P2P Directories for Distributed Web Search: From Each According to His Ability, to Each According to His Needs[*]

Matthias Bender, Sebastian Michel, Gerhard Weikum
Max-Planck-Institut für Informatik
66123 Saarbrücken
{mbender, smichel, weikum}@mpi-inf.mpg.de

Abstract

A compelling application of peer-to-peer (P2P) system technology would be distributed Web search, where each peer autonomously runs a search engine on a personalized local corpus (e.g., built from a thematically focused Web crawl) and peers collaborate by routing queries to remote peers that can contribute many or particularly good results for these specific queries. Such systems typically rely on a decentralized directory, e.g., built on top of a distributed hash table (DHT), that holds compact, aggregated statistical metadata about the peers which is used to identify promising peers for a particular query. To support an a-priori unlimited number of peers, it is crucial to keep the load on the distributed directory low. Moreover, each peer should ideally tailor its postings to the directory to reflect its particular strengths, such as rich information about specialized topics that no or only few other peers would also cover. This paper addresses this problem by proposing strategies for peers that identify suitable subsets of the most beneficial statistical metadata. We argue that posting a carefully selected subset of metadata can achieve almost the same result quality as a complete metadata directory, for only the most relevant peers are eventually involved in the execution of a given query. Additionally, asking only relevant peers will result in higher precision, as the noise introduced by poor peers is reduced. We have implemented these strategies in our fully operational P2P Web search prototype Minerva, and present experimental results on real-world Web data that show the viability of the strategies and their gains in terms of high search result quality at low networking costs.

1 Introduction

1.1 Background: The Case for P2P Web Search

The peer-to-peer (P2P) approach facilitates the sharing of huge amounts of data in a distributed and self-organizing way. These characteristics offer enormous potential benefit for P2P-based Web search, powerful in terms of scalability, efficiency, and resilience to failures and dynamics. Additionally, such a search engine offers great opportunities for collaborative search and recommendations, and can potentially benefit from the intellectual input (e.g., bookmarks, query logs, click streams, etc.) of a large user community participating in the data sharing network. Finally, but perhaps even more importantly, a P2P Web search engine can also facilitate pluralism in informing users about internet content, which is crucial in order to preclude the formation of information-resource monopolies and the biased visibility of content from economically powerful sources.

The crucial challenge in developing successful P2P Web search engines is based on reconciling the following high-level, conflicting goals: on the one hand, delivering high quality results with respect to IR measures like precision/recall, and, on the other hand, providing scalability in the presence of a very large, rapidly evolving peer population and the very large amounts of data that must be communicated in order to meet the first goal. We put forward our fully implemented P2P Web search prototype Minerva [5], whose architecture, design, and implementation address these goals.

Minerva pursues the following architecture for a P2P Web search federation. Each peer is fully autonomous and has its own local search engine and a local index that can be built from the peer's own crawls or imported from external sources and tailored to the user's thematic interest profile. Peers may share summaries of their local indexes (or specific fragments of local indexes) by posting metadata into a P2P-network-based directory. The conceptually global but physically distributed directory, which is layered on top of a

---

[*] Karl Marx, 1818-1883, German philosopher.

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

Chord-style Distributed Hash Table (DHT) [29], holds compact, aggregated information about the peers' local indexes and only to the extent that the individual peers are willing to disclose.

When a peer issues a query, it is first executed locally on the peer's own personalized index. If the result is not fully satisfactory, the query can be forwarded to a judiciously chosen, small set of remote peers, based on the thematic similarity of these peers to the query originator or the query itself, as well as peer-quality measures based on local index sizes, term-frequency statistics, or estimates of the peers' overlap with the index of the query originator [4, 15, 20]. The decision for this *query-routing* step, also known as *peer selection* or *resource selection*, is based on statistical metadata that is kept in the P2P directory and is easily accessible by all peers.

1.2 Problem Statement: P2P Postings According to Abilities and Needs

In order to build a scalable system that supports an a-priori unlimited number of peers, it is crucial to limit the amount of metadata posted to the P2P directory in order to ensure a moderate base load caused by the directory maintenance. Moreover, it is important that the statistical information in the directory reflects the *specific abilities and strengths* of the individual peers, so that the directory lookups by peers issuing a query can indeed obtain additional insight that helps satisfying these peers' *specific needs* for the given query topics.

Consider, for example, a peer P_0 with a user specifically interested in soccer, and assume that the peer has a sizable local index with the most important Web pages about the topic. A query such as "German goalkeeper England premiere league" about information on German goalkeepers who play in the British Premiere League would have to be routed to remote peers with rich corpora and a strong focus on soccer. But the routing decision should avoid selecting peers whose contents has high overlap with the local index of our peer P_0 - a problem we have addressed in other papers [4, 20] - or peers that are just generically good sources for sports information but do not stand out as authorities about soccer in England. As the query routing decision is based on statistical information on term frequencies and related measures, it may be difficult to identify the specifically good peers in the masses of generically strong peers whose statistical measures in the P2P directory make them appear promising.

