P2P Content Search: Give the Web Back to the People[*]

Matthias Bender[†], Sebastian Michel[†], Peter Triantafillou[‡], Gerhard Weikum[†], Christian Zimmer[†]

[†]**Max Planck Institute for Informatics, 66123 Saarbruecken, Germany**
{*mbender, smichel, weikum, czimmer*}*@mpi-inf.mpg.de*
[‡]**RACTI and CEID, University of Patras, 26500 Rio, Greece**
peter@ceid.upatras.gr

ABSTRACT

The proliferation of peer-to-peer (P2P) systems has come with various compelling applications including file sharing based on distributed hash tables (DHTs) or other kinds of overlay networks. Searching the content of files (especially Web Search) requires multi-keyword querying with scoring and ranking. Existing approaches have no way of taking into account the *correlation* between the keywords in the query. This paper presents our solution that incorporates the queries and behavior of the users in the P2P network such that interesting correlations can be inferred.

1. INTRODUCTION

The proliferation of peer-to-peer (P2P) systems has come with various compelling applications, most notably, file sharing and IP telephony. File sharing involves file name lookups and other simple search functionality such as finding files whose names, short strings, contain a specified word or phrase. Such simple queries can be executed in a highly efficient and scalable way, based on distributed hash tables (DHTs) or other kinds of overlay networks and dissemination protocols. However, these approaches are unsuitable for searching also the *content* of files such as Web pages or PDF documents. For the latter, much more flexible multi-keyword querying is needed, and, most importantly, the fact that many queries may return thousands of different matches calls for *scoring and ranking*, the paradigm of Web search engines and information retrieval (IR).

1.1 Why P2P Web Search?

In fact, full-fledged Web search is more or less exclusively under the control of centralized search engines. Lately, various projects have been started to build and operate a P2P web search network [12, 20, 25, 30–32] including our own Minerva project [4–6], but so far these endeavors are fairly small-scale. Ironically, Web search and Internet-scale file content search seem to be perfect candidates for a P2P approach, for several reasons: 1) the data is originally highly distributed, residing on millions of sites (with more and more private people contributing, e.g., by their blogs); 2) a P2P network could potentially dwarf even the largest server farm in terms of processing power and could thus enable much more advanced methods for linguistic data analysis, statistical learning, or ontology-based background knowledge and reasoning (all of which are out of the question when you have to serve hundred millions of queries per day on a, however big but centralized, server farm); 3) there is growing concern about the world's dependency on a few quasi-monopolistic search engines and their susceptibility to commercial interests, spam or distortion by spam combat, biases in geographic and thematic coverage, or even censorship. These issues have led to the postulation that "the Web should given back to the people" [14].

[*]Partially supported by the EU within the 6th Framework Programme under contract 001907 *Dynamically Evolving, Large Scale Information Systems* (DELIS).

1.2 Challenges

Comprehensive Web search based on a P2P network has been considered infeasible from a scalability viewpoint [19]. Recent work, however, indicates that the scalability problems could be overcome [3, 5, 18, 30], either by distributing a conceptually global keyword index across a DHT-style network or by having each peer compile its local index at its own discretion (using the peer's own "native" data sources or performing thematically focused Web crawls and other data extraction according to the peer's interest profile). In addition, various acceleration techniques can be employed. For example, [26] pursues a multi-level partitioning scheme, a hybrid between partitioning by keyword and partitioning by document. [8] uses view trees for result caching to improve the P2P search efficiency.

From a query processing and IR viewpoint, one of the key issues is *query routing*: when a peer poses a query with multiple keywords and expects a high-quality top-10 or top-100 ranked result list, the P2P system needs to make a judicious decision to which other peers the query should be forwarded. This decision needs statistical information about the data contents in the network. It can be made fairly efficiently in a variety of ways, like utilizing a DHT-based distributed directory [32], building and maintaining a semantic overlay network (SON) with local routing indexes [1, 11], or using limited forms of epidemic gossiping [12].

However, efficiency of P2P query routing is only one side of the coin. Of course, we also expect good search result quality, that is, good effectiveness in IR terminology, measured in terms of precision and recall. The goal is to be as good as the best centralized search engines, but the P2P approach faces the challenge that the index lists and statistical information that lead to good search results are scattered across the network.

