Technology
10 W 31st Street
Chicago, IL 60616 USA
{waigen, linhnt, ophir}@ir.iit.edu

ABSTRACT

Peer-to-peer file-sharing systems suffer from the over-specification of query results due to the fact that query processing is conjunctive and the descriptions of shared files are sparse. Ultimately, longer queries, which should yield more accurate results, do the opposite. To alleviate this problem, we consider alternative means of query processing. That is, results are sent from the server to the client only if they are deemed relevant based on cosine similarity. Based on our results, these alternatives can increase query accuracy by 40% at virtually no cost.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *search process.*

General Terms

Performance, Design, Experimentation.

Keywords

Query processing, peer-to-peer, search.

1. INTRODUCTION

Peer-to-peer (P2P) file-sharing is a leading search application where millions of users share petabytes of data [6]. Due to this application's scale, it is vital that queries retrieve relevant results.

Two characteristics of P2P file-sharing, however, make accurate result retrieval difficult: sparse description of shared files and conjunctive query processing [5]. Sparse description is a consequence of the fact that most files are described by solely their filenames, which are limited to about 200 bytes. Conjunctive query processing is used because of its simple semantics and its conservative use of network bandwidth, which is at a premium in inter-networked applications. Together, these characteristics can lead to a decrease of query accuracy when even relevant results are excluded from the result sets returned to users.

To find more precise results, a user may increase the number of terms in a query. With length, however, queries may become so constrained that no instances of relevant results are returned, reducing overall accuracy and forcing the user to re-issue the query.

Consider a search for Mozart's *Clarinet Concerto* in the key A major by clarinetist Michele Zukovsky. We searched for this song on the eDonkey P2P file-sharing system with various combinations of candidate query terms, "Mozart, clarinet, A, major, Zukovsky." 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	
X	X				80
X	X	X	X		54
X	X	X	X	X	0
X	X			X	2

We retrieve the desired result only with an appropriate subset of terms; the last combination in Table 1 contained only relevant results. This experiment demonstrates 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 a Web search engine (e.g., Google) resulted in more accurate results than did any of the sub-queries. We expect similar behavior from P2P file-sharing systems.

We address this problem by having the result servers return results based on alternatives to conjunctive query matching – cosine similarity in particular. Files whose descriptors "nearly match" the query are returned to a degree commensurate with their degree of match. The questions are how well this works in a P2P environment and at what cost.

It turns out that our query processing alternatives can improve accuracy by over 70% but with a seven-fold increase in cost. We show how cost can be tuned so that we have both an increase in accuracy with a reduction in cost.

2. 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

[9][11][14]. Such work assumes fixed environments, distinct from the highly dynamic and distributed one we assume.

Some systems employ distributed hash tables, or more recently, trees, to route queries in distributed environments [8][13]. 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 [15]).

One problem we are trying to address is "term mismatch" – where queries "fail" because they do not contain the same terms as a descriptor [12] – which is particularly problematic in the P2P file-sharing environment. One solution to this problem is to use techniques to transform or expand the terms in either the query or the descriptor [2]. Such techniques, however, are particularly vulnerable to semantic drift due to the sparsness of available terms and lack of global statistics.

A related technique is to mask out terms from a query, preserving only those most likely to match a relevant result, thereby reducing the likelihood of result over-specification [10]. Although this technique improves overall accuracy, result sets suffer from a low precision as well skew toward the unmasked query terms.

There is a similar problem in the database community, where queries with over-specific predicates fail because they return empty result sets [4][16]. Query reconciliation in this case consists of weakening or eliminating some of the selection conditions. This is in the same spirit as our work but can employ different techniques due to the fixed, centralized nature of databases.

3. QUERY PROCESSING SPECIFICATION

In typical P2P file-sharing systems (e.g., Gnutella) peers collectively share a set of (binary) files by maintaining local replicas of a subset of them. Each replica is represented by a user-tuned *descriptor*, which also contains an identifying *key* (e.g., an MD5 of SHA-1 hash on the file's bits). All replicas of the same file share the same key. A client's query is routed to all reachable *servers* based on its "time-to-live." A server compares each query to its local descriptors; a query *matches* a replica if the replica's descriptor contains all of the query's terms. For all matches, the server returns its server identifier and the matching replica's descriptor. This information allows the client to distinguish and, if desired, download the associated file.