Key to a solution is to identify the statistical features that characterize a peer's specific content and strength, and makes it stand out among the peer community. Then only these features should be posted to the P2P directory. As most peers are distinguished authorities only for a few topics (characterized, e.g., by a set of terms), this posting strategy would also drastically reduce the storage and bandwidth load necessary for directory maintenance. Conversely, as the directory is used to identify only the *most promising* peers for a particular query, the fact that much of the general statistics about peers is no longer published in the directory should hardly affect the recall of search results.

1.3 Contribution and Outline

In this work, we develop different strategies that enable a peer to identify those parts of its local index that make it superior to other peers, i.e., for which it is likely to contain better results than other peers. Conversely, the peer can also identify those parts of its index that are inferior to the indexes of other peers. Thus, the peer can refrain from publishing statistics about its "weaknesses" and concentrates the posting efforts on its specific abilities and strengths. This classification and the resulting posting strategies are based on comparing statistics about the local index against estimated statistics for the complete network. The global estimates are computed in a low-cost manner within the distributed directory and can be efficiently looked up or proactively disseminated across all peers.

In this paper, the most important of these local-versus-global comparisons are for the *document frequency* measure, also known as *df*. The *local df* of a peer for a given term t is the number of different documents containing t that the peer has in its local index. The *global df* for t is the network-wide number of distinct documents that contain t; obviously it is all but trivial to estimate *global df* values in the presence of overlapping collections. Our idea outlined above then translates as follows: all peers identify the terms for which their *local df* values are significantly above the average *local df*, as derived from the *global df* estimate and information about the current number of peers in the system. This is an example of our strategies, other measures such as *mutual information (MI)* can be used as well. The paper's contributions are that it develops this family of P2P posting strategies, investigates the suitable statistical measures and the underlying estimation problems, presents a practically viable solution with special care about networking costs, and demonstrates the benefits in experiments with real Web data.

The remainder of the paper is organized as follows. Section 2 reviews recent related research from different fields of study. This includes a brief overview of the P2P network protocol Chord, which forms the basis of our system design; readers who are familiar with Chord may skip this part. The system architecture of Minerva, our P2P Web search prototype, is introduced in Section 3. Section 4 presents our method for estimating *global df* values, which forms the basis for one of the P2P posting strategies. Section 5 is the paper's technical main contribution; it develops a family of peer-specific strategies for P2P posting and discusses their properties. Experimental results on real-word Web data are presented in Section 6, assessing the performance of different strategies. Section 7 concludes this work and points at future research.

2 Related Work

2.1 Peer-to-Peer Architectures

The efficient location of nodes and data keys (e.g., file names) in a P2P architecture is a fundamental problem that has been tackled from various directions. Early (but nevertheless popular) systems like Gnutella rely on unstructured architectures in which a peer forwards messages to all known neighbors. Typically, these messages include a *Time-to-live (TTL)* tag that is decreased whenever the message is forwarded to another peer. Even though studies show that this *message flooding* (or *gossiping*) works remarkably well in most cases, there are no guarantees that all relevant nodes will eventually be reached. Additionally, the fact that numerous unnecessary messages are sent interferes with our goal of a highly scalable architecture.

Recent research on P2P systems, thus, favors structured overlay networks with guarantees about message routing path lengths as well as lookup efficiency, and strong behavior regarding scale and dynamics (i.e., failures and churn) which can be guaranteed with high probability. Systems such as Chord [29], CAN [24], Pastry [26], P2P-Net [7], or P-Grid [2] are typically based on various forms of distributed hash tables (DHTs) and support mappings from keys, e.g., titles or authors, to locations in a decentralized manner such that routing scales well with n, the number of peers in the system. Typically, an exact-match key lookup can be routed to the proper peer(s) in at most $O(log\ n)$ hops, and no peer needs to maintain more than $O(log\ n)$ routing information. These architectures can also cope well with failures and the high dynamics of a P2P system as peers join or leave the system at a high rate and in an unpredictable manner. However, the approaches are limited to exact-match, single keyword queries on keys. This is insufficient when queries should return a ranked result list of the most relevant approximate matches in the spirit of IR models.

2.2 Distributed IR and Web Search

Many approaches have been proposed for distributed IR, most notably, CORI [9], the decision-theoretic framework [23], GlOSS [14], and methods based on statistical language models [27]. In principle, these methods could be applied to a P2P setting, but they fall short of various critical aspects: they incur major overhead in their statistical models, they do not scale up to large numbers of peers with high dynamics, and they disregard the crucial issue of collection overlap.

Galanx [31] is a P2P search engine implemented using the Apache HTTP server and BerkeleyDB. A site's Web servers are the peers of this architecture; pages are stored only where they originate from, thus forming an overlap-free network. PlanetP [11] is a publish-subscribe service for P2P communities, supporting content ranking search. The global index is replicated using a gossiping algorithm. Odissea [30] assumes a two-layered search engine architecture with a global index structure distributed over the nodes in the system. It actually advocates using a limited number of nodes, in the spirit of a server farm. GridVine [1] addresses the problem of building scalable semantic overlay networks and identifies strategies for their traversal using P-Grid [2]. P2P-Diet [16] consists of super-peers and client-peers and aims to support both ad-hoc and continues queries. Pepper [22] is a hierarchical peer-to-peer system that supports searching and browsing. In Pepper, super-peers use the decision-theoretic framework [13] for resource selection. None of this prior work considers the problem of estimating the *global df* value in the presence of peers with overlapping local contents.