For example, consider two- or three-keyword queries such as "Michael Jordan", "native American music", or "PhD admission". A standard, efficient and scalable, approach would decompose each of these queries into individual terms such as "native" and "American" and "music", identify the best peers for each of the terms separately, and finally combine them, e.g., by intersection or some form of score aggregation in order to derive a candidate list of peers to which the query should be forwarded. The result of this "factorization" would often lead to mediocre results as the best peers (and files located on those peers) for the entire query may not be among the top candidates for any of the individual keywords.

The root cause of the above problem is that the outlined "factorized" method for P2P query routing and processing has no way of taking into account the *correlation* between the keywords in the query. We miss out on the fact that, for example, "PhD" and "admission" are statistically correlated in the corpus, and, even worse, that the best matches for the entire query should exhibit a higher-than-average frequency of *both* terms (ideally within some proximity window). Standard search engines do not necessarily consider these correlations either, but they process index lists on the overall document space directly, whereas the P2P system first needs to identify other peers for query routing in order to access index lists and then sees only partitions of the global index space. Thus, the necessarily coarser aggregation granularity of routing indexes or the distributed directory causes an additional penalty for a P2P approach. On the other hand, directly simulating the centralized algorithms in the P2P network would incur undue communication costs.

One may argue that critical correlations of the above kind typically occur in composite names or phrases, as suggested by our examples. Although this is indeed often the case, the observation alone does not provide a solution. It is virtually impossible to foresee all phrases or names or correlated term pairs that will appear in important user queries, and brute-force precomputation of statistical measures for all possible pairs of terms is not a viable option.

1.3 Solution Outline

The solution that we have developed and advocate in this paper is based on the postulation that we cited above as a motivation for the general direction of P2P search engines: *give the Web back to the people*! We simply observe the queries and behavior of the thousands or millions of users that we expect to be active in a P2P network. More specifically, we monitor queries on the peers where they are initially posed and post them to the distributed directory, where they are aggregated so that interesting correlations can be inferred. The key point to emphasize here is that a querying peer whose query keywords exhibit correlations has a good chance to efficiently find a directory entry that helps to identify the best peers with files that match multiple terms of the query.

A potential caveat to our approach could be that user monitoring and query logging is a breach with user privacy. But it is exactly the key strength of the P2P approach that, unlike with a centralized Web search engine that logs queries and click-stream information, every peer is in a perfect position to define its own policies, would reveal critical data at its discretion, and has full control over the local software to enforce its specified policies.

2. SYSTEM ARCHITECTURE

Our *Minerva* system [4] is a fully operational distributed search engine consisting of autonomous peers where each peer has a local document collection from the peer's own (thematically focused) Web crawls or imported from external sources that fall into the userŠs thematic interest profile. The local data collection is indexed by inverted lists, one for each keyword or term (e.g., word stems) containing document identifiers like URLs and relevance scores based on term frequency statistics. A conceptually global but physically distributed directory [7], which is layered on top of a Chord-style [29] distributed hash table (DHT), manages aggregated information about the peersŠ local knowledge in compact form. Unlike [19], we use the Chord DHT to partition the term space, such that every peer is responsible for the statistics and metadata of a randomized subset of terms within the directory. We do *not* distribute the actual index lists or even documents across the directory. The directory entries for the terms may be replicated across multiple peers to ensure failure resilience and to improve availability. The Chord DHT offers a *lookup* method to determine a peer that is responsible for a particular term. This way, the DHT allows very efficient and scalable access to the global statistics for each term.

2.1 Directory Maintenance

In the publishing process, each peer distributes per-term summaries (*Posts*) of its local index to the global directory. The DHT determines a peer currently responsible for this term and this peer maintains a *PeerList* of all posts for this term. Each post includes the peer's address together with statistics to calculate IR-style measures for a term (e.g., the size of the inverted list for the term, the maximum and average score among the termŠs inverted list entries, and other statistical measures). The query processor uses these statistics to identify the most promising peers for a particular query. In order to deal with churn, we employ proactive replication of directory information to ensure a certain degree of replication, and we use a time-to-live technique that invalidates Posts that have not been updated (or reconfirmed) for a tunable period of time.

