Fachrichtung 6.2 – Informatik
Naturwissenschaftlich-Technische Fakultät I
Universität des Saarlandes

Max-Planck-Institut für Informatik, Saarbrücken
AG 5 Databases and Information Systems Group
Prof. Dr.-Ing. Gerhard Weikum

On the Usage of Global Document Occurrences in Peer-to-Peer Information Systems

Odysseas Papapetrou

Master thesis in Computer Science at the University of Saarland, Saarbrücken

October 2005

I hereby certify that the work contained in this Master thesis is my own work, unless explicitly mentioned and cited.

Odysseas Papapetrou
Saarbrücken, October 2005

Acknowledgments

I would like to thank Prof. Dr. Gerhard Weikum, for his guidance in the completion of this research. Special thanks belong to my tutors, Sebastian Michel and Matthias Bender. Working with them was a wonderful experience. Furthermore, I would like to thank Kerstin Meyer Ross, the IMPRS coordinator, as well as the IMPRS institution, providing me the stipend for my M.Sc. degree. Last, but not least I would like to thank my wife, Katerina Ioannou, for the moral support and advices.

Note: A preliminary version of this work is submitted for publication in the proceedings of the 7th International Conference on Cooperative Information Systems (CoopIS'05). The work is co-authored with Sebastian Michel, Matthias Bender and Prof. Dr. Gerhard Weikum.

On the Usage of Global Document Occurrences in Peer-to-Peer Information Systems

Summary

There exist a number of approaches for query processing in Peer-to-Peer information systems that efficiently retrieve relevant information from distributed peers. However, very few of them take into consideration the overlap between peers. As the most popular resources (e.g., documents or files) are often present at most of the peers, a large fraction of the documents eventually received by the query initiator are duplicates.

In this work we develop a technique based on the notion of *global document occurrences* (GDO), that, when processing a query, penalizes frequent documents increasingly as more and more peers contribute their local results. We argue that the additional effort to create and maintain the GDO information is reasonably low, as the necessary information can be piggybacked onto the existing communication. Our experiments indicate that our approach significantly decreases the number of peers that have to be involved in a query to reach a certain level of recall and, thus, decreases user-perceived latency and the wastage of network resources.

1 Introduction

The peer-to-peer (P2P) approach, which has become popular in the context of file-sharing systems such as Gnutella or KaZaA, allows handling huge amounts of data in a distributed and self-organizing way. In such a system, all peers are equal and all of the functionality is shared among all peers so that there is no single point of failure and the load is evenly balanced across a large number of peers. These characteristics offer enormous potential benefits for search capabilities in terms of scalability, efficiency, and resilience to failures and dynamics. Additionally, such a search engine can potentially benefit from the intellectual input (e.g., bookmarks, query logs, etc.) of a large user community.

One of the key difficulties, however, is to efficiently select promising peers for a particular information need. While there exist a number of strategies to tackle this problem, most of them ignore the fact that popular documents are typically present at a reasonable fraction of peers. In fact, experiments show that often promising peers are selected because they share the same high-quality documents. For instance, consider a query for all songs by a famous artist like Madonna. If, as in many of today's systems, every selected peer contributes its best matches only, you will most likely end up with many duplicates of popular and recent songs, when instead you would have been interested in a bigger variety of songs. The same scenario holds true in an information retrieval context where returning only the k best matches for a query is even more common. Popular documents then are uselessly contributed as query results by each selected peer, wasting precious local resources and disqualifying other relevant documents that eventually might not be returned at all. The size of the combined result eventually presented to the query initiator (after eliminating those duplicates), thus, is unnecessarily small.

We propose a technique based on the notion of *Global Document Occurrences* (GDO) that, when processing a query, penalizes frequent documents increasingly as more and more peers contribute their local results. The same approach can also be used prior to the query execution, when selecting promising peers for a query. We discuss the additional effort to create and maintain the GDO information and present early experiments indicating that our approach significantly decreases the number of peers that have to be involved in a query to reach a certain level of recall. Thus, taking overlap into account when performing query routing is a great step towards the feasibility of distributed P2P search.

This introduction is followed by an overview of related research in the different fields that we touch with our work. Section 3 gives a short introduction on Information Retrieval basics necessary for the remainder of this paper. Section 4 presents the architecture of MINERVA, our distributed P2P search engine that was used for our experiments. Section 5 introduces the notion of GDO and discusses its application at several stages of the querying process. Section 6 illustrates a number of experiments to show the potential of our approach. Section 7 concludes and briefly discusses future research directions.

2 Related work

Recent research on P2P systems, such as Chord [1], CAN [2], Pastry [3], P2P-Net [4], or P-Grid [5] is based on various forms of distributed hash tables (DHTs) and supports mappings from keys, e.g., titles or authors, to locations in a decentralized manner such that routing scales well with the number of peers in the system. Typically, in a network of n nodes, 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 [6].

In recent years, many approaches have been proposed for collection selection in distributed IR, among the most prominent the decision-theoretic framework by [7], the GlOSS method presented in [8], and approaches based on statistical language models [9, 10]. [11] gives an overview of algorithms for distributed IR style result merging and database content discovery. [7] presents a formal decision model for database selection in networked IR. [12] investigates different quality measures for database selection. [13, 14] study scalability issues for a distributed term index. None of the presented techniques incorporates overlap detection into the selection process.

[15] describes a permutation-based technique for efficiently estimating set similarities for informed content delivery. [16] proposes a hash-based synopsis data structure and algorithms to support low-error and high-confident estimates for general set expressions. Bloom [17] describes a data structure for compactly representing a set in order to support membership queries. [18] proposes compressed Bloom filters that improve performance in a distributed environment where network bandwidth is an issue.

[19] describes the use of statistics in ranking data sources with respect to a query. They use probabilistic measures to model overlap and coverage of the mediated data sources, but do not mention how to acquire these statistics. In contrast, in an earlier work [20] we assume these statistics being generated by the participating peers (based on their local collections) and present a DHT based infrastructure to make these statistics globally available.

[21] considers novelty and redundancy detection in a centralized, document-stream based information filtering system. Although the technique presented seems to be applicable in a distributed environment for filtering the documents at the querying peer, it is not obvious where to get these documents from. In a large-scale system, it seems impossible to query all peers and to process the documents.

[22, 23] also worked on overlap statistics in the context of collection selection. They present a technique to estimate coverage and overlap statistics by query classification and data mining and use a probing technique to extract features from the collections. Expecting that data mining techniques will be very heavy

for the envisioned, highly-dynamic application environment, we adopt a different philosophy.

In a prior work [24] we propose a Bloom filter based technique to estimate the mutual collection overlap. While there we use Bloom filters to estimate the mutual overlap between peers, we now use the number of global document occurrences of the documents in a collection to estimate the contribution of this collection to a particular query. These approaches can be seen as orthogonal and can eventually be combined to form even more powerful systems.