2.3 Chord - A Scalable P2P Lookup Service

Chord [29] is a distributed lookup protocol that provides the functionality of a distributed hash table (DHT) by supporting the following $lookup$ operation: given a key, it maps the key onto a node. For this purpose, Chord uses consistent hashing [17]. Consistent hashing tends to balance load, since each node receives roughly the same number of keys. Moreover, this load balancing works even in the presence of a dynamically changing hash range, i.e., when nodes fail or leave the system or when new nodes join.

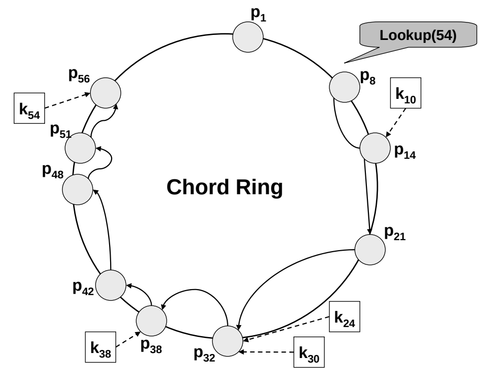

Figure 1. Chord Architecture

The idea behind Chord is as follows: all nodes p_i and all keys k_i are mapped onto the same cyclic ID space. In the following, we use keys and peer numbers as if the hash function had already been applied, but we do not explicitly show the hash function for simpler presentation. Every key k_i is assigned to its closest successor p_i in the ID space, i.e., every node is responsible for all keys with identifiers between the ID of its predecessor node and its own ID. For example, consider Figure 1. Ten nodes are distributed across the ID space. Key k_{54}, for example, is assigned to node p_{56} as its closest successor node.

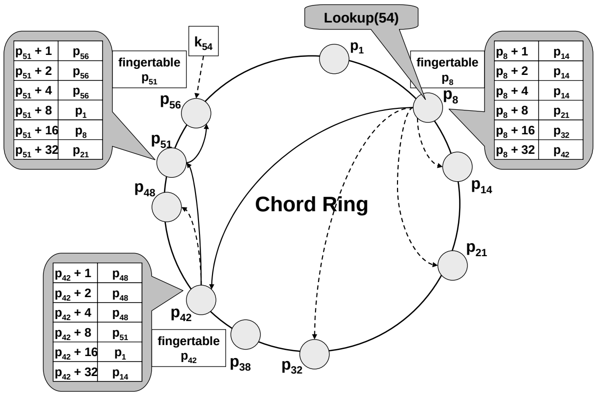

Figure 2. Chord Lookups Using Finger Tables

A naive approach of locating the peer responsible for a key would be to follow the successor pointers on the ID circle. To accelerate lookups, Chord maintains additional routing information: each peer p_i maintains a routing table called *finger table*. The m-th entry in the table of node p_i contains a pointer to the first node p_j that succeeds p_i by at least 2^{m-1} on the identifier circle. This scheme has two important characteristics. First, each node stores information about only a small number of other nodes, and knows more about nodes closely following it on the identifier circle than about nodes farther away. Secondly, a node's finger table does not necessarily contain enough information to *directly* determine the node responsible for an arbitrary key k_i. However, since each peer has finger entries at power of two intervals around the identifier circle, each node can forward a query at least halfway along the remaining distance between itself and the target node. This property is illustrated in Figure 2 for node p_8. It follows that the number of nodes to be contacted (and, thus, the number of messages to be sent) to find a target node in an n-node system is $O(\log n)$.

Chord implements a stabilization protocol that each peer runs periodically in the background and which updates finger tables and successor pointers in order to ensure that lookups execute correctly as the set of participating peers changes. But even with routing information becoming stale, system performance degrades gracefully.

Chord can provide lookup services for various applications, such as distributed file systems or cooperative mirroring. However, Chord by itself is not a search engine, as it only supports single-term exact-match queries and does not support any form of ranking.

3 P2P Web Search: The Minerva Prototype

This paper assumes an architecture of a P2P Web search federation as follows. Each peer is fully autonomous and has its own local search engine and a local index that can be built from the peer's own crawls or imported from external sources and tailored to the user's thematic interest profile. The index contains inverted index lists for each term holding *(docID, score)* pairs. The score reflects the significance of *docID* for this term.

Peers may share their local indexes (or specific fragments of local indexes) by posting metadata into a P2P network. This metadata contains compact statistics and quality-of-service information, and effectively forms a conceptually global (but physically distributed) directory.

A peer uses the global directory to identify candidate peers that are most likely able to provide good query results. A query posed by a user can first be executed on the user's own peer, but can be additionally forwarded to these other peers for better result quality. The results obtained from there are merged by the query initiator.

The rationale for an appropriate query routing strategy, i.e., the selection of the most promising peers among the, possibly large, set of candidates, is based on the following three observations:

1. The query initiator should prefer peers that are likely to hold highly relevant information for a particular query.
2. On the other hand, the query should be forwarded to peers that offer a great deal of *complementary results*.
3. Finally, this process should incur acceptable overhead.

Minerva is a fully operational distributed search engine that we have implemented and that serves as a valuable testbed for our work.