Formally, let O be the set of files, M be the set of all terms, and P be the set of peers. Each file $o_i \in O$ has a key associated with it, denoted k_i, such that $k_i = k_j$ if and only if $o_i = o_j$.

Associated with each file o_i is a set of terms, $T_i \subseteq M$, that validly describe it. Each term $t \in T_i$ has a *strength of association* with o_i, denoted soa(t, o_i), where $0 \leq \text{soa}(t, o_i) \leq 1$ and $\sum_{t\in T_i}\text{soa}(t, o_i) = 1$. The strength of association t has with o_i describes the relative likelihood that it is used to describe o_i, assuming that all terms are independent. The distribution of soa values for a file o_i is called the *natural term distribution* of o_i. Intuitively, an average person will describe o_i with terms from T_i with a distribution described by o_i's natural term distribution.

A peer $p_j \in P$ is defined as a pair, (R_j, g_j), where R_j is p_j's set of replicas (i.e., its *local repository*) and g_j is p_i's unique system identifier (e.g., its IP address). Each replica $r \in R_j$ is p_j's copy of some file $o_i \in O$ and has an associated locally maintained *descriptor*, $\text{d}(r) \subseteq M$, which is a multiset of terms. Each descriptor d(r) also contains k_i, the key of file o_i, which we denote by k(r). The maximum number of terms in a descriptor is limited by the system-defined number of bytes descriptors are allocated.

A query $Q_i \subseteq T_i$ for file o_i is also a multiset of terms. The terms in Q_i follow o_j's natural term distribution. When Q_i arrives at a server p_j, p_j returns result set $U_j = \{(\text{d}(r), g_j) \mid r \in R_j \text{ and } Q_i \subseteq \text{d}(r) \text{ and } Q_i \neq \emptyset\}$: all results' descriptors must contain all query terms in accordance to the conjunctive matching criterion.

The client that issues Q receives result set $U = \cup_{pj\in P}U_j$, and groups individual results in U by descriptor key, forming $G = \{G_1, G_2, \ldots\}$, where $G_i = (\text{d}(G_i), \text{k}(G_i), l_i)$. $\text{d}(G_i) = \{\oplus d(r) \mid (\text{d}(r), g) \in G_i\}$ is the group's descriptor, defined as the multiset sum of all of the descriptors of the results contained in G_i. ($\oplus$ denotes the multiset sum operation.) k(G_i) is the key of G_i, identifying the file that G_i represents. $l_i = \{g_j \mid (\text{d}(r), g_j) \in U \text{ and } \text{k}(r) = \text{k}(G_i)\}$ is the list of servers that returned the results in G_i.

The client assigns a rank score to each group with function $F_i \in F$, defined as F: $2^M \times 2^M \times Z \times Z \rightarrow R^+$. If $F_i(\text{d}(G_j), Q, |G_j|, \text{time}_j) > F_i(\text{d}(G_k), Q, |G_k|, \text{time}_k)$, 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, $|G_j|$ is the number of results contained in G_j (the group's *size*) and time$_j$ is the *creation time* of the G_j (i.e., the time when the first result in G_j arrived at the client). Note that in this paper, "result" may refer to an individual result or a group of results. The proper meaning should be clear from the context.

Typically, ranking is performed by "group size," as a large group suggests relevance to a query and multiple sources can better ensure a quick, successful download:

$$F_{\text{gsize}}(\text{d}(G_j), Q, |Gj|, \text{time}_j) = |G_j|.$$

In practice, descriptors are generally implemented via filenames, but a small amount of descriptive information may be embedded in the actual binary of the replica (e.g., ID3 data embedded in mp3 files [3]). Furthermore, when a file is downloaded, the descriptor of this new replica is initialized as a duplicate of one of the servers' descriptors from the result set, but can be subsequently tuned by the user as well.

4. QUERY PROCESSING ALTERNATIVES

Conjunctive queries are problematic because they are potentially overly selective, excluding desired results. Given a query Q with term, t, the probability that a descriptor d(r_p) for a replica r_p of file p contains t is

$$1-\left(1-soa(t,p)\right)^{\|d(r_p)\|}, \tag{1}$$

where $\|\text{d}(r_p)\|$ is the number of (not necessarily unique) terms in d(r_p). Let Q' be the set of unique terms of Q. For d(r_p) such that $\|\text{d}(r_p)\| \geq |Q'|$, the probability that d(r_p) contains Q is

$$\prod_{t_i\in Q'}\left(1-\left(1-soa(t_i,p)\right)^{\|d(r_p)\|-i}\right) \tag{2}$$

As Q grows, the probability that it is contained by d(r_p) decreases exponentially. Although the conjunctive matching criterion is effective at filtering out results that are not relevant – identified with different natural term distributions – it may also filter out desired results because their descriptors are too small or happen, by chance, to not contain all of Q.