3 Information Retrieval Basics

Information Retrieval (IR) systems keep large amounts of unstructured or weakly structured data, such as text documents or HTML pages, and offer search functionalities for delivering documents relevant to a query. Typical examples of IR systems include web search engines or digital libraries; in the recent past, relational database systems are integrating IR functionality as well.

The search functionality is typically accomplished by introducing measures of similarity between the query and the documents. For text-based IR with keyword queries, the similarity function typically takes into account the number of occurrences and relative positions of each query term in a document. Section 3.1 explains the concept of *inverted index lists* that support an efficient query execution and section 3.2 introduces one of the most popular similarity measures, the so-called *TF*IDF* measure. For further reading, we refer the reader to [6, 25].

3.1 Inverted Index Lists

The concept of inverted index lists has been developed in order to efficiently identify those documents in the dataset that contain a specific query term. For this purpose, all terms that appear in the collection form a tree-like index structure (often a b^+-tree or a trie) where the leafs contain a list of unique document identifiers for all documents that contain this term (Figure 1). Conceptually, these lists are combined by intersection or union for all query terms to find candidate documents for a specific query. Depending on the exact query execution strategy, the lists of document identifiers may be ordered according to the document identifiers or according to a score value to allow efficient pruning.

3.2 $TF * IDF$ Measure

The number of occurrences of a term t in a document d is called *term frequency* and typically denoted as $tf_{t,d}$. Intuitively, the significance of a document increases with the number of occurrences of a query term. The number of documents in a collection that contain a term t is called *document frequency* (df_t); the *inverse document frequency* (idf_t) is defined as the inverse of df_t. Intuitively, the relative importance of a query term decreases as the number of documents that

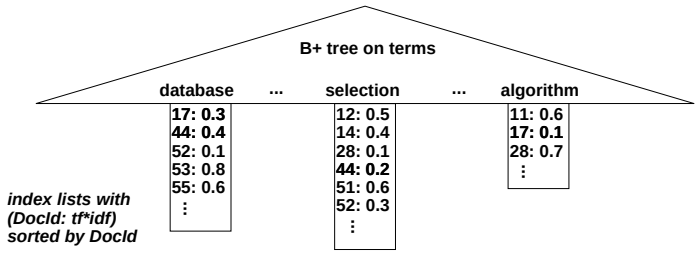

Fig. 1. B+ Tree of Inverted Index Lists

contain this term increases, i.e., the term offers less differentiation between the documents. In practice, these two measures may be normalized (e.g., to values between 0 and 1) and dampened using logarithms. A typical representative of this family of $tf * idf$ formulas that calculates the weight $w_{i,f}$ of the i-th term in the j-th document is

$$w_{i,j} := \frac{tf_{i,j}}{max_t\{tf_{t,j}\}} * log(\frac{N}{df_i})$$

where N is the total number of documents in the collection.

In recent years, other relevance measures based on statistical language models and probabilistic IR have received wide attention [7, 26]. For simplicity and because our focus is on P2P distributed search, we use the still most popular $tf * idf$ scoring family in this paper.

4 MINERVA

We briefly introduce MINERVA[1], a fully operational distributed search engine that we have implemented and that serves as a valuable testbed for our work[20, 27]. We assume a P2P collaboration in which every peer is autonomous and has 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 lists with URLs for Web pages that contain specific keywords.

A conceptually global but physically distributed directory, which is layered on top of a Chord-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. We only use the most basic DHT functionality, *lookup(key)*, that returns the peer currently responsible for *key*. Doing so, we partition the term space, such that every peer is responsible for a randomized subset of terms within the global directory. For failure resilience and availability, the entry for a term may be replicated across multiple peers.

Directory maintenance, query routing, and query execution work as follows (cf. Figure 2). In a preliminary step (step 0), every peer publishes a summary

[1] Project homepage available at http://www.minerva-project.org

(*Post*) about every term in its local index to the directory. A hash function is applied to the term in order to determine the peer currently responsible for this 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 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). These statistics are used to support the query routing process, i.e., determining the most promising peers for a particular query.

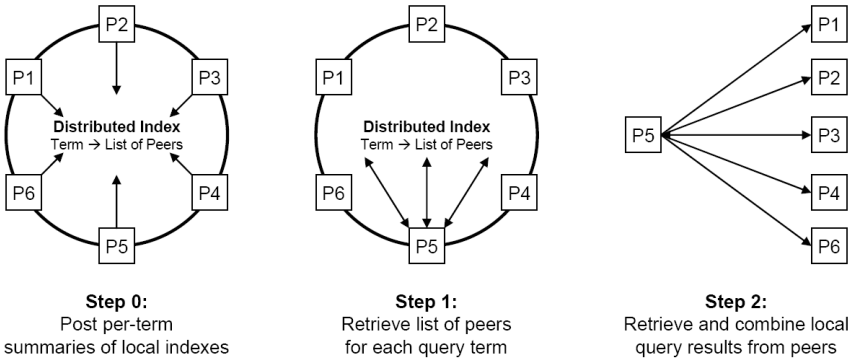

Fig. 2. MINERVA System Architecture

The querying process for a multi-term query proceeds as follows: a query is executed locally using the peer's local index. If the result is considered unsatisfactory by the user, the querying peer retrieves a list of potentially useful peers by issuing a *PeerList request* for each query term to the underlying overlay-network directory (step 1). Using database selection methods from distributed IR and metasearch [11], a number of promising peers for the complete query is computed from these PeerLists. This step is referred to as *query routing*. Subsequently, the query is forwarded to these peers and executed based on their local indexes (*query execution*; step 2). Note that this communication is done in a pairwise point-to-point manner between the peers, allowing for efficient communication and limiting the load on the global directory. Finally, the results from the various peers are combined at the querying peer into a single result list.

The goal of finding high-quality search results with respect to precision and recall cannot be easily reconciled with the design goal of unlimited scalability, as the best information retrieval techniques for query execution rely on large amounts of document metadata. Posting only compact, aggregated information about local indexes and using appropriate query routing methods to limit the number of peers involved in a query keeps the size of the global directory manageable and reduces network traffic, while at the same time allowing the query execution itself to rely on comprehensive local index data. We expect this ap-

proach to scale very well as more and more peers jointly maintain the moderately growing global directory.

The approach can easily be extended in a way that multiple distributed directories are created to store information beyond local index summaries, such as information about local bookmarks, information about relevance assessments (e.g., derived from peer-specific query logs or click streams), or explicit user feedback. This information could be leveraged when executing a query to further enhance result quality.