A conceptually global but physically distributed directory, which is layered on top of a Chord[29]-style Dynamic Hash Table (DHT), holds compact, aggregated information about the peers' local indexes and only to the extent that the individual peers are willing to disclose. Unlike [19], we use the DHT to partition the term space, such that every peer is responsible for the metadata from all peers in the network for a randomized subset of terms. We do *not* distribute index lists or even documents across the directory. For failure resilience and availability, the responsibility for a term may be shared and replicated across multiple peers.

Directory maintenance, query routing, and query processing work as follows. In a preliminary step (step 0), every peer publishes statistical information (*Posts*) about every term in its local index to the directory. Chord is used in order to determine the peer currently responsible for a term. This peer maintains a *PeerList* of all postings for this term from peers across the network. Posts contain contact information about the peer who posted this summary together with statistics to calculate IR-style relevance measures for a term (e.g., the size of the inverted list for the term, the maximum average score among the term's inverted list entries, or some other statistical measure). The query initiator retrieves the PeerLists for all query terms from the distributed directory. It then combines the given information to find the most promising peers for the current query. For efficiency reasons, the query initiator can decide to not retrieve the complete PeerLists (which could be a problem if there are thousands of peers in the system) but only a subset, say the top-k peers from each list, or more appropriately the top-k peers over all lists, calculated by a distributed top-k algorithm like [21].

The goal in the query routing process is to select promising peers for a query based on the previously published per-term statistics (Posts). We want to complete the entire selection process before sending the query to the remote peers so that all remote peers can be queried in parallel to avoid additional latencies. In other words, we do not rely on the actual query results of a peer when selecting subsequent peers, but only on the previously published statistics.

Query routing has been a research issue for many years [8, 23, 4]. For Minerva, we have implemented a number of different strategies that are based on the statistical information contained in the Posts, such as CORI [8] that focuses on the number of documents that a peer can potentially add to the query results as a quality measure. A number of different strategies for Minerva have been evaluated in previous work [6].

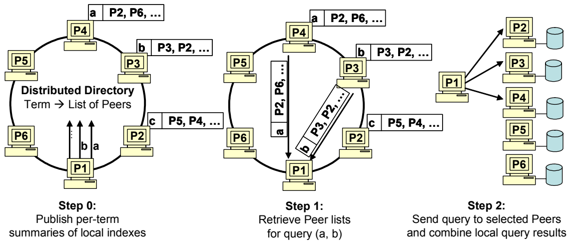

Figure 3. Minerva System Architecture

3.1 Cost Analysis

Most of the **network cost** is caused during the posting process, i.e., when a peer publishes its per-term statistics. Conceptually, each Post consists of the term it represents, an IP address and port number, plus collection-specific statistical information (e.g., collection size) and term-specific statistical information (e.g., document frequency). Typically, such a Post accounts for approximately 50 bytes. Where applicable, we use a straight-forward batching of Posts to further decrease the number of messages. For all messages, we also apply *gzip* compression to additionally decrease the message payload size.

After the dissemination of the Posts, peers executing a query perform so-called *PeerList requests* to retrieve a list of peers that have published statistics about the specific query terms. The network load depends on the number of query terms and the length of the result list.

The **storage cost** at the directory peers storing the Posts is also directly dependent on the number of Posts, the size of a Post, and the number of Peers in the system. In an n peer network storing Posts of m distinct terms, each peer is responsible for an expected m/n PeerLists. For example, in a system with $50,000$ terms and $10,000$ peers, each peer is responsible for the maintenance of an average of 5 PeerLists. This number decreases even further as more and more peers join the system, because they typically do not add a significant number of new terms. In a worst case scenario (every peer has posted information for all terms), a directory peer would thus be responsible for 50.000 Posts or 2.5 MB, which we consider a negligible storage effort.

The **computational cost** incurred at the directory peers is negligible, as it basically only consists of maintaining a list of Posts for each term that it is responsible for within the distributed directory.

4 Global df estimation

Given the large-scale data distribution, one of the key technical challenges in P2P Web search is *result merging*, i.e., the process of effectively combining local query results from different sources. While document scoring and ranking is a challenging problem already in centralized systems, additional difficulty in a distributed environment stems from the fact that most of the popular document scoring models, such as TFIDF or BM25 [25], use collection-wide statistical information for this purpose. Most prominently, both use *document frequencies (df)*, i.e. the number of documents in the collection that contain a query term[1]. The local usage of

---

[1] Note the difference to the notion of *peer* or *collection frequencies* that estimate the number of *collections* that contain a query term. The *document frequency*, instead, represents the total number of distinct documents that contain a term.

collection-specific *df* values in these scoring models results in document scores that are incompatible across collections and, thus, make result merging difficult. On the other hand, if *global df* values could be applied, the document scoring and ranking would be ideal in the sense that it is identical to the document ranking that would be produced by a hypothetical combined collection.

Early research on distributed information retrieval systems typically assumed disjointly partitioned collections. In such a setting, the *global df* value is simply the sum over all *local df* values. However, we envision autonomous peers that independently gather thematically focused collections through web crawls or similar techniques. In such a setting, studies show a skewed distribution of documents across the collections, with popular documents contained in a large fraction of collections. Thus, summing up the *df* values across collections would inevitably lead to biased *df* values (and, thus, document scores) , as popular documents are repeatedly accounted for [18]. Additionally, thematically focused collections show a high variance of df values for the same term among each other (whereas randomly partitioned collections show a rather uniform distribution of df values for the same term). This further increases the necessity of a score normalization across peers.