2.2 Query Execution

A query with multiple terms is processed as follows. In the first step, the query is executed locally using the peerŠs local index. If the user considers this local result as unsatisfactory, the peer issues a *PeerList request* to the directory for looking up potentially promising remote peers, for each query term separately. From the retrieved lists, a certain number of most promising peers for the complete query is computed (e.g., by simple intersection of the lists), and the query is forwarded to these peers. This step is referred to as *query routing*. For efficiency reasons, the query initiator can decide to not retrieve the complete PeerLists, but only a subset, say the top-k peers from each list based on some relevance measure, or the top-k peers over all lists, calculated by a distributed top-k algorithm like [10, 22].

For scalability, the query originator typically decides to ship the query to only a small number of peers based on an expected benefit/cost ratio. Once this decision is made, the query is executed completely on each of the remote peers, using whatever local scoring/ranking method and top-k search algorithm each peer uses. Each peer returns its top-k results (typically with k equal to 10 or 100), and the results are merged into a global ranking, using either the query originator's statistical model or global statistics derived from directory information (or using some heuristic result merging method [21]).

2.3 Query Routing

The prior literature on query routing has mostly focused on IR-style statistics about the document corpora, most notably, CORI [9], the decision-theoretic framework by [23], the GlOSS method presented in [16], and methods based on statistical language models [27]. These techniques have been shown to work well on disjoint data collections, but are insufficient to cope with a large number of autonomous peers that crawl the Web independently of each other, resulting in a certain degree of overlap as popular information may be indexed by many peers. We have recently developed an *overlap-aware* query routing strategy based on compact statistical synopses to overcome this problem [3].

A missing key functionality in our architecture described so far is the discovery and consideration of correlations among terms in a query. This will be remedied by our new method presented in the subsequent sections.

3. LEARNING CORRELATIONS FROM QUERIES

Our system design provides a natural way to learn term correlations from the peers' queries. To this end we extend the responsibilities of the peer(s) that maintain the Posts for one term; in addition, they keep track of the terms that are correlated to their main term. This is feasible at practically no additional costs. As query processing starts with the PeerList retrieval and needs to send a message to one peer for each query term, it is easy and inexpensive to send also the entire query to these peers by piggybacking a few additional bytes on the messages that are sent anyway. For example, a query such as "native American music" is now sent to three peers, one for each term. Each of these returns a PeerList for only one of the three terms (the one that it is responsible for), but the complete query is remembered by all three.

Collecting query logs at these directory peers is inexpensive up to some point when the storage demand for the logs may become prohibitive. To limit the storage consumption and to filter the logs for statistically significant information, each peer periodically performs a data analysis, using frequent-itemset mining techniques [2, 17]. These techniques identify the term combinations with sufficiently high support and strong correlation, aka. association rules. Subsequently, only these frequent itemsets are remembered and the complete log can be truncated or completely pruned. The computational cost of the data mining is typically linear to quadratic in the size of the query log, and there is a rich body of optimization techniques [13] to make this practically viable even on low-end peer hardware.

The frequent term combinations are then disseminated to other peers using epidemic gossiping techniques, again mostly in a piggybacked manner. This advises the peers in the network that they should not only post their statistical information for single terms but also for strongly correlated term combinations according to the disseminated association rules. To cope with the typically high dynamics of P2P systems in terms of evolving query patterns, standard techniques for aging the statistical data can be employed. For example, query logs may be compiled using moving-window techniques, and exponential-decay techniques can be used to combine fresh with previous results of the frequent-itemset mining and to gradually age the network-wide knowledge of association rules.

Note that query logs are not the only source of estimating term correlations. For example, peers may use local thesauri and dictionaries to identify phrases (composite nouns or proper names), thus realizing the high correlation of the individual terms in queries such as "Michael Jordan" or "soccer world championship". The posting, collecting, and mining of such correlations in the P2P network follows the same algorithm that we described above. In this paper, we concentrate on exploiting correlations from query logs, however.

To validate our hypothesis that queries are useful to recognize correlations, we analyzed a query collection of more than 1.6 million queries compiled from one of the TREC tasks [20]. About 90% of the queries contained 2, 3, or 4 terms so that term correlation should be recognizable. Overall the queries contained about 200 000 different terms. We ran the frequent-itemset mining on this collection, with different levels of support from 0.001 percent (i.e., at least 10 occurrences) to 0.1 (i.e., at least 1000 occurrences). Figure 1 shows the number of distinct term combinations that resulted from this task for different support levels using a logarithmic scale.