4.1 Query Routing

Database selection has been a research topic for many years, e.g. in distributed IR and metasearch [11]. Typically, the expected result quality of a collection is estimated using precomputed statistics, and the collections are ranked accordingly. Most of these approaches, however, are not directly applicable in a true P2P environment, as

- the number of peers in the system is substantially higher (10^x peers as opposed to 10-20 databases)
- the system evolves dynamically, i.e. peers enter or leave the system autonomously at their own discretion at a potentially high rate
- the results from remote peers should not only be of high quality, but also complementary to the results previously obtained from one's local search engine or other remote peers

In [20, 28], we have adopted a number of popular existing approaches to fit the requirements of such an environment and conducted extensive experiments in order to evaluate the performance of these naive approaches.

As a second step, we have extended these strategies using estimators of mutual overlap among collections [24] using bloom filters [17]. Preliminary experiments show that such a combination can outperform popular approaches based on quality estimation only, such as CORI [11].

We also want to incorporate the fact that every peer has its own local index, e.g., by using implicit-feedback techniques for automated query expansion (e.g., using the well-known IR technique of pseudo relevance feedback [29] or other techniques based on query logs [30] and click streams [31]). For this purpose, we can benefit from the fact that each peer executes the query locally first, and also the fact that each peer represents an actual user with personal preferences and interests. For example, we want to incorporate local user bookmarks into our query routing [28], as bookmarks represent strong recommendations for specific documents. Queries could be exclusively forwarded to thematically related peers with similarly interested users, to improve the chances of finding *subjectively* relevant pages.

Ultimately, we want to introduce a sophisticated $benefit/cost$ ratio when selecting remote peers for query forwarding. For the benefit estimation, it is intuitive to consider such measures as described in this section. Defining a meaningful

cost measure, however, is an even more challenging issue. While there are techniques for observing and inferring network bandwidth or other infrastructural information, expected response times (depending on the current system load) are changing over time. One approach is to create a distributed Quality-of-Service directory that, for example, holds moving averages of recent peer response times.

4.2 Query Execution

Query execution based on local index lists has been an intensive field of research for many years in information retrieval. A good algorithm should avoid reading inverted index lists completely, but limit the effort to $O(k)$ where k is the number of desired results. In the IR and multimedia-search literature, various algorithms have been proposed to accomplish this. The best known general-purpose method for top-k queries is Fagin's threshold algorithm (TA) [32], which has been independently proposed also by Nepal et al. [33] and Güntzer et al. [34]. It uses index lists that are sorted in descending order of term scores under the additional assumption that the final score for a document is calculated using a monotone aggregation function (such as a simple sum function). TA traverses all inverted index lists in a round-robin manner, i.e., lists are mainly traversed using sorted accesses. For every new document d encountered, TA uses random accesses to calculate the final score for d and keeps this information a in document candidate set. Since TA additionally keeps track of a higher bound for documents not yet encountered, the algorithm terminates as soon as this bound assures that no unseen document can enter the candidate set. Probabilistic methods have been studied in [35] that can further improve the efficiency of index processing.

As our focus is on the distributed aspect of query processing, we will not focus on query execution in this paper. Our approaches to be introduced in the upcoming sections are orthogonal to this issue and can be applied to virtually any query execution strategy.

5 Global Document Occurrences (GDO)

We define the *global document occurrence* of a document d ($GDO(d)$ for short) as the number of peers that contain d, i.e., as the number of occurrences of d within the network. This is substantially different from the notion of *global document frequency* of a term t (which is the number of documents that contain t) and from the notion of *collection frequency* (which is typically defined as the number of collections that contain documents that contain t).

The intuition behind using GDO when processing a query is the fact that GDO can be used to efficiently estimate the probability that a peer contains a certain document and, thus, the probability that a document is contained in at least one of a set of peers. Please note the obvious similarity to the $TF * IDF$ measure, which weights the relative importance of a query term t using the number of documents that contain t as an estimation of the popularity of t, favoring rare terms over popular (and, thus, less distinctive and discriminative)

terms. Similarly, the GDO approach weights the relative popularity of a document within the union of all collections. If a document is highly popular (i.e., occurs in most of the peers), it is considered less important both when selecting promising peers (query routing) and when locally executing the query (query execution). In contrast, rare documents receive a higher relative importance.

5.1 Mathematical Reasoning

The proposed approach will get clearer if we describe the reasoning behind it. Suppose that we are running a single-keyword query, and that each document d in our collection has a precomputed relevance to a term t (noted as $DocumentScore(d,t)$). When searching for the top-k documents, a P2P system would ask some of its peers for documents, which determine the relevant documents locally, and merge the results.

This independent document selection has the disadvantage that it does not consider overlapping results. For example, one relevant document might be so common, that every peer returns it as result. This reduces the recall for a query, as the document is redundant for all but the first peer. In fact, massive document replication is common in real P2P systems, so duplicate results frequently occur. This effect can be described with a mathematical model, which can be used to improve document retrieval.

Assuming a uniform distribution of documents among the peers, the probability that a given peer has a certain document d can be estimated by

$$P_H(d) = \frac{GDO(d)}{\#\text{peers}}.$$

Now consider a sequence of peers $< p_1, \ldots, p_\lambda >$. The probability that a given document d held by p_λ is fresh, i.e. not already occurs in one of the previous peers, can be estimated by

$$P_F^\lambda(d) = (1 - P_H(d))^{\lambda-1}.$$

This probability can now be used to re-evaluate the relevance of documents: If it is likely that a previously queried peer has already returned a document, the document is no longer relevant. Note that we introduce a slight inaccuracy here; we only used the probability that one of the previously asked peers has a document, not the probability that it has also returned the document. Thus we would be interested in the probability that a document has not been returned before $P_{NR}^\lambda(d)$. However the error introduced is reasonably small: for all documents $P_{NR}^\lambda(d) \geq P_F^\lambda(d)$. For the relevant documents $P_{NR}^\lambda(d) \approx P_F^\lambda(d)$, as the relevant documents will be returned by the peers. Therefore we only underestimate (and, thus, punish) the probability for irrelevant documents, which is not too bad, as the they were irrelevant anyway.

Now this probability can be used to adjust the scores according to the GDO. The straightforward usage would be to discard a document d during retrieval with a probability of $(1 - P_F^\lambda(d))$, but this would produce non-deterministic

behavior. Instead we adjust the DocumentScores of a document d with regard to a term t by aggregating the scores and the probability; for simplicity, we multiply them in our current experiments.

$$DocumentScore'(d,t) = DocumentScore(d,t) * P_F^{\lambda}(d)$$

This formula reduces the scores for frequent documents, which avoids duplicate results. Note that $P_F^{\lambda}(document)$ decreases with λ, thus frequent documents are still returned by peers asked early, but discarded by the subsequent peers.

