Dissertation zur Erlangung des Grades des Doktors der

Ingenieurwissenschaften der Naturwissenschaftlich-Technischen Fakultäten der

Universität des Saarlandes

Advanced Methods for Query Routing in Peer-to-Peer Information Retrieval

Matthias Bender

Supervisor:
Prof. Dr.-Ing. Gerhard Weikum

30th July 2007

Tag des Kolloquiums:

- 12. Juli 2007

Dekan:

- Prof. Dr.-Ing. Thorsten Herfet

Mitglieder des Prüfungsausschusses:

- Prof. Dr. Reinhard Wilhelm (Universität des Saarlandes),
- Prof. Dr.-Ing. Gerhard Weikum (Max-Planck-Institut für Informatik und Universität des Saarlandes)
- Prof. Peter Triantafillou, Ph.D. (RACTI und University of Patras, Griechenland)
- Prof. Dr. techn. Dipl.-Ing. Wolfgang Nejdl (Institut für Verteilte Systeme (KBS), Hannover)
- Prof. Peter Druschel, Ph.D. (Max-Planck-Institut für Softwaresysteme und Universität des Saarlandes)
- Dr.-Ing. Ralf Schenkel (Max-Planck-Institut für Informatik)

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Saarbrücken, den 23. Mai 2007

(Unterschrift)

Contents

Abstract

One of the most challenging problems in peer-to-peer networks is *query routing*: effectively and efficiently identifying peers that can return high-quality local results for a given query. Existing methods from the areas of distributed information retrieval and metasearch engines do not adequately address the peculiarities of a peer-to-peer network.

The main contributions of this thesis are as follows:

1. Methods for query routing that take into account the mutual overlap of different peers' collections,
2. Methods for query routing that take into account the correlations between multiple terms,
3. Comparative evaluation of different query routing methods.

Our experiments confirm the superiority of our novel query routing methods over the prior state-of-the-art, in particular in the context of peer-to-peer Web search.

Kurzfassung

Eines der drängendsten Probleme in Peer-to-Peer-Netzwerken ist *Query-Routing*: das effektive und effiziente Identifizieren solcher Peers, die qualitativ hochwertige lokale Ergebnisse zu einer gegebenen Anfrage liefern können. Die bisher bekannten Verfahren aus dem Bereich der verteilten Informationssuche sowie der Metasuchmaschinen werden den Besonderheiten von Peer-to-Peer-Netzwerken nicht gerecht.

Die Hautbeiträge dieser Arbeit teilen sich in folgende Schwerpunkte:

1. Query-Routing unter Berücksichtigung der gegenseitigen Überlappung der Kollektionen verschiedener Peers,
2. Query-Routing unter Berücksichtigung der Korrelationen zwischen verschiedenen Termen,
3. Vergleichende Evaluierung verschiedener Methoden zum Query-Routing.

Unsere Experimente bestätigen die Überlegenheit der in dieser Arbeit entwickelten Verfahren gegenüber den bisher bekannten Verfahren, insbesondere im Kontext von Peer-to-Peer-Websuche.

Summary

Peer-to-peer networks of cooperative, but autonomous machines are a promising foundation for powerful information systems, for example, Web search or massively distributed data mining. Such systems can benefit from the enormous computational power of peer-to-peer collaborations. On the other hand, their typically loose and volatile organization makes the effective and efficient recovery of information difficult. This thesis assumes an architecture of peers containing local data collections, in which the peers want to allow other peers to search their local data. For this purpose, they are willing to share statistical information describing their local data collections. Query originators can access a distributed directory storing this statistical information to identify the most promising peers for their particular queries. This process is referred to as *query routing*.

The methods that have previously been applied to query routing had originally been developed for smaller and more stable network. This thesis introduces some popular representatives, namely CORI, gGlOSS, DTF, and methods based on statistical language models. An evaluation identifies CORI as a particularly robust and at the same time viable method to estimate the expected result quality of a collection.

In our architecture with peers autonomously acquiring their local data collections, other aspects besides the expected result quality come to the fore. For example, the local data collections commonly show mutual overlap, as popular content is typically indexed by several peers. Therefore, query routing methods designed for peer-to-peer information systems should be able to systematically identify peers with high-quality, and at the same time mutually complementary local data. For this purpose, this thesis develops query routing methods taking into account the mutual overlap of different peers' collections by enriching the statistical information with compact synopses describing the local collections of a peer. A number of different synopses — Bloom filters, hash sketches, min-wise independent permutations — are evaluated with regard to their applicability to this problem, and methods estimating the degree of two collections' mutual overlap based on those synopses are developed. Experiments show that, due to such methods, fewer peers than before have to be involved in the query execution in order to retrieve the same number of relevant results.

Another characteristic problem of peer-to-peer information systems is the granularity of the statistical information, which are typically limited to a per-term scope in order to avoid putting the scalability of the distributed directory at risk due to the dimensionality of the search space. Query routing based on such statistical information is unavoidably based on an unrealistic independence assumption of individual terms. This disregards significant information regarding

the correlations between multiple terms. Our novel method taking into account the correlations between terms addresses this system-immanent weakness by appropriately combining the existing term-specific synopses of multiple terms to gain knowledge on the expected result quality of a collection with respect to a set of terms. Additionally, this thesis introduces a method to efficiently identify term combinations with a high potential for improvements by means of query log analyses, in order to facilitate a special handling of such combinations. The expressiveness of the derived knowledge for a given query exceeds the expressiveness of the original term-specific synopses significantly and, thus, enables a more effective query routing, as experiments show.

Comparative experiments conducted on several real-world data sets quantify the expected improvements realized by the novel methods presented in this thesis, both individually and combined.

Zusammenfassung

Peer-to-Peer-Netzwerke zwischen kooperativen, aber autonomen Rechnern sind eine vielversprechende Grundlage für mächtige Systeme zur Informationssuche, z.B. zur Websuche oder zur massiv verteilten Datenanalyse. Solche Systeme können dabei vom gewaltigen Rechenpotential des Peer-to-Peer-Verbundes profitieren. Andererseits verursacht dessen typischerweise lockere und volatile Organisation jedoch Komplikationen beim effizienten und effektiven Wiederfinden von Informationen. Die vorliegende Arbeit adressiert eine Architektur, in der alle Peers über lokale Daten verfügen und anderen Peers die Suche darauf ermöglichen wollen. Zu diesem Zweck sind sie bereit, statistische Informationen über die Qualität ihrer lokalen Datenbestände (Kollektionen) zur Verfügung zu stellen. Die anfragenden Peers ermitteln anhand dieser Informationen, die in einem verteilten Verzeichnis zugänglich gemacht werden, die besten Informationsquellen und entsprechende Peers für Ihre Anfragen. Dieser Schritt wird in der Literatur als *Query-Routing* bezeichnet.

Die bisher hierfür angewandten Verfahren wurden ursprünglich für kleinere und statischere Netzwerke entwickelt. Die vorliegende Arbeit stellt einige namhafte Vertreter vor: CORI, gGlOSS, DTF, sowie Verfahren basierend auf generierenden Sprachmodellen. Eine Evaluation identifiziert insbesondere CORI als robuste und gleichzeitig praktikable Methode zur Abschätzung der zu erwartenden Resultatsgüte einer Kollektion.

In unserer Architektur mit Peers, die ihre lokalen Kollektionen autonom aufbauen, treten neben der Betrachtung der erwarteten Resultatsgüte weitere Aspekte in den Vordergrund. So überlappen sich die lokalen Kollektionen häufig, da populäre Inhalte typischerweise von mehreren Peers indexiert werden. Daher sollten speziell für Peer-to-Peer-Informationssysteme entwickelte Methoden zum Query-Routing gezielt Peers mit qualitativ hochwertigen und gleichzeitig komplementären lokalen Daten identifizieren können. Die von uns entwickelten Verfahren zum Query-Routing unter Berücksichtigung gegenseitiger Überlappung erweitern daher die statistischen Zusammenfassungen um kompakte Synopsen zur Beschreibung der lokalen Kollektion eines Peers. Es werden verschiedene Synopsen — Bloom Filter, Hash Sketches, Min-wise Independent Permutations — auf ihre Eignung für diesen Zweck hin evaluiert, und es werden Methoden zur Abschätzung der gegenseitigen Überlappung zweier Kollektionen basierend auf diesen Synopsen entwickelt. Experimente belegen, dass durch diese Methoden weniger Peers als bisher an der Anfrageausführung beteiligt werden müssen, um die gleiche Menge an relevanten Resultaten zu erzielen.

Ein weiteres charakteristisches Problem von Peer-to-Peer-Informationssystemen ist die Granularität der statistischen Zusammenfassungen. Diese beschränken sich typischerweise auf term-spezifische Zusammen-

fassungen, um die Skalierbarkeit des verteilten Verzeichnisses nicht durch die Dimensionalität des Suchraums zu gefährden. Query-Routing auf solchen Zusammenfassungen basiert daher zwangsläufig auf der unrealistischen Annahme der Unabhängigkeit der einzelnen Terme. Es gehen wichtige Informationen verloren, die Auskunft über die Korrelationen zwischen verschiedenen Termen in den Daten geben. Unsere Verfahren zum Query-Routing unter Berücksichtigung von korreliert auftretenden Termen adressieren diese system-immanente Schwäche, indem sie die vorhandenen term-spezifischen Synopsen von mehreren Termen geeignet kombinieren. Die so gewonnenen Erkenntnisse geben Auskunft über die erwartete Güte einer Kollektion für Wortkombinationen. Zusätzlich stellt diese Arbeit ein Verfahren vor, das Wortkombinationen mit besonders hohem Gewinnpotential anhand von Query-Logs effizient identifiziert und so eine gesonderte Behandlung solcher Kombinationen ermöglicht. Die Aussagekraft der gewonnenen Informationen für eine konkrete Anfrage geht weit über die Aussagekraft der ursprünglichen Zusammenfassungen pro Einzelterm hinaus und ermöglicht so ein effektiveres Query-Routing, wie Experimente belegen.

Vergleichende Experimente auf verschiedenen realen Datenmengen liefern quantitative Aussagen über die durch die vorgestellten Methoden einzeln und kombiniert zu erwartenden Verbesserungen gegenüber dem bisherigen Stand der Wissenschaft.

Chapter 1

Introduction

1.1 Motivation

In recent years, the peer-to-peer (P2P) paradigm has been receiving increasing attention. While becoming popular mainly in the context of file-sharing applications (Gnutella, BitTorrent) or IP telephony (Skype), the P2P paradigm is rapidly making its way into distributed data management and information retrieval (IR) due to its ability to handle huge amounts of data in a decentralized and self-organizing way. These characteristics offer enormous potential benefit for information systems powerful in terms of scalability, efficiency, and resilience to failures and dynamics. Additionally, such an information system can potentially benefit from the intellectual input of a large user community participating in the data sharing network. Finally, but perhaps even more importantly, a P2P information system 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 challenge addressed in this thesis is to exploit P2P technology for efficient, reliable, large-scale content sharing and delivery to build P2P information systems. While there exist a number of protocols to build up and maintain such a collaboration of nodes, such architectures are typically limited to exact-match queries on keys and are insufficient for text queries that consist of a variable number of keywords, and they are absolutely inappropriate for full-fledged Web search where keyword queries should return a ranked result list of the most relevant approximate matches.

Practically viable P2P information systems must reconcile the following high-level, potentially conflicting goals: on the one hand, delivering high quality results with respect to precision/recall, and, on the other hand, providing scalability in the presence of a very large peer population and the very large amounts of data that must be communicated in order to meet the first goal.

Our targeted system consists of a number N of peers, p_j, $j = 1, ..., N$, forming a network G. In general, peers are assumed to be independently harvesting data, e.g., by performing Web crawls. A peer p_j constructs and stores a local search index, consisting of *index lists*, $I_j(t)$ over each term t (aka. "attribute" or "keyword") of interest. Thus, the peers store, share, and are used to deliver

index lists contents. Each inverted index list for a term t, $I_j(t)$, consists of a number of *(docID, score)* pairs, where score is a real number in $(0, 1)$ reflecting the significance of the document with the unique identifier *docID* for term t based on statistical scoring models.

A search request in such a system, initiated at a peer p_{init}, has the form of a top-k query, $q(T, k)$, which consists of a nonempty set of terms, $T = \{t_1, t_2, ..., t_m\}$, and an integer k. The answer to this query should yield a ranked list representing the k data items most relevant to the query, e.g., represented by URLs / URIs.

We assume that all peers have precomputed statistical *summaries* on their local index contents. These are organized on a per-term basis and would typically include measures such as the number of documents that the peer's local index contains for a given term, the average term frequency in these documents, and so on. Additionally, these summaries may contain compact synopses representing the documents that each peer holds. These summaries are then posted to a conceptually global, but physically distributed directory conveniently accessible by every peer, with $O(\log N)$ communication costs per term where N is the network size.

Assuming the existence of a such a directory, one of the fundamental functionalities a P2P information system must provide is to identify the most appropriate peers for a particular query, i.e., those peers that are expected to hold high-quality results for the query in their local indexes and, thus, should be involved in the query processing. This task is commonly referred to as *query routing*, sometimes also as *resource* or *collection selection*.

We stress that this task is more challenging than it may appear at first sight: the set of peers to be contacted is not simply the set of all peers that store relevant index data. Such a set could contain a very large number of peers and contacting all of them is prohibitive.

While there exist a number of approaches for query routing in the literature on distributed IR, these were typically designed for a stable and rather small set of collections (e.g., in the context of metasearch engines) and fall short of addressing the peculiarities of a large-scale, highly dynamic P2P system.

1.2 Main Contributions

1.2.1 Overlap-Aware Query Routing

A key shortcoming of the existing strategies is their — at first sight by all means intuitive — approach to base their decisions on the expected result quality of the peers. However, with autonomous peers harvesting information at their own discretion, peers may have highly overlapping local data contents as popular information is indexed by a large number of peers individually.

Thus, the rationale for the overlap-aware query routing strategies is based on the observation that a query should be forwarded to peers that are expected to contribute not only high-quality, but also *complementary* results. If a remote peer returns more or less the same high-quality results that the query initiator already obtained from other candidates, then the whole approach of collaborative P2P search would be pointless. An integrated quality- and overlap-aware query routing method should be able to estimate the content richness of candi-

date target peers, in terms of the similarity and relevance of the peers' contents to the given query, and the degree of novelty that a candidate peer offers relative to the initial results that are already known to the query originator.

From a research perspective, the key challenges in order to achieve overlap-awareness lie in

- defining appropriate *metrics* that allow an estimation of the expected benefit that the inclusion of a peer will bring to the result set,
- representing local index data by means of compact synopses that support operations to estimate the above metrics for a given collection, but also support *aggregation* of synopses for several collections, and
- designing scalable *algorithms* for overlap-aware query routing in P2P information systems that can benefit from these synopses efficiently.

We address the first issue by introducing the notion of *novelty* that combines standard set operations (union, intersection, cardinality) as a metric for the expected contribution of a collection with regard to a given reference collection. We evaluate a number of statistical synopses (Bloom filters, hash sketches, min-wise independent permutations) from the literature regarding their general accuracy and their particular support for the necessary intermediate combination and aggregation steps. We present the IQN algorithm (*I*ntegrated *Q*uality and *N*ovelty) that chooses target peers in an iterative manner, performing two steps in each iteration: first, the Select-Best-Peer step identifies the most promising peer regarding a combination of result quality and novelty. This step is driven by the statistical synopses that are obtained from the directory. Then, the Aggregate-Synopses step conceptually aggregates the selected peer's content with the previously selected peers' data collections. This aggregation is actually carried out on the compact synopses, not on the full data. The two-step selection procedure is iterated until given performance and/or quality goals are satisfied (e.g., a predefined number of peers is reached, or a desired recall is estimated to be achieved).

Our experiments show that the efficiency and effectiveness of the IQN routing method crucially depends on appropriately designed compact synopses describing the collection of a peer. The synopses must be small to keep network bandwidth consumption and storage costs low, yet they must offer low-error estimations of quality and novelty measures. Furthermore, to support the intermediary aggregation step introduced above, it must be possible to iteratively combine multiple synopses published by different peers in order to derive a synopsis for a hypothetical combined collection. We have developed such methods based on Bloom filters, hash sketches, and min-wise independent permutations as peer synopses. Extensive experiments have shown that these novel algorithms indeed combine very low overhead with high accuracy for quality-novelty estimation, and the IQN query routing strategy outperforms prior methods for query routing.

1.2.2 Correlation-Aware Query Routing

The summaries describing a peer in the distributed directory are usually organized on a per-term basis, indicating the expected result quality of a peer's

collection for a given term. This limitation is considered unavoidable, as statistics on all term pairs would incur a quadratic explosion, leading to a breach with the goal of scalability. On the other hand, completely disregarding correlations among terms is a major impediment: for example, consider the following extreme scenario. Assume peer p_1 contains a large number of data items for each of the two terms a and b separately, but none that contains both a and b together. Judging only by per-term statistics, state-of-the-art query routing approaches would reach the conclusion that p_1 is a good candidate peer for the query $(\{a, b\}, k)$, whereas the actual result set would be empty. This is because the summaries describe the expected quality for each term *separately*, where the most promising candidate peers for the entire query should really exhibit a higher-than-average frequency of data items that contain both terms *at the same time*.

This thesis develops and evaluates two approaches to address this issue:

- *sk-STAT* estimates the desired multi-term statistics from the existing per-term statistics with additional computational efforts and at higher networking costs, and

- *mk-STAT* enhances the distributed directory to include also explicit statistical information about judiciously chosen sets of multiple terms

The caveat of mk-STAT is that it faces the necessity to identify those valuable term sets to avoid the dimension explosion mentioned above. It does so by mining locally gathered query logs, to improve the performance of frequently queried term combinations. This discovery phase additionally performs an in-depth statistical analysis of the degree of correlation observed within the peers' data collections for these term combinations. One of our novel contributions is how to make this analysis efficient and scalable.

sk-STAT, on the other hand, can readily deal with all possible term sets, as it only relies on combinatorial operations on the existing single-term statistics. However, it has higher bandwidth requirements at query time, as larger amounts of single-term statistics have to be shipped to estimate the statistics for the term sets.

1.2.3 Minerva (P2P Web Search Software)

In the course of this thesis, a prototype software has been developed, initially to serve as a testbed for our experiments, in order to evaluate our novel methods for query routing, but also to demonstrate the feasibility of P2P Web search in general. After a number of live demonstrations at scientific conferences [BMPC07, BMT+05b, MBT+05] and project workshops, the interest in our prototype software has increased to a level where we have decided to make the prototype publicly available under an open source software license. The most current release version, coined MinervaLight, combines the (previously separate) focused Web crawler BINGO! [STS+03], the local search engine TopX [TSW05], and Minerva under one common user interface. The crawler unattendedly downloads and indexes Web data, where the scope of the *focused* crawl can be tailored to the thematic interest profile of the user. The result of this process is a local search index, which is used by TopX to evaluate user queries.

In the background, MinervaLight continuously computes compact statistical synopses that describe a user's local search index and publishes that information to a conceptually global, but physically fully decentralized directory, implementing the general system architecture outlined before. MinervaLight offers a search interface where users can submit queries to Minerva. Our novel query routing strategies are used to identify the most promising peers for each query based on the statistical synopses in the directory. The query is forwarded to those judiciously chosen peers and evaluated based on their local search indexes. These results are sent back to the query initiator and merged into a single result list, which is eventually displayed to the user.

The latest stable version of the software can be downloaded from *http://www.minerva-project.org*.

1.3 Outline

The remainder of this thesis is organized as follows. Chapter 2 gives a general introduction to the background and state-of-the-art regarding peer-to-peer computing and IR. Chapter 3 explores the possible design space for P2P Web search engines and introduces Minerva, our P2P Web search software. Existing approaches to query routing are featured in Chapter 4 together with an evaluation.

As the main contributions of this thesis, Chapters 5 and 6 present novel approaches to query routing that take into account the mutual overlap of peers' collections (Chapter 5) and the correlations between terms that can be observed in both data and queries (Chapter 6). While each of these chapters features an individual experimental evaluation, Chapter 7 evaluates the potential of a comprehensive combined approach with overlap- and correlation-awareness. Further possible enhancements to query routing based on the global authority of the documents contained in each collection and other miscellaneous approaches are introduced subsequently in Chapter 8.

Finally, Chapter 9 concludes this thesis and points out possible future research directions.

Chapter 2

Background and State-of-the-art

2.1 Peer-to-Peer (P2P) Computing

In recent years, the peer-to-peer (P2P) paradigm has been receiving ever-increasing attention and has become a hype paradigm for communication on the Internet. While becoming popular mainly in the context of file sharing applications such as Napster, Gnutella, or BitTorrent, the P2P paradigm can be used to access any kind of distributed data and is rapidly making its way into distributed data management and offering possibilities for previously unseen Internet applications. The traditional client-server approach, in contrast, requires a tremendous amount of effort and resources to meet the increasing challenges of the continuously growing Internet. Due to its centralized nature, the client-server network model usually results in high loads for these centralized entities that easily become bottlenecks and so-called *single-points-of-failures*, where the failure of one entity actually shuts down the functionality of the system. Consequently, such systems can easily be attacked, e.g., by denial-of-service attacks. Additionally, dedicated servers are often difficult and expensive to administrate and to relocate due to their strategic placement within the Internet infrastructure.

The concept of P2P computing promises to offer enormous potential benefits to issues such as scalability, security, reliability, efficiency, flexibility, and resilience to failures and dynamics, through a fundamental shift of paradigms [SW05].

So what exactly is a P2P system? An exact definition is hard to give and hindered by the fact that some of the early applications entitled peer-to-peer are not even true peer-to-peer in a strict sense. Wikipedia currently defines a peer-to-peer network as

“ a network that relies primarily on the computing power and bandwidth of the participants in the network rather than concentrating it in a relatively low number of servers. P2P networks are typically used for connecting nodes via largely ad hoc connections. [...] A pure peer-to-peer network does not have the notion of clients or servers, but only equal peer nodes that simultaneously function as both ”clients”and ”servers”to the other nodes on the network. ”

[Ora01, SW05] give a more technical definition of a P2P system:

"[a Peer-to-Peer system is] a self-organizing system of equal, autonomous entities (peers) [which] aims for the shared usage of distributed resources in a networked environment avoiding central services."

Both definitions share the idea of decentralization and point at decentralized resource usage and decentralized self-organization as potential benefits.

2.1.1 The Lookup Problem

However, decentralization as the particular strength of P2P systems also sparks their predominant challenge, often referred to as the *lookup problem*:

Where to store, and how to find a certain data item in a distributed system without any centralized control or coordination [BKK+03].

In contrast to traditional, centralized client-server-style systems, where the data is provided by dedicated physical entities that are explicitly referenced (e.g., by means of a Uniform Resource Locator [URL]), P2P systems store data in multiple, distant, transient, and unreliable locations within the network. One of the predominant challenges of a P2P system, thus, is to efficiently locate data that is stored in the network. Its ability to do so even in the case of node failures and the resulting resilience in the presence of network dynamics constitute the potential benefits of a P2P system.

The research literature commonly classifies the existing approaches to tackle the lookup problem in a fully decentralized environment, such as a P2P collaboration, into *structured* and *unstructured* approaches [SW05]. There is no common agreement on whether a third family of approaches, often called *Centralized P2P Architectures*, should be considered a P2P architecture at all. Additionally, there exist all kinds of hybrid approaches that try to combine their respective advantages, most notably the so-called *super-peer architectures*. The upcoming sections present the approaches in more detail.

2.1.2 Centralized P2P Architectures

Napster, which was arguably the first occurrence of the P2P paradigm in a broader public perception, elegantly circumnavigated the *lookup problem* by instantiating an architecture in which a centralized entity provides a directory service to all participating peers, effectively forming a star network. The fact that all peers that join the system register their data (mostly music files in the early days) with this centralized instance allowed a comfortable way for other peer to locate any data in the network by presence of a physically centralized directory[1]. The fact that only pointers to decentralized peers are stored at the centralized entity (instead of the actual data) conceptually decreases the load at the central entity; the fact that (after relevant data has been located by means of the directory) each peer could directly communicate with other peers that store the data in a decentralized manner, completely bypassing the centralized directory entity, drives the perception of Napster as a P2P system.

In a "true" P2P system, however, it should be possible to remove any entity from the network without loss of functionality (so that, by this definition, Napster is actually *not* a P2P system). Instead, a peer should conceptually fill both

[1] A fact that later also allowed for the easy shutdown of Napster by legal authorities.

roles as server and client, i.e., content provider and content consumer, over time, such that the functionality of the system is spread equally over all the peers in the network.

2.1.3 Unstructured P2P Architectures

In order to locate data, *unstructured P2P architectures* no longer rely on any central entity or any other form of explicit knowledge about the location of data within the network. Instead, each node recursively forwards requests to all other peers that it is aware of (*neighbors*), in an attempt to locate all relevant data in the network. In order to reach all appropriate peers, thus, a node broadcasts each message it receives to other peers, regardless of whether they store relevant data or not. This approach is known as *message flooding* and effectively leads to a breadth-first search strategy. Each message is assigned a *Time-to-live* (TTL) value, which a peer decreases by one when forwarding a message, to avoid infinite loops and to control the number of messages being generated by one query being issued.

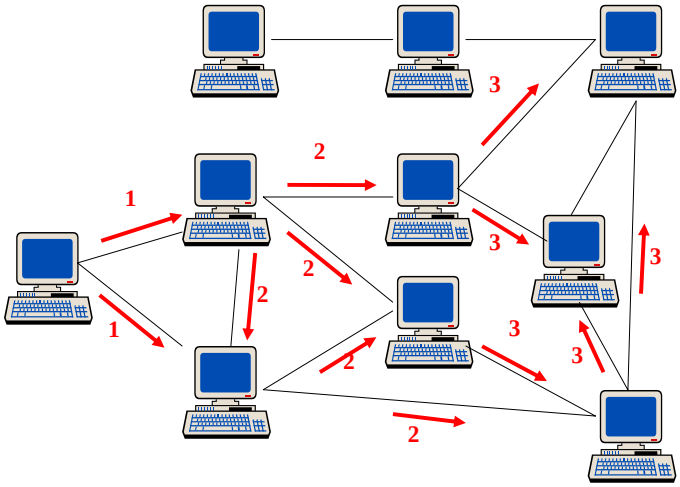

Figure 2.1: Message flooding in an unstructured P2P network

This general design is illustrated in Figure 2.1. The query is issued by the leftmost peer and forwarded to its two neighbors, as indicated by the red arrows labeled *1*. These peers again forward the query to their neighbors. Note that some messages unnecessarily address peers that have already received the query at different times and/or from different peers.

Several studies of real-world networks have shown that the nodes of such a network form graphs with small diameters, typically in a range from five to seven, supporting the so-called *small-world-phenomenon* [Kle00], which has also been observed in regular social networks. In such a network, a relatively small TTL value per message can locate any data of interest with high probability.

However, this technique still represents a *lossy* protocol as it cannot guarantee the successful location of data, e.g., due to higher graph diameters or disconnected graph partitions. The number of messages caused by a single request is significant; it depends on the degree of the nodes in the network (i.e., the number of neighbors of a node) and the chosen TTL value.

The advantages of this approach are the facts that it is not necessary to proactively maintain the network, e.g., upon node joins and arrivals, and that the node state is limited to $O(1)$, maintaining only pointers to an upper-bounded number of neighbors. Also note that there is no enforcement of the storage location for data items, as they can be located anywhere in the network. In other words, the data stored on a node is unrelated to the node's position in the network.

Popular implementations of this paradigm include Freenet [CMH+02] and early version of the Gnutella protocol [gnu00].

Flooding is also a fundamental message dissemination strategy in unreliable networks, such as mobile ad-hoc networks (*MANETs*). While plain flooding algorithms provoke a high number of unnecessary messages, causing network contention, packet collisions, and wasting energy, several probabilistic and epidemic approaches have been studied to optimize flooding.

2.1.4 Super-Peer Architectures

An important lesson learned from the deployment of Gnutella is that the performance characteristics of the peers (processing power, bandwidth, availability, ...) is not evenly distributed over all peers, decreasing the theoretical benefits of perfect decentralization. This fact is exploited by *super-peer architectures*, where a small subset of peers takes over specific responsibilities in the network, e.g., aggregation or routing tasks. In a way, the super-peers are the distributed successors of the centralized entity in Napster-style architectures. Conceptually, only the super-peers form a P2P network, and all other peers connect to this backbone via one of the super peers, which act in the spirit of database mediators aggregating the content of downstream peers. Routing is conducted in a two-phase mode, which routes a request within the super-peer backbone at first, and then distributes it to the peers connected via the super-peers. While dedicating specific nodes potentially limits the self-organizing capabilities of a P2P network, super-peer architectures have been proven a way to alleviate the performance issues of pure unstructured topologies.