In [3], we propose a robust and scalable approach towards estimating *global df* values using *hash sketches* [12], that we will briefly sketch in this section. Given the system design introduced above with a hash-based assignment of terms to responsible directory peers, it is very natural for the these peers to maintain additional data that supports the *global df* estimation for the terms they are responsible for. We propose that every peer includes a hash sketch representing its index list for the respective term when publishing its (term-specific) Post, so that a directory peer can compute an estimate of the *global df* values for the terms it is responsible for (as the hash sketch synopses representing the index lists of all peers for a particular term are *all* sent to the same directory peer). This peer can, by means of inexpensive bit-wise operations, calculate a moving-window estimate for the *global df* for the terms it is responsible for from these synopses.

Having calculated such *df* estimates, we propose two methods for the dissemination of these values and their usage in the query process:

- Each time a peer contacts a remote directory node, e.g., during the posting process, it retrieves the current *df* estimates from the directory peer and uses these values to re-compute its local scores. In that way, all document scores (even though locally calculated) are directly comparable, as they share the same statistical information. The disadvantage is the necessity to repeatedly re-compute a large number of local scores, which poses a high computational burden on the peers.
- The query initiator collects the estimates as piggybacked information when retrieving the PeerLists from the directory peers[2]. The query initiator then includes the df values when sending the query to peers selected in the query routing phase. These remote peers can use the df estimates on-the-fly (as weights during index scans) to compute their local query results, to produce globally comparable scores

We advocate the second option and present a detailed study of its low overhead cost in [3].

5 Peer Strategies for P2P Directory Posting

This section discusses a number of different strategies to decrease the effort necessary to build and maintain the metadata repository. A key observation is the fact that, at query time, the directory is used to identify the *most promising* peers for a particular query. If the metadata for peers that are *not* selected had not been published to the directory, the directory load could have been decreased without sacrificing result quality. We propose ways to identify which Posts are promising enough to be published and stored in the metadata repository in an attempt to decrease the load of the directory in several ways:

- it decreases the network traffic, as less Posts are sent around at publishing time,
- it decreases the storage cost at the directory peers, and
- it decreases the traffic when retrieving the PeerLists for a particular query.

We argue that, if selecting the metadata carefully and in a suitable way, the reduced amount of metadata in the directory will *not* result in a degradation of the result quality in terms of result recall, but actually lead to an increase in result precision, as the noise introduced by poor peers is reduced.

5.1 Threshold Strategies

We propose a number of different threshold strategies to select the Posts that are of value to the system.

1. **Absolute Threshold.** Peers only publish Posts of terms for which their index contains at least *threshold* documents. This is computationally easy for the local peers (most naively, a simple filter when publishing), it does not involve any additional communication with the directory peers and is, obviously, independent from

---

[2] Remember that the df estimate for a particular term is maintained at the same peer that maintains the respective PeerList

any *global df* estimates. In fact, experiments show that most terms that appear only once or twice in a reasonably sized collection are due to typing errors or other artefacts that are not likely to become query terms. Refraining from publishing metadata for these terms, thus, does not decrease result quality at all.

2. **Relative Threshold.** Peers only publish Posts of terms for which their index contains at least *threshold* percent of all documents containing this term (estimated by the *global df* estimation described above). This technique allows a more flexible pruning of metadata for terms that (as in practice), differ highly in popularity. For example, consider the terms *network* and *latex*. As there are certainly more documents containing the term *network*[3], it would be difficult to find one suitable absolute threshold that fits both terms. With a relative threshold, however, we can adjust the threshold automatically to the absolute popularity of the term. Pruning unnecessary Posts at publishing time for this strategy is also computationally reasonable.

3. **Top-x Quantile.** Based on the relative threshold, a peer only publishes the top-x quantile of terms with the highest ratio of *local df* to *global df*, where x is 90% or higher (i.e., the 10% strongest terms). Intuitively, a peer selects those terms in its local index that are relatively more frequent than in a hypothetically average collection. However, doing so is computationally slightly more expensive, as it involved to steps: first, the ratio of *df*'s has to be computed for all terms, before all terms have to be re-sorted in order to identify the Posts to send.

Notice that the first two strategies can not only be enforced at peers publishing their metadata, but also at the directory peer receivings the respective metadata. Doing this does obviously not decrease the network traffic incurred when publishing the metadata, but it adds even more opportunities to carefully select the metadata most relevant to the system. For example, the directory peer can adapt the threshold values with respect to the number of peers that have published metadata for a term, i.e., it could decrease the threshold if too few peers are publishing metadata for a term or, vice-versa, increase the threshold if there is enough metadata describing peers of reasonable quality.