4.1 Proposed Alternatives

We propose the application of *cosine similarity* at the server for matching incoming queries with shared files [1]. Cosine similarity is a commonly used metric in information systems to compare the similarity between two documents, when each is modelled as a term-frequency vector:

$$\text{cosSim}(D_1, D_2) = \frac{V(D_1) \cdot V(D_2)}{\|V(D_1)\| \|V(D_2)\|} \tag{3}$$

In the expression above, the inner product of the vector representations of documents D_1 and D_2 is normalized by the product of their lengths, so *cosSim* has a normalized maximum value of 1. If the *cosSim*(Q, d) of an incoming query Q to a descriptor d is greater than some threshold τ, then d and Q match.

As stated in the Section 2, alternatives to query performance include having the client mask out query terms (to reduce the query's selectivity) [10] or using disjunctive matching. Cosine similarity improves these alternatives in the following ways:

- It yields better result quality than conjunctive matching because it is not overly selective with long queries.
- It yields better result quality than query masking because it does not skew matching results toward particular query terms.
- It is more efficient than disjunctive matching because it is better at matching relevant results.

The parameter that must be tuned, however, is τ. A low τ value would make *cosSim* too permissive, like disjunctive matching. Too high a τ value makes *cosSim* too selective, like conjunctive matching.

We also tested techniques such as having the server compute the largest subset of query terms that match at least one descriptor (which was inspired by work on query relaxation in database query processing [16]). For a variety of reasons (e.g., inconsistent matching over different servers) these techniques underperformed *cosSim*, so, due to space limitations, we exclude them.

5. EXPERIMENTAL RESULTS

We simulate the performance of a P2P file-sharing system to test the large-scale performance of our query processing methods. In accordance with the accepted model described in [7] and observations presented in [8], we include in our experimental model interest categories C, a partitioning of O into sets $C_i \in C$, where $C_i \subseteq O$, and $\cup_i C_i = O$. Interest categories are used to model constraints on user interests.

Each category C_i has popularity, b_i, with a Zipf-skewed distribution. At initialization, each peer p_j is assigned some interests $I_j \subseteq C$, based on b_i.

Each file within an instance of an interest category varies in popularity, which is also Zipf skewed. This re-skewing of popularities models individual user interests and governs the likelihood that a peer who has interest in the category that contains o_m is initialized with a replica of o_m. Each replica allocated at initialization has a randomly initialized descriptor subject to its natural term distribution. Peer p_j's interest categories also constrain its searches; p_j only searches files from $\cup_n L_n$ for all $L_n \in I_j$.

We use Web data to simulate our language model (i.e., term distributions and interest categories). Web data are a convenient choice because they constitute a grouping of terms (we use terms' relative frequencies within a Web document to simulate the natural term distribution for a file) and a grouping of documents (we use Web domains to simulate interest categories).

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 as well as other experimental results are found on our Web site or are available on request [17].

Table 2 Query length distribution.

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	1,000
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

Each peer is initialized with a random number of interest categories and a random number of file replicas from these categories. Descriptors for these replicas are initialized independently based on each file's natural term distribution. Queries are issued for files a peer does not already have in the peer's interest category.

Query terms are picked randomly based on the desired file's natural term distribution, similar to a technique described in [1]. The query length distribution shown in Table 2 was derived from observations of query logs we collected over several days on the Gnutella network in the Spring and Summer of 2006 and match that of the literature [6]. Without loss of generality, queries are flooded to all peers. 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 [6].

Although other behavior is possible, we assume that the user identifies and downloads the desired result group with a probability $1/r$, where $r \geq 1$ is its position in the ranked set of results. If the desired result is not in the result set, $r = \infty$ and the top-ranked result, using group size ranking described in Section 4, is selected.

We tried other ranking functions (e.g., cosine similarity), but group size ranking results are representative.