5.2 Apply GDO to Query Routing

In most of the existing approaches to query routing, the quality of a peer is estimated using per-term statistics about the documents that are contained in its collection. Popular approaches include counting the number of documents that contain this term (*document frequency*), or summing up the document scores for all these documents (*score mass*). These term-specific scores are combined to form an aggregated *PeerScore* with regard to a specific query. The peers are ordered according to their PeerScore to form a peer *ranking* that determines an order in which the peers will be queried.

The key insight of our approach to tackle the problem of retrieving duplicate documents seems obvious: the probability of a certain document being contained in at least one of the involved peers increases with the number of involved peers. Additionally, the more popular the document, the higher the probability that it is contained in one of the first peers to contribute to a query. Thus, the impact of such documents to the PeerScore should decrease as the number of involved peers increases.

If a candidate peer in the ranking contains a large fraction of popular documents, it would be increasingly unwise to query this peer at later stages of the ranking, as the peer might not have any fresh (i.e., previously unseen) documents to offer. In contrast, if no peers have been queried yet, then a peer should not be punished for containing popular documents, as we certainly *do* want to retrieve those documents. We suggest an extension that is applicable to almost all popular query routing strategies and calculates the PeerScore of a peer depending on its position in the peer ranking.

For this purpose, we modify the score of each document in a collection with different biases, one for each position in a peer ranking[2]. In other words, there is no longer only *one* DocumentScore for each document, but rather several DocumentScores corresponding to the potential ranks in a peer ranking. Remember from the previous section, that the DocumentScore of a document d with regard to term t is calculated using the following formula:

$$DocumentScore'(d,t,\lambda) = DocumentScore(d,t) * P_F^{\lambda}(d)$$

[2] Please note that, for techniques that simply count the number of documents, the scores of all relevant documents are initially set to 1.

where λ is the position in the peer ranking (i.e., the number of peers that have already contributed to the query before), and $P_F^{\lambda}(d)$ is the probability that this document is not contained in any of the previously contributing collections.

From this set of DocumentScores, each peer now calculates *separate* term-specific scores (i.e., the scores that serve as subscores when calculating PeerScores in the process of Query Routing) corresponding to the different positions in a peer ranking by combining the respectively biased document scores. In the simplest case where the PeerScore was previously calculated by summing up the scores for all relevant documents, this means that now one of these sums is calculated for every rank λ:

$$score(p, t, \lambda) = \sum_{d \in D_p} DocumentScore'(d, t, \lambda)$$

where D_p denotes the document collection of p. Instead of including only one score in each term-specific post, now a *list* of the term-specific peer scores $score(p, t, \lambda)$ is included in the statistics that is published to the distributed directory. Figure 3 shows some extended statistics for a particular term. The numbers shown in the boxes left to the scores represent the respective ranks in a peer ranking. Please note that the term-specific score of a peer decreases as the document scores for its popular documents decrease with the ranking position. Previous experiments have shown that typically involving only 2-3 peers in a query already yields a reasonable recall; we only calculate $score(p, t, \lambda)$ for $\lambda \leq 10$ [20] as we consider asking more than 10 peers very rare and not compatible with our goal of system scalability. The calculation itself of this magnitude of DocumentScores is negligible.

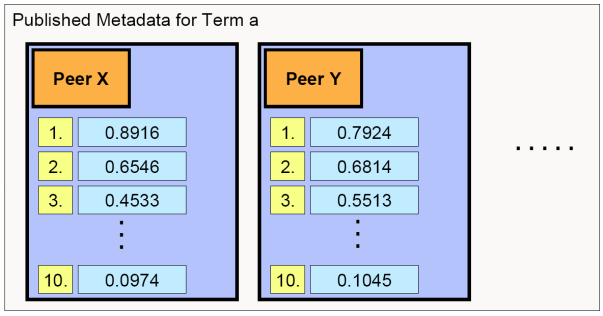

Fig. 3. Extended Term-specific scores for different ranking positions

Please also note that this process does not require the selected peers to locally execute the queries sequentially, but it allows for the parallel query execution of all peers involved: after identifying the desired number of peers and their ranks in the peer ranking, the query initiator can contact all other peers simultaneously and include their respective ranks in the communication. Thus, the modification of the standard approach using GDOs does not cause additional latencies.

The additional network resource consumption needed for our proposed approach is relatively small if the GDO distributed directory is conducted in a clever manner. Instead of distributing the GDO counters across the peers using random hashing on unique document identifiers, we propose to maintain the counters at peers that are responsible for a representative term within the document, (e.g., the first term or the most frequent term). Doing so, the peers can easily piggyback the GDO-related communication when publishing the Posts and, in turn, they can immediately receive the current GDO values for the same documents. The GDO values are then cached locally and used to update the local DocumentScores, that will eventually be used when publishing our Posts again. The Posts themselves become slightly larger as more than one score values are now included in each Post; but this typically fits within the existing network message avoiding extra communication.

5.3 Apply GDO to Query Execution

The peers that have been selected during query routing can additionally use GDO-based biases to penalize popular documents during their local query execution. The later a peer is involved in the processing of a query, the higher punishing impact this GDO-based bias should have as popular documents are likely to be returned from prior peers. For this purpose, each peer re-weights the DocumentScores obtained by its local query execution with the GDO-values for the documents.

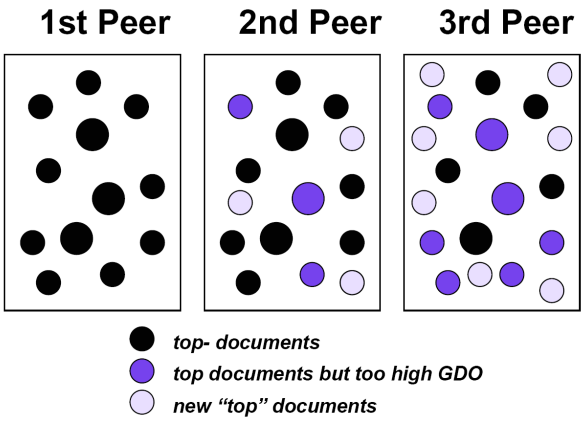

Fig. 4. The impact of GDO-enhanced query execution.

Figure 4 shows the impact of the GDO-based local query execution.

The additional cost implied by our approach within the query execution step is negligible. As the GDO values are cached locally as described in a previous section, the DocumentScores can easily be adjusted on-line using a small number

of basic arithmetic operations. Alternatively, all the position-dependent GDO-based document scores can be pre-calculated and cached locally, as in the case of the GDO values. Either of the approaches is inexpensive and feasible.