5.2 Poisson-based Strategies

A more sophisticated strategy could involve statistical information about the distribution of *local df* values in the list of peers that have published metadata for a term. Typically, a large fraction of peers in the system contain a term only a few times, whereas a small portion of peers account for the majority of documents. We can easily approximate this distribution, e.g., using a *Poisson distribution* [10], at the directory peers. As Poisson distributions can be represented by only one floating point value (its mean), and more powerful *Two-Poisson Mixes* by three floating point values, it is cheap to disseminate these values to the peers and let them publish their metadata only if the peer belongs to the top *threshold* percent of peers.

This strategy fits perfectly with our initial motivation that we are mainly interested in the most promising peers for a term. Using such a distribution allows us to estimate a peer's *rank* for a term within the network, while the threshold strategies introduced in the previous subsection cannot tell us anything about this rank, but only rely on statistical metadata about the index terms disregarding their actual distribution across the peers in the network.

5.3 MI-based Strategy

Another strategy is based on the *mutual information (MI)* that is exhibited by the peers' different *local df* values for different terms. MI is an information-theoretic concept, also known as *relative entropy*, which is popular as a feature-selection criterion for statistical learners (e.g., classifiers). Here, the distribution that we are interested in capturing and characterizing by its MI value is the joint distribution of a document containing a given term and being locally held by a given peer. For a fixed peer and a fixed term this is:

$$\sum_{x\in\{0,1\},y\in\{0,1\}} P[X=x \wedge Y=y] log_2 \frac{P[X=x \wedge Y=y]}{P[X=x]P[Y=y]}$$

with binary random variables X and Y denoting that a document contains the term ($X = 1$) or not ($X = 0$) and that a document is in the peer's local index ($Y = 1$) or not ($Y = 0$).

Assume that the total number of documents in the network is gN, the number of documents in the peer's local index is lN (contained in gN), the *local df* of the term is $ldf \leq lN$, and the *global df* of the term is $gdf \leq gN$. Then we estimate:

$$P[X=1 \wedge Y=1] = \frac{ldf}{gN}$$

$$P[X=1 \wedge Y=0] = \frac{gdf-ldf}{gN}$$

$$P[X=0 \wedge Y=1] = \frac{lN-ldf}{gN}$$

$$P[X=0 \wedge Y=0] = \frac{gN-lN-(gdf-ldf)}{gN}$$

$$P[X=1] = \frac{gdf}{gN} \quad P[Y=1] = \frac{lN}{gN}$$

---

[3] Google as of Nov 18: 1,760,000,000 vs. 24,800,000

For computing the MI values, a peer looks up the estimated *gdf* values and the estimate of gN in the P2P directory (or uses its locally cached estimated from the last dissemination) and then carries out the simple calculation above. Conceptually this is done for each term, but in practice the peer can easily prune many terms if their ratio of local to global *df* values is low. The remaining terms are then ranked by their MI values in descending order, and the peer would use the highest-ranked terms for postings to the P2P directory.

5.4 Noise Reduction

An interesting side effect of peers not publishing metadata for terms with low *local df* values is the reduction of noise in the final query result. Assume a local query execution strategy that ranks local documents using a TFIDF-style scoring function with the collection-wide *local df* values, as in many of today's P2P Web search approaches. As a document's TFIDF score for a particular query depends inversely on the *df* values for the query terms (i.e., the higher the *df* values, the lower a document's score), documents from peers with very few documents for at least one query term receive unjustified high local score values. If the result merging process uses these local scores from different peers, the scores are inherently incomparable, as documents from peers with low *df* values have received high local score values and, thus, are ranked high in the final result list. This effect is even amplified by the fact that (as for commonly used logarithmic dampening) differences for low absolute *df* values have a higher impact on the *idf*-subscore than the same difference for higher values. Thus, peers with very low *df* values tend to place their few results into the top portion of the final result list. If such peers refrain from posting their metadata, they will (depending on the actual query routing strategy) typically no longer be selected to contribute their local results, so that the precision in a top-k result list of a globally executed query (where k denotes the maximum number of query results after result merging) experienced without involving these peers is actually *higher* than the precision if they were involved (measured against a hypothetical centralized collection).

Remember that most of our strategies prevent peers from publishing metadata for a term if the *local df* value is small with respect to the *global df*. Doing so, we can ensure that even (and in particular) for less frequent terms peers *do* publish their metadata, even if their *local df* values are low.

Figure 4 shows a refined rank distance [6] between the query results of the Minerva query execution (without any threshold) versus a reference result of a hypotetical combined collection of all peers as a function of the number of remote peers chosen for some queries. In particular, the peers (as selected by the query routing algorithm) use their *local df* value for TFIDF-style score computation and the result merging is based on these local scores. As expected, the rank distance decreases at first, while some high-quality peers (with high *local df* values) contribute their results. At some point, however, the peers that contribute local results add high scoring results due to their low *local df* values, that unjustifiedly make it into the top portion of the query result. This leads to an increase in rank distance, that is, a decrease in precision. However, it can also be seen that the ideal number of peers to stop varies among the queries and query terms. This supports our quest for a flexible pruning strategy that can effectively cope with these variations.