Performance is measured using mean reciprocal rank score (MRR) [1]. MRR is appropriate for known-item search, which is the assumed behavior in P2P file-sharing systems [11]. MRR assumes that there is a single, identifiable desired result, and uses the ranking of this result in the result set to compute an accuracy score. The score for an individual query is the reciprocal of the rank at which the desired result is returned. MRR is defined as

$$MRR = \frac{1}{N_q}\sum_{i=1}^{N_q}\frac{1}{rank_i} \tag{4}$$

where N_q is the number of queries issued by the client and $rank_i$ is the rank of the desired file in query i's result set.

For reference, we also present precision and recall results, which have slightly different meanings here than they do in traditional Information Retrieval (IR) since result replication is a factor in P2P file-sharing, but not typically in IR. Let S_A be the set of replicas of the desired file available throughout the file-sharing system, and S_R be the result set of the query. Precision and recall are defined as:

$$precision = \frac{|S_A \cap S_R|}{|S_R|} \qquad recall = \frac{|S_A \cap S_R|}{|S_A|} \tag{5}$$

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

We test four query matching techniques. Let Q be a query and d be a descriptor in the following query matching technique definitions:

- Conjunctive – Q matches d if $Q \subseteq d$.
- Disjunctive – Q matches d if $Q \cap d \neq \emptyset$.
- Client masking – Let Q' be a subquery of Q as defined in [10]. Q' is matched with d conjunctively.
- Cosine similarity – See definition in Section 4.1.

5.1 Result Quality

The accuracies of the matching techniques are shown in Figure 1. Disjunctive query matching (disj), client masking (cmask) and cosine similarity with $\tau = 0.1$ (cos10) improves MRR by 80%, 40% and 80%, respectively, over conjunctive query processing. Cosine similarity performance degrades with higher τ values. The reason that the alternatives outperform conjunctive matching is because the alternatives perform better on longer queries as shown in Figure 2.

An examination of the precision and recall graphs in Figure 3 and Figure 4 justifies the performance differences. Although precision increases with query length with conjunctive matching, recall decreases drastically. With the other techniques, recall is either stable or increases with query length. In general, the techniques with higher recall perform better. (This explains why cosine similarity performance degrades with τ.)

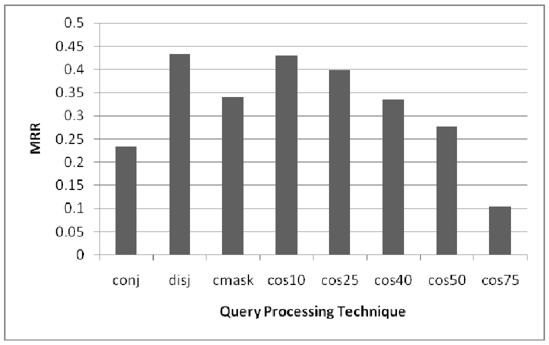

Figure 1. MRR.

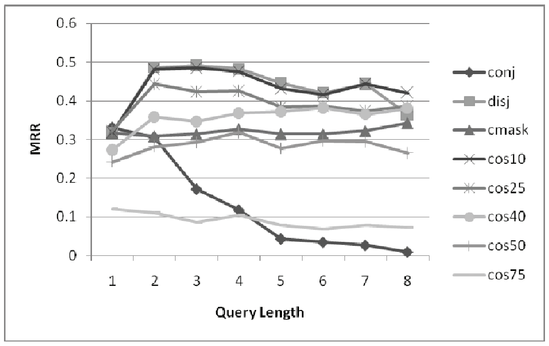

Figure 2. MRR over query length.

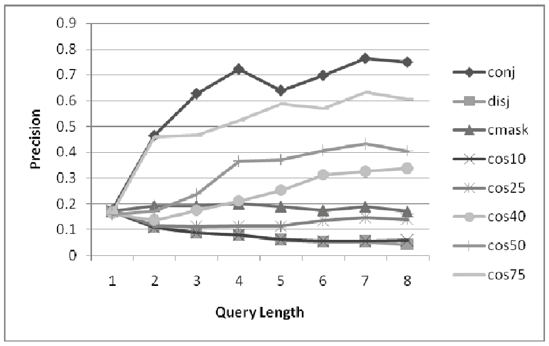

Figure 3. Precision over query length.

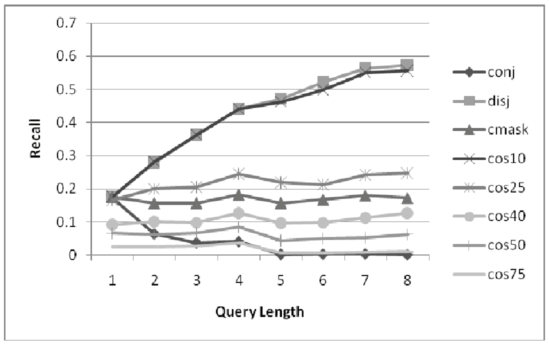

Figure 4. Recall over query length.