2.1.5 Structured P2P Architectures

Both of the above architectures expose bottlenecks which hinder their scalability to an a-priori unlimited number of peers. In centralized architectures, the linear storage complexity of the directory entity is prohibitive, whereas in unstructured architectures the communication overhead caused by message flooding is a significant shortcoming. Instead, an efficient and scalable approach requires a sub-linear increase in the storage and search complexity as more and more peers enter the system.

Structured P2P architectures superimpose certain overlay structures to map nodes and data items into a common address space, enabling a unique mapping from data items to nodes given the current state of the network. For this

purpose, each node manages a small number of pointers to carefully selected other peers (typically $O(logN)$, where N is the number of nodes in the network); routing along these paths eventually leads to the globally agreed-on peer that is currently *responsible* for a given data item, commonly with $O(logN)$ message hops. Distributing the responsibilities as uniformly as possible over the nodes in the network promises balanced storage and retrieval loads among all nodes.

On top of this routing functionality, it is straightforward to implement a hash-table-like data structure that allows the insertion and retrieval of *(k, value)*-pairs describing a key k: for insertion or retrieval of a *(k, value)*-pair, turn to the peer currently responsible for k in the network as defined by the structured P2P network. This peer stores and maintains all appropriate *(k, value)*-pairs for k from across the directory. Note that, in contrast to unstructured P2P architectures, the placement of data is no longer arbitrary, but determined by the underlying architecture. Therefore, the routing is no longer lossy, i.e., a data item that is stored in the network can be guaranteed to be found.

The term *distributed hash table* (DHT) has been coined for such a functionality in a P2P network and is commonly used as a synonym for structured P2P architectures in general. Some researchers, however, strictly distinguish between structured P2P routing primitives on one side and the DHT interface of inserting and retrieving data as the next layer of functionality on the other side.

Various geometries have been proposed for use as structured overlay topologies. These geometries include hypercubes (CAN [RFH+01]), rings (Chord [SMK+01], Pastry [RD01]), tree-like structures (P-Grid [Abe01], PRR [PRR97]), and butterfly networks (Viceroy [MNR02]). A special case are random graphs [MS05]. The general resilience and proximity properties of these different geometries have been studied in [GGG+03].

The Chord Protocol

The elegance of the Chord protocol [SMK+01] as a representative of structured P2P architectures stems from its simplicity and clarity. Data items and nodes are all mapped to a unique one-dimensional identifier space. Identifiers in Chord are l-bit numbers, i.e., integer values in the range $[0, 2^l - 1]$, forming a cyclic identifier space modulo 2^l. An identifier of a data item is referred to as a *key*, that of a node as an *ID*.

The responsibility of maintaining the data items *(k, value)* associated with key k lies at the nearest node on the identifier circle whose ID is greater or equal to k. Such a node n is called the *successor* of key k. Thus, a node n is responsible for all key identifiers between its predecessor's identifier (exclusive) and its own identifier (inclusive), and each *(k, value)*-pair is located and managed on a single, well-defined node.

Figure 2.2 illustrates an identifier circle with $l = 6$, i.e., identifiers in the range $[0, 63]$. For example, key k_{54} is maintained by node p_{56} as its next successor on the ID circle, whereas both keys k_{24} and k_{30} are maintained by node p_{32}.

The key to efficient lookup and modification operations on this data is to efficiently solve the *lookup problem* introduced before, i.e., to quickly locate the node responsible for a particular key. Most naively, every peer might store a pointer to its successor node on the identifier circle. When a key is being looked up, each node forwards the query to its successor, until one node determines

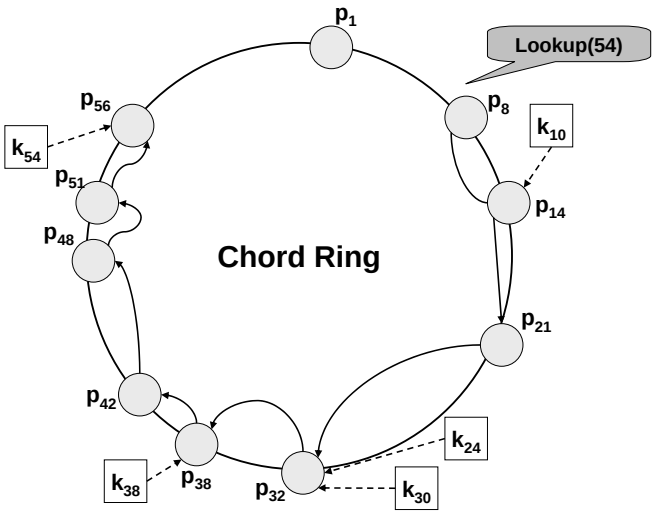

Figure 2.2: Chord ring

that the key lies between its own ID and the ID of its successor. Thus, the key must be hosted by its successor. Consequently, the successor is communicated as the result of the query back to the originator. While maintaining only a minimum amount of state at each node ($O(1)$), this form of key location leads to an expected number of messages linear in the number of nodes in the network, which is not considered scalable. This naive approach is also illustrated in Figure 2.2 for node p_8 issuing a lookup request for key k_{54}. The request is forwarded along the ID circle linearly until the responsible peer p_{56} can be identified.

Chord keeps additional state at each node to enable more scalable lookups. Each node maintains a routing table, the *finger table*, pointing to other nodes on the identifier circle. The m-th entry in the finger 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, leading to a finger table with at most l distinct entries (independent of the actual number of keys or nodes). This scheme has two important characteristics. First, each node only maintains state about a logarithmic number of other nodes, and knows more about nodes closely following it on the identifier circle than about nodes farther away. Second, a node's finger table does not necessarily contain enough information to *directly* determine the node responsible for an arbitrary key k. 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 2.3 for nodes p_8, p_{42}, and p_{51}. 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 peers runs periodically in the background and which updates Chord's finger tables and successor pointers in order to ensure that lookups execute correctly as the set of participating peers changes. The stabilization requires an additional predecessor pointer, as each node p_i requests its successor, $succ(p_i)$, to return its predeces-

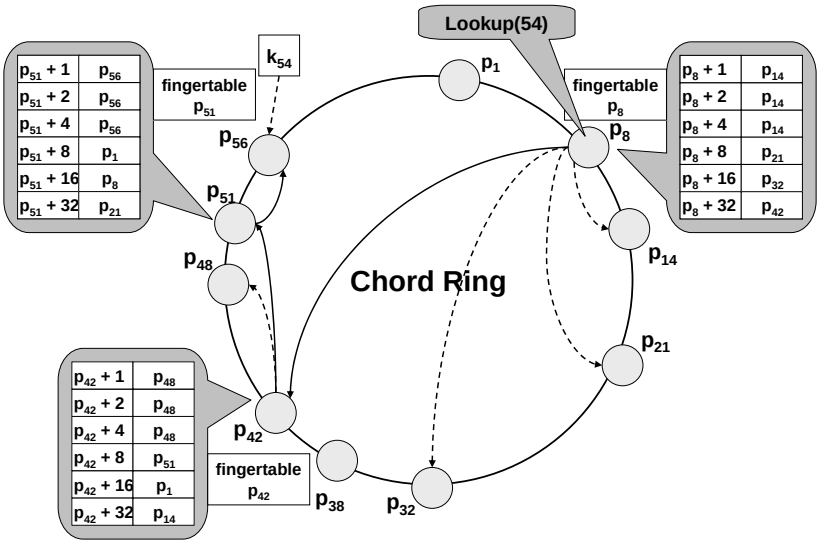

Figure 2.3: Scalable lookups using finger tables

sor $pred(succ(p_i))$. If p_i equals $pred(succ(p_i))$, p_i and $succ(p_i)$ agree on being each other's respective predecessor and successor. In contrast, the fact that $pred(succ(p_i))$ lies between p_i and $succ(p_i)$ indicates that $pred(succ(p_i))$ recently joined the identifier circle as p_i's successor. Thus, node p_i updates its successor pointer to $pred(succ(p_i))$ and notifies $pred(succ(p_i))$ of being its predecessor. At this stage, all successor pointers are up to date and queries can be routed correctly.

As the impact of outdated finger table entries on lookup performance is small, Chord updates finger tables only lazily by periodically picking a finger table entry i randomly and performing a lookup to find the true node that currently succeeds node p by 2^{i-1}.

Chord addresses node failures by checking all communication with remote nodes for timeouts. To further ensure routing stability in the presence of multiple simultaneous node failures, each node maintains not only a pointer to its immediate successor, but a list of the first r successors. When a node detects a failure of its successor, it reverts to the next live node in its successor list. The successor list is also maintained by the stabilization protocol. [LNBK02] gives a theoretical analysis of Chord's stability in the face of concurrent joins and multiple simultaneous node failures.

The failure of a node not only puts the routing stability at risk, but also makes the data managed by this node unavailable. Such data loss can be prevented by replicating the data to other peers. In Chord, the successor of a failed node becomes responsible for the keys and data of the failed node. Thus, an obvious replication strategy to prevent data loss is to replicate data to successor nodes, using the successor list.

2.2 Information Retrieval

Information retrieval (IR) systems keep collections of unstructured or weakly structured data, such as text documents or HTML pages, and offer search functionalities for delivering documents *relevant* (useful) 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. This section introduces some basic information retrieval concepts; a more detailed introduction can for example be found in [MYL02, Cha02].

2.2.1 Document Representation

Documents are usually represented by the words (or *terms* in IR jargon) contained in them, where a small set of so-called *stop words* not containing any semantic information (a, the, of, ...) is typically omitted. Words with a common root (e.g., beauty, beautiful, beautify) are often registered in a combined way and denoted with their common prefix only, a concept known as *stemming*. After stop word removal and stemming, each document can be logically represented by a vector of n term occurrences, where n is the total number of distinct terms in the set of all documents in a collection. In this *vector-space model*, for a document represented by a vector $(w_1, w_2, ..., w_n)$, each w_i is the weight (or *score*) indicating the importance of term i in a document[2]. The weight assigned to a term is usually based on the following two factors: the number of occurrences of term t in document d is called *term frequency* and typically denoted as $tf_{t,d}$. Intuitively, the weight of a term within a document increases with the number of occurrences. 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 term decreases as the number of documents that 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. Eventually, multiplying these two values yields the weight of a term in a document.

This family of similarity functions is often referred to as *tf*idf*; a typical representative calculates the weight $w_{i,j}$ of term t_i in document d_j as

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

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

In recent years, other relevance measures based on statistical language models [PC98, SJCO02] and probabilistic IR have been investigated [Fuh99]. [Cha02, MS99] give a good overview over various similarity functions.

2.2.2 Query Representation and Execution

A query consists of a set of (possibly weighted) terms that can again be represented by the appropriate n-vector. Query execution can easily be performed by applying the dot product function

[2] Most of the entries in the vector will be zero because most terms are absent for typical documents.

$$dot(q,d) = \sum_{k=1}^{n} q_i * d_i$$

where q and d are the query and document vectors, respectively. For a particular query, the dot product intuitively sums up the importance scores for those terms of the document that are contained in the query vector, weighted by their respective weights within the query. The *cosine* function can be used to overcome the tendency of the dot function to favor longer documents having many terms, capturing the angle between the two vectors

$$dot(q,d)/(|q| * |d|)$$

where |q| and |d| denote the lengths (L_2-norms) of the vectors.

A nice side effect is that, for non-negative weights, the cosine function returns values between 0 and 1, making an additional score normalization unnecessary.

2.2.3 Measuring Retrieval Effectiveness

The two most popular measures for retrieval effectiveness are *precision* and *recall*, which are defined as follows:

$$precision = \frac{\#\ of\ relevant\ docs\ retrieved}{total\ \#\ of\ retrieved\ docs}$$

$$recall = \frac{\#\ of\ relevant\ docs\ retrieved}{total\ \#\ of\ relevant\ docs}$$

For each query, the set of relevant documents is identified in advance, e.g., using human relevance assessment techniques or using another retrieval technique as a baseline algorithm.

Depending on the concrete application context, one or the other measure is of greater importance: some applications cannot accept irrelevant documents, they will tend to reduce the number of retrieved documents at the cost of sacrificing recall. Other applications focus on all relevant documents being returned; those applications will tend to increase the number of retrieved documents at the expense of lower precision.

A measure capturing this interdependency is the *F1 measure*, which combines precision P and recall R with equal importance into a single parameter for optimization and is defined as

$$F1 = \frac{2 * P * R}{P + R}$$

The F1 measure is a good candidate for optimization, given the fact that one can get a perfect precision score by not returning any documents, or a perfect recall score by returning all documents. A powerful document retrieval system will yield all truly relevant documents and only truly relevant documents, maximizing precision and recall at the same time, and therefore maximizing the F1 score, which falls in the range from 0 to 1 (with 1 being the best score).

2.2.4 Top-k Query Processing

For a given query, computing the dot product or the cosine measure for all documents is a computationally expensive task. Instead, there are strategies to efficiently identify only the top-k documents (i.e., the k documents with the highest score for that particular query) without evaluating the full dot product for all documents.

In order to efficiently support this process, the concept of *inverted index lists* has been developed [ZM06]. All terms that appear in the collection form a tree-based index structure (often a B^+-tree or a trie) where the leaves contain a list of unique document identifiers for exactly those documents that contain this term (Figure 2.4). Depending on the exact query execution strategy, the lists of document identifiers may be ordered according to the document identifiers or according to their term scores to allow efficient pruning.

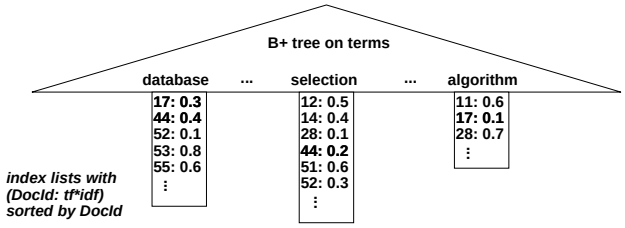

Figure 2.4: B+ tree of inverted index lists

An efficient algorithm should avoid reading inverted index lists completely, but would ideally limit the effort to $O(k)$ steps where k is the number of desired results. In the IR and multimedia-search literature, various algorithms have been proposed to accomplish this task. The best known general-purpose method for top-k queries is Fagin's threshold algorithm (TA) [FLN01], which has been independently proposed also by Nepal et al. [NR99] and Güntzer et al. [GBK00]. It uses index lists that are sorted in descending order of term weights 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 in the remaining index lists to calculate the final score for d and keeps this information in a document candidate set of size k. Since TA additionally keeps track of an upper score bound for documents not yet encountered, the algorithm terminates as soon as this bound assures that no previously unseen document can enter the candidate set. Extensions of this scheme further reduce the number of index list accesses (especially random accesses) by a more sophisticated scheduling. Also, probabilistic methods have been studied that can further improve the efficiency of index processing [TSW05, BMS$^+$06].

Chapter 3

P2P Web Search and the Minerva Prototype

The proliferation of P2P architectures has come with various compelling applications, most notably, file sharing and IP telephony. File sharing involves file name lookups and other simple search functionalities such as finding files whose names (short strings) contain a specified word or phrase. Such simple queries can be executed in a highly efficient and ultra-scalable way, based on distributed hash tables. However, the research on structured P2P architectures so far is mainly limited to exact-match queries on single keys and does typically not support any form of ranking. This is insufficient for text queries that consist of a variable number of keywords, and it is absolutely inappropriate for full-fledged Web search where keyword queries should return a ranked result list of the most relevant approximate matches [Cha02]. Thus, these approaches are unsuitable for searching also the *content* of files, such as Web pages or PDF documents. For the latter, much more flexible multi-keyword querying is needed, and, most importantly, the fact that many queries may return thousands of different matches calls for scoring and ranking, the paradigm of Web search engines and information retrieval.

In fact, full-fledged Web search today is more or less exclusively under the control of centralized search engines. As of June 2006, 86% of the more than 6 billion Web searches a month in the US are conducted by one of the three major players (Google 44,7%, Yahoo 28,5%, Microsoft/MSN 12,8 %)[1]. For Germany, Google is even more dominant with a market share of 86% on its own[2]. Lately, various projects have been started to build and operate a P2P Web search network (e.g., [CAPMN03, LC03, RV03, SMwW+03, TD04, GWJD03, MEH05, NOT02, LKP+06]) including our own Minerva project [BMT+05b, BMPC07], but so far these endeavors have not yet attracted a critical user mass — despite the fact that Web search and Internet-scale file content search seem to be perfect candidates for a P2P approach for several reasons:

- deploying a single, general-purpose centralized Web search engine is unrealistic in terms of processing power, keeping it up-to-date, and limited

[1] http://www.comscore.com/press/release.asp?press=984
[2] http://www.webhits.de/deutsch/index.shtml?/deutsch/webstats.html

visibility of *Deep Web* sources to outside crawlers. In fact, it has been argued that the Web is increasing at a much faster rate than the indexing capability of any centralized search engine [HT99, SE00, WMYL01]

- the data is highly distributed "by nature", residing on millions of sites (with more and more private people contributing, e.g., with private blogs)
- a P2P network could potentially dwarf even the largest server farm in terms of processing power and could, thus, enable much more advanced methods for linguistic data analysis, statistical learning, or ontology-based background knowledge and reasoning (all of which are out of the question when you have to serve hundred millions of queries per day on a, however big but centralized, server farm)
- there is growing concern about the world's dependency on a few quasi-monopolistic search engines and their susceptibility to commercial interests, spam or distortion by spam combat, biases in geographic and thematic coverage, or even censorship. Especially the last issue has led to the postulation that the Web should be returned to the people [GM05]
- besides its unprecedented computational power, a P2P Web search engine can potentially benefit from the intellectual input of a large user community, as every peer's behavior is driven by a human user

While many Internet users today are probably more than happy with the results returned by these centralized services[3], their weaknesses are easily apparent and directly following from the above arguments for a decentralized system:

- insufficient coverage of the Web
- inaccessibility of Deep Web sources by crawlers
- risk of *information monopolies*
- risk of infiltration and/or censorship

The next section elaborates on the huge design space and the challenges of distributed search. Section 3.2 presents the system architecture of Minerva, a P2P Web Search prototype developed in the course of this work, and positions it as one particular incarnation within this design space.

3.1 Design Space and Challenges

We identify the following key environment characteristics and desirable performance features that greatly influence the key design choices for a P2P Web search engine.

1. *Peer Autonomy*: There are two specific aspects of autonomy of concern. First, whether a peer is willing to relinquish the storage/maintenance of its index lists, agreeing that they will be stored at other peers. For instance, a peer may insist on storing/maintaining its own index lists, worrying about possible problems (e.g., index-list data integrity, availability, etc). Second, a peer may not be willing to store index lists produced by other peers.

[3] *To google* after all has to become a synonym for searching the Web.

2. *Sharing Granule*: Influenced from the autonomy levels as above and performance concerns, the shared data can be at the level of complete index lists, portions of index lists, or even simply index list metadata appropriately defined.

3. *Scalability*: For popular terms, there may be a very large number of peers locally storing index lists. Accessing all such peers may not be an option. Hence, designing with scalability in mind must foresee the development of mechanisms that can select the best possible subset of relevant peers, in a sense that the efficiency of operation and result quality remain at acceptable levels. Complementarily, peers storing popular index lists may form retrieval bottlenecks hurting scalability. Hence, designing for scalability also involves novel strategies for distributing index list information that facilitates a large number of peers pulling together their resources during query execution, forming in essence large-capacity, virtual peers.

4. *Latency*: Latency may conflict with scalability. For example when, for scalability reasons, query processing is forced to visit a number of peers which collectively form a large-capacity virtual peer, query response time may be adversely impacted.

5. *Exact vs. Approximate Results*: Approximate results may very well be justified at large scales. For scalability and efficiency reasons, contacting all peers storing relevant index list data might be infeasible, which makes exact and complete answers unrealizable.

Approaches to comprehensive Web search based on a P2P network have long been considered infeasible, or at least being a grand challenge, from a scalability viewpoint [LLH+03]. Those early approaches typically spread inverted index lists across the directory such that each peer is responsible for maintaining a subset of index lists. Such a system design allows for exact and complete execution of top-k style aggregation queries over the P2P network. However, in spite of query execution closely resembling the operation modes of a centralized engine, the bandwidth requirements and also latency issues rendered the immediate application of existing TA-style techniques (which were originally developed with a centralized system in mind) infeasible. The concurrent dissertation [Mic07] elaborates on efficient large-scale top-k query aggregation algorithms for distributed systems, e.g., [CW04, MTW05a, MTW05b], and presents novel approaches to overcome these challenges.

The system design that is adopted for this thesis is different. Instead of spreading inverted index lists across the directory, we use only pointers to promising peers (enriched with compact statistical metadata describing the index contents of that peer) and utilize these pointers to efficiently answer multiple-keyword queries. We leverage the extensive local indexes (which would be impossible to efficiently share across all peers) for query execution, in order to benefit from sophisticated features that are currently considered infeasible in the above setup, such as phrase matching or proximity searches. Section 3.2 presents this system architecture in more detail.

The upcoming subsections introduce and discuss the two major challenges of distributed Web search in this architecture of choice, namely *query routing* and *result merging*.

3.1.1 Query Routing

One of the key issues to make P2P Web search feasible is *query routing*: when a peer poses a query with multiple keywords and expects a high-quality top-10 or top-100 ranked result list, the P2P system needs to make a judicious decision to which other peers the query should be forwarded. This decision needs statistical information about the data contents in the network. It can be made fairly efficiently in a variety of ways, like utilizing a DHT-based distributed directory [WGD03], building and maintaining a semantic overlay network (SON) with local routing indexes [CGM02, ACMHP04], or using limited forms of epidemic gossiping [CAPMN03]. This thesis builds on the first option, i.e., techniques that try to estimate the expected quality of a peer for a particular query from statistics that are stored in a conceptually global, but physically distributed directory based on a DHT infrastructure.

Chapter 4 reviews existing algorithms for query routing that have originally been designed for distributed IR, but not for the large-scale and high dynamics of a P2P system. Nevertheless, they offer interesting insights and serve as a valuable baseline for improvements. As the main contribution of this work, Chapters 5, 6, and 8 present novel query routing approaches that were designed to deal with the peculiarities of a P2P federation of autonomous peers, and can form even stronger hybrids with the existing algorithms from distributed IR.

3.1.2 Result Merging

The second key issue (which is only of peripheral interest to this thesis) is *result merging*: when the peers that have been selected during query routing return their top-ranked local results, these results have to be combined into a single, comprehensively ranked result list, which is eventually displayed to the user. As we are dealing with the local query execution results of a large number of autonomous peers, each peer individually has the freedom to deploy its favorite document scoring model, rendering scores mutually incompatible and incomparable. Even if they had agreed on a common document scoring model, most of them rely on (global) statistical knowledge that is not readily available for a large-scale distributed system. The obvious solution, the usage of local statistics, again leads to scores that are inherently incomparable across peer boundaries.

We have identified four general approaches to tackle result merging:

- **objective document scoring:** if all peers agree on a common scoring function that *exclusively* utilizes "objective" ingredients (e.g., term frequencies), scores are immediately comparable across peer boundaries, and result merging simply involves sorting the combined result list of all peers by document scores. Additionally, if the same documents are returned by several peers, any combination of document scores could be applied (e.g., $sum()$, $min()$, $max()$ [CCH92]).

- **work with individual rankings:** instead of trying to perform the merging step based on document scores, merge the local result rankings by traversing them. Beyond the most obvious approach to pick one document at a time from each list in a round-robin manner, one could easily come up with more sophisticated techniques that bias the scheduling of

rankings in favor of certain peers, e.g., according to their overall quality determined by the query routing phase.

- **initiator re-scoring:** the query initiator might choose to consider the result lists retrieved from the remote peers as a candidate set only and apply any document scoring model to score and eventually rank the documents. This process could either be based on raw statistics that are shipped along with the result lists (as an extreme case, the actual documents), or on subsequent retrieval of all candidate documents by the query initiator.

- **estimating global statistics:** in order to overcome the incompatibility of scores due to the unavailability of (global) scoring ingredients (most notably global (inverse) document frequencies), the peer collaboration tries to estimate these values and local peers use these estimates to produce compatible document scores. More details on an algorithm to globally estimate (inverse) document frequencies can be found in Section 8.3.

3.2 The Minerva System

3.2.1 System Architecture

The P2P Web search prototype system Minerva[4] was developed in the course of this work to serve as a valuable research testbed and to evaluate novel approaches. It assumes a P2P collaboration in which every *data 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 index lists with URLs for Web pages that contain specific keywords.

A conceptually global but physically distributed directory, which is loosely layered on top of a dynamic hash table (DHT)[5], holds compact, aggregated information about the peers' local indexes and only to the extent that the individual peers are willing to disclose. The DHT partitions the key space, such that each *directory peer* is responsible for maintaining the metadata about a randomized subset of keys. The two roles are not distinct; typically, the same physical peer would act both as a data peer and as a directory peer (there are no dedicated directory peers). For failure resilience and availability, the metadata may be replicated across multiple directory peers.

Directory maintenance, query routing, and query processing work as follows. First, every peer publishes a number of term-specific statistical summaries (*Posts*) describing its local index to the directory (cf. Figure 3.1 (left)). Posts contain contact information about the peer who published this summary together with statistical information to support appropriate query routing strategies (e.g., the size of the inverted list for the key, the maximum average score among the key's inverted list entries, or various other statistical synopses). The *lookup()* functionality of the DHT is used to determine the directory peer that

[4] Minerva is the Roman goddess of wisdom, and also the icon of the Max-Planck Society.

[5] While Chord is used for illustrative purposes throughout this thesis, Minerva has also been deployed on top of other DHT's, e.g., Pastry.

is currently responsible for a term. That peer maintains a *PeerList* of all Posts for this term from peers across the network.

If the results of a local query execution are unsatisfactory, the user can utilize the metadata directory to identify more promising peers for this particular query as follows (cf. Figure 3.1 (middle)): for each query term, the query initiator identifies the peer that is currently maintaining the appropriate PeerList, using the *lookup()* functionality of the DHT. Subsequently, the query initiator retrieves the relevant Posts by issuing *PeerList requests* directly to these peers. The statistical synopses contained in the Posts are used to perform query routing, i.e., to identify the most promising peers for that particular query based on the peers' metadata.

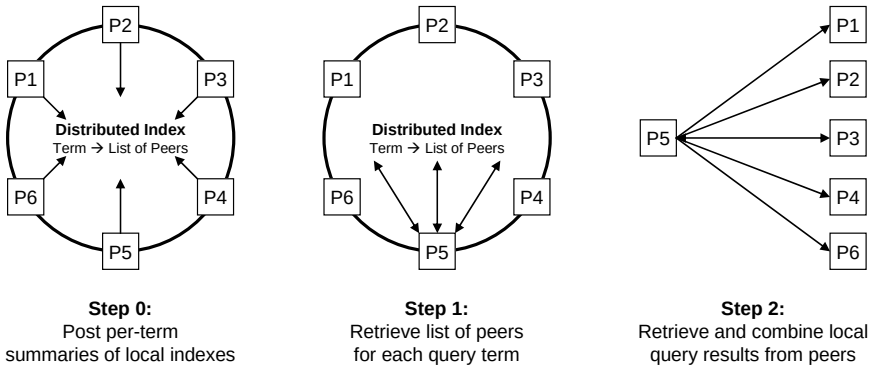

Figure 3.1: Minerva posting, routing, and querying steps

After a small subset of peers has been selected, the query is forwarded and executed based on their local indexes (cf. Figure 3.1 (right)). Note that this communication is carried out in a pairwise point-to-point manner between the peers, allowing for efficient communication and limiting the load on the global directory. Finally, the remote peers return their local results to the query initiator, where they are combined into a single result list (result merging).

The goal of finding high-quality search results with respect to precision and recall in general 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.

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 [BMWZ04], information about relevance assessments (e.g., derived from peer-specific query logs or click streams) [KLFW06], or (implicit or explicit) user feedback. This information could be leveraged when executing a query to further enhance result quality.

3.2.2 Implementation

The implementation of Minerva is realized platform-independently, using Java 5, and closely follows the conceptual design as above. Figure 3.2 illustrates the software architecture of a peer. Each peer resides on top of a globally distributed directory which is organized as a distributed hash table (DHT). We decided for a loose coupling, not relying on peculiarities of any concrete DHT implementation, but only assuming a *lookup()* functionality that can provide a mapping from keys to peers. While early versions of Minerva relied on a home-brewed reimplementation of the Chord protocol, more recent versions run as a FreePastry application [RD01], using Pastry's network routing mechanisms and Past's storage functionalities to become resilient to network dynamics and node failures. Each peer maintains a *PastryNode*, implementing the *PastryApplication* interface, and is registered at a Pastry *Endpoint*. Once registered, the PastryNode delivers incoming messages to the registered applications. These interdependencies are again illustrated in Figure 3.3. There exist two different implementations of Past, *PastImpl* and *GCPastImpl*. GCPastImpl is an extension of PastImpl that offers garbage collection based on timestamps. Minerva uses this latter version in order to prune out-dated metadata objects after a specific time interval.