As another source of queries, we investigated the query log [28] of the Excite search engine from the year 1999 [1]. The log consists of more than 2 million queries and we analyzed all queries with up to 10 terms. The results confirmed our finding that real web queries exhibit significant correlations.

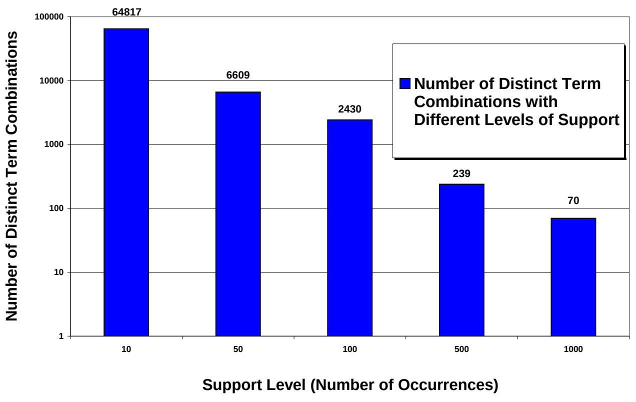

Figure 1: Term combinations above support level

4. EXPLOITING CORRELATIONS IN QUERIES

The directory information about term correlations can be exploited for query routing in several ways. If we treat association rules, i.e., strongly correlated term combinations, as keys for the DHT-based overlay network (e.g., by bringing the terms in a rule into lexicographic order and using a special delimiter between terms to produce a single string), then

[1]Thanks to Dr. Amanda Spink for providing the log.

a query initiator can locate the responsible directory peer by simply hashing the key and using standard DHT lookup routing. This way the directory entry directly provides the query initiator with a query-specific PeerList that reflects the best peers for the entire query. If this PeerList is too short, the query initiator always has the fallback option of decomposing the query into its individual terms and retrieving separate PeerLists for each term. However, this approach has the disadvantage that, whenever the fallback option is needed, the necessary directory lookups entail a second message round and increase the query latency.

A modified variant that avoids the above drawback is based on a simple twist to the directory organization. We still collect PeerLists for high-correlation term combinations, but this information is always posted to the peer that is responsible for the first term in the lexicographic ordering of the terms. This way, when the PeerList for an entire query is insufficient, the same directory peer provides the fallback option for the first term without any additional communication. If we consider replication of directory entries as inexpensive, we actually post the PeerList for a term combination with m terms to *all m peers* that are responsible for one of the terms (with m typically being 2 to 5). Then we can simply use the standard query routing: look up the directory for each query term separately, but whenever a directory peer has good PeerLists for the entire query, this information is returned to the query initiator, too, together with the per-term PeerLists. This does not cause any additional communication costs, and provides the query initiator with the best available information on all individual terms and the entire query.

This last consideration also points to a highly efficient solution for cases when there is no correlation information for a full query, but the query contains term pairs or subsets of terms that are highly correlated and have directory entries. Again, the query initiator contacts all peers responsible for the directory entries of one of the individual terms in the query, but the full query is piggybacked on the lookup message. When a directory peer receives the request, it decomposes the query into its subsets and tests whether it has a PeerList for one or more of these subsets. It then returns the PeerLists for those term combinations that have sufficiently good and long PeerLists and are not covered by other subsets with more terms.

The final fallback position of simply returning the PeerList for the single term that led to contacting this directory peer is automatically included in the outlined strategy. Note that the decomposition into subsets is a purely local operation, and for typical queries with 2 to 5 terms it is certainly not expensive. For "heavy" queries with 20 or more keywords (which may be automatically generated by search tools using query expansion based on thesauri, user feedback, or other "knowledge sources"), fast heuristics for identifying PeerLists for a decent fraction of the subsets or simply limiting the subsets to term pairs or combinations of three terms would be adequate solutions.

5. COST ANALYSIS

This section presents a simple "back-of-the-envelope" cost analysis for the query execution that serves as a feasibility check with regard to scalability. We focus on message costs, as storage and local-processing costs are relatively uncritical in our setting. We assume that we have $P = 1,000,000$ peers, each with $N = 10,000,000$ documents that contain $M = 100,000$ distinct terms (the concrete numbers are for illustration of the magnitudes that we would expect in a real system). We further assume that all queries have $Q = 5$ terms, and that every peer issues $\lambda = 10$ queries per hour. Each term, peer ID, or URL is assumed to have a length of 10 bytes (with appropriate compression of URLs). PeerLists are assumed to contain 100 peer IDs each.