5.4 Maintaining the GDO values

The approach introduced above builds on top of a directory that globally counts the number of occurrences or each document. When a new peer joins the network, it updates the GDO values for all its documents (i.e., increment the respective counters) and retrieves the GDO values for the computation of its biased scores at a low extra cost. Similarly, before a peer leaves the network, it reduces the GDO values for all its documents.

We propose the usage of the existing distributed DHT-based directory to maintain the GDO values in a scalable way. In a naive approach, the document space is partitioned across all peers using globally unique document identifiers, e.g., by applying a hash function to their URLs and maintaining the counter at the DHT peer that is responsible for this identifier (analogously to the term-specific statistics that are maintained independently in parallel). This naive approach requires two messages for each document per peer (one when the peer enters and one when the peer leaves the network), which results to $O(n)$ messages for the whole system, where n is the number of document instances.

In an effort to reduce the number of messages required for maintaining the distributed GDO directory, we change the hashing function used in distributing the GDO counters. For each document, we maintain its GDO at the peer that is responsible for a representative term within the document, (e.g., the first term or the most frequent term). We can then easily piggyback the GDO-related communication at the messages created when publishing the Posts; they will both have the same recipient peer. In turn, the response message can include the current GDO values for the same documents from the distributed directory. The GDO values are then cached locally and used to update the local DocumentScores, that will eventually be used when publishing our Posts again. The Posts themselves become slightly larger as more than one score value is now included in a Post; however this typically fits within the existing network message avoiding extra communication.

The latter approach almost completely avoids additional messages. In fact, when a peer enters the network, no additional messages are required for the GDO maintenance, as all messages are piggybacked in the process of publishing Post objects to the directory. Most importantly, there is no extra overhead in running the $Peer - lookup$ function at the DHT for finding the responsible peer for each GDO counter; the responsible peers for each document are already discovered from the process of publishing the Posts.

To cope with the dynamics of a Peer-to-Peer system, in which peers join and leave the system autonomously and without prior notice, we go one step further and propose the following technique. Each object in the global directory is assigned a TTL (time-to-live) value, after which it is discarded by the maintaining peer. In turn, each peer is required to re-send its information periodically. This

fits perfectly with our local caching of GDO values, as these values can be used when updating the Post objects. This update process, in turn, again updates the local GDO values.

6 Experiments

6.1 Benchmarks

We have generated two synthetic benchmarks. The first benchmark includes 500 peers and 10000 documents, while the second benchmark consists of 1000 peers and 10000 documents. In both the benchmarks, the 10000 documents are created by randomly assigning 100 terms to them, so that each document gets exactly 4 terms. The term-specific scores for the documents follow a Zipf[36] distribution ($\alpha = 0.8$). The assumption that the document scores follow Zipf's law is widely accepted in information retrieval literature.

The *document replication* follows a Zipf distribution too ($\alpha = 0.8$). This means that most documents are assigned to a very small number of peers (i.e., have a low GDO value) and only very few documents are assigned to a large number of peers (i.e., have a high GDO value). Please note that, although the GDOs and the document scores of the documents are both following a Zipf distribution, the two distributions are not connected. This means that we do not expect a document with a very high importance for one term to be also highly replicated, or the other way around. We do not believe that this would create real-world document collections as we know from personal experiences that the most popular documents are not necessarily the most relevant documents for any possible relevant query.

6.2 Evaluated Strategies

In our experimental evaluation, we compare 7 different strategies. All strategies consist of the query routing part and the query execution part. For query routing, our baseline algorithm for calculating the PeerScore of a peer p works as follows:

- $score(p,t) = \sum_{d \in D_p} DocumentScore(d,t)$, i.e., the (unbiased) score mass of all relevant documents in p's collection D_p
- $PeerScore(p,q) = \sum_{t \in q} score(p,t)$, i.e., the sum over all term-specific scores for all terms t contained in the query q

For the query execution part, the synthetically created DocumentScores were derived by summing up the (synthetically assigned) term-specific scores described above. The top-20 documents for the query were detected, and returned to the query initiator.

At both stages, query routing and query execution, we had to choose between a standard (non-GDO) approach or our GDO-enhanced approach, yielding a total of four strategies: (a) the baseline (GDO-free) approach, (b) GDO-based query routing, but normal query execution, (c) GDO-based query execution but

normal query routing, and (d) the full power GDO-based approach, where the GDO biases both query routing and query execution. The GDO values were provided to each strategy using global knowledge of our data.

In the evaluation we also include a greedy near-optimal algorithm. This algorithm in each step queries the most promising peer, acquires the results, and then broadcasts them to all the peers, so that they are not used in the query routing or query execution again for the same query. Then, all the peers re-evaluate themselves for the query, to calculate their new *PeerScore* for that query. The performance of this algorithm serves as a rough indication of the upper bounds of the performance of any distributed query processing algorithm. While the algorithm is straight-forward in implementation, it has practical difficulties for real-life usage, such as an overwhelming network usage and an increased delay due to its serialized nature (no two peers can be queried at the same time). It should be clear to the reader that this approach does not yield optimal results; there are cases where a cleverer (a more modest) selection of peers can result to more unique documents. Yet, the results produced are approaching the optimal results, which could only be found with exponential cost.

In addition, we employ two other strategies that use a Mod-κ sampling-based query execution technique to return fresh documents: In the query routing and query execution process, the peers consider and return only documents with $(DocumentId \ mod \ \kappa) = \lambda$. κ is typically equal to the total number of peers that are going to be queried (i.e. top-10), and λ is the number of peers that have already been queried. In the case that the peer does not have enough documents to complete the required number of documents to a query (in the query execution step only), it also includes documents that do not satisfy the equation $(DocumentId \ mod \ \kappa) = \lambda$, ordered descending in their DocumentScore for the query. In our tests, we experiment with $\kappa = 5$ and $\kappa = 10$.

6.3 Evaluation Methodology

We run several queries (a total of 20) using the seven strategies introduced above. Our queries have from 1 to 4 randomly selected keywords (average 2.5 keywords per query). In each case, we send the query to the top-10 peers suggested by each approach, and collect the local top-20 documents from each peer. Additionally, we run the queries on a combined collection of all peers to retrieve the global top-100 documents that serves as a baseline for our strategies.

We use the following metrics to assess the quality of each strategy:

- the number of *distinct* retrieved documents, i.e., after eliminating duplicates
- the score mass (for the query) of all the retrieved distinct documents[3]
- the number of distinct retrieved top-100 documents
- the score mass (for the query) of the retrieved distinct top-100 documents
- the number of replicated documents retrieved from each approach (the first occurrence of a multiply-returned document is not counted as a replica)

[3] Note that, by our experimental design, the same document is assigned the same score at different peers.

6.4 Results

The experiments are conducted on both the benchmark collections. The GDO-enhanced strategies show significant performance gains. In all our measures, the full power GDO-based approach performs significantly better than the baseline approach. In fact, it approaches the near-optimal results, obtained from the greedy algorithm.