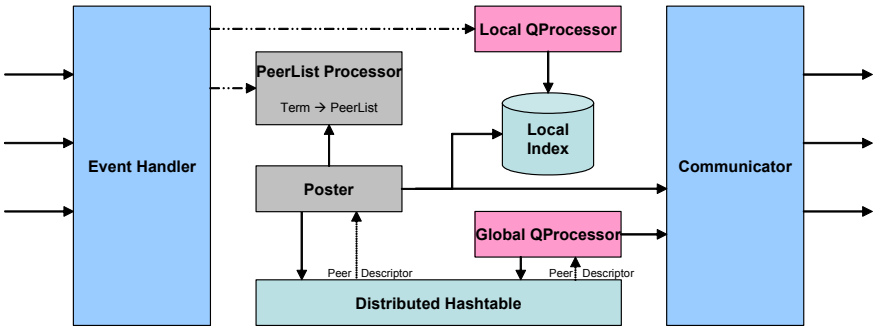

Figure 3.2: Minerva implementation

As a reply to *lookup(key)* requests, the DHT layer returns a *PeerDescriptor* object containing the information (e.g., IP address and port) necessary to contact the peer currently responsible for *key*. A *Communicator* is instantiated with this data to carry out the actual communication with remote peers. Each peer runs an *Event Handler* listener that receives incoming messages and forwards them to the appropriate local components.

Every peer has a local index that can be imported from external crawlers and indexers, such as our focused crawler BINGO! [STS+03]. Minerva can store the local index in any database system capable of executing standard SQL commands, and provides fully implemented JDBC interfaces for Oracle, MySQL, and Cloudscape/Derby. The index can be used for local query execution by our own *Local QueryProcessor* component or by more sophisticated packages for local query execution, such as the TopX engine [TSW05]. Additionally, the *Poster* component uses the local index to produce the term-specific summaries (*Posts)* that are published to the global directory using the Communicator and

the *lookup()* functionality of the DHT to identify the peer currently responsible for the term.

Each peer originally implemented a *PeerList Processor* to maintain the incoming Posts, i.e., all Posts from across the network regarding the subset of terms that the peer is currently responsible for. Since Past is designed to handle such inserts natively, the most recent version of Minerva uses the appropriate functionality of Past, benefiting from its sophisticated mechanisms regarding redundant storage and caching. However, Past currently does not offer a bulk insertion functionality, which is highly desirable in order to send multiple metadata objects to the same peer in one single message, avoiding unnecessary message overhead. We have extended Past with such a functionality.

When the user initiates a query using Minerva, the *Global QueryProcessor* component uses the *lookup()* functionality of the DHT to find the peer responsible for each query term and retrieves the respective PeerLists using Communicator components. After appropriately processing these lists, the Global QueryProcessor forwards the complete query to selected peers, which in turn process the query using their Local QueryProcessors and return their results. Finally, the Global QueryProcessor merges these results and presents them to the user.

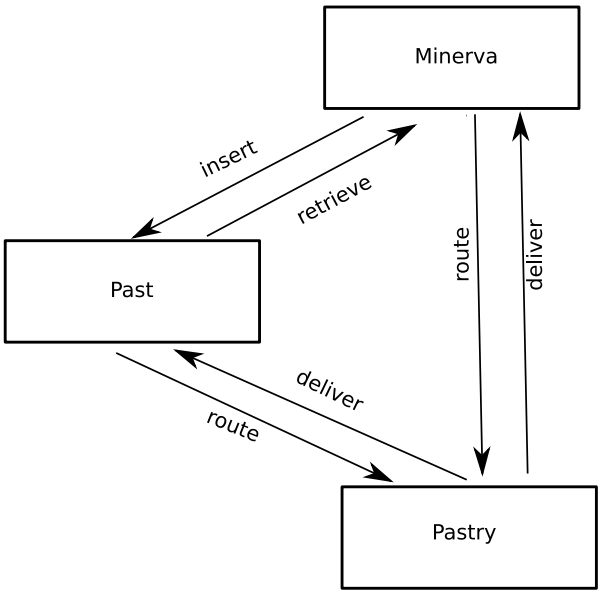

Figure 3.3: Past, Pastry, and Minerva interdependencies

Besides the insert/retrieve methods of Past, there are several cases where Minerva peers communicate directly, i.e., beyond the simple DHT lookup functionality. The most prominent and obvious case is the actual query execution when the query initiator sends the query to a few peers selected based on statistical metadata previously retrieved from the DHT directory.

3.2.3 Minerva in Action

This section showcases the usage of Minerva by means of some screenshots taken from the latest stable version, coined MinervaLight[6]. It also serves as a short manual demonstrating the *data lifecycle* within Minerva, i.e., from harvesting Web data, over publishing metadata to the distributed directory, to executing distributed queries.

MinervaLight combines the (previously separate) focused crawler BINGO! (to harvest Web data), the local search engine TopX, and Minerva under one common user interface. The crawler unattendedly downloads and indexes Web data, where the scope of the *focused* crawl can be tailored to the thematic interest profile of the user. The result of this process is a local search index, which is used by TopX to evaluate user queries.

In the background, MinervaLight continuously computes compact statistical synopses that describe a user's local search index and publishes that information to a conceptually global, but physically fully decentralized directory. MinervaLight offers a search interface where users can submit queries to Minerva. Sophisticated query routing strategies are used to identify the most promising peers for each query based on the statistical synopses in the directory. The query is forwarded to those judiciously chosen peers and evaluated based on their local indexes. These results are sent back to the query initiator and merged into a single result list.

Importing Bookmark Files

In order to populate the local index of each peer, MinervaLight allows the import of bookmark files to build a local index according to the users' personal interest profiles (cf. Figure 3.4). These files contain URLs, which by means of an intuitive XML syntax can optionally be categorized into a personal topic hierarchy; MinervaLight determines the discriminative features of the documents in each category to build a classifier, so it can also categorize all new pages it encounters during the Web crawl accordingly. All applicable content (currently txt, html, and pdf) is added to the local search index. Optionally, the crawl can also be limited to a certain crawl depth or to certain hosts.

Alternatively, the user can manually enter a set of URLs as starting points for the Web crawl.

Crawling the Web

MinervaLight subsequently automatically starts its crawling activity; users can follow the progress of the Web crawl (cf. Figure 3.5), including the categorization of the encountered documents. The user may interrupt the Web crawl at any point or let it continue to run as a background process.

The left part of the user interface actually shows the topic hierarchy that was imported with the bookmark file in the previous step. All pages that are encountered at the current Web crawling phase are automatically assigned to one category (or left uncategorized, if no category fits well enough).

[6] Available at: *http://www.minerva-project.org*

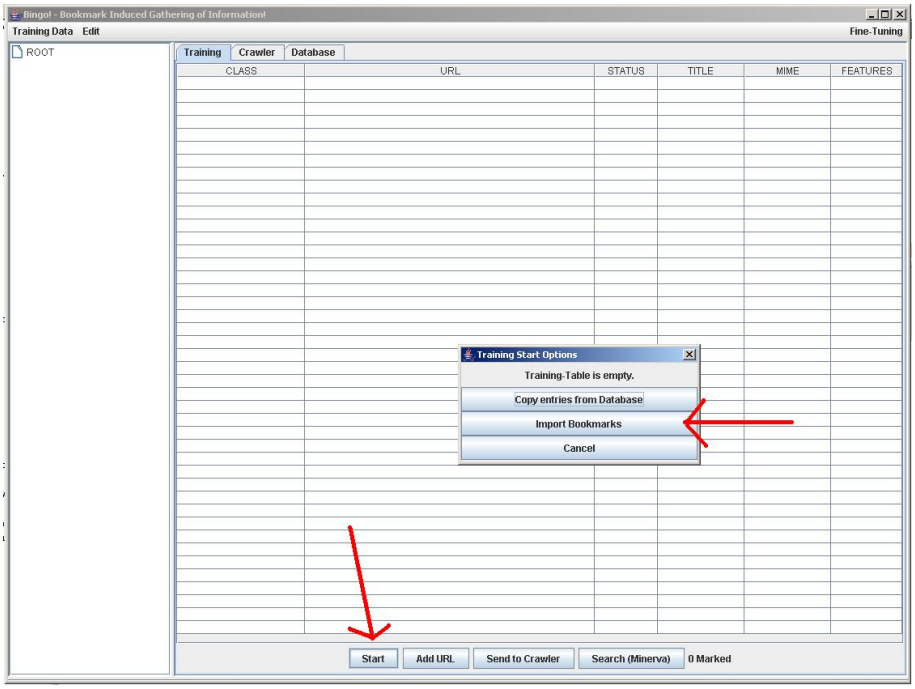

Figure 3.4: Import bookmark files

Instantiating Minerva

Having built up a local search index, the next step is to either create a new Minerva network, or to join an existing network by means of providing any existing node as a bootstrap. For this purpose, MinervaLight automatically tries to determine the most important network settings, and the user typically only needs to provide a nickname as a unique identifier within the network (cf. Figure 3.6).

Updating the Minerva Index

Users can update the data that Minerva uses to build its statistical information with the latest documents obtained when crawling the Web by means of an intuitive "Update Index" button at the bottom of the window (cf. Figure 3.7).

Publishing Metadata

The next step is to disseminate the statistical metadata describing the locally indexed data to the distributed directory (cf. Figure 3.8). Users can also inspect the share of metadata they receive from the network, i.e., their "portion" of the distributed directory.

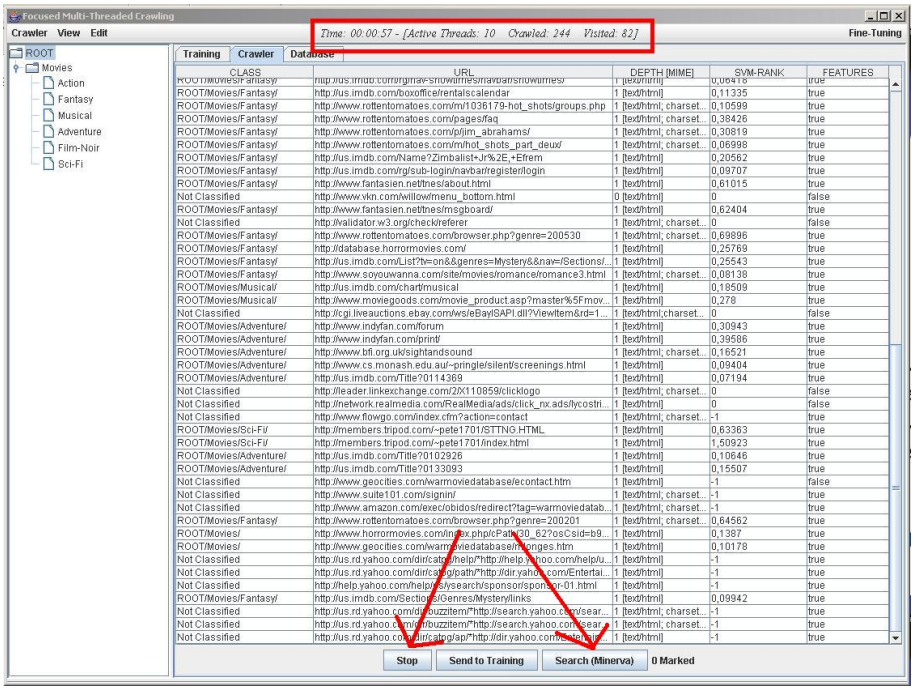

Figure 3.5: Web crawl

Executing Queries

Finally, the user can enter a keyword query into an input field. Minerva will automatically retrieve the applicable metadata for this query from the directory, identify the most promising peers for this query, forward the query accordingly, and eventually return a ranked list of relevant results (cf. Figure 3.9). Users can browse the result list, including the document titles and further document metadata, and open the result document in a Web browser.

Annotating Documents

An additional feature of Minerva is to let the user annotate (*tag*) local as well as remote documents; annotations are *tag=value*-pairs that can for example be used to express the personal opinion on the quality of the document (*quality=good*). Users can simply right-click any result that is displayed in the query results pane, choose *Add New Tag for URL*, and enter arbitrary annotations (cf. Figure 3.10). For the users' convenience, tags previously used to annotate the same document are automatically suggested.

Retrieving Annotations for Document

Existing annotations to documents can be leveraged by the user in different ways. One obvious way is to inspect the annotations that have previously been assigned by other users in the network to a given document. This use-case is readily supported by Minerva; right-clicking any result that is displayed in

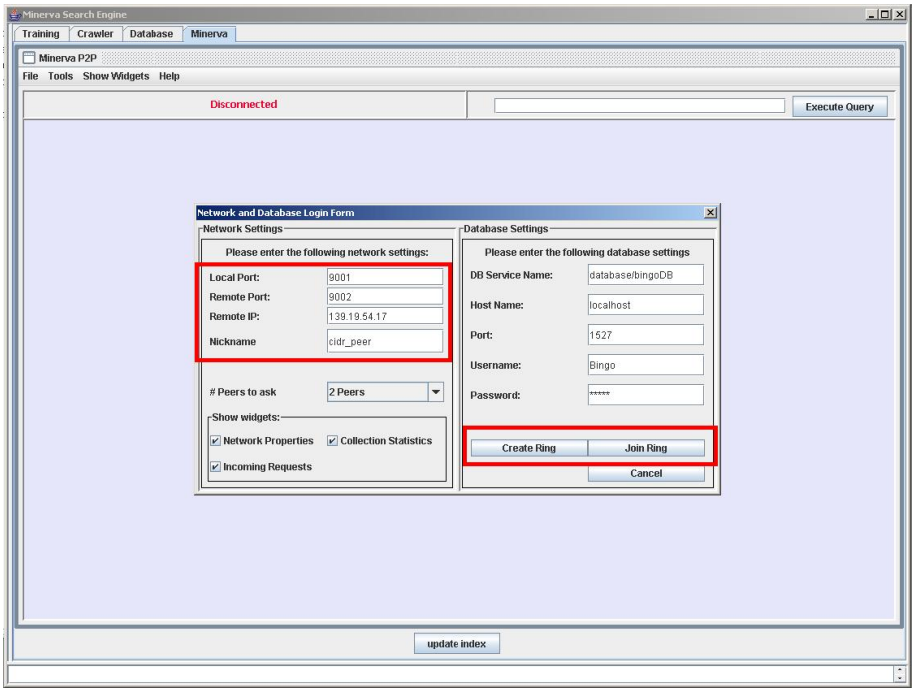

Figure 3.6: Instantiate Minerva

the query results pane and choosing *Retrieve All Tags for URL* retrieves all applicable annotations for the given document from the directory and displays them to the user.

Querying for Annotated Documents

Existing annotations can also intuitively be used to refine user queries. By simply entering the appropriate pairs into the query form, e.g. *quality=good* (possibly in combination with plain keywords), Minerva applies its regular query routing approaches to find relevant results that have been tagged accordingly.

3.3 Other P2P Web Search Prototypes

In the following we briefly discuss some prior and ongoing projects towards P2P Web search.

Galanx [WGD03] is a P2P search engine implemented using the Apache HTTP server and BerkeleyDB. The Web site servers are the peers of this architecture; pages are stored only where they originate from. In contrast, our approach leaves it to the peers to what extent they want to crawl interesting fractions of the Web and build their own local indexes.

PlanetP [CAPMN03] is a publish-subscribe service for P2P communities, supporting content ranking search. PlanetP distinguishes local indexes and a global index to describe all peers and their shared information. The global index

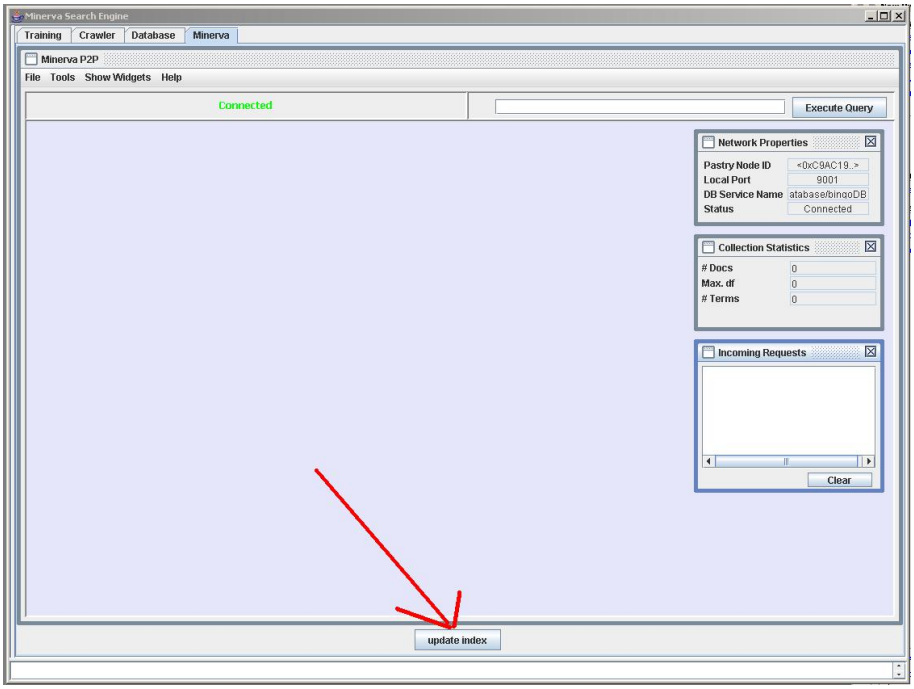

Figure 3.7: Update Minerva index

is replicated using a gossiping algorithm.

Odissea [SMwW+03] assumes a two-layered search engine architecture with a global index structure distributed over the nodes in the system. A single node holds the complete, Web-scale, index for a given text term (i.e., keyword or word stem). Query execution uses a distributed version of Fagin's threshold algorithm [Fag02]. The system appears to cause high network traffic when posting document metadata into the network, and the presented query execution method seems limited to queries with at most two keywords. The paper actually advocates using a limited number of nodes, in the spirit of a server farm.

The work presented in [RV03] adopts an architecture very similar to Minerva, but — to our knowledge — has never been implemented. The results presented in the paper seem to be based on simulations that also support our assumption that a Minerva-like architecture does in fact scale and is well within reasonable bandwidth limits. Again to our knowledge, there are no more recent follow-up papers.

The eSearch system [TD04] is a P2P keyword search system based on a hybrid indexing structure in which each node is responsible for certain terms. Given a document, eSearch selects a small number of important terms in the document and publishes the complete term list for the document to nodes responsible for those top terms. This selective replication of term lists allows a multi-term query to proceed local to the nodes responsible for query terms, but the document granularity may interfere with the goal of unlimited scalability.

[GWJD03] focuses on XML data repositories and postulates that, upon completion of the query, regardless of the number of results or how they are ranked

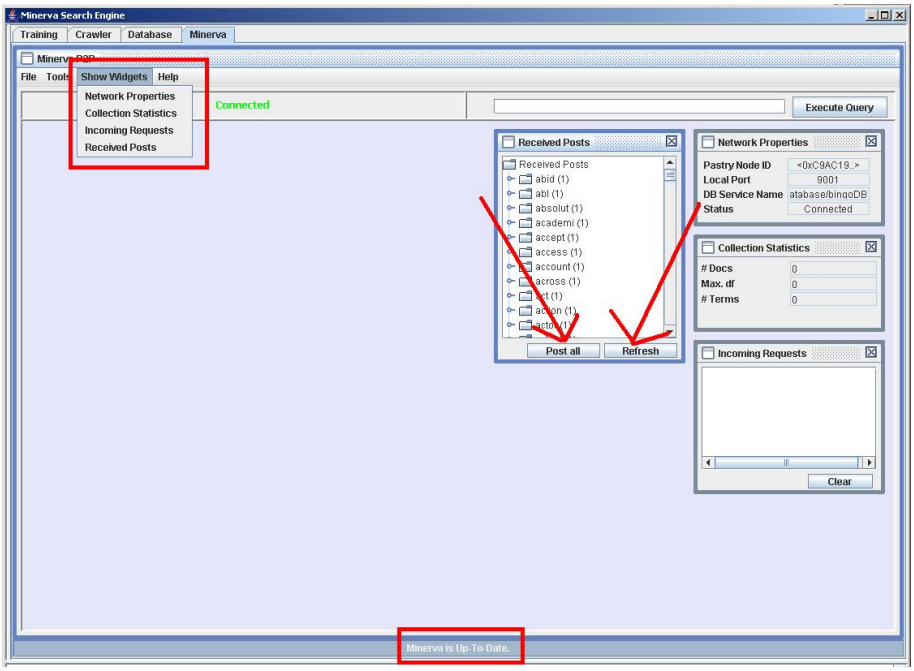

Figure 3.8: Publish metadata

and presented, the system guarantees that all the relevant data sources known at query submission time have been contacted. For this purpose, a distributed catalog service is designed that maintains summaries of all nodes.

Rumorama [MEH05] is an approach based on the replication of peer data summaries via rumor spreading and multicast in a structured overlay. Rumorama achieves a hierarchization of PlanetP-like summary-based P2P-IR networks. In a Rumorama network, each peer views the network as a small PlanetP network with connections to peers that see other small PlanetP networks. Each peer can choose the size of the PlanetP network it wants to see according to its local processing power and bandwidth.

Alvis [LKP+06] is a prototype for scalable full-text P2P-IR using the notion of *highly discriminative keys* for indexing, which claims to overcome the scalability problem of single-term retrieval in structured P2P networks. Alvis is a fully-functional retrieval engine built on top of P-Grid. It provides distributed indexing, retrieval, and a content-based ranking module. While the index size is even larger than the single term index, the authors bring forward that storage is available in P2P systems as opposed to network bandwidth.

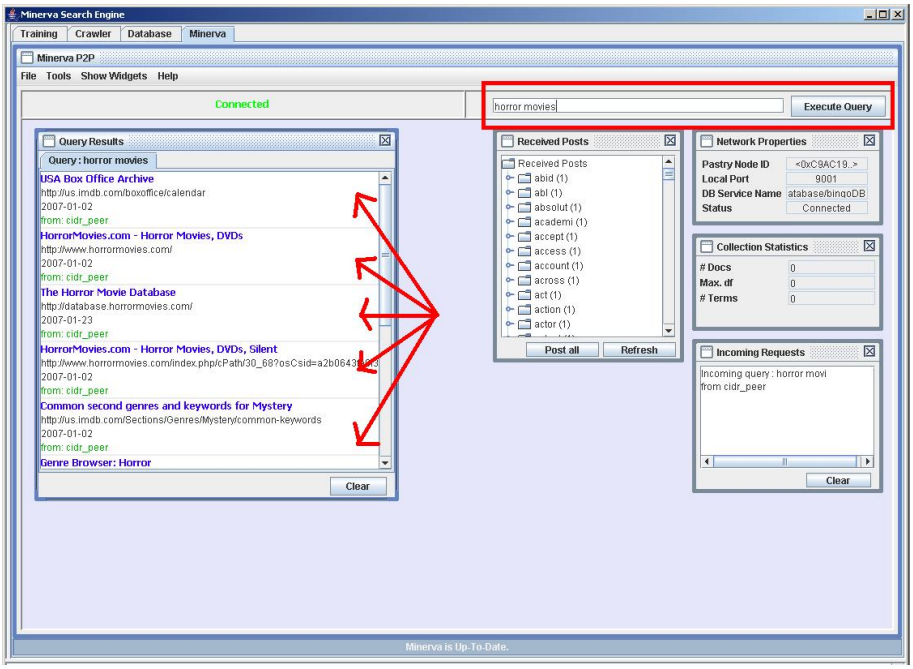

Figure 3.9: Execute query

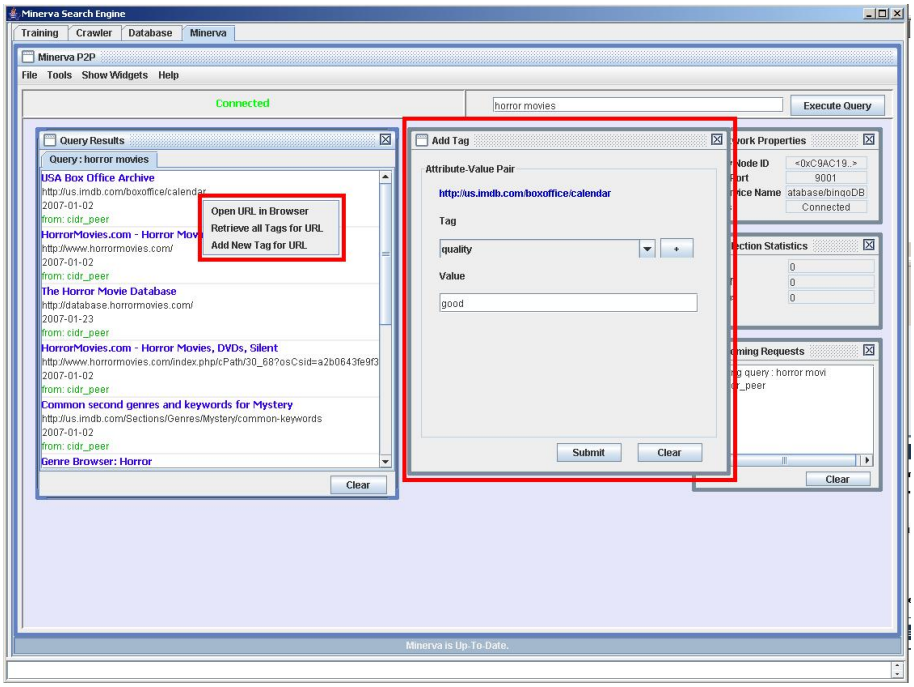

Figure 3.10: Annotate document

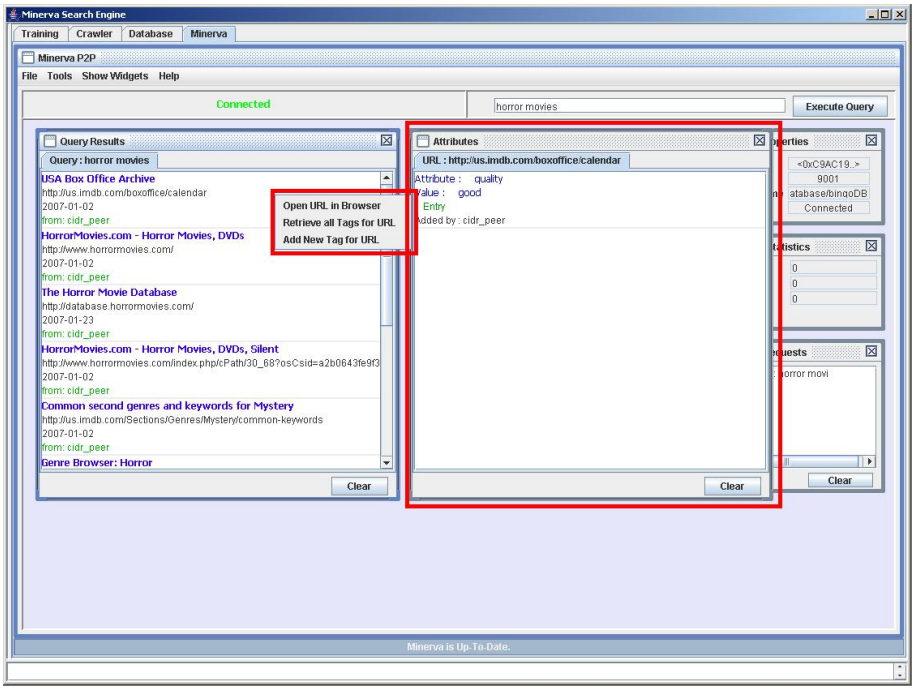

Figure 3.11: Retrieve annotations

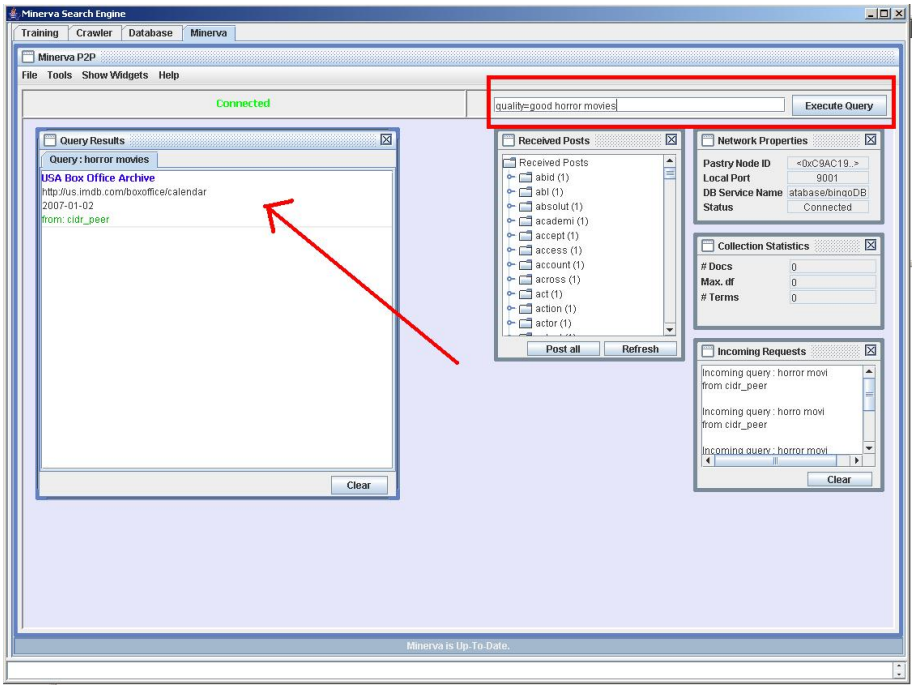

Figure 3.12: Query annotated documents

Chapter 4

Evaluation of Existing Approaches to Query Routing

As discussed in the previous chapter, given the large-scale data distribution of a P2P search network, one of the key technical challenges is *query routing*, which is the process of efficiently selecting the most promising peers (from a potentially very large set of peers storing relevant data) for a particular information need. The similar problem of resource selection has been studied extensively in the literature in the context of metasearch engines. These approaches were typically designed for a small and rather static set of search engines and did not consider the challenges present in a P2P network, such as peer autonomy, network dynamics and high inter-node latencies. Nevertheless, they serve as a starting point and as a baseline for this work on query routing in a P2P search collaboration.