When a peer issues a query, the directory lookup requires exactly $Q = 5$ messages, one for each term, and the entire query is piggybacked on these messages, which results in a message size of (1+Q)*10 = 60 bytes (effective payload, disregarding protocol overhead). Each of the contacted directory peers replies with at least one PeerList (the one for the individual term that led to the lookup) and typically two PeerLists (the single-term PeerList and the one for the entire query if the term combination of the query is among the frequent itemsets), yielding a typical network load of Q*2*100*10 = 10,000 bytes for one query-routing decision in total. If the query initiator then determines the 10 most promising peers by combining the different PeerLists and retrieves the top-100 query results from each of these peers (merely URLs, not full documents), this results in additional 10*100*10 = 10,000 bytes. So the total network load per query is 20,000 bytes. With $N = 1,000,000$ peers each with a query rate of $\lambda = 10h^{-1}$, this results in $N * \lambda * 20,000 = 200 * 10^9$ bytes per hour or approximately $5.66 * 10^6$ bytes per second, less than 100 MBytes/s for the entire P2P network (i.e., spread across many network links). As for network latency, the entire query execution requires two message round-trips through the DHT overlay network, the first round involving Q recipients and the second round involving 10 recipients.

The bottom line of this crude analysis is that the approach does not present any critical performance challenges and seems very viable also from a scalability viewpoint.

6. EXPERIMENTS

6.1 Experimental Setup

For the experimental evaluation of our approach, we used the GOV document collection [24] with more than 1.2 million Web documents from the Internet domain .gov (i.e., US government and other official institutions), containing more than 5 million different terms (including numbers and names). Subsequently this corpus serves as our *reference collection* for a centralized search engine against which we compare our P2P techniques. Our primary quality measure is the *relative recall*, the query results that a small number of peers in the network can together achieve relative to the recall from the centralized reference collection. We created 750 peers, each holding about 1500 documents, by partitioning the overall document collection without any overlap between peers. All peers were running on a single 64-bit Opteron server in our research lab.

For the query workload we used fifty queries from the topic-distillation task of the TREC 2003 Web Track benchmark [24]; examples are "juvenile delinquency", "legalization Marijuana", "shipwrecks", "wireless communications", typically short but nontrivial queries (significantly more difficult than the most popular queries of commercial search engines, as shown, for example, on [15]). As we wanted to measure the influence of term correlations, we were mostly interested in longer queries that should ideally have more than two keywords. Therefore, we manually expanded the queries by additional terms from the query description (an official part of the TREC benchmark queries). For example, the original query "wireless communications" was expanded to "wireless communications broadcasting transmission radio" and the query expansion for "shipwrecks" resulted in "shipwrecks accident". The characteristics of our 50 benchmark queries are shown in Table 1.

Property	Avg	Min	Max	S
Query length	3.59	2	5	1.03
top-25 ideal results	23.65	3	25	5.12
overall results	709.98	3	3285	785.93
Peers with Posts for all query terms	634.31	35	750	209.92
Peers with result documents	340.69	3	738	245.50

Table 1: Properties of test queries (S denotes the standard deviation).

6.2 Quality Measure

In our experiment we compared the distributed P2P query execution with a centralized search engine managing the reference collection. For each query, the centralized search engine computed the 25 top-ranked documents referred to as *ideal results* by applying standard IR scores based on the term frequency tf and the inverse document frequency idf. Multi-term queries are evaluated in conjunctive mode; so only documents that contain all query terms qualify for the final ideal result. In the P2P setting, the local peer engines first perform the peer selection step that will be explained in the following subsection. Having the most promising peers selected, the query is executed and the query initiator merges the result documents of the best peers. The recall of the P2P query execution is the fraction of the ideal result that the merged P2P result yields. As the query execution cost increases with the number of peers that participate in executing a query, we also discuss the benefit/cost ratio of relative recall to the number of contributing peers.