Figures 5.a and 5.b show the number of distinct documents retrieved from each approach, in the 500-peers and the 1000-peers setup respectively. As expected, the full power GDO-based approach (when GDO used for both query routing and query execution) performs significantly better than the GDO-free approach. It returns more than double relevant documents than its GDO-free counterpart. Even disabling the GDO-based enhancement in either query routing or query execution, the approach is still significantly better than the GDO-free approach. Not surprisingly, Mod-5 and Mod-10 approaches are very keen in returning fresh documents; they are very effective in avoiding replicas. They outperform all the other approaches except the non-implementable greedy approach. However, the documents returned from the Mod-κ approaches are of very low document score with our query - see Figure 6.

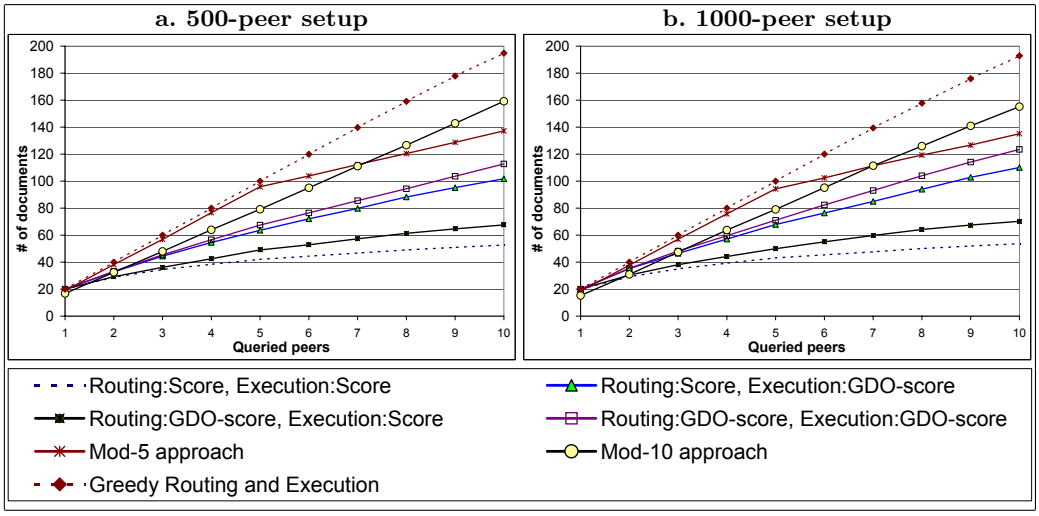

Fig. 5. Number of retrieved relevant documents

Figure 6 compares the aggregated score masses for the retrieved documents in each approach. Again, the full power GDO-based approach performs significantly better than the GDO-free approach, returning documents with over 33% more score mass. Applying GDO in only one of the two steps again has a significant contribution in the performance. The Mod-5 and Mod-10 approaches are now performing worse than the full power GDO-based approach. Even more

interesting, combining the score mass with the number of relevant documents returned from each approach, we realize that the documents originally returned from the Mod-5 and Mod-10 approaches had a moderate-to-low relevance score for our query; the average document score, even at the first most promising peer was below 0.15 with the Mod-κ approaches, while the respective average score in the baseline and GDO approaches was twice as much. This indicates that the retrieved documents from the Mod-κ approaches were only slightly relevant; yet, they were counted as relevant from our evaluation, thus increasing the number of returned relevant documents (Figure 5).

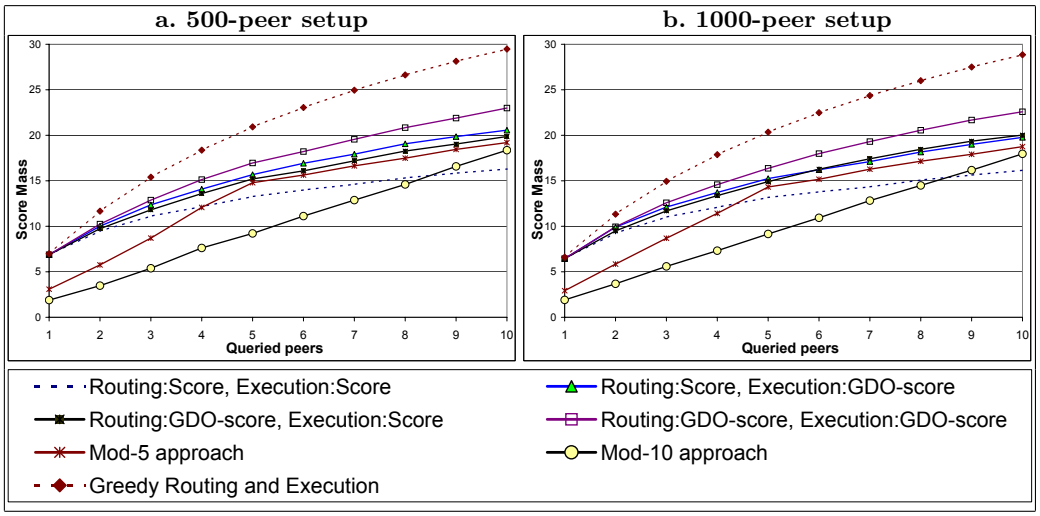

Fig. 6. Score mass of retrieved relevant documents

Figure 7, comparing the number of the top-100 returned documents at each approach, also yields interesting results. The number of the top-100 returned documents from the full power GDO-based approach was about 10% better than the GDO-free approach. An interesting observation was that the GDO-based query execution results in less top-100 documents, compared to the GDO-free counterpart; thus, the best approach for retrieving top-100 documents appears to be when employing the GDO-based scores only for query routing. We expect that this behaviour is due to some very frequent *low-ranked* top-100 documents[4]. These documents get replaced from some less popular documents, slightly less relevant, which do not belong in the top-100 documents. While the loss is of insignificant practical value, due to the hard-line approach used for selecting the top-100 documents, the results show a noticeable difference for the top-100 related measures. It is important to note that the same behaviour occurs in the

[4] The reader is reminded that the top-100 documents are selected using their relevance rank with the query from the global document collection

1000-peer setup, for the same reason. Finally, the top-100 documents returned from the Mod-κ approaches were very few, a lot worse than the original GDO-free approach. The same conclusions are obtained from analyzing the aggregate score mass for the top-100 returned documents (Figure 8).