Good surveys over existing approaches to query routing can be found in [MYL02, NF06, BMWZ05], which serve as a reference for the following sections describing a (non-exhaustive) set of popular approaches.

4.1 Existing Approaches

4.1.1 CORI

The **Co**llection **R**etrieval **I**nference Network (CORI) [CLC95, SJCO02] based on inference networks reduces the task of query routing to a document retrieval task, where a *superdocument* as the representative of a collection is the concatenation of all documents of one collection, containing all distinct terms in the collection. If a term appears in k documents in the collection, the term appears k times in the superdocument. The set of all superdocuments forms a special-purpose collection that is used to identify the most promising collections for a given query. In principle, standard approaches (e.g., tf*idf and cosine [cf. Section 2.2]) could now be applied. From the viewpoint of regular document scoring and retrieval, term frequencies are replaced by document frequencies, and document frequencies by *collection frequencies*, i.e., the number of collections that

contain a specific term. The approach taken by CORI applies inference networks [TC91] to compute the ranking score of a collection with respect to query q as the estimated belief that the database contains useful documents. The belief is essentially the combined probability that the database contains useful documents due to each query term.

More specifically, the belief that the information need expressed by a query term t is satisfied by searching collection c is determined by

$$T := \frac{df}{df + 50 + 150 * \frac{cl}{avgcl}}$$

$$I := \frac{log(\frac{N+0.5}{cf})}{log(N+1)}$$

$$Pr(t|c) := 0.4 + 0.6 * T * I$$

with N being the total number of collections, cf the collection frequency, cl the number of terms in the collection, and $avgcl$ the average number of terms in a collection. The values of some constants have already been inserted into the formulae as determined by empirical experiments [CLC95, SJCO02]. Note that, essentially, $Pr(t|c)$ resembles the tf*idf score of term t in the superdocument of c. Finally, the belief that c contains useful documents with respect to query q can be computed as a sum over all query terms, possibly weighted by the significance of a term in representing the query.

Compared to the original application of inference networks to document scoring, document nodes are replaced by the superdocuments, yielding a moderate-sized network. The frequency values are typically higher, but that does not affect the computational complexity. However, in a highly dynamic network of (nevertheless cooperating) peers, the proper estimation of N, $avgcl$, and cf is a non-trivial problem; a possible approach to overcome this issue will be discussed in Section 8.3.

4.1.2 gGlOSS

The gGlOSS (**g**eneralized **Gl**ossary **o**f **S**ervers' **S**erver) system [GGMT94, GGM95] represents the usefulness of a collection with respect to term t by the document frequency df_t for t and the sum of the weights of t over all documents in the collection, W_t. Additionally, each query is associated with a threshold T that indicates that only documents whose similarities with the query are higher than T are of interest.

gGlOSS suggests two different methods to calculate the usefulness of a collection. We focus on the method based on the *high-correlation assumption*, as it has been shown to be superior to the *disjoint assumption* for scenarios like ours where both precision and recall are of interest [GGMT99]. The high-correlation assumption states that, for any given collection, if query term t_i appears in at least as many documents as query term t_j, then every document containing t_j also contains t_i. Under this assumption, more specifically, the usefulness of collection c for a query q with threshold T is computed as follows: Let terms be arranged in ascending order of document frequencies, i.e., $df_i \leq df_j$ for any $i < j$, where df_i is the document frequency of term t_i. This means that every

document containing t_i also contains t_j for any $j > i$. There are df_1 documents having similarity $\sum_{i=1}^{k} q_i * \frac{W_i}{df_i}$ with q, where k is the number of query terms and q_i the weight of term t_i in the query. In general, there are $df_j - df_{j-1}$ documents having similarity $\sum_{i=j}^{k} q_i * \frac{W_i}{df_i}$ with q, $1 \leq j \leq k$, and df_0 is defined to be 0. Let p be an integer between 1 and k that satisfies $\sum_{i=p}^{k} q_i * \frac{W_i}{df_i} > T$ and $\sum_{i=p+1}^{k} q_i * \frac{W_i}{df_i} \leq T$. Then the estimated usefulness of a collection is calculated as

$$\sum_{j=1}^{p}(df_j - df_{j-1}) * \left(\sum_{i=j}^{k} q_i * \frac{W_i}{df_i}\right)$$

$$= \sum_{j=1}^{p} q_j * W_j + df_p * \sum_{j=p+1}^{k} q_j * \frac{W_j}{df_j}$$

As the usefulness of a collection is sensitive to the threshold T, gGlOSS can, unlike CORI, differentiate a collection with many moderately similar documents from a collection with a few highly similar documents. However, the estimated usefulness may be inaccurate due to the high-correlation assumption that does not hold in most real-world scenarios.

4.1.3 Decision-Theoretic Framework

In contrast to the above collection selection approaches which only consider the similarity of a collection to a query, the decision-theoretic framework (DTF) [NF03, Fuh99] estimates real "costs" from different sources (e.g., monetary costs, communication time, ...). As the actual costs are unknown in advance, *expected* costs for a collection c when s_c documents are retrieved for a query q are regarded instead:

$$EC_c(s_c, q) :=$$

$$E[r_c(s_c, q)] * C^+ + [s_c - E[r_c(s_c, q]] * C^- + C^t * EC^t(s_c) + C^m * EC^m(s_c)$$

where $E[r_c(s_c.q)]$ is the expected number of relevant documents among the s_c top-ranked documents and $s_c - E[r_c(s_c, q)]$ is the expected number of non-relevant documents, EC^t denotes the expected "time" costs, and EC^m is the expected monetary costs. In addition, C^+, C^-, C^t, and C^m are user-specific parameters which allow a user to specify her own selection policy, e.g., fast and cheap results. Non-relevant documents have higher costs (wasted time) than relevant ones, thus $C^+ < C^-$.

A user also specifies the total number n of documents to be retrieved out of m collections, and the task is to compute an optimum solution for $s_1, ..., s_m$, the number of documents retrieved from collections $1...m$, such that $\sum_{i=1}^{m} s_i = n$ and $\sum_{i=1}^{m} EC_i(s_c, q)$ is minimal.

Relevance costs are computed in two steps:

1. First, the expected number $E(rel|q, c)$ of relevant documents in a collection is computed based on statistical aggregations of the collection.

2. Then, a linearly decreasing approximation of the recall-precision function is assumed for computing the expected number $E[r_c(s_c, q)]$ of relevant retrieved documents.

For the first step, the statistical aggregations consist of the collection size $|c|$ and the average (expectation) $\mu_t = E[w(d,t)|d \in c]$ of the indexing weights $w(d,t)$ (for document d and term t). For a query with term weights $a(q,t)$ (summing up to one), the expected number $E(rel|q,c)$ of relevant documents in c for a query q can be estimated as:

$$E(rel|q,c) = \sum_{d \in c} Pr(rel|q,d) \approx \sum_{d\ inc} \sum_{t\ inq} a(q,t) * w(d,t) = |c| * \sum_{t \in q} a(q,t) * \mu_t$$

where $Pr(rel|q,d)$ denotes the probability that document d is relevant. In a second step, $E(rel|q,c)$ is mapped onto the expected number $E[r_c(s_c,q)]$ of relevant retrieved documents employing a linearly decreasing recall-precision function:

$$\frac{E[r_c(s_c,q)]}{s} = precision(recall) := 1 - recall = 1 - \frac{E[r_c(s_c,q)]}{E(rel|q,c)}$$

DTF has a better theoretic foundation (selection with minimum costs) than traditional resource ranking algorithms like CORI, considers additional cost sources like time and money, and computes the number of peers to be queried as well as the number of documents which should be retrieved from each of these libraries. In contrast, heuristic methods like CORI compute a ranking of peers, and additional heuristics are needed for determining the number of peers and the number of documents to be retrieved. While the retrieval quality of DTF has been shown to be competitive with CORI, it requires several input parameters, e.g., for expected costs, which are hard to determine. For a rather static set of well-managed peers that may involve monetary cost, e.g., in a Digital Library setting, the parameter choice may be more feasible, but we consider the application of DTF in a setting with highly-dynamic, unmanaged, and autonomous peers very difficult or even infeasible.

4.1.4 Statistical Language Models

Language Modeling has a long record of successful applications in the fields of speech recognition and statistical natural language processing and has also been applied to different problems in information retrieval, e.g. in [SC99, PC98, SJCO02, ZL01, MLS99]. All these approaches share the common idea that each document is seen as a sample generated from a special language. After estimating a language model for each document beforehand, the relevance for a document to a given query (*document-query* similarity) is computed as the likelihood that the query can be generated from the language model for that document. More formally, the likelihood for a query Q to be generated from a document D is computed as

$$P(Q|D) = \prod_{t \in Q} \delta P(t|D) + (1-\delta)P(t|C))$$

where t is a query term in Q, P(t|D) is the probability for a query term t to appear in D, P(t|C) is the probability for t to be used in the collection C to which D belongs, and δ is a weighting parameter. Note that the notion of smoothing introduced by the term $P(t|C)$ (in particular when $P(t|D)$ is zero) is similar to the *idf* term in *tf*idf* document scoring.

In order to take advantage of these scores, the idea when computing collection scores is to collapse (potentially a sample of) the documents for a collection together as one *superdocument* and to perform the similar computation for the *collection-query* similarity as

$$P(Q|C) = \prod_{t \in Q} \delta P(t|C) + (1 - \delta) P(t|G)$$

where $P(t|G)$ is the global language model. Collections with the largest generation probabilities $P(Q|C)$ will be selected as the most relevant collection. The approach presented in [XC99], based on Kullback-Leibler (KL) divergence, can also be reduced to the same equation, with an additional query-specific constant.

Compared to the CORI or gGlOSS, language modeling tends to be better justified by probability theory. However, the choice of δ and also the computation of the global language model remain serious problems. While δ has typically been set to values between 0.5 and 0.7, query sampling has commonly been applied to overcome the problem of constructing a global collection.

4.2 Experiments

4.2.1 Experimental Setup

Experiments are conducted on collections that have been created by Web crawls originating from manually selected crawl seeds on the topics Sports, Computer Science, Entertainment, and Life, leading to 10 thematically focused collections. Additionally, one reference collection was created by combining all collections and eliminating duplicates. Table 4.1 gives details about the collections. Note the overlap between the 10 original collections.

For the query workload we consider the 7 most popular queries on AltaVista, as reported by http://www.wordtracker.com for September 21, 2004, and 3 additional queries that were specifically suitable for our corpus. Table 4.2 lists all queries.

For each query we obtain an *ideal peer ranking* as follows: the query is executed on the reference collection with the measures introduced in Section 2.2 to obtain a reference query result. Subsequently, the query is executed on each of the collections individually, using the same strategy. These local results are compared to the reference query result using the *rank distance* function described in detail in Section 4.2.2. We order the peers in descending order of these distances to obtain the ideal peer ranking.

We evaluate different query routing strategies by comparing their peer selection results (peer rankings) to this ideal peer ranking, again using our rank distance function. Figure 4.1 illustrates this experimental setup.

We have a number of system parameters that influence the experimental results:

Collection	# Documents	Collection Size in MB
Computer Science	10459	137
Life	12400	144
Entertainment	11878	134
Sport	12536	190
Computer Science mixed	11493	223
Computer Science mixed	13703	239
CS Google	7453	695
Sport Google	33856	1,086
Life Google	16874	809
Entertainment Google	18301	687
Σ	168953	4,348
Combined Collection	142206	3,808

Table 4.1: Collection statistics

Max Planck Light Wave Particle*	Einstein Relativity Theory*
Lauren Bacall	Nasa Genesis
Hainan Island	Carmen Electra
National Weather Service	Search Engines
John Kerry	George Bush Iraq*

Table 4.2: List of queries (* denotes queries *not* taken from WordTracker)

- Number of peers returned by peer selection strategies
- Number of documents retrieved from each peer
- Number of documents retrieved from combined collection (ideal document ranking)
- Number of peers in ideal peer ranking

All experiments have been conducted using 10 peers running as separate processes on a single notebook with a Pentium M 1.6GHz processor and 1GB main memory. All peers share a common Oracle $10g$ database that is installed on a Dual-Pentium Xeon 3GHz processor with 4GB main memory. The peers are connected to the database through a 100MBit network.

4.2.2 Rank Distance Function

Formally, a ranking σ is a bijection (i.e., a permutation) from a domain D_σ to $[k]$ where $|D_\sigma| = k$ and $[k] := \{1, ..., k\}$. Metrics comparing permutations have intensively been studied for a long time. One of the most prominent metrics is *Spearman's footrule metric* [DG77, Dia88] that calculates the difference between two permutations σ_1 and σ_2 with $D = D_{\sigma_1} = D_{\sigma_2}$ as follows:

$$F(\sigma_1, \sigma_2) := \sum_{i \in D} |\sigma_2(i) - \sigma_1(i)| \tag{4.1}$$

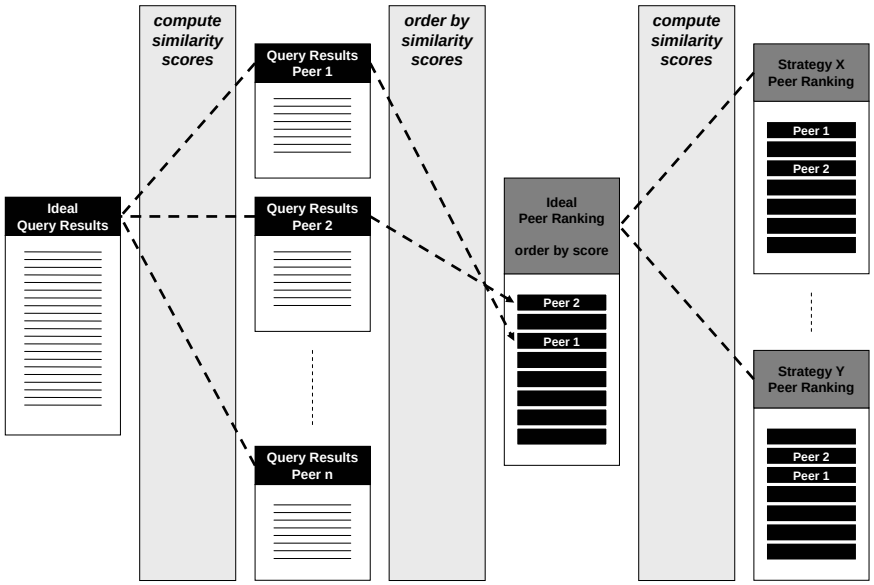

Figure 4.1: Experimental setup

In our case, the domains D_{σ_1} and D_{σ_2} are not necessarily identical. Thus, we can neither apply Spearman's footrule metric (4.1) nor other popular techniques like *Kendall's tau* [KG90].

[FKS03] gives an overview about metrics on comparing top-k lists representing incomplete rankings by presenting modifications of the well-known metrics for permutations. One possibility to apply metrics for permutations on top-k lists is to extend the top-k lists to complete rankings over a shared domain, that is, the union of the domains of the top-k lists.

We propose the following formula to calculate the distance between two rankings σ_1 and σ_2:

$$F'(\sigma_1, \sigma_2) := \sum_{i \in D_{\sigma_2}} |\sigma_2(i) - \sigma_1(i)| \tag{4.2}$$

Although this formula closely resembles Spearman's footrule metric, it sums only over elements that are contained in D_{σ_2}. However, (4.2) is only valid if D_{σ_1} is a superset of D_{σ_2} because $\sigma_1(i)$ is undefined for all $i \in D_{\sigma_2} \setminus D_{\sigma_1}$. Thus, we need to define an extension ${\sigma_1}'$ of σ_1 as follows:

$${\sigma_1}'(i) := \begin{cases} \sigma_1(i) & i \in D_{\sigma_1} \\ |D_{\sigma_1}|+1 & i \notin D_{\sigma_1} \end{cases}$$

Note that we do not need an extension of σ_2, because we sum only over elements in D_{σ_2}. This causes F' to be asymmetric.

Another important issue appears when trying to evaluate several rankings of different sizes to one reference ranking. To avoid unfair comparison of short

result lists with good matches against long result lists with not so good matches at low ranks, we required a minimum size for each query result list from each peer and supplemented shorter lists by "dummy" documents that would correspond to the rank $|D_{\sigma_1}| + 1$ in the reference collection.

4.2.3 Experimental Results

Figure 4.2 shows the rank distances between the peer rankings returned by the peer selection strategies and the ideal peer ranking. The distances for each strategy are averaged over the 10 queries in our benchmark, based on the rank distance definition given in Section 4.2.2. For these test queries, CORI produced the best results and clearly outperformed gGlOSS and the approach based on language modeling. An ad-hoc $cdf - ctf^{max}$ approach (cf. [BMWZ05]) also worked remarkably well provided that α was not set too low. The poor performance of the language modeling approach contradicts the findings by [SJCO02], but this effect may be heavily dependent on the specific nature of the queries and the distributed corpora (e.g., the degree to which corpora of different peers may overlap).

Using the same set of collections and queries, we studied the recall relative to the top-30 documents from the combined collection when we query the peers in descending order of their positions in the ideal peer ranking. Figure 4.3 shows that sending the query to the best peer only already yielded an average of about 50% of all relevant documents, whereas the inferior peers typically do not contribute any new documents to the query result. Note that this does not mean they do not *contain* any relevant documents, but rather that their relevant documents have already been contributed by other peers before. So taking into account the potential overlap of the peers' local contents is crucial and will be addressed in a later chapter. Moreover, CORI is robust strategy and is typically not inferior no any other strategy, in particular for a small number of peers.

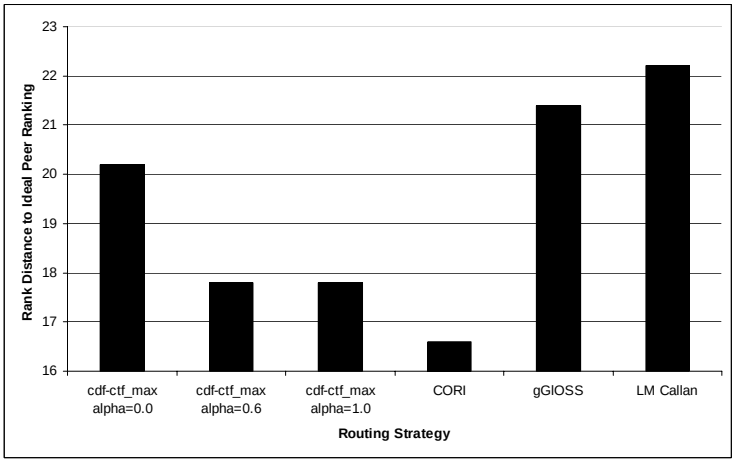

Figure 4.2: Experimental results

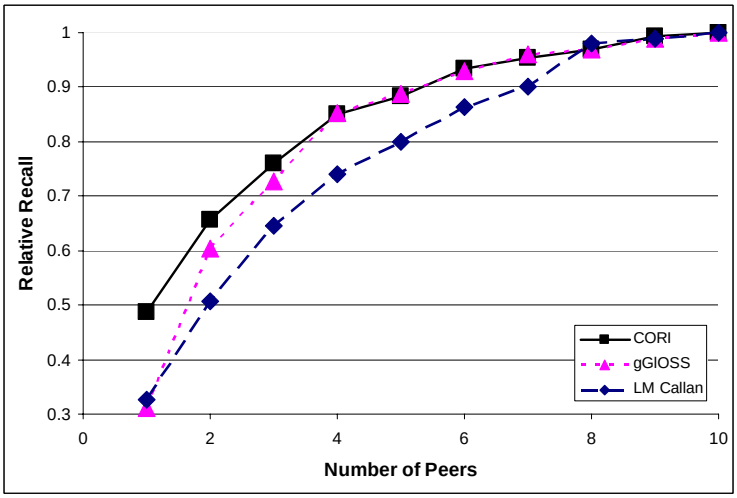

Figure 4.3: Experimental results

4.3 Conclusion

There exist several approaches to query routing that are able to identify promising information sources for a given query based on precomputed statistics. Even though the exact results vary in different publications, our experimental are totally accordant with other scientific works that identify CORI as a particular robust approach that works well in almost all application settings without the need for many hard-to-tune parameters (cf., e.g., [PF03, NF03]). None of the other methods could be shown to constantly outperform CORI, but often only in carefully designed experiments. Therefore, this thesis will use CORI as a baseline to quantify the improvements by our novel query routing strategies that aim at better capturing the peculiarities of a peer-to-peer collaboration.

Chapter 5

Overlap-Aware Query Routing

The approaches presented in the previous chapter typically ignore the fact that popular documents are replicated at a significant number of peers. Thus, these strategies often result in promising peers being selected because they share the same high-quality documents. Consider a single-attribute query for all songs by Mikis Theodorakis. If, as in many of today's systems, every selected peer contributes its best matches only, the query result will most likely contain many duplicates (of popular songs), when instead users would have preferred a much larger variety of songs from the same number of peers. Other application classes with similar difficulties include P2P sensor networks or network monitoring [HHL+03]. What is lacking is a technique that enables the quantification of how many *novel* results can be contributed to the query result by each of the prospective peers.

Contacting all prospective peers during query execution and exchanging the full information necessary to determine collection novelty is unacceptable due to the high cost in latency and network bandwidth. This chapter envisions an iterative approach based on compact statistical synopses, which all peers have precomputed and previously published to a (decentralized and scalable) directory implemented by a distributed hash table (DHT). The algorithm, coined *IQN routing* (for integrated quality and novelty) [MBTW06, BMT+05a], performs two steps in each iteration: first, the *Select-Best-Peer* step identifies the most promising peer regarding result quality *and* novelty based on the statistics that were posted to the directory. Then, the *Aggregate-Synopses* step conceptually aggregates the chosen peer's document collection with the previously selected peers' collections (including the query initiator's own local collection). This aggregation is actually carried out on the corresponding synopses obtained from the directory. It is important to note that this decision process for query routing does not yet contact any remote peers at all (other than for the, very fast DHT-based, directory lookups). The two-step selection procedure is iterated until some performance and/or quality constraints are satisfied (e.g., a predefined number of peers has been chosen).

The effectiveness of the IQN routing method crucially depends on appropriately designed compact *synopses* for the collection statistics. To support

the Select-Best-Peer step, these synopses must be small (for low bandwidth consumption, latency, and storage overhead), yet they must offer low-error estimations of the novelty by the peers' collections. To support the Aggregate-Synopses step, it must be possible to combine synopses published by different peers in order to derive a synopsis for the aggregated collection.

This chapter considers three kinds of synopses that each peer builds up and posts on a per-term basis, representing the globally unique identifies of documents (e.g., URLs or unique names of MP3 files) that a peer holds in its collection: Bloom filters [Blo70], hash sketches [FM85], and min-wise independent permutations [BCFM00]. These techniques have been invented for approximate, low-error representation of sets or multisets. We show how they can be adapted to a P2P setting and exploited for the highly effective IQN query routing.

We assume that each peer p_j locally maintains inverted index lists $I_j(t)$ with entries of the form $(docID, score)$, and posts for each term (or attribute value in a structured data setting) a set synopsis that captures the relevant documents that the local index of that peer contains for the term. These postings are kept in the DHT-based P2P directory for very efficient lookup by all peers in the network.

The specific contributions of this chapter are as follows:

- conducting a systematic study of Bloom filters, hash sketches, and min-wise permutations to characterize the suitability for the specific purpose of supporting query routing in a P2P system.
- developing the new IQN query routing algorithm that reconciles quality and novelty measures. We show how this algorithm combines multiple per-term synopses to support multi-keyword or multi-attribute queries in an efficient and effective manner.
- carrying out a systematic experimental evaluation, using real-life data and queries from TREC benchmarks, that demonstrate the benefits of IQN query routing (based on min-wise permutations) in terms of result recall (a standard IR measure) and query execution cost.

The rest of the chapter is organized as follows. Section 5.1 discusses related work. Section 5.2 introduces the different types of synopses and presents our experimental comparison of the basic techniques. Section 5.3 develops the IQN routing method in detail. Section 5.4 discusses special techniques for handling multi-dimensional queries. Section 5.5 describes possible extensions to exploit histograms on score distributions. Section 5.6 presents our experimental evaluation of the IQN routing method versus CORI as a state-of-the art approach (cf. Chapter 4), before Section 5.7 concludes this chapter.

5.1 Related Work

Fundamentals for statistical synopses of (multi-)sets have a rich literature, including work on Bloom filters [Blo70, FCAB00], hash sketches [FM85], and min-wise independent permutations [BCFM00]. We will overview these in Section 5.2.

There is relatively little work on the specific issue of overlap and novelty estimation. [ZCM02] addresses redundancy detection in a centralized information

filtering system; it is unclear how this approach could be made scalable in a highly distributed setting. [NKH03, HK05] present techniques to estimate coverage and overlap statistics by query classification and use a probing technique to extract features from the collections. The computational overhead of this technique makes it unsuitable for a P2P query routing setting where estimates must be made within the critical response-time path of an online query.

5.2 Collection Synopses for Information Retrieval

5.2.1 Measures

Consider two sets, S_A and S_B, with each element identified by an integer key (e.g., $docID$). The *overlap* of these two sets is defined as $|S_A \cap S_B|$, i.e., the cardinality of the intersection.

The notions of *Containment* and *Resemblance* have been proposed as measures of mutual set correlation [BCFM00] and can be used for our problem setting.

$$Containment(S_A, S_B) = \frac{|S_A \cap S_B|}{|S_B|}$$

is used to represent the fraction of elements in S_B that are already known to S_A;

$$Resemblance(S_A, S_B) = \frac{|S_A \cap S_B|}{|S_A \cup S_B|}$$

represents the fraction documents that S_A and S_B share with each other. If the intersection $|S_A \cap S_B|$ is small, so are containment and resemblance, and S_B can be considered a useful information source from the viewpoint of S_A. Note that resemblance is symmetric, while containment is not. Also, given $|S_A|$ and $|S_B|$ and either one of Resemblance or Containment, one can calculate the other [BCMR04].

However, none of these notions can fully capture the requirements of our system model. Specifically, we expect peers to have widely varying index list sizes. Consider now, for example, two collections S_A and S_B with $|S_A| \ll |S_B|$ and a reference collection S_C. Since $|S_A|$ is small, so is $|S_A \cap S_C|$, yielding low containment and resemblance values, even if $S_A \subset S_C$. If we preferred collections with low containment or resemblance, we would prefer S_A over S_B, even though S_A might not add *any* new documents. To overcome this problem, we propose the notion of *novelty* of a set S_B with regard to S_A, defined as

$$Novelty(S_B|S_A) = |S_B - (S_A \cap S_B)|$$

5.2.2 Synopses

In the following, we briefly overview three relevant statistical synopses methods from the literature, focusing on estimating resemblance. In Section 5.3.2 we will show how to use resemblance to estimate our proposed novelty measure.

Bloom Filters

A Bloom filter [Blo70] is a bit array of length m that uses k independent hash functions $h_1, ..., h_k$ with range $1, ..., m$. When mapping a set S to a Bloom filter the hash functions are applied to the elements in S and the corresponding bits are set to 1. More formally, for each $x \in S$ and for each hash function h_i the bit at position $h_i(x)$ is set to 1. Note that a bit can be set to 1 multiple times (for different x). Given a Bloom filter for a set S, a membership query works as follows: for a given element y, the test if y is contained in S is conducted by applying all hash functions to y and checking if all corresponding bits are set. If not all bits are set, y is definitely not in S. If all bits are set, y is either in the Bloom filter or it is a *false positive*. Figure 5.1 illustrates a Bloom filter: Two items, x_1 and x_2, are inserted into the (initially empty) Bloom filter. Subsequently, when issuing a membership test for y_1 and y_2 we conclude that y_1 is not contained in the filter since not all corresponding bits are set, whereas y_2 seems to be contained in the filter, i.e., it is contained with high probability or it is a false positive. [BM05] gives a more detailed overview over the usage of Bloom filters.