Naturally, recall increases and precision decreases with query length with disjunctive matching. Longer disjunctive queries match more results, both relevant and irrelevant. Recall and precision are steady with masking as term used in the single-term query is generally the term left unmasked.

Cosine similarity matching works like either disjunctive or conjunctive matching depending on the value of τ. With low τ values (i.e., low selectivity), longer queries have higher recall and lower precision when using cosine similarity matching, just like disjunctive matching. The opposite is true with high τ values.

What makes cosine similarity matching attractive is that neither its recall nor precision suffer from the extreme low values for longer queries as does conjunctive or disjunctive matching, resulting in better overall performance. Moreover, the particular mix of recall and precision can be controlled with a single parameter, τ.

5.2 Efficiency of the Techniques

The price that must be paid for increased recall is increased cost in terms of the number of results returned to the client. In addition to accuracy, we use cost to demonstrate the superiority of cosine similarity query matching.

When comparing cost shown in Figure 5 to the result quality shown in Figure 1, we see a correlation. The two most effective query matching techniques, disjunctive and cos10, cost the most.

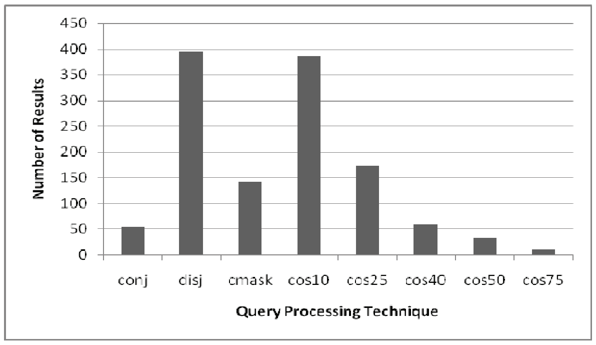

Figure 5. Average number of results per query.

However, cmask, cos25 and cos40 are good compromises between quality and cost. cos25 has an MRR that is only 8% lower than disj, but at less than half the cost. cos40 has an MRR that is similar to that of cmask, but with a cost that is almost 60% lower. In fact, the cost of cos40 is similar to that of conj, but with a 42% greater query accuracy. Cosine similarity query matching therefore allows us to tune the query quality we desire with a trade-off in cost: cos25 nearly matches our best performance but at less than half the cost, while cos40 matches our best cost, but with a 42% improvement in accuracy. (Likewise, cos50 reduces cost by 40% with an 18% increase in accuracy compared with conj.)

6. CONCLUSION

We consider the problem of query processing in fully distributed P2P environments. The current practice of conjunctive query processing turns out to be too restrictive, overspecifying results to the point that no relevant results are returned by long queries. Previous solutions, such as disjunctive query processing, yield more accurate query results, but at a high cost.

Using cosine similarity result matching instead yields a query accuracy on par with the best existing techniques, but with lower cost. It can improve query accuracy by 80% if cost is not an issue. If cost is an issue, it can improve accuracy by 42% with no cost, or, surprisingly, by 18% with a cost reduction of 40%.

Our next step is to analyze the effect that the results yielded by the various query matching techniques have on the effectiveness of various client-side ranking functions.

REFERENCES

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

[2] 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.

[3] M. Nilsson. Id3v2 web site. www.id3.org. 2007.

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

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

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

[7] 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.

[8] 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.

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

[10] W. G. Yee, L. T. Nguyen, and O. Frieder. Masked Queries for Search Accuracy in Peer-to-Peer File-Sharing Systems. In *Proc. IEEE IPDPS*, 2007.

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

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

[13] 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.

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

[15] G. Skobeltsyn, T. Luu, I. Podnar Zarko, M. Rajman, K. Aberer: Web text retrieval with a P2P query-driven index. SIGIR 2007: 679-686.

[16] P. Godfrey, Minimization in Cooperative Response to Failing Database Queries, International Journal of Cooperative Information System (IJCIS), World Scientific, 6(2):95-149, June 1997.

[17] IIT P2P Information Retrieval System Web Site. www.ir.iit.edu/~waigen/pirs.