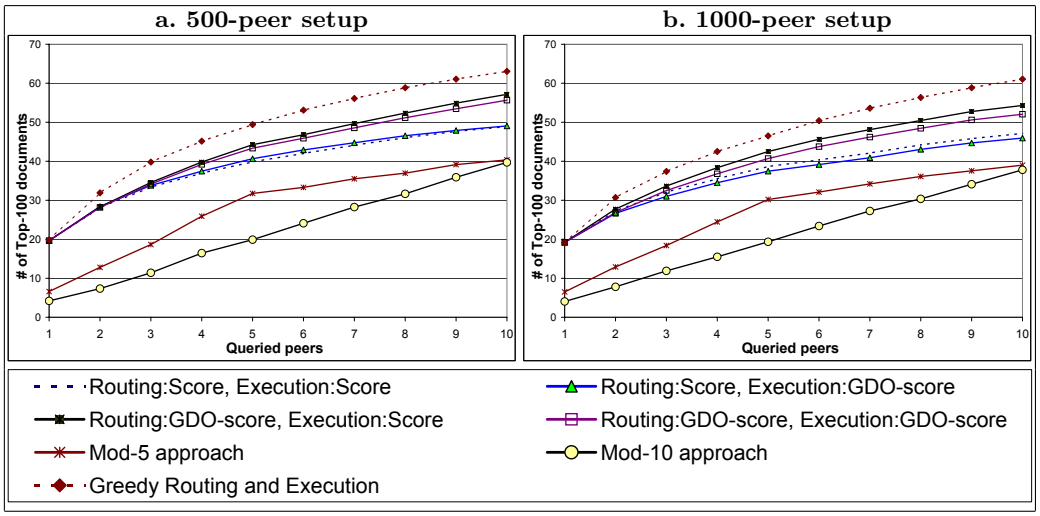

Fig. 7. Number of retrieved top-100 documents with regard to the number of queried peers

We also compare the number of the replicated documents in each approach(Figure 9); these are the documents returned to the query initiator more than once for the same query (the first occurrence of such a document is not counted as a replica). This measure should be as low as possible; ideally 0. Since multiple replicas of the same document do not contribute on the quality of the answer, we want them to be replaced from other *unseen* relevant documents. The full power GDO-based version detects and avoids half of the replicas occurring in the baseline (GDO-free) approach. It is also notable that even the GDO-based query routing as well as the GDO-based query execution alone can positively affect the performance; the former by proposing peers with mostly novel results, and the latter by proposing mostly novel documents from the selected peers. However, as expected, even the full power GDO-based version is not capable in detecting and avoiding *all* the replications. It is in fact a lot worse than the greedy algorithm, whose results resemble the optimal ones (completely avoid replications). Not surprisingly, the Mod-k approaches can also avoid all the replications in the first k peers (yet, not without a sacrifice in the quality of the retrieved documents).

Note that in our experiments we were retrieving 20 documents from each peer, for the top-10 peers, which resulted to a total of 200 documents for each query. However, there were some single-keyword queries for which the actual distribution of the relevant documents did not permit the retrieval of as much as

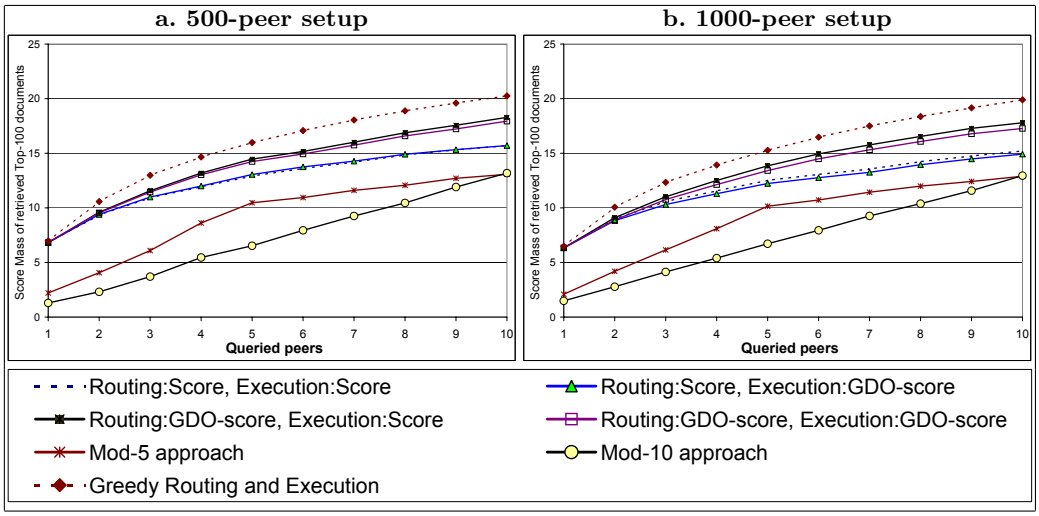

Fig. 8. Score Mass of retrieved top-100 documents with regard to the number of queried peers

200 distinct documents by asking only 10 peers. In these cases, the peers returned only the relevant documents they had, according to their approach, which were fewer than 20. Thus, the number of the distinct relevant documents and the number of the replicated documents cannot be mathematically correlated, and both of them are measured in the experiments independently.

The overall conclusion of the experimental evaluation is that the GDO-based scoring in both query routing and query execution has a significant positive impact in improving the number and the quality of the retrieved documents. Unlike the Mod-κ approaches, it manages to retrieve a large number of unique *yet highly relevant* documents. Compared to the baseline approach, it presents a significant improvement in recall and avoids more than half of the replicated results.

7 Conclusion and Future Work

This work presents an approach towards improving the query processing in Peer-to-Peer Information Systems. The approach is based on the notion of Global Document Occurrences (GDO) and aims at increasing the number of uniquely retrieved high-quality documents without imposing significant additional network load or latency. Our approach can be applied both at the stage of query routing (i.e., when selecting promising peers for a particular query) and when locally executing the query at these selected peers. The additional cost incurred for building and maintaining the required statistical information is small and our approach is expected to scale very well with a growing network. Early ex-

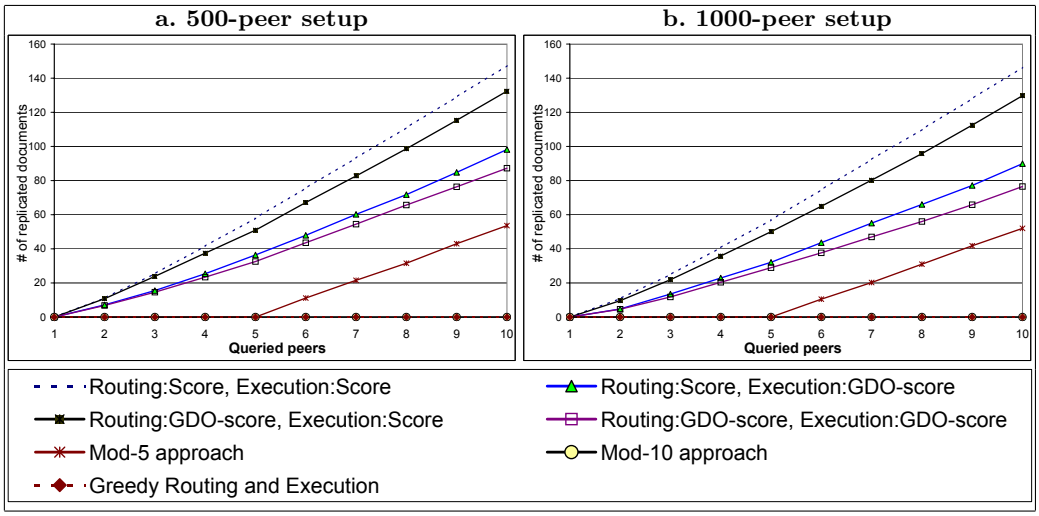

Fig. 9. Replicated documents with regard to the number of queried peers

periments show the potential of our approach, significantly increasing the recall experienced in our settings.