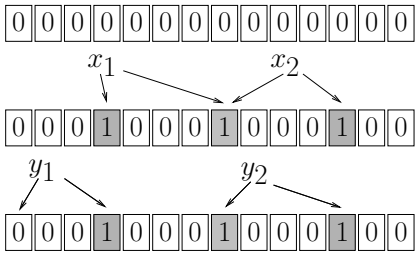

Figure 5.1: Insertion and membership test in a Bloom filter

The probability of a *false positive* after n items have been inserted into a Bloom filter with k hash functions is

$$(1 - (1 - \frac{1}{m})^{kn})^k \approx (1 - e^{-kn/m})^k$$

Min-Wise Independent Permutations

Min-wise independent permutations, or MIPs for short, have been introduced in [BCFM00]. This technique assumes that the set elements can be ordered (which is trivial for integer keys) and computes N random permutations of the elements. Each permutation uses a linear hash function of the form $h_i(x) := a_i * x + b_i \ mod \ U$ where U is a big prime number and a_i, b_i are fixed random numbers. By ordering the resulting hash values, we obtain a random permutation. For each of the N permutations, the MIPs technique determines the minimum hash value, and stores it in an N-dimensional vector, thus capturing the minimum set element under each of these random permutations. Its fundamental rationale is that each element has the same probability of becoming the minimum element under a random permutation.

An unbiased estimate of the pair-wise resemblance of sets using their N-dimensional MIPs vectors is obtained by counting the number of positions in

which the two vectors have the same number and dividing this by the number of permutations N [BCMR04]. Essentially, this holds as the matched numbers are guaranteed to belong to the intersection of the sets.

A heuristic form of approximating also the intersection and union of two sets (and to obtain a synopsis for the combination to continue to work with) would combine two MIPs vectors by taking, for each position, the maximum and minimum of the two values. The ratio of the number of distinct values in the resulting aggregated MIPs vector to the vector length N provides an estimate for the intersection and union cardinalities, but in the intersection case, this is no longer a statistically sound unbiased estimator.

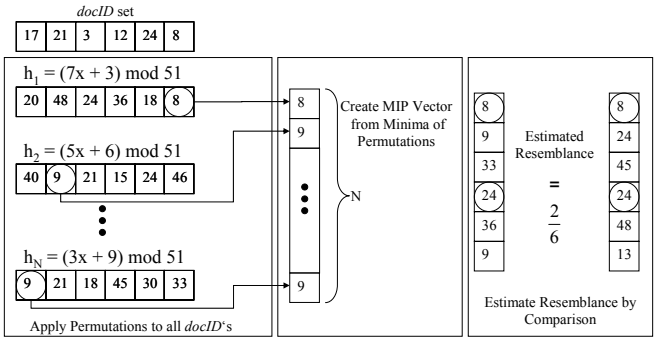

Figure 5.2: Example of min-wise independent permutations

Hash Sketches

Flajolet and Martin in [FM85] proposed hash sketches, a statistical tool for probabilistically estimating the cardinality of a multiset S (i.e., to count the number of distinct items in a multiset). Hash sketches rely on the existence of a pseudo-uniform hash function $h() : S \rightarrow [0, 1, \ldots, 2^L)$, which spreads input values pseudo-uniformly over its output values. Durand and Flajolet further improved hash sketches [DF03] (*super-LogLog counting*) by reducing the space complexity for maintaining hash sketches and relaxing the requirements on the statistical properties of the hash function.

In their essence, hash sketches work as follows. They use the function $\rho(y) : [0, 2^L) \rightarrow [0, L)$ which designates the position of the least significant 1-bit in the binary representation of y; that is,

$$\rho(y) = \min_{k \geq 0} bit(y, k) \neq 0, \; y > 0 \tag{5.1}$$

and $\rho(0) = L$, where $bit(y, k)$ denotes the k-th bit in the binary representation of y (bit-position 0 corresponds to the least significant bit). Estimating n, the number of distinct elements in a multiset S, proceeds as follows. For all $d \in S$, apply $\rho(h(d))$ and record the least-significant 1-bits in a bitmap vector $B[0 \ldots L]$. Since $h()$ distributes values uniformly over $[0, 2^L)$, it follows that

$$P(\rho(h(d)) = k) = 2^{-k-1} \tag{5.2}$$

With the above process, note that in the bit vector B hosting the hash sketch, $B[0]$ is expected to be set to one in $\frac{n}{2}$ of the cases, $B[1]$ in $\frac{n}{4}$ times, etc. From this follows that the quantity $R(S) = max_{d \in S} \rho(d)$ constitutes an estimation of the value of $\log n$. The statistical error can be reduced to very small quantities by utilizing multiple bit vectors B_i, recording $\rho(h(d))$ for some item $d \in S$ to only one of the vectors B_i, producing an R_i estimate for each vector B_i, and averaging over the R_i estimates; the standard deviation of this estimation is $\frac{1.05}{\sqrt{m}}$, for m bitmap vectors[DF03].

5.2.3 Experimental Characterization

We evaluated the above synopses in terms of their general ability to estimate mutual collection resemblance. For this purpose, we randomly created pairs of synthetic collections of varying sizes with an expected overlap of 33%.

For a fair and realistic comparison, we restricted all techniques to a synopsis size of 2,048 bits, and from this space constraint we derived the parameters of the various synopses (e.g., the number N of different permutations for MIPs). We report the average relative error (i.e., the difference between estimated and true resemblance over the true resemblance, averaged over 50 runs with different synthesized sets).[1]

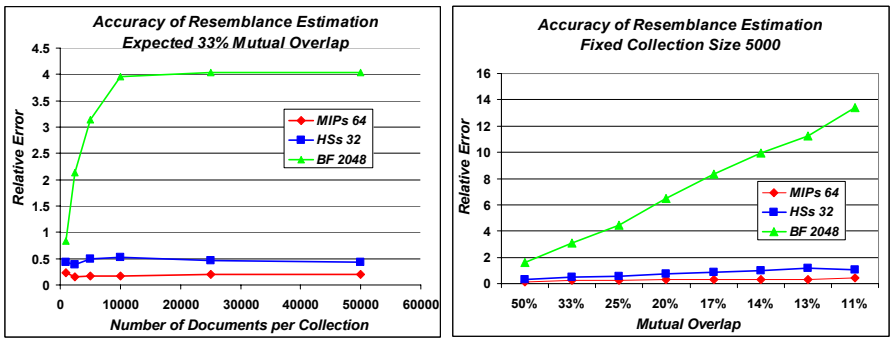

Figure 5.3: Relative error of resemblance estimation

Figure 5.3 (left) shows the relative error as a function of the set cardinality. We see that MIPs offer accurate estimates with little variance and that their error is almost independent of the collection sizes. Hash sketches are also robust with respect to the collection sizes, but on average have a higher error. Bloom filters perform worse even with small collections, because (given their size of 2,048 bits) they are overloaded, i.e., they would require more bits to allow for accurate estimates.

Next, we created synthetic collections of a fixed size (10,000 elements), and varied the expected mutual overlap. We again report on average relative error. The results, shown in Figure 5.3 (right) are similar to the observations above: Bloom filters suffer again from overload; MIPs and hash sketches offer accurate estimates with a low variance for all degrees of overlap.

[1] The expectation values, i.e., the averages over the estimated resemblance values, are more or less perfect (at least for MIPs and hash sketches) and not shown here. This is no surprise as the estimators are designed to be unbiased.

5.2.4 Discussion

A qualitative comparison for selecting the most appropriate synopsis of the peer collections should be based on the following criteria:

- low estimation error
- small space requirements for low storage and communication costs
- the ability to aggregate synopses for different sets in order to derive a synopsis for the results of set operations like union, intersection, or difference
- the ability to cope with synopses of heterogeneous sizes, e.g., to combine a short synopsis for a small set with a longer synopsis for a larger set.

Bloom filters can provide tunably accurate estimations of resemblance between two sets. They also facilitate the construction of aggregate synopses for the union and intersection of sets, by simply taking the bit-wise OR and bit-wise AND of the filters of the two sets. From these, it is in turn straightforward to derive a novelty estimator. A major drawback of Bloom filters is that they cannot work when different sets have used different-sized filters.

This either leads to very high bandwidth and storage overhead (when forcing all collections to be represented by an a-priori maximum filter size) or to high errors (when using inappropriately small size filters, due to very high false positive probability).

MIPs and hash sketches can offer set resemblance estimation with small errors with reasonable space and bandwidth requirements. For the numbers chosen in our experiments, MIPs work even more accurately (i.e., with a lower variance) than hash sketches for different combinations of collection sizes and degrees of overlap, for sets with cardinalities from a few thousand up to millions of elements.

For hash sketches, we are not aware of ways to derive aggregated synopses for the intersection of two sets (whereas union is straightforward by bit-wise OR). This somewhat limits their flexibility in some application classes with conjunctive multi-dimensional queries (cf. Section 5.4). Moreover, they share with Bloom filters the disadvantage that all hash sketches need to have the same bit lengths in order to be comparable.

MIPs are at least as good as the other two techniques in terms of error and space requirements. In contrast to both Bloom filters and hash sketches, they can cope, to some extent, with heterogeneous sizes for resemblance estimation. When comparing two MIPs vectors with N_1 and N_2 permutations, we can simply limit ourselves to the $min(N_1, N_2)$ common permutations and obtain meaningful estimates. Of course, the accuracy of the estimator may degrade this way, but we still have a working method and our experiments in Section 5.6 show that the accuracy is typically still good enough.

5.3 Enhancing Query Execution using Novelty Estimation

5.3.1 The IQN Query Routing Method

Good query routing 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.

For the first aspect, we utilize the statistical metadata about the peers' local content quality that all peers post to the distributed directory (based on local IR measures like tf*idf-based scores, scores derived from statistical language models, or PageRank-like authority scores of documents). For the second aspect, each peer additionally publishes term-specific synopses that can be used to estimate the mutual term-specific novelty. For the third aspect, we ensure that the synopses are as compact as possible and we utilize them in a particularly cost-efficient way for making routing decisions.

The Integrated Quality Novelty (IQN) method based on these rationales starts from the local query result that the query initiator can compute by executing the query against its own local collection and builds a synopsis for the result documents as a *reference synopsis* against which additionally considered peers are assessed. Alternatively to the local query execution, the peer may also construct the reference synopsis from its already existing local per-term synopses. In this section we will simplify the presentation and assume that queries are single-dimensional, e.g., use only one keyword; we will discuss how to handle multi-keyword or multi-attributed queries in Section 5.4.

IQN adds peers to the query processing plan in an iterative manner, by alternating between a *Select-Best-Peer* and an *Aggregate-Synopses* step.

The Select-Best-Peer step uses the query-relevant PeerList from the directory, fetched before the first iteration, to form a candidate peer list and identify the best peer that is not yet included in the execution plan. Quality is measured in terms of an IR relevance metric like CORI (cf. Chapter 4).

Novelty is measured by the candidate peers' synopses, also fetched from the directory upfront, using the techniques of the previous section with further details provided below. The candidate list is sorted by the product of quality and novelty.

Each IQN iteration selects the best quality*novelty peer, adds it to the query processing plan, and removes it from the candidate list.

The Aggregate-Synopses step aims to update the expected quality of the result under the condition that the query will be processed by all peers that were previously selected including the one chosen in the current iteration. For this purpose, IQN aggregates the synopsis of the last selected peer and the references synopsis, where the latter already captures the results that can be expected from all peers chosen in previous iterations. The result forms the reference synopsis for the next iteration. The details of the synopses aggregation depend on the kind of synopsis structure and is discussed in the following subsection. Note that IQN always aggregates only two synopses at a time, and also needs to estimate only the novelty of an additionally considered peer against the reference synopsis. The algorithm is designed so that pair-wise novelty estimation is all it needs.

The two steps, Select-Best-Peer and Aggregate-Synopses, are iterated until some specified stopping criterion is satisfied. Good criteria would be reaching a

certain number of maximum peers that should be involved in the query, or estimating that the combined query result has at least a certain number of (good) documents. The latter can be inferred from the updated reference synopsis.

5.3.2 Estimating Pair-wise Novelty

We show how to utilize the synopses based on MIPs, hash sketches, and Bloom filters to select the next best peer in an iteration of the IQN method. For simplicity, *best* refers to highest novelty here. In a real-world application like Minerva, the peer selection process will be based on a combination of novelty and quality as explained in the previous subsection.

Exploiting MIPs

MIPs can be used to estimate the resemblance R between S_A and S_B as seen in Section 5.2.2. Given $|S_A|$ and $|S_B|$, we estimate the overlap between S_A and S_B as $|S_A \cap S_B| = \frac{R*(|S_A|+|S_B|)}{(R+1)}$ and use this overlap estimation to calculate our notion of novelty using the equation from the definition: $Novelty(S_B|S_A) := |S_B - (S_A \cap S_B)| = |S_B| - |(S_A \cap S_B)|$. This assumes that the initial reference synopsis from which IQN starts is given in a form that we can estimate its cardinality (in addition to having its MIPs representation). This is guaranteed as the query initiator's local query result forms the seed for the reference synopsis.

Exploiting Hash Sketches

Hash sketches can be used to estimate the cardinality of the union of two sets. Using the equation $|S_A \cap S_B| = |S_A| + |S_B| - |S_A \cup S_B|$, we can derive the overlap $|S_A \cap S_B|$ and subsequently our notion of novelty. Given hash sketches for all candidate peers and an (initially empty) hash sketch representing the result space already covered, one can create a hash sketch for the union of two sets by a bit-wise OR operation, as the document that is responsible for a set bit will also be present in the combined collection. Inversely, if none of the documents in either collection has set a specific bit, there will also be no document in the combined collection setting this particular bit: $HS_{A\cup B}[i] = HS_A[i]\ OR\ HS_B[i]\ \forall i : 1 \leq i \leq n$.

Exploiting Bloom Filters

Given Bloom filter representations of the reference synopsis and of the additionally considered peer's collection, we need to estimate the novelty of peer p to the query result. For this purpose, we first compute a Bloom filter bf for the set difference by taking the bit-wise difference, that is: $bf[i] := bf_p[i] \wedge /bf_{ref}[i]$. This is not an accurate representation of the set difference; the bit-wise difference may lead to additional false positives in bf, but our experiments did not encounter dramatic problems with false positives due to this operation (unless there were already many false positives in the operands because of short bitvector lengths). Finally, we estimate the cardinality of the set difference from the number of set bits in bf.

5.3.3 Aggregate Synopses

After having selected the best peer in an iteration of the IQN method, we need to update the reference synopsis that represents the result space already covered with the expected contribution from the previously selected peers. This is conceptually a *union* operation, since the previous result space is increased with the results from the selected peer.

Exploiting MIPs

By design of MIPs, it is possible to form the MIPs representation for the union of two MIPs-approximated sets by creating a vector, taking the position-wise *min* of the vectors. This is correct as for each permutation, the document yielding the minimum for the combined set is the minimum of the two minima. More formally, given $MIPs_A[]$ and $MIPs_B[]$, one can form $MIPs_{A\cup B}[]$ as $MIPs_{A\cup B}[i] = min\{MIPs_A[i], MIPs_B[i]\}\ \forall i : 1 \leq i \leq n$.

A nice property of MIPs that distinguishes this technique from hash sketches and Bloom filters is that this MIPs-based approximation of unions can be applied even if the MIPs vectors of the two operands have different lengths, i.e., have used a different number of permutations. In a large-scale P2P network with autonomous peers and high dynamics, there may be many reasons why individual peers want to choose the lengths of their MIPs synopses at their own discretion. The only agreement that needs to be disseminated among and obeyed by all participating peers is that they use the same sequence of hash functions for creating their permutations. Then, if two MIPs have different lengths, we can always use the smaller number of permutations as a common denominator. This sacrifices accuracy in the resulting MIPs, but still yields a viable synopsis that can be further processed by the IQN algorithm (and possibly other components of a P2P search engine).

Exploiting Hash Sketches

Similarly, one can create a hash sketch for the union of two sets by a bit-wise OR operation, as described in Section 5.3.2.

Exploiting Bloom Filters

For Bloom filters, forming the union is straightforward. By construction of the Bloom filters, one can create the Bloom filter for the combined set from the Bloom filters of two collections by again performing a bit-wise OR operation: $BF_{A\cup B}[i] = BF_A[i]\ OR\ BF_B[i]\ \forall i : 1 \leq i \leq n$.

5.4 Multi-Dimensional Queries

As the synopses posted by the peers are on a per-term basis, there is a need to combine the synopses of all terms or query conditions for a multi-dimensional query appropriately. This issue primarily refers to the Aggregate-Synopses step of the IQN method (once we have an overall synopsis for capturing multi-keyword result estimates, the Select-Best-Peer step is the same as before). We have developed two techniques for this purpose, a per-peer aggregation method

and a per-term aggregation method. They will be discussed the following subsections. We start, however, by discriminating two kinds of queries, conjunctive and disjunctive ones, and discussing their requirements for synopses aggregation.

5.4.1 Conjunctive vs. Disjunctive Queries

Two query execution models are common in information retrieval: disjunctive queries and conjunctive queries. Conjunctive queries require a document to contain *all* query terms (or a file to satisfy all specified attribute-value conditions), while disjunctive queries search for documents containing *any* (and ideally many) of the terms. Both query types can be either with ranking of the results (and would then typically be interested only in the top-k results) or with Boolean search predicates. While conjunctive queries have become common in simple IR systems with human interaction such as Web search engines and are much more frequent in database querying or file search, disjunctive query models are often used in environments with large, automatically generated queries or in the presence of query expansion. The latter is often the case in intranet search, corporate knowledge management, and business analytics.

The choice of one of these query models has implications for the creation of per-peer synopses from the original term-specific synopses. In the Select-Best-Peer stage of IQN, a peer's novelty has to be estimated based on all terms of a specific query. For conjunctive queries, the appropriate operation on the per-term synopses would, thus, be an intersection. For Bloom filters this is straightforward: we represent the intersection of the two sets by simply combining their corresponding Bloom filters (i.e., bitvectors) using a bitwise *AND*. However, we are not aware of any method to create meaningful intersections between synopses based on hash sketches, and for MIPs the literature does not offer any solutions either. For hash sketches a very crude approach would be to use unions also for conjunctive queries; this would at least yield a valid synopsis as unions are supersets of intersections. But, of course, the accuracy of the synopses would drastically degrade. This is certainly an inherent disadvantage of hash sketches for our P2P query routing framework. For MIPs the same crude technique would be applicable, too, but there is a considerably better, albeit somewhat ad hoc, heuristic solution. When combining the minimum values under the same permutation from two different MIPs synopses, instead of using the minimum of the two values (like for union) we could use the maximum for intersection. The resulting combined MIPs synopsis is no longer the MIPs representation that we would compute from the real set intersection, but it can serve as an approximation. It is a conservative representation because the true minimum value under a permutation of the real set intersection can be no lower than the maximum of the two values from the corresponding MIPs synopses.

For a disjunctive query model, in contrast, the *union* operation suffices to form an aggregated per-peer synopsis from the term-specific synopses of a peer. This follows since any document being a member of any of the peer's index lists qualifies for the result. In Section 5.3.3 we have introduced ways of creating such synopses from the synopses of both sets.

In the following, we present two strategies for combining per-term synopses of different peers to assess their expected novelty for a multi-key query with respect to a reference set and its synopsis. For Bloom filters or MIPs, these can handle both conjunctive or disjunctive queries; for hash sketches a low-error

aggregation method for conjunctions is left for future work.

5.4.2 Per-Peer Collection Aggregation

The per-peer aggregation method first combines the term-specific set representations of a peer for all query terms (using union or intersection, depending on the query type and the underlying type of synopsis). This builds one query-specific combined synopsis for each peer, which is used by IQN, to estimate the peer's novelty with respect to the aggregated reference synopsis of the previously covered result space. After selecting the most promising peer, its combined synopsis is aggregated with the reference synopsis of the current IQN iteration.

5.4.3 Per-Term Collection Aggregation

The per-term aggregation method maintains *term-specific* reference synopses of the previously covered result space, $\sigma_{prev}(t)$, one for each term or attribute-value condition of the query. The term-specific synopses $\sigma(p,t)$ of each peer p, considered as a candidate by IQN, are now used to calculate *term-specific* novelty values. For the entire query, these values are combined (e.g., summed up) over all terms in the query. The summation is, of course, a crude estimate of the novelty of the contribution of p for the entire query result. But this technique leads to a viable peer selection strategy.

Per-peer aggregation, discussed in the previous subsection, seems to be more intuitive and accurate, but the per-term aggregation method offers an interesting advantage: there is no need for an intersection of set synopses, even in the conjunctive query model. Instead, the magic lies in the aggregation of the term-specific novelty values. We will explore this idea in Chapter 6 for exploiting term correlation measures mined from the P2P system.

5.5 Extensions

5.5.1 Score-conscious Novelty Estimation using Histograms

In the previous sections we have focused on techniques that treat collections as a *set* of documents. This might be useful in P2P file sharing applications but in ranked retrieval we can do better. Observe that we are more interested in the mutual overlap that different peers have in the higher-scoring portions of an index list. We employ histograms to put documents of each index list into cells, where each cell represents a score range of an index list.

Synopses are now produced separately for each histogram cell. We calculate the weighted novelty estimate between two statistics by performing a pairwise novelty estimation over all pairs of histogram cells, i.e., we estimate the novelties of all histogram cells of a peer's synopses with regard to the cells of another peer's synopses and aggregate these novelty values using a weighted sum, where the weight reflects the score range (i.e., we assign a higher weight for overlap among high-scoring cells).

5.5.2 Adaptive Synopses Lengths

As mentioned before, a large-scale P2P setting with high churn dictates that different peers may want to use synopses of different lengths. The MIPs-based techniques do indeed support this option (although it has a price in terms of potential reduction of accuracy).

In P2P Web search, an important scenario is the situation where each peer wants to invest a certain budget B for the total space that all its per-term synopses require in total, primarily in order to limit the network bandwidth that is consumed by posting the synopses to the directory. Although each individual synopsis is small, peers should batch multiple posts that are directed to the same recipient so that message sizes do indeed matter. Especially when directory entries are replicated for higher availability and when peers post frequent updates, the network efficiency of posting synopses becomes a critical issue.

In this framework, a peer with a total budget B has the freedom to choose a specific length len_j for the synopsis of term j, such that $\sum_{j=1}^{M} len_j = B$ where M is the total number of terms.

This optimization problem is reminiscent of a knapsack problem. A heuristic approach that we have pursued is to choose len_j in proportion to a notion of *benefit* for term j at the given peer. Natural candidates for the benefit weights could be the length of the index list for term j, giving higher weight to lists with more documents, or the number of list entries with a relevance score above some threshold, or the number of list entries whose accumulated score mass equals the 90% quantile of the score distribution.

5.6 Experiments

5.6.1 Experimental Setup

One pivotal issue when designing our experiments was the absence of a standard benchmark. While there are benchmark collections for centralized Web search, it is not clear how to distribute such data across peers of a P2P network. Some previous studies partitioned the data into many small and disjoint pieces; but we do not think this is an adequate approach for P2P search with no central coordination and highly autonomous peers. In contrast, we expect a certain degree of overlap, with popular documents being indexed by a substantial fraction of all peers, but, at the same time, with a large number of documents only indexed by a tiny fraction of all peers.

For our experiments we use the GOV document collection, a crawl of the .gov internet domain used in the TREC 2003 Web Track benchmark [Tex]. This data comprises about 1.5 million documents (mostly HTML and PDF). All recall measurements that we report below are relative to this centralized reference collection. So a recall of x percent means that the P2P Web search system with IQN routing found in its result list x percent of the results that a centralized search engine with the same scoring/ranking scheme found in the entire reference collection.

For our P2P testbed, we partition the whole data into disjoint fragments, and then we form collections placed onto peers by using various strategies to combine fragments. In one strategy, we split the whole data into f fragments and created collections by choosing all subsets with s fragments, thus, ending up with $\binom{f}{s}$

collections, each of which was assigned to one peer. In a second strategy, we have split the entire dataset into 100 fragments and used the following sliding-window technique to form collections assigned to peers: the first peer receives r (subsequent) fragments f_1 to f_r, the next peer receives the fragments $f_{1+offset}$ to $f_{r+offset}$, and so on. This way, we systematically control the overlap of peers.

For the query workload we took 10 queries from the topic-distillation part of the TREC 2003 Web Track benchmark [Tex]. These were relatively short multi-keyword queries, representative examples being "forest fire" or "pest safety control".

All experiments were conducted on the Minerva testbed described in Chapter 3, with peers running on a PC cluster. We compared query routing based on the CORI method which is merely quality-driven (see Chapter 4) against the quality- and novelty-conscious IQN method. Recall that CORI is among the very best database selection methods for distributed IR. We measure the (relative) recall as defined above, for a specified number of peers to which the query was forwarded. In the experiments we varied this maximum number of peers per query. This notion of recall directly reflects the benefit/cost ratio of the different query routing methods and their underlying synopses.

5.6.2 Experimental Results

Figure 5.4 shows the recall results (micro-averaged over all our benchmark queries), using the $\binom{f}{s}$ technique (left) and the sliding-window technique (right). More specifically we chose $f = 6$ and $s = 3$ for the left chart, which gave us $\binom{6}{3} = 20$ collections for 20 peers, and we chose $r = 10$ and $offset = 2$ for 50 collections on 50 peers in the sliding-window setup.

The charts show recall results for 4 variants of IQN: using MIPs or Bloom filter synopses with two different lengths. The shorter synopsis length was 1024 bits (32 permutations); the longer one was 2048 bits (64 permutations).

Figure 5.4 clearly demonstrates that all IQN variants outperform CORI by a substantial margin: in some cases, the recall for a cost-efficient, small number of peers, e.g., 5 peers, was more than 3 times higher, a very significant gain. Also note that in the more challenging sliding-window scenario, the IQN methods needed about 5 peers to reach 50% recall, whereas CORI required more than 20 peers.

In the comparison of the two different synopses techniques, our expectation from the stand-alone experiments (cf. Section 5.2) that MIPs can outperform Bloom filters were fully reconfirmed, now in the full application setting of P2P Web search. Especially for the smaller synopsis length of 1024 bits, the MIPs-based IQN beats Bloom filters by a significant margin in terms of recall for a given number of peers. In terms of number of peers required for achieving a given recall target, again the improvement is even more prominent. For example, IQN with 1024-bit Bloom filters required 9 peers to exceed 60 % recall, whereas IQN with MIPs synopses of the same length used only 6 peers. Doubling the bit length improved the recall of the Bloom filter variant, and led to minor gains for MIPs.

As the network cost of synopses posting (and updating) and the network cost and load per peer caused by query routing are the major performance issues in a P2P Web search setting, we conclude that IQN, especially in combination with short MIPs synopses, is a highly effective means of gaining efficiency, reducing

the network and per-peer load, and thus improving throughput and response times of the entire P2P system.

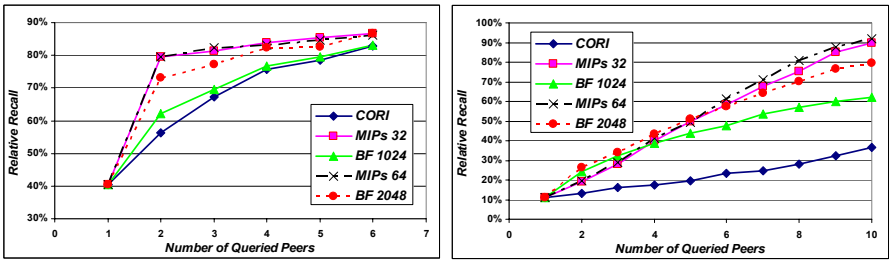

Figure 5.4: Recall as a function of the number of peers involved per query

5.7 Conclusion

This chapter has described the novel IQN query routing method for large-scale P2P systems, with applications in file and Web search. We have characterized and experimentally studied the strengths and weaknesses of three prominent types of statistical synopses, and we have shown how these basic techniques can be incorporated into and effectively leveraged for P2P query routing.

The experiments have proven the high potential of novelty-aware collection selection. It can drastically decrease the number of collections that have to be queried in order to achieve good recall. Depending on the actual degree of overlap between the collections, we have seen remarkable improvements especially at low numbers of queried peers. This fits exactly with our scenario of P2P Web search where we want to put low limits in the number of peers involved in a query.

Chapter 6

Correlation-Aware Query Routing

For scalability, the summaries that are used for query routing have peer granularity, not data item granularity; so they capture, for example, the best peers for certain keywords, attribute values, or topics, but not the best specific data items. Moreover, the summaries are usually organized on a per-keyword (or per-attribute-value) basis, indicating how good a peer's collection is for a given keyword. For tractability, there is no information about keyword sets, phrases, or other forms of correlation between multiple keywords. This limitation to *per-key peer summaries* seems unavoidable, as statistics on all keyword pairs would incur a quadratic explosion and a challenging issue of distributed parameter estimation over a very-high-dimensional and extremely sparsely populated feature space, leading to a breach with the goal of scalability. On the other hand, completely disregarding correlations among keys is a major impediment: together with the restriction to peer rather than document summaries, it may lead to poor search result quality in the P2P setting.