6.3 Peer Selection

The peer selection step distinguishes the two cases with and without term-combination Posts: the standard peer selection combines PeerLists for all query terms containing single-term-specific Posts. IR measures like the maximum term frequency $tf^{max}(t)$ (the maximum number of occurrences of a term t in documents of a peer) and the collection document frequency $cdf(t)$ (the number of documents of a peer containing a term t) are used to compute a peer ranking as described in [6]. In contrast, the case with term-combination Posts looks up the full-query Posts at the corresponding directory peers and uses the following statistics to compute the peer ranking: the collection document frequency $cdf(q)$ of the full query (the number of documents within a peer containing all terms of the query q), and 2) the sum of maximum term frequencies $\sum_{t \in q} tf^{max}(t)$ over all terms t in the query q.

6.4 Experimental Results

In the experiment we systematically increased the number of remote peers that the query initiator chooses to involve in the query execution. We compared the two strategies: standard peer selection versus correlation-aware peer selection for term combinations. Figure 2 shows the relative recall numbers for this setup (averaged over all 50 queries).

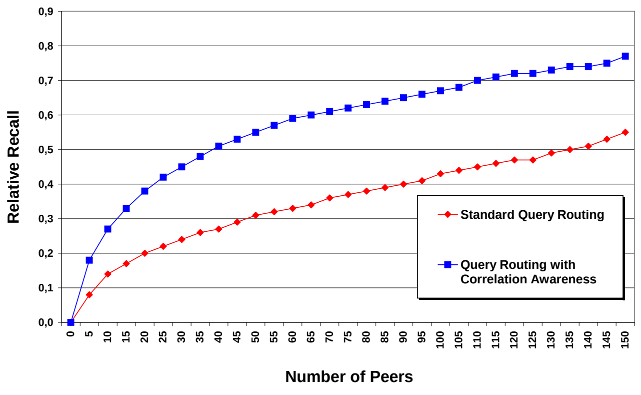

Figure 2: Experimental Results

The chart clearly shows that the use of multi-term Posts improves recall and the benefit/cost ratio by a substantial margin. To reach 50% recall, the standard query routing strategy needs to contact about 130 peers, whereas the correlation-aware strategy can reduce this number to 45 peers for the same recall. As a concrete example of the improvements, cosnider the query "shipwreck accident". Overall, there were 46 result documents containing both terms spread over 44 different peers, but 437 peers contained documents with "shipwreck" alone or with "accident" alone. The standard peer selection needs 80 peers to reach a recall of 60% whereas the new correlation-aware routing with multi-term Posts for the complete query obtains all relevant results (i.e., 100% recall) by contacting exactly the 44 relevant peers.

7. CONCLUSION

Our ongoing research efforts in the area of P2P content search are driven by the desire to "give the Web back to the people" [14], enabling ultra-scalable and robust information search that is immune to the quasi-monopolies and potential biases of centralized commercial search engines. This paper has explored the theme of leveraging "the power of users" in a P2P Web search engine, by exploiting term correlations that appear in query logs. We believe that considering user and community behavior, in implicit or explicit form, is one potential key towards better search result quality for advanced expert queries (as opposed to mass-user queries that are well served by commercial search engines).

REFERENCES

[1] Karl Aberer, Philippe Cudré-Mauroux, Manfred Hauswirth, and Tim Van Pelt. Gridvine: Building Internet-Scale Semantic Overlay Networks. In *International Semantic Web Conference*, 2004.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining Association Rules Between Sets of

Items in Large Databases. In *SIGMOD Conference*, 1993.

[3] Matthias Bender, Sebastian Michel, Peter Triantafillou, Gerhard Weikum, and Christian Zimmer. Improving Collection Selection with Overlap-Awareness. In *SIGIR*, 2005.

[4] Matthias Bender, Sebastian Michel, Peter Triantafillou, Gerhard Weikum, and Christian Zimmer. Minerva: Collaborative P2P Search. In *VLDB*, 2005.

[5] Matthias Bender, Sebastian Michel, Gerhard Weikum, and Christian Zimmer. Bookmark-driven Query Routing in Peer-to-Peer Web Search. In *Workshop on P2P IR*, 2004.

[6] Matthias Bender, Sebastian Michel, Gerhard Weikum, and Christian Zimmer. The Minerva Project: Database Selection in the Context of P2P Search. In *BTW*, 2005.

[7] Matthias Bender, Sebastian Michel, Christian Zimmer, and Gerhard Weikum. Towards Collaborative Search in Digital Libraries using Peer-to-Peer Technology. In *DELOS*, 2004.