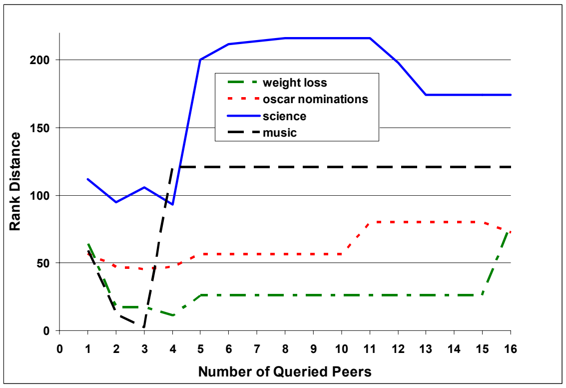

Figure 4. Noise Reduction

6 Experiments

We could not use any standard benchmark setup, as we are not aware of any document corpora with a distribution of documents that matches our vision of peers autonomously crawling the web, as they typically partition the documents *randomly* (as opposed to topically) and *disjointly* (as opposed to experiencing significant replication of popular documents). We have used the BINGO! focused crawler [28] with general purpose web pages (such as *http://cnn.com*) as crawl seeds to create a total of 10 topically focused collections on topics like sports, politics, or arts. From each of the 10 topics, 4 peers where created by picking 75% of the respective documents, ending up with a total of 40 peers. For the query workload we took 17 queries from Google's *Zeitgeist*[4] statistics that fitted our topics, such as *Pamela Anderson*, *national hurricane center*, or *arafat*.

As a baseline for our experiments, we created a combined collection of all 40 peers (253,875 documents, 45,731,219 features [after stemming and stopword removal]). We report on *relative recall* with respect to executing the query on this combined collection.

---

[4] http://www.google.com/press/zeitgeist.html

All experiments were conducted on the Minerva testbed described in Section 3, with peers running on a PC cluster.

6.1 Performance Gains

We measure the total number of Posts in the system as a function of the threshold value. As can be seen in Tables 1 and 2, even small *threshold* values result in a significantly decreased number of Posts for both relative and absolute threshold strategies. The effect is even stronger for absolute thresholds, as a substantial fraction of the number terms contained in a collection only appear once or twice and are artefact terms, e.g., due to typing errors. For such a term, however, a peer easily exceeds a *relative* threshold of 20% of all documents in the network that contain the term (because it *only* occurs at this peer); this is why the relative threshold strategy does not decrease the number of Posts as much as an absolute threshold. This makes a strong case for a hybrid-strategy that combines the strengths of both approaches. Refraining from publishing metadata for these needless terms saves network bandwidth at publishing time and at querying time (when retrieving the PeerLists), and decreases the storage load at the directory peers.

Absolute Threshold	0	5	10	20
Total # of Posts	4,747,517	964,274	651,437	430,080
Percent	100.00%	20.37%	13.72%	9.06%

Table 1. Absolute Threshold

Relative Threshold	0%	5%	10%	20%
Total # of Posts	4,747,517	3,926,424	3,391,943	2,810,033
Percent	100.00%	82.70%	71.45%	59.19%

Table 2. Relative Threshold

6.2 Recall

Figures 5 and 6 show the average relative recall as a function of the number of remote peers chosen in the query routing process for several threshold values. As expected, we experience the best relative recall if no threshold is applied. Increasing the thresholds causes a decrease in recall, due to the unavailability of metadata information, that otherwise would have been available. The (at first sight surprising) effect that an absolute threshold of 20 does hardly influence the recall - in spite resulting in only 9% of all metadata - is due to the two facts that (1) it reduces largely typing errors and other artefacts that are not likely to become query terms, and (2) the query load, taken from Google's *Zeitgeist*, largely consist of highly popular terms, for which an absolute threshold of 20 was mostly exceeded.

For a relative threshold of 5% we can see that the decrease of recall is nearly negligible while the decrease of the global metadata directory is already remarkable (cf. Subsection 6.1). Note at this point that a threshold of 20% for a particular term means that a peer posts metadata if and only if it maintains more than 20% of all globally available documents that contain this term. This is an extremely high threshold that will not be considered in a real-world scenario with thousands of peers.

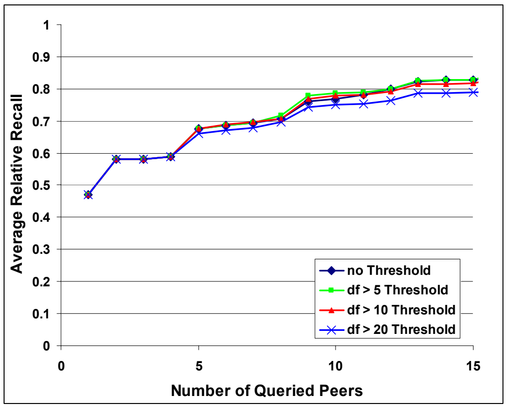

Figure 5. Absolute Threshold

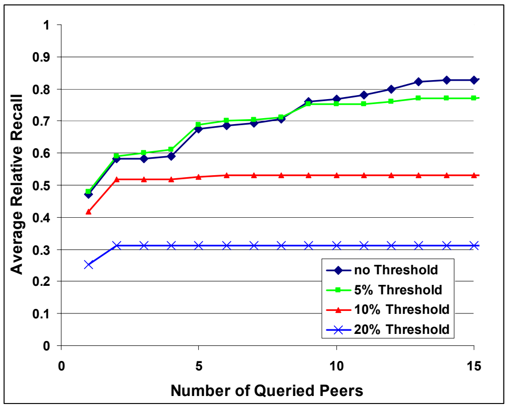

Figure 6. Relative Threshold

7 Conclusion and Outlook

We have described different strategies to limit the amount of metadata that peers publish to the distributed directory. We have experimentally quantified the decrease of network and storage load that can be achieved using these strategies and also examined the impact on the query result quality, utilizing a measure of relative recall versus a combined collection. While suggesting suitable specific threshold values is highly dependent on various system parameters, such as the number of peers and the size of the peers, the experiments underline our initial assumption that making the peers publish only their most discriminative metadata decreases the burden of the metadata directory significantly without sacrificing result quality.