In the following we refer to individual keywords or values as *keys* and to key combinations that exhibit correlation or other mutual relationships as *key sets*. Note that dealing with key sets in queries and routing indices is different from distributed search structures for partial-match queries [AAS05, LNS96], as the latter is limited to low-dimensional spaces with fixed dimensions, whereas in our setting, arbitrary sets of keys from a very-high-dimensional feature space may appear together in a query.

Given the system architecture of Minerva as presented in Chapter 3, in general, query routing for a single-key query with key a proceeds as follows. First, the query initiator issues a request for the statistical summaries regarding the query key a to the underlying overlay network, thus contacting the directory peer for the key, $p(a)$. After the retrieval of this PeerList and its associated information, the query initiator uses the per-key statistical summaries to identify a set of promising data peers for the given query and forwards the query accordingly. Eventually, the query initiator merges the query results individually returned by the selected data peers. The state-of-the-art idea to deal with multi-key queries is to consider the intersection of the PeerLists for the query keys, i.e., to send the query only to (a subset of) data peers that indeed pub-

lished statistics for *all* queried keys. However, this approach may fail miserably. A peer appearing in both PeerLists for the two keys of a multi-key query is only guaranteed to have data items regarding a or b separately, but *not* necessarily data items regarding both a and b simultaneously. Consider the following extreme scenario. Assume peer p_1 containing a large number of data items for each of the two keys a and b separately, but none that contains both a and b together. Judging only by the single-key statistics for a and b, we might reach the conclusion that p_1 is a good candidate peer for the query ab, whereas the actual result set would be empty!

The above dilemma is illustrated by the following example. Consider two- or three-keyword queries such as "Anna Kournikova", "native American music", or "PhD admission". A standard approach to query routing would decompose each query into individual keywords such as "native", "American", and "music", identify the best peers for each of the keywords separately, and finally combine them (e.g., by intersection or some form of aggregating the summary scores) in order to derive a candidate list of peers to which the query should be forwarded. This approach may lead to mediocre query results as the best peers for the entire query may not be among the top candidates for the individual keywords. In a worst case scenario, these peers might not have a single data item that matches *all* keywords at once. Hence, we miss out on the fact that, for example, "PhD" and "admission" are statistically correlated in the underlying corpora and that the best matches for the entire query should exhibit a higher-than-average frequency of *both* keywords.

Table 6.1 summarizes the notation we will be using throughout the rest of this chapter, which develops and evaluates two conceptually diverse approaches to address the above stated problem: *sk-STAT* tries to estimate the desired multi-key statistics from the existing single-key statistics with additional computational efforts and at higher networking costs, and *mk-STAT*, enhancing the distributed directory to explicitly include also statistical information about judiciously chosen sets of multiple keys, namely, those that exhibit particularly high correlation or other forms of strong association among the individual keys in the key set.

Our methods can be used with a large variety of P2P overlay networks, including DHTs but also arbitrary graph topologies with requests being routed among peers based on peer-local routing indices. In the DHT case, the statistical information that drives our query routing covers the entire P2P network, and is stored in a decentralized directory that is physically implemented by the DHT. In the routing-indices case, the statistical information known to one peer covers the peer's neighbors or some efficiently reachable subgraph of the network (e.g., all peers reachable from the nearest super-peer), and is stored locally at the peer itself. For easier presentation, we will restrict ourselves to the DHT case employed by Minerva for the rest of this chapter.

While mk-STAT in principle is the more powerful method, it faces the necessity to identify those valuable key sets that are most likely to enable improvements, as it is practically infeasible to build and disseminate statistics for all possible key sets for combinatorial complexity. Instead, the discovery of interesting key sets is initiated by mining locally gathered query logs, to improve the performance of frequently queried key combinations. This discovery phase can optionally trigger an in-depth statistical analysis of the correlations within the peers' data collections. One of our novel key contributions is how to make this

Symbol	Quantity
$\|X\|$	Number of distinct items in a multi-set X
N	Number of nodes in the system
D	Set of data items in the system
D_i	Set of data items on peer i
a, b	Individual keys
ab	Key set (both of a and b)
$D(a)$	Set of data items in D containing term a
$D_i(a)$	Set of data items in D_i containing term a
$df(a)$	Frequency of key a in D $(= \|D(a)\|)$
$df_i(a)$	Frequency of key a in D_i $(= \|D_i(a)\|)$
$HS(a)$	Hash sketch representing data items in $D(a)$
$HS_i(a)$	Hash sketch representing data items in $D_i(a)$
$p(a)$	Directory peer responsible for key a

Table 6.1: Notation summary

analysis efficient and scalable.

We show that our approach is highly scalable by piggybacking all network communication for gathering and disseminating statistical information on messages that need to be sent between peers anyway (for their regular query traffic).

sk-STAT, on the other hand, can readily deal with all possible key sets, as it only relies on combinatorial operations on the existing single-key statistics. However, it has higher bandwidth requirements at query time, as larger amounts of these single-keyword statistics have to be shipped to estimate the statistics for key sets.

For both approaches, we employ hash sketches (HS) [FM85] as compact synopses for capturing key- and key-set-specific collection quality, that we combine efficiently for different keys and from different peers in a distributed setting. The information gained is harnessed by the query routing process, utilizing the DHT infrastructure for efficiency, and leads to significantly better peer selection decisions for subsequent queries. Our experimental studies at the end of this chapter demonstrate the viability of the method and its performance improvements over the prior state-of-the-art.

The chapter is based on our own work in [MBN+06] and is organized as follows. We give a brief overview of related work in Section 6.1 Section 6.2 introduces the Distributivity Theorem for hash sketches, a major building block in both of our approaches. Section 6.3 discusses measures of key correlations. Sections 6.4 and 6.5 present our algorithms, sk-STAT and mk-STAT, for utilizing statistics on key sets to improve the query routing process. Section 6.6 discusses the scalability properties of our methods. Section 6.7 presents experimental results with two major setups: one based on Gnutella-style file sharing data, one based on Web data. Section 6.8 concludes this chapter.

6.1 Related Work

To our knowledge, the only recent works that consider term correlations in the context of P2P search are our own work [BMT+06, MBN+06] and [PRL+07].

[BMT+06] only considers frequent key combinations in query logs, does not consider data statistics, and uses simple techniques for disseminating statistics in the network, with a very preliminary performance study. [PRL+07] proposes a framework for discriminative keys, which includes correlated term combinations; however, it does not give any algorithms for managing the corresponding statistics in a distributed setting and for correlation-aware query routing.

6.2 Distributivity of Hash Sketches

A key property of hash sketches (cf. Section 5.2.2) with great implications for the efficiency of large-scale network applications (including distributed IR) lies in the ability to combine them. We can derive the hash sketch of the union of an arbitrary number of multisets from the hash sketches of each multiset by taking their bit-wise OR. Thus, given the compact hash-sketch based synopses of a set of multisets, one can instantly estimate the number of distinct items in the union of these multisets.

More formally, we can claim that, if $\beta(S)$ is the set of bit positions $\rho(h(d))$ for all $d \in S$, then $\beta(S_1 \cup S_2) = \beta(S_1) \cup \beta(S_2)$. Notice that, if both original collections carry a random document, the document will conceptually be counted only once, effectively providing duplicate-insensitive (i.e. distinct item) counting for the union of the original multisets.

Furthermore, hash sketches can be used to estimate the cardinality of the intersection (overlap) of two sets. First, recall that

$$|S_A \cap S_B| = |S_A| + |S_B| - |S_A \cup S_B| \tag{6.1}$$

Second, one can derive the hash sketch for $S_A \cup S_B$, and thus compute the cardinality of $|S_A \cap S_B|$ by utilizing the combination method outlined above,

The above can be generalized to more than two sets, using the inclusion-exclusion principle and the sieve formula by Poincaré and Sylvester:

$$|\bigcup_{i=1}^{n} S_i| = \sum_{k=1}^{n} (-1)^{k+1} \sum_{\substack{I \subseteq \{1,\ldots,n\}, i \in I \\ |I|=k}} |\bigcap S_i| \tag{6.2}$$

6.3 Measures of Key Correlation

In this section we introduce the measure for capturing relatedness among the keys of a key set, co-occurring either in a query or in a data item, and we develop the correlation model that will drive the extended synopses construction and query routing as explained in this chapter. For the ease of presentation in this chapter, we restrict ourselves to key pairs, as we expect the major benefit when we move from single keys to correlated pairs.

One of the obvious choices for standard measures, the correlation coefficient, has the drawback that its estimation requires knowledge (or an estimate) of the total number of data items in the network. Moreover, we may encounter situations where it is important to capture that key b is related to key a, but the reverse direction is uninteresting. For example, in popular Web queries the term "soccer" often implies that the same query contains also the term "Germany" (because of the soccer world championship taking place in Germany),

Key A	Key B	P(A|B)	P(B|A)
andy	roddick	0.5106	0.0216
anna	kournikova	0.9613	0.0655
berlin	marathon	0.0611	0.0126

Table 6.2: Selected conditional probabilities

but the reverse direction has a much weaker association from a user viewpoint. This discussion motivates considering the conditional probability that a random data item contains key a given that it contains b, an asymmetric measure of relatedness, for which we have the following estimator:

$$\hat{P}(A|B) = \frac{df(ab)/|D|}{df(b)/|D|} = \frac{df(ab)}{df(b)} \tag{6.3}$$

where

$$df(ab) = df(a) + df(b) - df(a \cup b) \tag{6.4}$$

and $df(a \cup b)$ can be estimated by taking the bitwise OR of $HS(a)$ and $HS(b)$ (see Section 6.2). Obviously, a nice property of this measure is that we can estimate it without knowing (or estimating) $|D|$.

A design dimension orthogonal to the issue of which correlation measure we choose is the consideration of key sets in *queries vs. data items*. Both queries and data are sources of interesting correlations. For queries we can collect, either locally at each peer or globally partitioned based on the DHT, comprehensive query logs and apply frequent itemset mining techniques [AIS93, FSGM+98] so as to extract statistically significant key sets that exhibit a high degree of mutual association among their keys. We will show in Section 6.5 that such techniques are feasible within our P2P architecture without incurring extra communication costs. For data items, a similar approach is conceivable but may be more difficult to implement without incurring extra messages. Moreover and most importantly, we are not really interested in correlated keys within data items per se, unless there are actually queries about these keys. Thus, we pursue a two-stage approach:

1. we *discover* correlated key pairs in queries as an indication that special support for such key pairs may be needed and justified;

2. we *assess* the identified key pairs as to their relatedness in the data items and take special action only if both the discovery and the assessment step are positive for a candidate key pair.

In the discovery step, a key pair is of interest if it is sufficiently frequent and its correlation is high in the query logs. Using the conditional probability estimator of Equation 6.3, we identify the key pair (a, b) as interesting if either one of $\hat{P}(A|B)$ or $\hat{P}(B|A)$ is above some specified threshold α.

In the assessment step the question is whether a key pair, identified in the discovery step, deserves special action for publishing statistical summaries to the distributed directory. At first glance, it may seem that we can use the same

principle as in the discovery step: select key pairs for which the conditional-probability estimate is above some threshold - with the estimate based on data, not on queries. However, this intuition is flawed. Suppose that, for keys a and b, the estimate $\hat{P}(A|B)$ is close to one; so the keys are very strongly, positively correlated. Then the best peers for b alone are likely to contain key a, too. For high recall on these good matches for ab we do not need any special statistical metadata; we can use the statistics for b alone. It is just the opposite situation where we need additional information to find the best peers for a multi-key query: when the keys are either uncorrelated or exhibit strong negative correlation. The latter situation is the most interesting one: when the two keys in a pair ab don't expose a notable degree of correlation, i.e. there are only few data items in the network that contain both a and b, we cannot find them by selecting and combining the best peers for a and the best peers for b alone.

The conclusion from this discussion is that the assessment step considers a pair ab as interesting if both $\hat{P}(A|B)$ and $\hat{P}(B|A)$ are *below* some threshold β within the data items. Table 6.2 shows some conditional probability estimates for popular Google queries[1], based on a large collection that we have crawled recently. If we set β, for example, to 0.1 we would identify ("Berlin", "Marathon") as a valuable key pair, but would dismiss ("Anna", "Kournikova") as not sufficiently valuable, as the statistical summary available for *Kournikova* alone would suffice.

We see that the discovery step and the assessment step have different and *complementary* goals: finding highly correlated keys to identify demand for special support in the discovery step; finding uncorrelated keys or negatively correlated keys in the data as such key sets would be very poorly supported by the standard single-key statistical metadata and established methods for query routing. Note that, of course, the selection in the assessment step refers only to the candidates that were identified in the discovery step.

6.4 sk-STAT: Single-Key Statistics

Recall that the statistical metadata describing the index content of a peer p_i regarding a key a contains the hash sketch $HS_i(a)$, representing the peer's set of local data items $D_i(a)$. By the nature of the hash sketch synopses, the knowledge of $HS_i(a)$ and $HS_i(b)$ for two keys a and b provides a means for estimating the cardinality of the number of data items with at least one of the keys a or b, i.e., $|D_i(a) \cup D_i(b)|$.

Moreover, using Equation 6.1, we can also estimate $df(ab) = |\{d|a \in d \wedge b \in d\}|$, and this generalizes to key sets with more than two keys using the sieve formula. So we can indeed derive vital information for multi-key query routing from the existing single-key statistics in the distributed directory.

Consider a peer p_{init} initiating a query ab. In the sk-STAT approach, fully relying on the existing single-keyword statistics, p_{init} then can proceed as follows:

1. it contacts $p(a)$ and $p(b)$ to retrieve the statistical summaries published individually for the keys a and b, including all hash sketch synopses produced by the data peers,

[1] http://www.google.com/zeitgeist

2. for each remote peer p_i appearing in both PeerLists, p_{init} computes an estimation of $df_i(ab)$, indicating p_i's possible contribution to the query,
3. p_{init} possibly combines this measure with other indicators of p_i's result quality and novelty, and
4. p_{init} forwards the query ab to these selected peers and eventually merges their local results.

A major advantage of sk-STAT, compared to producing additional explicit multi-key statistics (as in the mk-STAT approach) is that the estimations can readily be performed for *all* possible key sets in the directory, and not only for judiciously selected valuable key sets, because sk-STAT only relies on the existing single-key synopses. On the other hand, some disadvantages of sk-STAT also become apparent:

- In order to find the truly best peers for ab from the existing single-key statistical summaries for a and b, a peer probably has to retrieve and inspect the PeerLists for a and b at much higher depth, compared to a multi-key approach where the prefix of a potentially very long list would quickly give you (with high probability) the best peers. The reason is exactly the fact that there is often no strong correlation between keys in the data and thus no correlation between the "rank" of a peer p in the PeerLists for a and b separately; so to identify with high confidence the ranking of some peer p for ab requires longer prefixes of PeerLists if not the entire lists including the hash sketches for every peer in each list. This effect leads to much higher network load on the directory peers and the query initiator at query time. If, like in mk-STAT, there readily exist statistical summaries for the combined key set ab, effective pruning becomes easy by fetching only the top entries from the PeerList for ab.
- While it is possible to estimate $df(ab)$ from the existing statistical summaries (i.e., an integer value estimating the cardinality of the combined set), it is *not* possible to derive the hash sketch synopsis actually describing the data items of a peer p_i for ab, as we are not aware of a way to meaningfully intersect hash sketches, which would be valuable as to also support the overlap-aware techniques presented in a previous chapter. Applying the sieve formula, on the other hand, may well degrade the accuracy of the estimated cardinality (i.e., increase the variance of the estimator).
- In order to properly aggregate the hash sketches, in particular for queries with more than 3 or 4 keywords, combining the hash sketches by the sieve formula requires nontrivial local data structures and entails non-negligible computational costs for the query initiator.

Our experiments have shown that the resource consumption of sk-STAT becomes a significant cost factor under high arrival rates of queries.

6.5 mk-STAT: Key Set Statistics

The obvious idea to overcome the problems of sk-STAT is to learn valuable key sets that frequently co-occur (e.g., in the data items or in user queries), create

and disseminate statistical summaries for those key sets explicitly, and harness this information in order to improve the query routing process. The core idea of mk-STAT is, thus, given a query *ab*, we find pre-prepared statistics describing the peers' local index quality for *ab*.

To this end we need to explore:

1. how to discover candidate key sets,
2. how to assess whether the key-set correlation in the data justifies the additional investment of multi-key statistical summaries,
3. how to notify data peers about these key sets, so they can start to create and disseminate appropriate statistics, and
4. how to leverage the additional multi-key statistics to improve the query routing process.

6.5.1 Query-Driven Key Set Discovery

The motivation for discovering key sets that frequently co-occur in query logs is to improve the search experience of actual users. As a matter of fact, it would be a waste of resources to create, disseminate, and store summaries for a key set that is never queried by a user. Consequently, we limit our efforts for discovering key sets with strongly related keys to the key sets observed in actual queries. In real-world search engines, the distribution of queries is highly skewed (i.e., a small fraction of distinct queries makes up a large fraction of the complete query load); so a careful choice of frequently queried key sets allow us to remarkably improve the search experience for a large fraction of queries (and, thus, for many users) with manageable effort.

The query routing process outlined earlier turns directory peers into "rendezvous points" for key combinations containing one of the keys they are responsible for, making query-driven discovery of frequently co-occurring key sets very efficient:

- when retrieving the statistical metadata for each query key q_i from $p(q_i)$, make the query initiator also send the actual query, i.e., *all* query keys, to $p(q_i)$.
- make the directory peers $p(q_i)$ keep a log of queries they receive. The size of the query logs that need to be kept can be bounded by periodically applying frequent-itemset mining techniques [AIS93, FSGM+98, HK00] and truncating the logs.

For example, a request for all statistical summaries regarding the query "Michael Jordan" would be sent to the two directory peers *p(Michael)* and *p(Jordan)*. Each of these would return its respective PeerList for the key it is responsible for and simultaneously log the query locally. Analyzing these logs, each directory peer can identify key sets that appear in queries with a frequency that is above some support threshold and/or that appear together above some confidence threshold.

6.5.2 Data-Driven Assessment

Discovery of key correlations on the basis of query logs alone is fully sufficient; after all, if there exist correlations among keys which are not (frequently) queried, they are of little consequence and the production and dissemination of appropriate multi-key statistics is of no (immediate) use. However, mining query logs and maintaining previously discovered key correlations in this way depends on a number of hard-to-tune thresholds (such as the support level of occurrences of keyword tuples in order to be deemed as "truly correlated") and requires non-negligible local computations. Furthermore, only a subset of the correlated keys discovered in queries may significantly benefit from additional multi-key statistics: as was discussed in Section 6.3, it is exactly the keys uncorrelated or negatively correlated in the data that mandate multi-key statistics, whereas the keys with high positive correlation also in the data do not really need these additional statistical summaries.

Assume that directory peer $p(a)$ wants to assess the relatedness between a and b based on the data items in the P2P network. For this purpose, peer $p(a)$ proceeds as follows:

1. First, $p(a)$ contacts $p(b)$ to retrieve the overall hash sketch $HS(b)$ for key b. This step requires $O(logN)$ message hops in an N-node P2P network, while the bandwidth consumption is minimal by the following technique: instead of shipping the individual hash sketch synopses $HS_i(b)$ of each peer p_i in its PeerList, the directory peer $p(b)$ locally computes the union of these hash sketches by bit-wise OR and transfers only *one combined* hash sketch synopsis representing df_b.

2. Then, $p(a)$ can compute the hash sketch synopsis representing the union of D_a and D_b by a simple bit-wise OR over the hash sketch synopses $HS(a)$ and $HS(b)$, yielding an estimator for the cardinality of the set consisting of all data items in the system that contain either a or b (i.e., $D(a \vee b)$).

3. The cardinality of $D(ab)$ (i.e. $df(ab)$, the set of documents that contain *both* keywords) can now easily be derived from the above using equation 6.4.

4. From this, $p(a)$ can finally compute the conditional probabilities $P(A|B)$ and $P(B|A)$ using equation 6.3.

The conditional probabilities provide us with a means to quantify the relatedness between two keys in the data and assess the utility of explicit key-set statistics (cf. Section 6.3). As discussed before, the query routing process benefits the most from explicit knowledge of statistical summaries for keys uncorrelated or negatively correlated in the underlying data. On the other hand, if a key set shows high conditional probabilities of co-occurrence within the data items, e.g., $P(A|B) > \alpha$, the single-key statistics readily available for b alone already yield promising peers also for the key set ab exactly *because* the existence of b in a data item strongly suggests the existence of a. In other words, a high $P(A|B)$ value is a heuristic to base query routing decisions on the statistics for b alone, the expected benefits from additional summaries for the multi-key set is small. Small $P(A|B)$ values, in contrast, show that the occurrences of a and b in the data items is largely independent or even negatively related, so that

query routing decisions can highly benefit from the existence of precomputed multi-key statistics for ab. Thus, we initiate the creation of a multi-key summary for a key pair ab if both $P(A|B)$ and $P(B|A)$ are below some threshold β (set, e.g., to 0.1).

Note that the quantitative degree of relatedness is not vital to mk-STAT. However, applying the thresholding can decrease the load on the data peers and the directory by limiting the number of key sets identified as valuable for the query routing process even beyond our initial approach to learn the sets from query logs.

6.5.3 Creating and Disseminating Summaries

When a key set has been identified to be a valuable candidate to produce multi-key statistics for, the data peers need to learn this fact in order to produce the appropriate multi-key statistics. The easiest way of doing so is to use the continuous process of metadata refreshment, i.e., peers periodically updating their summaries in the distributed directory: For any key a, when contacting $p(a)$ in order to update the statistical summary, a data peer retrieves information about such valuable multi-key sets containing a. Remember from Subsection 6.5.1 that all applicable key sets containing a can be identified at $p(a)$ and, thus, are available there. The data peer can subsequently start to produce multi-key statistics for those key sets, which can be published during the next round of metadata refreshments. This procedure has the salient property that it does not incur any additional messages compared to the standard single-key-based P2P system: both the notifications of data peers about interesting key sets and the publishing of multi-key statistics can be piggybacked on messages that need to be sent anyway.

Regarding the placement of multi-key statistics within the directory, a similar consideration suggests that the statistical summary for the key set ab should be stored at one of the directory peers already responsible for one of the keys in the set, or alternatively, all or at least multiple of these directory peers for higher availability. If we choose exactly one of the directory peers, i.e., either $p(a)$ or $p(b)$ for the two-keys case, a simple strategy is to pick the peer that is responsible for the smaller keyword in lexicographic order (i.e., $p(a)$ if we assume $a <_{lex} b$). Again, this has the nice advantage that no additional messages are needed, as any data peer publishing a summary for ab would also publish a summary for a alone and $p(a)$ has to be contacted anyway. The approach also simplifies the retrieval of the summaries for query routing purposes, as we will see in the next subsection.

Preferring the lexicographically smaller keyword does not lead to any critical load imbalances, as all keys are hashed and thus pseudo-randomly assigned anyway.

6.5.4 Enhanced Query Routing

Now that the summaries for ab are contained in the distributed directory and located at peer $p(a)$, a peer p_{init} initiating a multi-key query containing the keys a and b can proceed as usual by issuing requests for applicable summaries to $p(a)$ and $p(b)$. Recall that these requests include the fact that the full query is ab. Because the summaries for ab are kept at peer $p(a)$, $p(a)$ can easily

check whether multi-key summaries for the full query (or any multi-key subset containing more than one query key) are available and deliver the appropriate summaries back to the requester. If no summaries for any multi-key subset are available, every directory peer ships the single keyword summaries, so that baseline query routing can proceed as usual and falling back to the single-key case in those situations does not require any additional messages.

For queries with more than two keys, there is an additional complication: it could happen that there is no summary for the full key set Q of the query (either because this query was not discovered from the query log or the assessment step did not consider the full key set as sufficiently useful), but there are several *subsets* of Q that have explicit multi-key summaries. The situation is easy when there is a clear dominance among subsets, i.e., when one subset is a superset of another one. In this case, we would always prefer the statistical summary with the highest number of keys. If, however, there are incomparable subsets, say *abc* and *bcd* for a five-key query *abcde*, we have more options at hand. Currently, we resort to the simple heuristics that we select all maximal subsets among the available multi-key statistical summaries. This is efficient in terms of network costs because the entire query will be sent to all single-key directory peers anyway. So both $p(a)$ and $p(b)$ are contacted for the query routing decision and can return the statistical summaries for *abc* and *bcd* to the query initiator with no extra costs in communication. But this consideration opens up a space of optimization strategies; this issue is left for future work. Combining such incomparable but mutually related multi-key statistics is reminiscent of the recent work on multidimensional histograms with incomplete information [MMK+05], but our setting has the additional complexity of very-high-dimensional key space (e.g., keywords over text documents).

6.6 Scalability

Among the two suggested methods, sk-STAT is more light-weight when maintaining the P2P directory, but pays higher cost at query run-time, whereas mk-STAT has little overhead at query run-time but is more costly at publishing time. This section briefly discusses to what extent these tradeoffs affect the scalability of our methods. The critical question to address is whether the methods work well as the number of peers in the network grows (to say millions of peers) while the data volume per peer and the rate of query generation per peer remain constant.

The DHT-based distributed directory provides a scalable lookup infrastructure for queries in the P2P network. A query with m keywords triggers directory lookups at (at most) m directory peers. This holds for both sk-STAT and mk-STAT. When a single keyword becomes a bottleneck for the responsible directory peer (by being very frequent in a highly skewed query workload), we can simply replicate the directory peer and use random selection among replicas; this is well supported by DHTs and also other kinds of overlay networks. So there is no scalability bottleneck at query run-time, regarding the network traffic.

When a data peer wants to publish correlation information about two or more keys, it will actually send it only to the directory peer that is responsible for the lexicographically smallest key (unless we introduce replication). Moreover, there is no need to send this statistical piece of information eagerly; rather the data

peer can postpone the publishing until it needs to contact the same directory peer for a query-routing lookup anyway (for the same or a different key). So all publishing messages can effectively be piggybacked on messages that are sent on behalf of queries anyway. The messages become slightly bigger, but the added information is compact so that the message size remains reasonably small. In this setting, the number of messages and the network latency are the critical factors in the overall network performance. Thus, mk-STAT is scalable also from a networking cost viewpoint. The sk-STAT method, on the other hand, may indeed require more messages for accurate estimates at query-routing time, but is as efficient as mk-STAT at publishing time.

The only situation where the publishing cost may become critical is when a data peer wants to publish statistics at a high rate, but has a much lower query-generation rate. In this situation, piggybacking statistical summaries on directory lookups for query routing would not be practical. But this situation is very unlikely for two reasons. First, most P2P applications exhibit many more queries than updates. Second and most importantly, the statistical summaries that mk-STAT newly introduces capture query key correlations and are, thus, triggered by locally issued queries; therefore, a peer with few queries will not issue many statistical summaries of this kind either.

6.7 Experimental Evaluation

We evaluate the performance of sk-STAT and mk-STAT on two different datasets. We consider a dataset derived from an April 2003 crawl of a portion of the Gnutella network[2]. As an IR text retrieval scenario, we use real-world web data from a TREC[Tex] benchmark.

For both datasets, we compare the query result quality obtained by using a state-of-the-art CORI-style query routing approach (cf. Chapter 4] based on single-key frequencies[3] versus both of our approaches.

For our experimental comparison, we have created hypothetical combined collections over all peers' local data collections and identified all globally relevant items for each query on this collection. We report on *relative recall* w.r.t. this reference collection, as a function of the number of peers involved in the queries, i.e., the fraction of results that the reference collection would yield.

6.7.1 Gnutella Data

This dataset is derived from a crawl of a portion of the Gnutella network performed in April 2003, containing information about almost 850,000 music (and other media) files shared by more than 4,000 users, and more than 11,000 queries issued during that time period. As related studies have shown that the users' interests in such a network closely follow the chart trends, we use the US top-40 single charts of the week Apr 12, 2003 to identify key pairs and triplets that can be expected to appear frequently in user queries. For these key sets, all peers published additional, explicit metadata (mk-STAT).

Each user was considered a separate peer, so our P2P network contained about 4,000 peers, with the original assignment of files to peers (including many

[2]Available at http://www.comp.nus.edu.sg/∼p2p/

[3]We denote this approach as *standard* in all upcoming figures

duplicates of popular files at different peers).

As a benchmark query load, we took the original, observed user queries from the dataset, eliminating all queries that were obviously not related to music (but typically to adult content). A file was assumed relevant to a query if its filename contained all query terms. The quality measure used is *relative recall*, based on the number of distinct relevant files returned by the peers (thus, eliminating duplicates returned by more than one peer).