[8] Bobby Bhattacharjee, Sudarshan S. Chawathe, Vijay Gopalakrishnan, Peter J. Keleher, and Bujor D. Silaghi. Efficient Peer-to-Peer Searches using Result-Caching. In *IPTPS*, 2003.

[9] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching Distributed Collections with Inference Networks. In *SIGIR*, 1995.

[10] Pei Cao and Zhe Wang. Efficient Top-K Query Calculation in Distributed Networks. In *PODC*, 2004.

[11] Arturo Crespo and Hector Garcia-Molina. Semantic Overlay Networks for P2P Systems. Technical report, Stanford University, 2002.

[12] Francisco Matias Cuenca-Acuna, Christopher Peery, Richard P. Martin, and Thu D. Nguyen. Planetp: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. In *HPDC*, 2003.

[13] Min Fang, Narayanan Shivakumar, Hector Garcia-Molina, Rajeev Motwani, and Jeffrey D. Ullman. Computing Iceberg Queries Efficiently. In *VLDB*, 1998.

[14] H. Garcia-Molina. Panel Discussion P2P Search at *CIDR*, 2005.

[15] Google.com Search Engine. Google Zeitgeist. *http://www.google.com/press/zeitgeist.html*

[16] Luis Gravano, Hector Garcia-Molina, and Anthony Tomasic. Gloss: Text-Source Discovery over the Internet. *ACM*, 1999.

[17] Jiawei Han and Micheline Kamber. *Data Mining Concepts and Techniques*. Morgan Kaufman, 2001.

[18] Ryan Huebsch, Joseph M. Hellerstein, Nick Lanham, Boon Thau Loo, Scott Shenker, and Ion Stoica. Querying the Internet with Pier. In *VLDB*, 2003.

[19] Jinyang Li, Boon Thau Loo, Joseph M.Hellerstein, M. Frans Kaashoek, David R. Karger, and Robert Morris. On the Feasibility of Peer-to-Peer Web Indexing and Search. In *IPTPS*, 2003.

[20] Jie Lu and James P. Callan. Content-Based Retrieval in Hybrid Peer-to-Peer Networks. In *CIKM*, 2003.

[21] Weiyi Meng, Clement T. Yu, and King-Lup Liu. Building Efficient and Effective Metasearch Engines. *ACM*, 2002.

[22] Sebastian Michel, Peter Triantafillou, and Gerhard Weikum. Klee: A Framework for Distributed Top-K Query Algorithms. In *VLDB*, 2005.

[23] Henrik Nottelmann and Norbert Fuhr. Evaluating Different Methods of Estimating Retrieval Quality for Resource Selection. In *SIGIR*, 2003.

[24] National Institute of Standards and Technology. Text Retrieval Conference (TREC).

[25] Patrick Reynolds and Amin Vahdat. Efficient Peer-to-Peer Keyword Searching. In *Middleware*, 2003.

[26] Shuming Shi, Guangwen Yang, Dingxing Wang, Jin Yu, Shaogang Qu, and Ming Chen. Making Peer-to-Peer Keyword Searching Feasible Using Multi-Level Partitioning. In *IPTPS*, 2004.

[27] Luo Si, Rong Jin, James P. Callan, and Paul Ogilvie. A Language Modeling Framework for Resource Selection and Results Merging. In *CIKM*, 2002.

[28] Amanda Spink, Bernard J. Jansen, Dietmar Wolfram, and Tefko Saracevic. From E-Sex to E-Commerce: Web Search Changes. *IEEE Computer*, 35(3):107–109, 2002.

[29] Ion Stoica, Robert Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *SIGCOMM*, 2001.

[30] Torsten Suel, Chandan Mathur, Jo wen Wu, Jiangong Zhang, Alex Delis, Mehdi Kharrazi, Xiaohui Long, and Kulesh Shanmugasundaram. Odissea: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval. In *WebDB*, 2003.

[31] Chunqiang Tang and Sandhya Dwarkadas. Hybrid Global-Local Indexing for Efficient Peer-to-Peer Information Retrieval. In *NSDI*, 2004.

[32] Yuan Wang, Leonidas Galanis, and David J. de Witt. Galanx: An Efficient Peer-to-Peer Search Engine System.