We are currently working on experiments on real data obtained from focused web crawls, which exactly fits our environment of peers being users with individual interest profiles. Also, a more thorough study of the resource consumption of our approach in under way. One central point of interest is the directory maintenance cost; in this context, we evaluate strategies that do not rely on periodically resending all information, but on explicit GDO increment/decrement messages. Using a time-sliding window approach might allow us to even more accurately estimate the GDO values, with an even lower overhead.

References

1. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of the ACM SIGCOMM 2001, ACM Press (2001) 149–160
2. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: Proceedings of ACM SIGCOMM 2001, ACM Press (2001) 161–172
3. Rowstron, A., Druschel, P.: 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) 329–350
4. Buchmann, E., Böhm, K.: 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. (2004)
5. Aberer, K., Punceva, M., Hauswirth, M., Schmidt, R.: Improving data access in p2p systems. IEEE Internet Computing **6** (2002) 58–67

6. Chakrabarti, S.: Mining the Web: Discovering Knowledge from Hypertext Data. Morgan Kaufmann, San Francisco (2002)
7. Fuhr, N.: A decision-theoretic approach to database selection in networked IR. ACM Transactions on Information Systems **17** (1999) 229–249
8. Gravano, L., Garcia-Molina, H., Tomasic, A.: Gloss: text-source discovery over the internet. ACM Trans. Database Syst. **24** (1999) 229–264
9. Si, L., Jin, R., Callan, J., Ogilvie, P.: A language modeling framework for resource selection and results merging. In: Proceedings of CIKM02, ACM Press (2002) 391–397
10. Xu, J., Croft, W.B.: Cluster-based language models for distributed retrieval. In: Research and Development in Information Retrieval. (1999) 254–261
11. Callan, J.: Distributed information retrieval. Advances in information retrieval, Kluwer Academic Publishers. (2000) 127–150
12. Nottelmann, H., Fuhr, N.: Evaluating different methods of estimating retrieval quality for resource selection. In: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, ACM Press (2003) 290–297
13. Grabs, T., Böhm, K., Schek, H.J.: Powerdb-ir: information retrieval on top of a database cluster. In: Proceedings of CIKM01, ACM Press (2001) 411–418
14. Melnik, S., Raghavan, S., Yang, B., Garcia-Molina, H.: Building a distributed full-text index for the web. ACM Trans. Inf. Syst. **19** (2001) 217–241
15. Byers, J., Considine, J., Mitzenmacher, M., Rost, S.: Informed content delivery across adaptive overlay networks. In Proceedings of ACM SIGCOMM, 2002. (2002)
16. Ganguly, S., Garofalakis, M., Rastogi, R.: Processing set expressions over continuous update streams. In: SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, ACM Press (2003) 265–276
17. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM **13** (1970) 422–426
18. Mitzenmacher, M.: Compressed bloom filters. IEEE/ACM Trans. Netw. **10** (2002) 604–612
19. Florescu, D., Koller, D., Levy, A.Y.: Using probabilistic information in data integration. In: The VLDB Journal. (1997) 216–225
20. Bender, M., Michel, S., Weikum, G., Zimmer, C.: The MINERVA project: Database selection in the context of P2P search. In: BTW 2005. (2005)
21. Zhang, Y., Callan, J., Minka, T.: Novelty and redundancy detection in adaptive filtering. In: SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, ACM Press (2002) 81–88
22. Nie, Z., Kambhampati, S., Hernandez, T.: Bibfinder/statminer: Effectively mining and using coverage and overlap statistics in data integration. In: VLDB. (2003) 1097–1100
23. Hernandez, T., Kambhampati, S.: Improving text collection selection with coverage and overlap statistics. pc-recommended poster. WWW 2005. Full version available at *http://rakaposhi.eas.asu.edu/thomas-www05-long.pdf* (2005)
24. Bender, M., Michel, S., Triantafillou, P., Weikum, G., Zimmer, C.: Improving collection selection with overlap awareness in p2p systems. In: Proceedings of the SIGIR Conference. (2005)
25. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. The MIT Press, Cambridge, Massachusetts (1999)
26. Croft, W.B., Lafferty, J.: Language Modeling for Information Retrieval. Volume 13. Kluwer International Series on Information Retrieval (2003)

27. Bender, M., Michel, S., Weikum, G., Zimmer, C.: Minerva: Collaborative p2p search. In: Proceedings of the VLDB Conference (Demonstration). (2005)
28. Bender, M., Michel, S., Weikum, G., Zimmer, C.: Bookmark-driven query routing in peer-to-peer web search. In Callan, J., Fuhr, N., Nejdl, W., eds.: Proceedings of the SIGIR Workshop on Peer-to-Peer Information Retrieval. (2004) 46–57
29. Buckley, C., Salton, G., Allan, J.: The effect of adding relevance information in a relevance feedback environment. In: SIGIR, Springer-Verlag (1994)
30. Luxenburger, J., Weikum, G.: Query-log based authority analysis for web information search. In: WISE04. (2004)
31. Srivastava et al., J.: Web usage mining: Discovery and applications of usage patterns from web data. SIGKDD Explorations **1** (2000) 12–23
32. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. In: Symposium on Principles of Database Systems. (2001)
33. Nepal, S., Ramakrishna, M.V.: Query processing issues in image (multimedia) databases. In: ICDE. (1999) 22–29
34. Guntzer, U., Balke, W.T., Kiesling, W.: Optimizing multi-feature queries for image databases. In: The VLDB Journal. (2000) 419–428
35. Theobald, M., Weikum, G., Schenkel, R.: Top-k query evaluation with probabilistic guarantees. VLDB (2004)
36. Zipf, G.K.: Human Behaviour and the Principle of Least Effort: an Introduction to Human Ecology. Addison-Wesley (1949)