Figure 6.1 shows the results averaged over all remaining *distinct* queries for standard CORI-based query routing vs. sk-STAT and mk-STAT. Figure 6.2 considers only those queries that use at least one key pair or triplet from the chart-based training queries. Figure 6.3, finally, only considers those queries that can benefit from triplets derived from the charts. Additionally, all plots show a theoretic recall optimum that could be obtained from complete knowledge of all collections, which is of course infeasible in a large-scale P2P network.

The results clearly show the recall improvements obtained by our novel methods. The number of peers that need to be queried in order to reach a relative recall of 50 % decreases from about 50 (CORI-style query routing) to about 25 peers in our approaches.

mk-STAT outperforms sk-STAT, because it offers more accurate, explicit statistics for the term pairs. The fact that sk-STAT is able to estimate cardinalities for *all* key combinations cannot compensate for that.

The relative recall figures may appear low for an MP3 file-sharing network. Note, however, that the workload queries were not very selective; on average, 40 distinct files qualified for a query result, and in a few cases there were hundreds of results. With the original placement of files on peers, the results for a query are (averaged over all queries) distributed across 135 peers, including many duplicates. The minimum number of peers that together hold all distinct matches for a query was 30 peers, averaged over all queries, and sometimes more than 100 peers for some less selective queries. Thus, to achieve a relative recall of 100 % would require contacting at least 30 peers on average and in the order of 100 peers for the most expensive queries. However, this is a theoretical minimum and could be efficiently achieved only with a centralized or nearly-centralized super-peer directory structure, neither of which fits with an ultra-scalable P2P architecture. With a Gnutella-style overlay network where search requests are epidemically disseminated to neighbors, messages would be sent to many more peers, in fact, even more than the total number of peers that hold at least one match (i.e., 135 peers on average).

In practice, the user-perceived improvements can be even higher than shown in the figures, because the distribution of queries observed in the real-world query logs of the Gnutella dataset are highly skewed. A small portion of distinct queries make up a substantial fraction of the query load. As those frequent queries are typically exactly the queries that will actually benefit from the additional statistics (because the query-log analysis can identify them and trigger the production of appropriate statistics), our novel method has benefits for an over-proportional fraction of the queries and, thus, of the users.

6.7.2 Web Data

As Web data, we consider the GOV document collection [Tex] with roughly 1.2 million documents compiled by a Web crawl of the .gov internet domain and

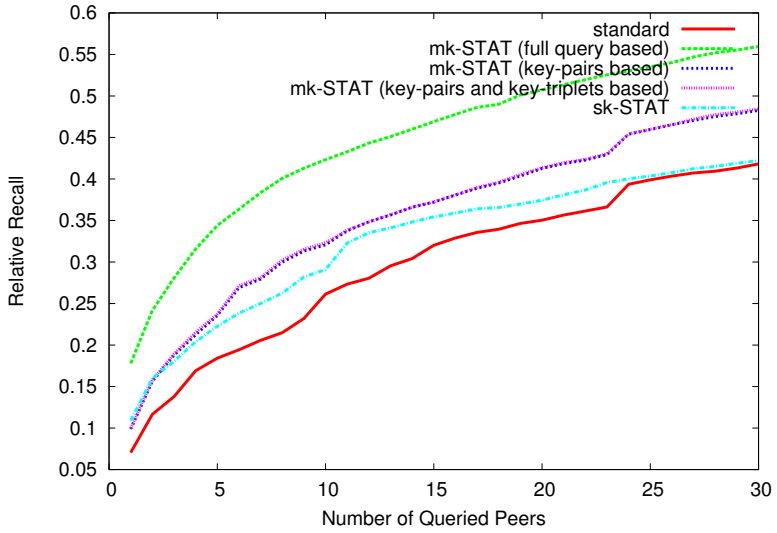

Figure 6.1: All queries

used in TREC benchmarks. To evaluate the distributed behavior, we created 750 peers by randomly assigning documents to them. The random placement was chosen as a stress test for the query routing methods. With thematic clustering, we could achieve much higher relative recall with fewer peers, but that would have simplified the query routing decisions for all methods, whereas we wanted to obtain insights into performance differences under stress conditions. This explains why the results given below show fairly small recall numbers. Note, however, that in Web search, users are typically satisfied with low recall as long as they have acceptable precision among the top-10 or top-20 ranks.

For the query workload we used queries from the topic-distillation track of the TREC 2003 Web Track benchmark, eliminating the single term queries, because we wanted to focus on the improvements for multi-keyword queries. The remaining queries, expanded to increase the document recall, are shown in Table 6.3.

For each query we obtained the top-20 results from the peers that were chosen by the query routing method and merged them into a global result list based on their locally computed scores. The relative recall measure was computed for the top-50 of the global result lists and averaged over all queries. That is, we report the overlap between the top-50 results of the P2P search engine and the top-50 results that a centralized engine would yield. We believe that is a reasonable measure of query routing effectiveness.

Figure 6.4 shows that mk-STAT clearly outperforms the other methods and is very close to the optimum. sk-STAT's performance degrades quickly. This is because, due to the small size of the data peers and the random placement of the data items, only very few peers have a reasonable number of relevant documents for a query, for which sk-STAT's estimation of multi-key statistics is

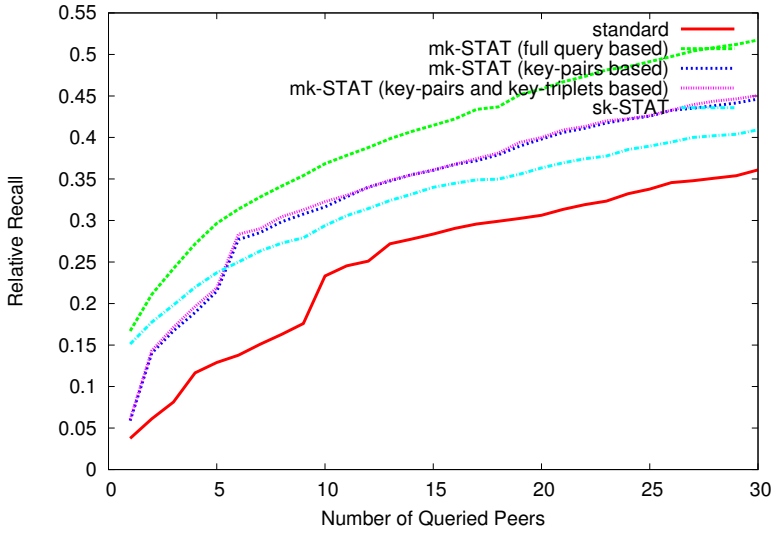

Figure 6.2: Queries with applicable triplet or pair

sensitive enough. Typically, after 4 or 5 peers, each additional peer has only one or two relevant documents to add. In this situation, the estimation accuracy of sk-STAT's combinatorial computation degrades significantly. In contrast, mk-STAT with its explicit multi-key statistics performed very well also for the peers with a very small number of relevant documents. Note that the low recall values reported are due to the random placement of documents on 750 data peers. As the estimated number of relevant documents for a query is evenly distributed over all peers, there is no single peer that can contribute a large fraction of the relevant documents. Nevertheless, mk-STAT manages to yield a recall of almost 20% for 100 out of these 750 peers. Our novel mk-STAT method in this scenario decreases the number of peers involved in a query necessary to reach a relative recall of 10% from 125 to less than 30.

6.8 Conclusion

We have developed efficient methods for capturing, disseminating, and exploiting statistics about correlated key sets in a P2P network. Our experimental studies have shown significant gains in terms of the benefit/cost ratio with benefit defined in terms of query recall and cost proportional to the number of peers that participate in executing a query.

Among the strategic issues that are left for future work is query optimization beyond the query routing decision. We plan to address adaptive run-time adjustments to execution plans and other aspects of dynamic query optimization in P2P systems.

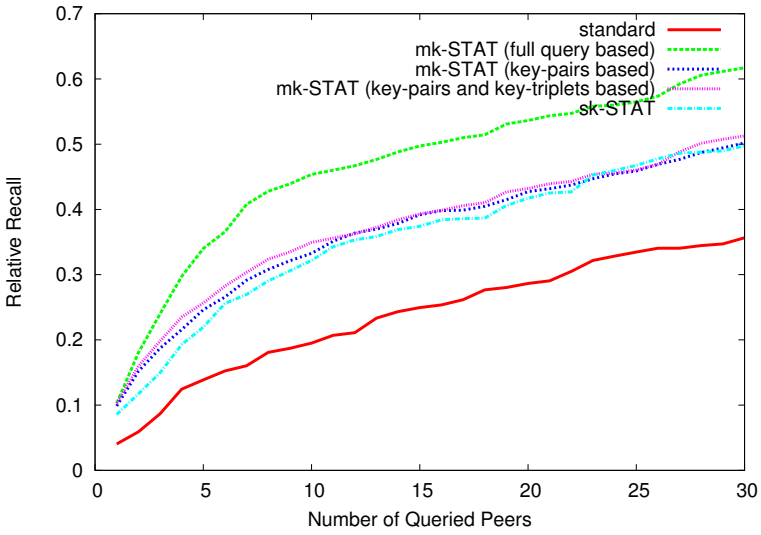

Figure 6.3: Queries with applicable triplet

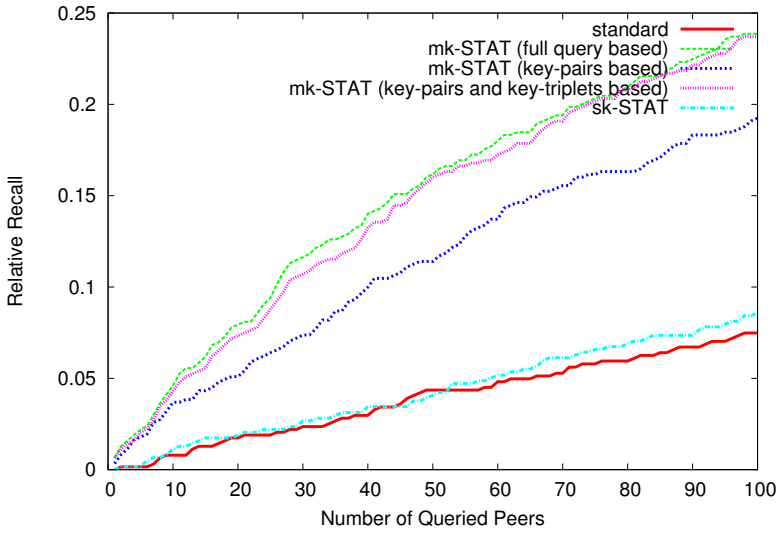

Figure 6.4: Relative recall for extended queries

children's literature book novel kid
forest fires dry flames
homelessness home prevalence
anthrax prevention quarantine
coin collecting numismatics
North Korea communist
Asbestos asbestosis
deafness in children youngsters
Cybercrime, internet fraud, and cyber fraud detection
legalization of marijuana reality
Lewis and Clark expedition historic
computer viruses software trojan
arctic exploration pole
Agricultural biotechnology cultivation
mining gold silver coal metal

Table 6.3: Extended queries

Chapter 7

Combined Methods

This chapter briefly outlines how straight-forward it is to combine the novel approaches from the previous chapters into an even more powerful combined framework. It also compares the effects of overlap-aware and correlation-aware query routing separately on different datasets and evaluates the additional benefits of a combined approach that benefits from overlap-awareness and correlation-awareness at the same time.

7.1 Combing Overlap- and Correlation-Awareness

The methods proposed in Chapter 5 for making query routing overlap-aware can be easily applied to term combinations in the mk-STAT approach, if the statistical summaries published by the data peers contain appropriate data set synopses, e.g., hash sketches or min-wise independent permutations. Note that this is typically the case anyway in order to support the data-driven assessment of term combinations.

Correlation-aware query routing based on mk-STAT would not need to fetch the individual hash sketches of the individual peers from the directory (recall that this is one of the advantages of mk-STAT over sk-STAT, as sk-STAT *does* indeed require all individual hash sketches). However, if the peer also wants to estimate the expected overlap in the expected result sets, the peer can easily fetch the individual peers' hash sketches as well. Note that this does not require any additional messages, but the size of the reply messages from the directory peers to the query initiator increases.

The upcoming section compares the benefits of overlap-aware and correlation-aware query routing individually and evaluates the impact of an even more sophisticated combined framework.

7.2 Experiments

We re-use the following datasets from the previous chapters:

- **CRAWLP**: approximately 250,000 Web pages were harvested using our focused crawler BINGO! on 10 topics, split each topic into 4 disjoint frag-

ments, and created 40 peers by creating all possible 3-subsets of fragments within each topic. The query load for this dataset consists of 28 queries taken from Google Zeitgeist corresponding to the time of the Web crawl (cf. Table 7.1).

- **(X)GOVS:** A random sample of approximately 63,000 documents taken from the .gov document collection has been randomly assigned to 100 (disjoint) fragments, which in turn have been assigned to 50 peers using a sliding-window approach ($r = 10$ and $offset = 2$; cf. Section 5.6.2). We use two different query loads on this dataset: while the first set of queries consists of all 50 queries from the topic-distillation track of the TREC 2003 Web Track benchmark (GOVS, cf. Table 7.2), we have additionally expanded all 50 queries using the query descriptions that come alone with the queries (XGOVS, cf. Table 7.3/7.4). Please note that, for the expanded queries, we use a disjunctive retrieval model, while all other measurements in this chapter use a conjunctive retrieval model.

andy roddick	star wars
american music awards	iraq
oscars	sports illustrated
thailand	hurricane charley
music	mel gibson
diane lane	nfl
gregory hines	berlin marathon
salt lake city	columbus day
fathers day	chicago marathon
marilyn monroe	matrix reloaded
emmy awards	baseball
haiti	lebron james
solar eclipse	real madrid
world series of poker	carmen electra

Table 7.1: Zeitgeist queries

The experimental evaluation in Chapter 4 has concluded that CORI is a robust heuristic approach which is typically not outperformed by other, better-founded theoretic approaches. We will therefore continue to use CORI as our baseline also for the upcoming experiments.

7.3 Overlap-Aware Query Routing

We have evaluated our family of overlap-aware query routing strategies based on different statistical synopses for a number of different synopses sizes. For both data sets, we show relative recall as a function of the number of peers involved in the execution of a query. Figures 7.1, 7.2, and 7.3 show the performance of overlap-aware routing using Bloom filters for both data sets; Figures 7.4, 7.5, and 7.6 do the same for min-wise independent permutations. It is clearly visible that larger synopses can represent the underlying data sets more precisely than smaller synopses, so that the overlap-aware query routing approaches can as a result obtain higher relative recall values. Figures 7.7, 7.8, and 7.9 compare the effectiveness of Bloom filters and min-wise independent permutations for the same synopses sizes for both data sets. Note that Bloom filters tend to

mining gold silver coal	juvenile delinquency
Lewis and Clark expedition	wireless communications
pest control safety	physical therapists
cotton industry	computer viruses
genealogy searches	Physical Fitness
folk art folk music	legalization of marijuana
Schizophrenia	Agricultural biotechnology
cell phones	Emergency and disaster preparedness assistance
Polygraphs	Shipwrecks
Cybercrime, internet fraud, and cyber fraud	children literature
cartography	Veteran's Benefits
Photography	Air Bag Safety
death penalty	Nuclear power plants
affirmative action	Early Childhood Education
Asbestos	Counterfeit money
deafness in children	wildlife conservation
food safety	Literacy
arctic exploration	global warming
coin collecting	weather hazards and extremes
National Public Radio/TV	North Korea
Electric Automobiles	homelessness
forest fires	Ozone layer
Bicycle trails	infant mortality
trains/railroads	robots
Bilingual education	anthrax

Table 7.2: TREC queries

outperform min-wise independent permutations of the same size in our setting, as the number of documents represented by each synopsis is typically small for our small data sets. In such a setting, even small Bloom filters can describe the data set very precisely. This hypothesis is supported by the fact that, as the number of peers increases (and more documents have to be represented), min-wise independent permutations catch up with Bloom filters.

7.4 Correlation-Aware Query Routing

We limit our evaluation of correlation-aware query routing methods to the mk-STAT approach as the more effective and efficient solution to correlation-aware routing. Additionally, we have limited ourselves to term *pairs* as the maximum granularity of mk-STAT synopses, no triplets or larger term sets, in order to assure the practical viability of the set identification and verification process. For both collections, we show relative recall as a function of the number of peers involved in the execution of a query averaged over all queries in the respective query workload (Figures 7.10, 7.11, and 7.12).

The retrieval effectiveness improvements realized by correlation-aware routing are on average smaller than the improvements seen for overlap-aware routing. There are however some details that deserve special attention: first, the query loads for both data sets contain a number of queries with only one query term, which inherently cannot benefit from correlation-aware routing. Second, in contrast to overlap-aware routing, the selection of the *first* peer is already

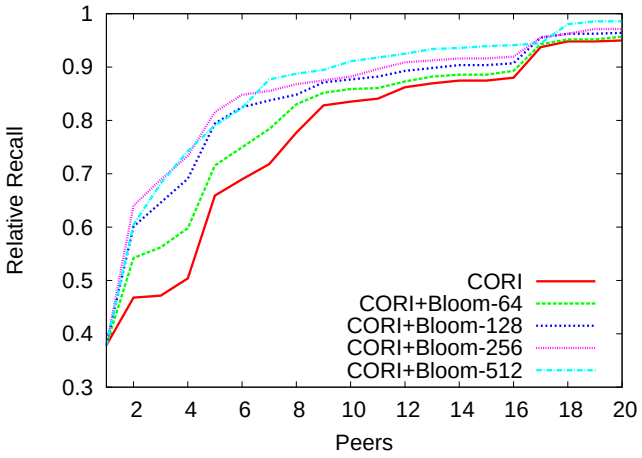

Figure 7.1: CRAWLP: Effectiveness of overlap-aware query routing (varying BF size)

positively influenced by correlation-awareness, while the overlap-aware strategies pick the most promising peer identically to CORI and only subsequently optimize the choice of peers by means of mutual overlap. The reason why correlation-awareness shows a significantly higher impact for the CRAWLP dataset than for the GOVS dataset is based on random document assignment which makes it hard for any query routing strategy to discriminate the peers, when each is expected to hold the very same (and low) number of relevant result. As expected, the impact for XGOVS is higher than for GOVS, as the expanded queries can benefit more from correlation-aware query routing.

7.5 Combined Approach

Figures 7.13, 7.14, and 7.15 show relative recall as a function of the number of peers involved in the execution of a query averaged over all queries for the baseline (CORI), overlap-aware routing (Bloom-256) and correlation-aware routing (mk-STAT), and a sophisticated combined approach leveraging both techniques at the same time for both data sets.

It is clearly visible that both our overlap-aware method and our correlation-aware method individually already outperform CORI. A combined approach combining both approaches in turn outperforms all individual enhancements, and naturally also CORI as the baseline method. Moreover, the performance increase realized by our combined approach over CORI is significant. For the CRAWLP data set, for example, the number of peers that are necessary to reach a relative recall of 70% decreases from five to two. For the GOVS data set, the increase is even more evident, as our combined method can reduce the number of peers in order to reach a relative recall of 70% from 16 peers to as few as

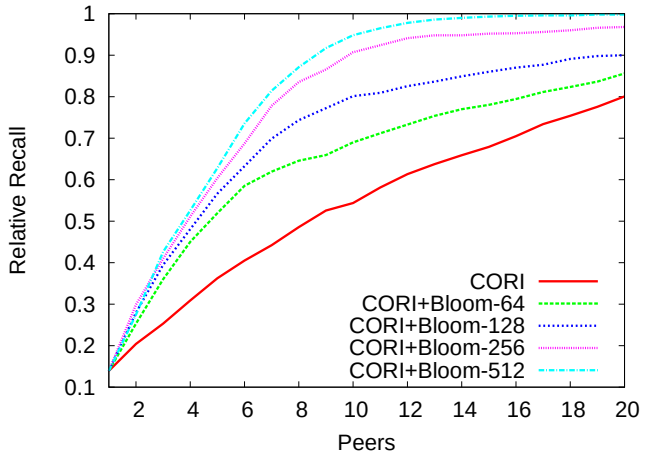

Figure 7.2: GOVS: Effectiveness of overlap-aware query routing (varying BF size)

seven peers, effectively saving more than 50% of the system's bandwidth and computational resources.

7.6 Discussion

The experiments show that the impact of our optimizations to the baseline CORI-style query routing is significant, but the actual impact highly depends on the nature of the data distribution and also on the nature of the queries. The random placement of documents onto peers in our experiments, which is rather unrealistic, is a stress test for any query routing strategy; when the expected distribution of features over the peers is uniform, making effective decisions based on hardly discriminative statistical summaries is a bothersome task. While this argument explains why correlation-aware query routing did not show significant improvements on the (randomly assigned) GOVS dataset, the mutual overlap of the collections enabled our overlap-aware query routing methods to significantly outperform the CORI baseline. Vice versa, for hardly overlapping collections, adding overlap-awareness to query routing will not show a significant impact. Many of the applications envisioned, such as digital libraries or personalized, human-driven peers, are likely to exhibit considerable overlap.

Both overlap-awareness and correlation-awareness approaches are based on any of our statistical synopsis (Bloom filters, min-wise independent permutations, hash sketches). All synopses can yield better estimates when increasing their sizes. While Bloom filters offer the most exact estimation, they are more sensitive to overload than the other synopses, i.e., their accuracy degrades faster as more and more documents are inserted into a fixed-size synopsis beyond their optimal load factor. Thus, they generally need more space to reasonably support

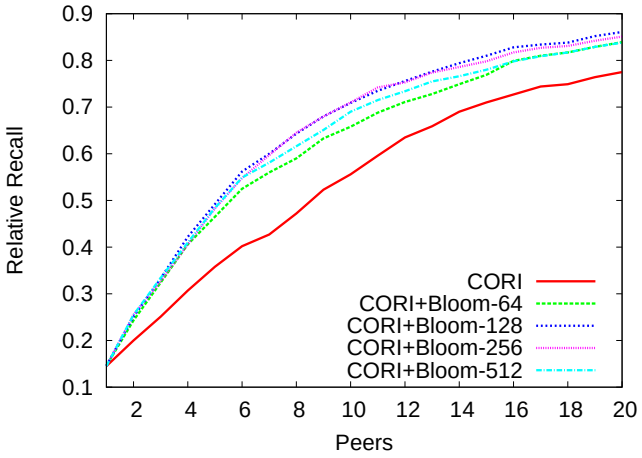

Figure 7.3: XGOVS: Effectiveness of overlap-aware query routing (varying BF size)

the representation of larger or highly skewed data collections.

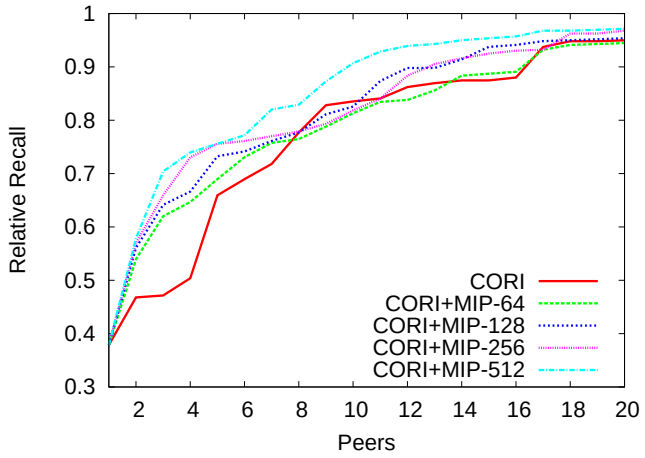

Figure 7.4: CRAWLP: Effectiveness of overlap-aware query routing (varying MIP size)

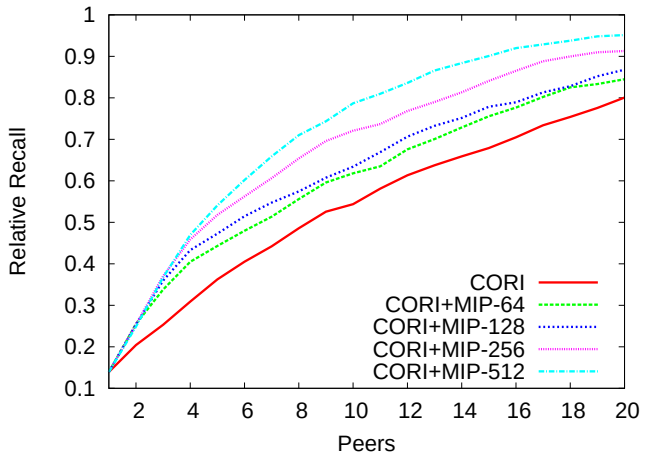

Figure 7.5: GOVS: Effectiveness of overlap-aware query routing (varying MIP size)

mining gold silver coal metal location mineral resources industry
juvenile delinquency youth minor crime law jurisdiction offense prevention
Lewis and Clark expedition historic explore
wireless communications radio broadcasting transmission electromagnetic waves use research technology regulations legislative
pest control safety epidemic contamination quarantine
physical therapists healer training licensing skills body
cotton industry growing harvesting cloth silky fiber plant fabric textile material
computer viruses software program malevolent worm trojan bug
genealogy searches family tree lineage bloodline descent ancestry pedigree origin parentage generation
Physical Fitness shape condition body training
folk art folk music ethnic traditional song ballad country western gospel singing
legalization marijuana cannabis drug soft leaves plant smoked chewed euphoric abuse substance possession control pot grass dope weed smoke
Schizophrenia disorder psychosis distortion reality disturbance social contact
Agricultural biotechnology farming cultivation land food grow crops microorganism bacteria industrial process genetically altered
cell phones cellular mobile hand-held radio transmitter receiver wireless telephone electronic signal sound
Emergency disaster preparedness assistance local state national crisis danger immediate action catastrophe extreme readiness help aid
Polygraphs requirement exam medical instrument physiological process pulse rate blood pressure respiration perspiration lie detector
Shipwrecks ship wreck accident sea capsizing boat nautical water
Cybercrime internet fraud cyber detection crime
children's literature youngster kid book writing novel
cartography mapmaking map chart
Veteran's Benefits ex-serviceman financial assistance
Photography picture taking telephotography
Air Bag Safety restraint automobile inflate collision

Table 7.3: TREC queries (expanded using query descriptions) — part 1

death penalty execution executing capital punishment hanging electrocution decapitation beheading crucifixion burning
Nuclear atomic power plants power station power house
affirmative action discrimination minority groups
Early Childhood child infancy babyhood Education instruction teaching pedagogy elementary
Asbestos fibrous amphibole asbestosis
Counterfeit imitation forgery fake false forged money paper coin
deafness deaf hearing loss deaf-mutism deaf-muteness in children child kids youngsters preschooler infant baby
wildlife living undomesticated conservation preservation conservancy environment
food nutrient foodstuff comestible edible eatable eat safety risklessness security
Literacy center ability read write human skills learn knowledge cognition
arctic north-polar north pole exploration geographical expedition discovery
global warming increase average temperature earth atmosphere climatic changes planetary worldwide heating
coin collecting numismatics numismatology coin collection
weather hazards and extremes peril risk jeopardy wind rain snow storm wave
National Public Radio/TV television telecasting broadcasting cable
North Korea Democratic People's Republic of Korea DPRK communist country
Electric Automobiles production car research progress fuel
homelessness combat vagrancy wandering livelihood home prevalence
forest fires woods burn flames dry summer
Ozone layer environment pollution ultraviolet rays industry
Bicycle trails mountain bike downhill sport offroad nature
infant mortality deathrate children neonatal
trains/railroads travel safety government industry
robots artificial machine production lane research
Bilingual education language learning skills school children
anthrax bacillus anthracis fever disease treatment prevention contagion quarantine

Table 7.4: TREC queries (expanded using query descriptions) — part 2

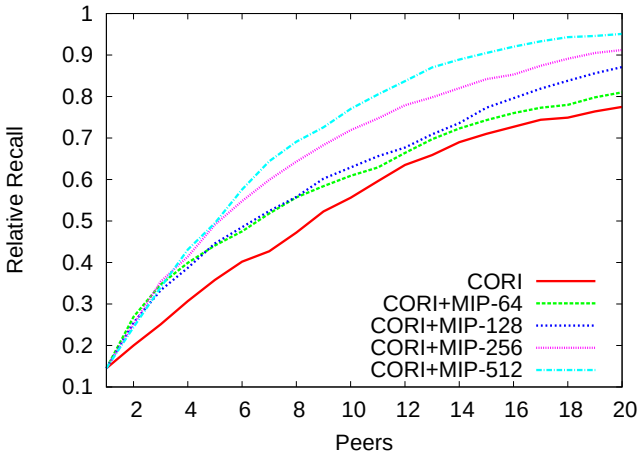

Figure 7.6: XGOVS: Effectiveness of overlap-aware query routing (varying MIP size)