We are currently working on experiments with strategies based on Poisson Mixes and MI, and we are also developing hybrid strategies that combine the respective advantages of all strategies. Additionally, we plan to conduct experiments on substantially larger peer populations in order to further support our results.

References

[1] K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. V. Pelt. Gridvine: Building internet-scale semantic overlay networks. In *International Semantic Web Conference*, pages 107–121, 2004.

[2] K. Aberer, M. Punceva, M. Hauswirth, and R. Schmidt. Improving data access in p2p systems. *IEEE Internet Computing*, 6(1):58–67, 2002.

[3] M. Bender, S. Michel, P. Triantafillou, and G. Weikum. Overlap-aware global *df* estimation in distributed information retrieval systems. In *Submitted for publication*, 2006.

[4] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Improving collection selection with overlap awareness in p2p search engines. In *SIGIR05*, Salvador, Brasil, 2005. ACM.

[5] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Minerva: Collaborative p2p search. In *VLDB*, pages 1263–1266, 2005.

[6] M. Bender, S. Michel, G. Weikum, and C. Zimmer. The MINERVA project: Database selection in the context of P2P search. In *BTW 2005*, 2005.

[7] E. Buchmann and K. Böhm. How to Run Experiments with Large Peer-to-Peer Data Structures. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium, Santa Fe, USA*, Apr. 2004.

[8] J. Callan. Distributed information retrieval. *Advances in information retrieval, Kluwer Academic Publishers.*, pages 127–150, 2000.

[9] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR1995*, pages 21–28. ACM Press, 1995.

[10] K. Church and W. Gale. Poisson mixtures. *Natural Language Engineering, 1(2), 163–190, (1995).*, 1995.

[11] F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen. PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. Technical Report DCS-TR-487, Rutgers University, 2002.

[12] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.

[13] N. Fuhr. A decision-theoretic approach to database selection in networked IR. *ACM Transactions on Information Systems*, 17(3):229–249, 1999.

[14] L. Gravano, H. Garcia-Molina, and A. Tomasic. Gloss: text-source discovery over the internet. *ACM Trans. Database Syst.*, 24(2):229–264, 1999.

[15] T. Hernandez and S. Kambhampati. Improving text collection selection with coverage and overlap statistics. In *Poster at WWW*, 2005.

[16] S. Idreos, M. Koubarakis, and C. Tryfonopoulos. P2p-diet: An extensible p2p service that unifies ad-hoc and continuous querying in super-peer networks. In *SIGMOD*, 2004.

[17] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *ACM Symposium on Theory of Computing*, pages 654–663, May 1997.

[18] L. S. Larkey, M. E. Connell, and J. P. Callan. Collection selection and results merging with topically organized u.s. patents and TREC data. In *CIKM 2000*.

[19] J. Li, B. Loo, J. Hellerstein, F. Kaashoek, D. Karger, and R. Morris. On the feasibility of peer-to-peer web indexing and search. In *IPTPS2003*, 2003.

[20] S. Michel, M. Bender, P. Triantafillou, and G. Weikum. IQN routing: Integrating quality and novelty in p2p querying and ranking. In *EDBT*, 2006.

[21] S. Michel, P. Triantafillou, and G. Weikum. KLEE: A framework for distributed top-k query algorithms. In *VLDB*, 2005.

[22] H. Nottelmann, G. Fischer, A. Titarenko, and A. Nurzenski. An integrated approach for searching and browsing in heterogeneous peer-to-peer networks. In *HDIR 2005*.

[23] H. Nottelmann and N. Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. In *SIGIR*, pages 290–297. ACM Press, 2003.

[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *SIGCOMM*, 2001.

[25] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR*, pages 232–241, 1994.

[26] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001.

[27] L. Si, R. Jin, J. Callan, and P. Ogilvie. A language modeling framework for resource selection and results merging. In *Proceedings of CIKM02*, pages 391–397. ACM Press, 2002.

[28] S. Sizov, M. Biwer, J. Graupmann, S. Siersdorfer, M. Theobald, G. Weikum, and P. Zimmer. The bingo! system for information portal generation and expert web search. *CIDR*, 2003.

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

[30] T. Suel, C. Mathur, J. Wu, J. Zhang, A. Delis, M. Kharrazi, X. Long, and K. Shanmugasunderam. Odissea: A peer-to-peer architecture for scalable web search and information retrieval. Technical report, Polytechnic Univ., 2003.

[31] Y. Wang, L. Galanis, and D. J. de Witt. Galanx: An efficient peer-to-peer search engine system. *Available at http://www.cs.wisc.edu/ yuanwang*.