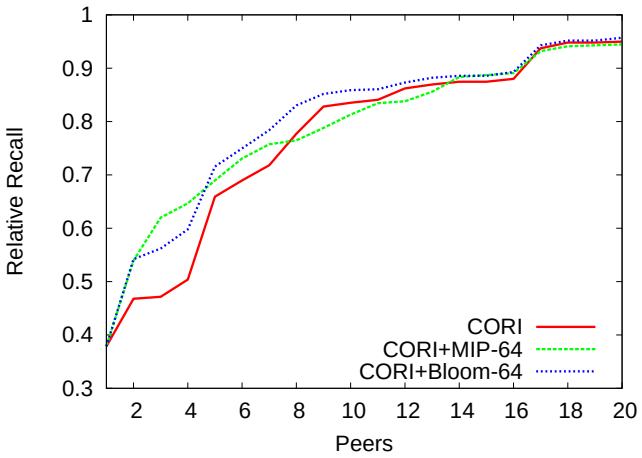

Figure 7.7: CRAWLP: Effectiveness of overlap-aware query routing (comparing BF and MIP)

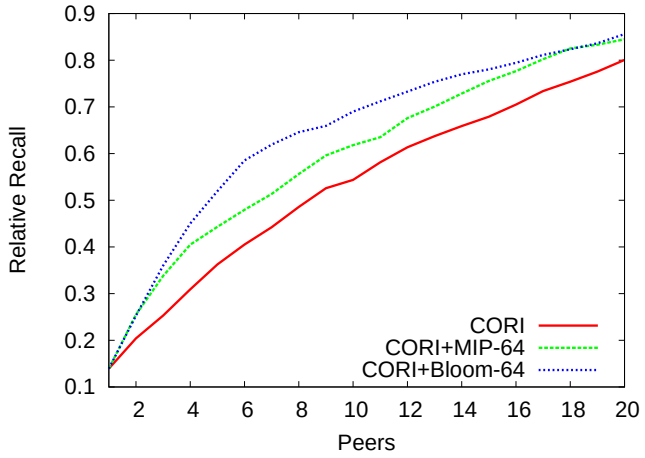

Figure 7.8: GOVS: Effectiveness of overlap-aware query routing (comparing BF and MIP)

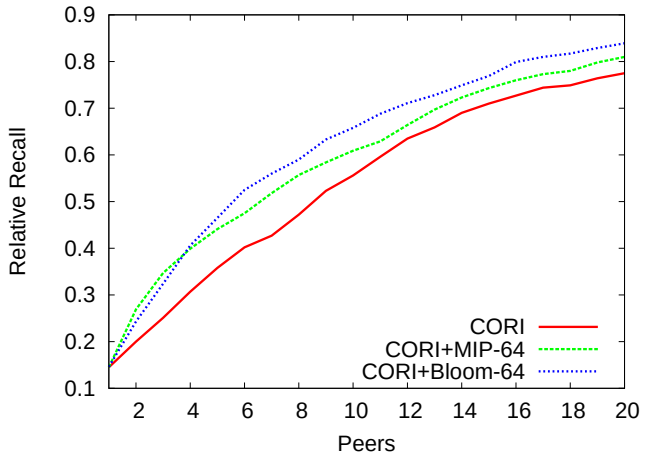

Figure 7.9: XGOVS: Effectiveness of overlap-aware query routing (comparing BF and MIP)

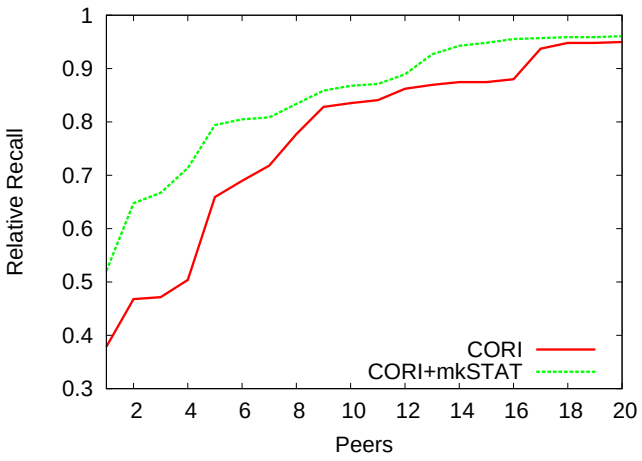

Figure 7.10: CRAWLP: Effectiveness of correlation-aware query routing

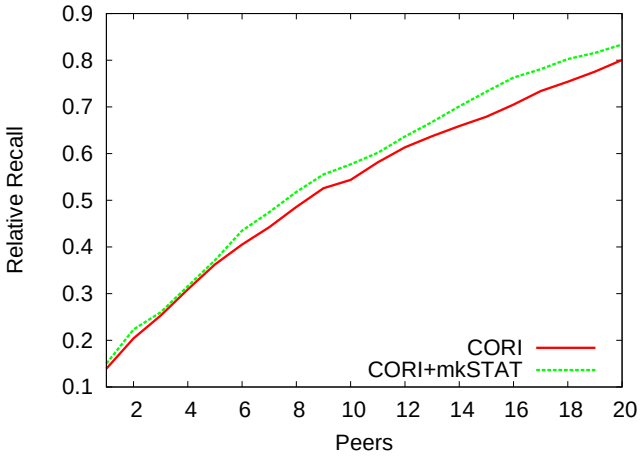

Figure 7.11: GOVS: Effectiveness of correlation-aware query routing

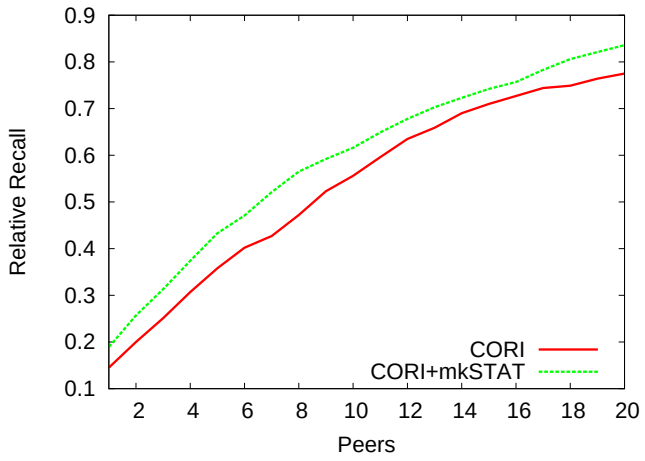

Figure 7.12: XGOVS: Effectiveness of correlation-aware query routing

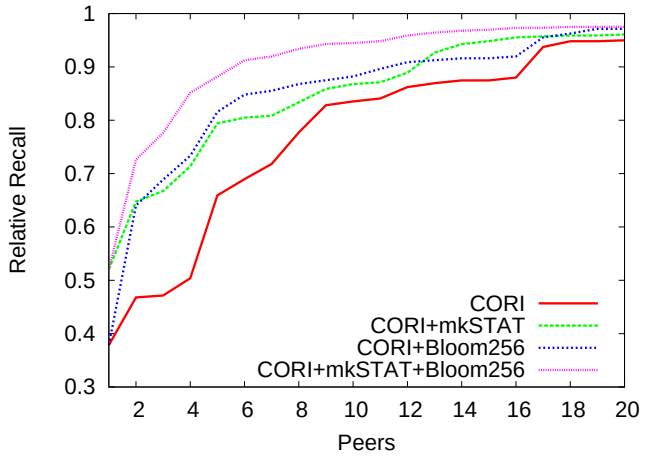

Figure 7.13: CRAWLP: Effectiveness of combined method

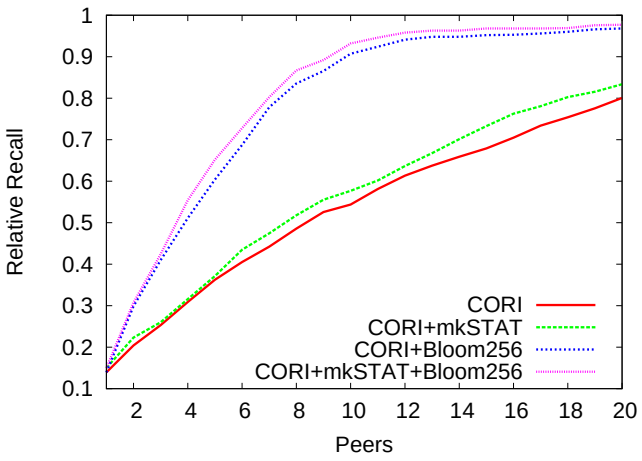

Figure 7.14: GOVS: Effectiveness of combined method

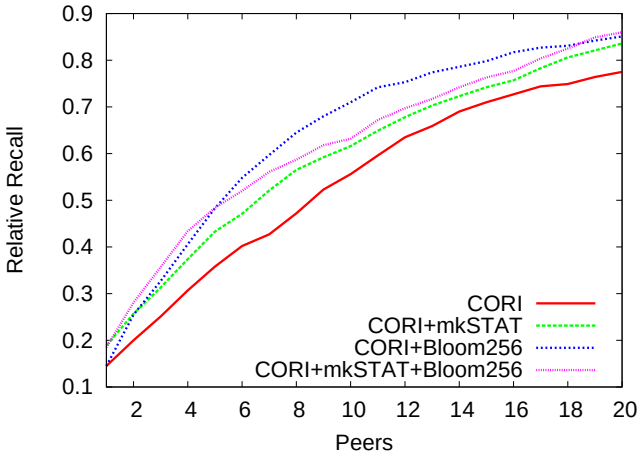

Figure 7.15: XGOVS: Effectiveness of combined method

Chapter 8

Extensions and Additional Optimizations

8.1 Authority-Aware Query Routing

The PageRank authority measure [BP98] is a powerful technique to improve the result ranking quality in centralized Web search. Its intuitive approach to favor documents from high-authority sources has been proven a pivotal ingredient to sophisticated document scoring models, beyond popular statistical information describing the documents, such as *tf*idf* or BM25 [RW94].

However, the computation of PageRank authority scores has traditionally required complete knowledge of the underlying Web graph, i.e., knowledge of all documents (nodes) and interconnecting hyperlinks (edges). While there exist a number of approaches to compute global PageRank authority scores in an environment with the Web graph partitioned *disjointly* over the peers [SYYW03, LAE+04, KHMG03, SSB03], our system architecture envisions a distributed system of autonomous peers that independently crawl the Web, effectively leading to (typically overlapping) partitions of the Web graph being stored at the peers. Recently, the first solution to efficiently compute authority scores that provably converge to global PageRank authority scores in such an environment in a completely decentralized manner has been proposed [PDMW06].

The existing approaches to query routing tend to prefer larger peers (i.e., peers containing more documents) over smaller peers, as larger peers are expected to have a higher probability of containing high-quality query results. But: size doesn't always matter!

For example, consider a general purpose news portal, like cnn.com, and a highly specialized portal for financial statements. While the general purpose portal may have more documents for a company like Enron, the authority of the financial portal may be even higher than the (already high) authority of cnn.com.

Conceptually, the query routing process closely resembles the local document scoring process when evaluating a query. If we visualize each peer as one large *superdocument* (the combination of all its local documents), query routing boils down to finding the top-k "documents" in the network. As PageRank authority scores have been shown to greatly improve this process, it suggests itself to

exploit global PageRank scores for query routing purposes, so to base query routing on an a scoring model that better captures the expected result quality of a peer.

The key idea is to improve the query routing process by preferring peers that have a high local PageRank score mass. This section presents different approaches that take into account the total PageRank score mass of a collection, query-specific portions of the PageRank score mass, and a hybrid framework that combines the authority score mass with other existing scoring models for query routing, such as CORI. We discuss the impact that both ingredients have on query routing and provide evidence gained from experiments that shows the potential impact on query result quality.

The remainder of this section builds upon our own work in [PMB06] and is organized as follows. Subsection 8.1.1 briefly introduces the concept of authority scores, such as PageRank. One possible (though fully orthogonal) way to compute global authority scores and its integration into the Minerva system architecture is introduced in Subsection 8.1.2. Subsection 8.1.3 discusses several approaches to exploit PageRank authority scores for the query routing process in detail. Subsection 8.1.4 develops a hybrid strategy that combines PageRank authority scores and CORI-style scores into a single framework for query routing. The results of an experimental evaluation are illustrated in Subsection 8.1.5, before Subsection 8.1.6 concludes this section and points at future research directions.

8.1.1 Authority Scores

The basic idea of the most popular measure of authority, PageRank, is the assumption that a link from document p to document q represents an implicit endorsement of q, which adds to q's authority. How much p contributes to the authority of q is proportional to the importance of p itself.

This recursive definition of authority is captured by the stationary distribution of a Markov chain that describes a random walk over the Web graph, starting at an arbitrary document and following a random outgoing link from the current page at every step. To ensure the ergodicity of this Markov chain (i.e., the existence of stationary page-visit probabilities), additional random jumps to uniformly chosen target pages are allowed with a small probability $(1 - \varepsilon)$. Formally, the PageRank of a page q is defined as:

$$PR(q) = \varepsilon \times \sum_{p|p \to q} PR(p)/out(p) + (1 - \varepsilon) \times 1/N$$

where N is the total number of pages in the Web graph, $PR(p)$ is the PageRank score of the page p, $out(p)$ is the outdegree of p, the sum ranges over all link predecessors of q, and $(1 - \varepsilon)$ is the random jump probability, with $0 < \varepsilon < 1$ and ε is usually set to a value like 0.85.

PageRank scores are usually computed by initializing a PageRank score vector with uniform values $1/N$, and subsequently applying power iterations with the previous iteration's values on the right-hand side of the above equation to recompute the left-hand side. This iteration process is repeated until sufficient convergence, i.e., until the PageRank scores exhibit only minor changes.

Several similar approaches have been proposed as authority measures, such as HITS [Kle99]. For distributed environments, [SYYW03, LAE+04, KHMG03, SSB03] compute authority scores in the presence of disjointly partitioned collections. [WA05] uses a layered Markov model for Web rank computation.

8.1.2 Distributed Authority Score Computation

This section illustrates how to incorporate global authority score computation into our generalized system architecture of a distributed collaboration of peers. We will show that the communication overhead of adding JXP authority score computation is negligible, as the necessary information exchange can largely be piggybacked onto existing communication.

While there exist a number of techniques to compute aggregated peer authority scores for disjoint collections, we highly advocate solutions that compute the actual PageRank scores for all documents individually. Knowledge of global PageRank scores for individual documents can be of great additional benefit for the local document scoring.

The exact algorithm to compute global PageRank scores for our environment, where the Web graph is partitioned over autonomous peers, is orthogonal to this work, as we simply exploit PageRank scores for query routing purposes. We outline JXP [PDMW06], an algorithm to compute global authority scores in a decentralized manner. [PDMW06] also gives a mathematical proof of the convergence of JXP scores to the global PageRank authority scores, i.e., the scores that would be obtained by a PageRank computation on a hypothetically centralized combined Web graph over all peers.

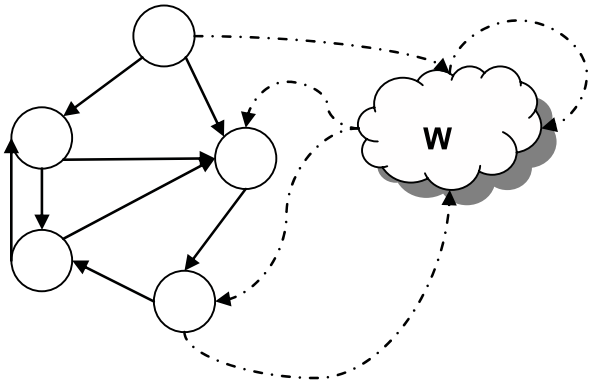

Figure 8.1: Local Web graph, augmented by world node

Running at each peer, JXP combines standard PageRank computations on the local portion of the Web graph with condensed knowledge on the rest of the network continuously being refined by meetings with other peers. The knowledge about the non-local partition of the Web graph is collapsed into a *single* dedicated node that is added to the local Web graph, the so-called *world node*[1]. It conceptually represents all non-local documents of the Web graph. As such,

[1]This is an application of the state-lumping techniques used in the analysis of large Markov models.

all documents of the local Web graph that point to non-local documents will create an edge to the world node (cf. Figure 8.1).

Meetings with other peers in the network are used to exchange local knowledge and to improve the local approximation of global authority scores, illustrated in Figure 8.2. As a peer learns about non-local documents pointing at a local document, a corresponding edge from the world node to that local document is inserted into the local Web graph[2]. Each peer locally maintains a list of scores for external documents that point to a local document. The weight of an edge from the world node to a local document reflects the authority score mass that is transferred from the non-local document; if this edge already exists, its weight is updated with the maximum of both scores. The world node contains an additional self-loop link, representing links within non-local pages. The JXP authority score of the world node itself reflects the JXP score mass of all non-local pages. Locally, each peer recomputes its local JXP scores by a standard PageRank power iteration on the local Web graph augmented by the world node.

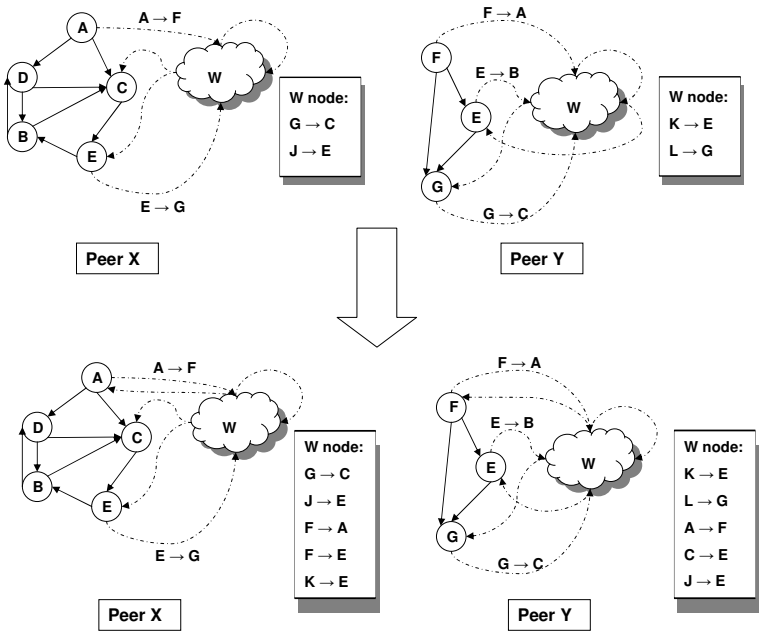

Figure 8.2: Exchanging local knowledge

The JXP algorithm is scalable, as the PageRank power iteration computation is always performed on small local graphs, regardless of the number of peers in the network. The local storage requirements at each peer are independent from the number of remote peers they have previously met and the size of the remote (or even the complete) Web graph, i.e., the size of the local Web graph *only* reflects the local crawl. The autonomy of peers is fully preserved by the

[2]Note that such a meeting does *not* increase the number of nodes of a peer's local Web graph.

asynchronous nature of communication and computation.

The existing Minerva system architecture illustrated before involves manifold steps of mutual peer communication that can naturally be exploited for the JXP authority score computation without arranging additional "meetings": when disseminating local metadata to directory peers, when retrieving directory data on the occasion of a global query execution, and when forwarding queries to the peers selected at query routing. The dissemination of metadata, for example, is a well-suited candidate for JXP meetings, as the communication endpoint is determined pseudo-randomly by the keys which are hashed to directory peers. The data that has to be exchanged for updating local JXP authority scores can easily be piggybacked onto this communication, avoiding extra network messages. The additionally incurred bandwidth requirements are also manageable.

8.1.3 Exploiting PageRank for Query Routing

This section introduces two ways to exploit PageRank authority scores for query routing purposes.

Total PR Mass

One way is to use the total PageRank score mass of a peer (i.e., the sum over all PageRank scores accumulated by the local documents), possibly normalized by the total size of the local collection. Given a local PageRank computation continuously running on a local peer, the collection score s_i is simply calculated as follows:

$$s_i = \sum_{d \in P_i} PR_d$$

where PR_d is the PageRank score of a document d. It is easy to disseminate this value along with the previous metadata to the distributed directory, most naively included in every metadata object.

However, the total PageRank score mass does not well reflect a peer's authority for a particular query; instead, the very same peers would be chosen *regardless of the actual query*, if they have posted any appropriate metadata. For example, consider the Web page of a computer scientist that has crawled the publications of leading CS university departments. While his local Web graph might have accumulated an above-average PageRank score mass, the peer is ill-suited to evaluate queries about sport, cars, or movies. So the total PageRank score pass of a collection is an inappropriate measure for the expected result quality for a particular query. We need a more expressive way of using authority scores for query routing sensible for concrete queries.

Term-specific PR Mass

In order to become query-specific, we want to describe the PageRank score mass of a peer in more detail. A query consists of several terms, and only those local documents that contain at least one of the query terms can be potentially relevant to this query. So the key idea is to compute term-specific PageRank score masses: for each term individually, we sum up the PageRank score masses

of only those documents that contain this term — just as CORI computes term-specific collection subscores.

Query routing based on such term-specific PageRank score masses is straightforward, as the collection score s_i for peer i is computed as:

$$s_i = \sum_{t \in Q, d: t \in d} PR_d$$

where PR_d is the PageRank score of a document d, and the summation is over all documents that contain at least one query term. Note that this scheme does *not* require a separate PageRank computation for every term, but simply sums up the regular PageRank values for the documents that contain the particular term at query time.

Also remember that our system architecture assumes the dissemination of term-specific metadata to a conceptually global, but physically distributed directory. It is straightforward to include the term-specific PageRank score masses with the appropriate piece of metadata for subsequent use in the query routing process.

While the existence of term-specific quality estimators allows for a fine-grained query routing approach by summing up only potentially relevant portions of the PageRank mass, it postulates term independence, as high score masses w.r.t. terms a and b alone don't guarantee a single high authority document for the combined query (a, b). This problem can be overcome using one of the correlation-aware extensions introduced in a previous chapter.

8.1.4 Combining Authority and Quality

There exist several popular strategies for query routing based on various kinds of statistical models, e.g., CORI (cf. Chapter 4). We will present a hybrid framework that can benefit from the best of both worlds by combining quality and authority scores for query routing.

We ignore the query-insensitive approach based on the *total* PageRank authority score mass of a peer and, instead, focus on term-specific PageRank authority score masses. We propose the following linear combination to compute s_i, the hybrid collection score of the i-th peer:

$$s_i = \sum_{t \in Q} \beta * CORI_{i,t} + (1 - \beta) * PR_{i,t}$$

where $CORI_{i,t}$ and $PR_{i,t}$ are the (both term-specific) CORI and PageRank subscores of peer i for term t, respectively.

This approach closely resembles the approach taken in local query execution, where also statistical quality measures describing a document (like *tf*idf* or BM25) are coupled with a document's PageRank score for a total document score. As extreme cases, $\beta = 1$ results in standard CORI-based query routing, while $\beta = 0$ results in query routing solely based on term-specific PageRank score masses.

Please note that for multi-term queries, conceptually, the PageRank score mass for documents containing both terms is accounted for twice. Standard CORI query routing conceptually suffers from the same problem, and we are not aware of literature that has identified this as a serious problem. Therefore,

we do not expect to experience such problems with the PageRank score masses either. Also, we currently do not weight the PageRank score masses of different terms differently, according to some term-specific importance measure, like the idf part does for *tf*idf* or like the $I_{i,t}$ part does for CORI. Experiments have not shown a major impact when the PageRank score mass was additionally weighted with idf values.

In order to account for the different absolute subscore values yielded by CORI and PageRank, we previously apply the following normalization to all values of $CORI_{i,t}$ and $PR_{i,t}$, generalized to $score$:

$$\frac{score - min_t(score)}{max_t(score) - min_t(score)}$$

where $min_t(score)$ and $max_t(score)$ refer to all applicable $score$ values regarding term t that a peer retrieves from the distributed directory during this query routing phase. This ensures a constant impact of both score components.

8.1.5 Experiments

Setup

The effectiveness of *any* query routing strategy builds on the hypothesis that peers are mutually discriminative, i.e., the quality measure of choice is not uniformly distributed over all peers. For example, consider an extreme case where the full Web graph is randomly and uniformly distributed over all peers. In this case, the metadata between all peers will only show minor variations, because all terms as well as all relevant or high-authority documents are expected to be uniformly partitioned.

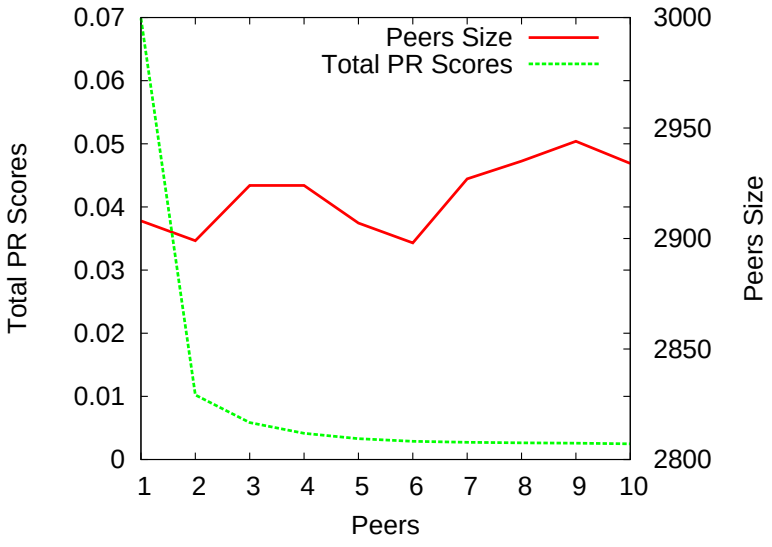

Figure 8.3: PageRank score masses vs. peer sizes

Fortunately, we believe that this is an unrealistic model for the real-world. For example, consider two news portals, cnn.com and a regional German news-

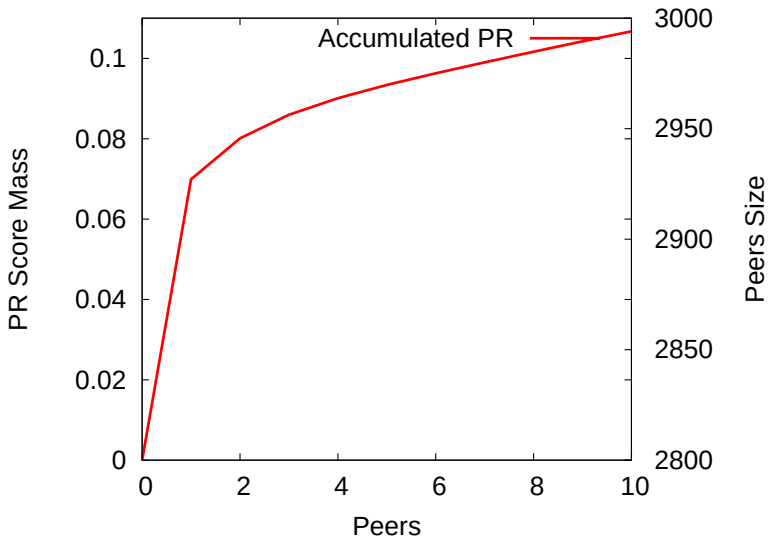

Figure 8.4: Accumulated PageRank score masses

paper. While both portals publish different stories and/or the same stories with different importance, the expected quality of articles on Israel is higher on cnn.com, the expected quality of articles about the German Social Security System might be higher for the German newspaper. Needless to say that, for both examples, the PageRank score mass is higher for cnn.com. Similarly, with users of different skill-levels and different prior knowledge on their fields of interest starting Web crawls from different crawl seeds, the distribution of PageRank authority score mass over the peers is not expected to be uniform — not for the total PageRank score mass, and even less for the term-specific PageRank score mass.

For this experiment we use real-world Web data from 10 topically focused collections harvested by a focused web crawler. Details of the 10 collections can be found in Table 8.1. Several prior experiments have shown that identifying those peers with the "right" topic (for a specific query) is an easy task for all query routing strategies, i.e., separating the sport peers from the politics peers for a query about soccer is not very challenging. However, to find the best order *within* the peers of the right topic is a challenging task.

	Topic	**Number of docs**
Collection 1	Travel	29485
Collection 2	Arts	25093
Collection 3	Finance	29681
Collection 4	Health	22226
Collection 5	Natural Science	18125
Collection 6	Music	20332
Collection 7	Movies	29612
Collection 8	Nature	18714
Collection 9	Sports	22238
Collection 10	Politics	35254

Table 8.1: Collection attributes

For this purpose, we have restricted ourselves to exactly one topic (namely

movies) and distributed only the documents related to movies over a total of 10 peers. With a number of queries related to movies, we want to study the behavior of different strategies regarding the hard task to discriminate peers that share the same topic.

Figure 8.3 plots the total PageRank score mass for each of the 10 resulting peers, ordered by their PageRank mass, together with their respective collection sizes. For example, Peer 2 has a total PageRank score mass of approximately 0.01 and contains approximately 2825 documents. It can clearly be seen that the total PageRank score masses accumulated at each peer differ highly and do not correspond to the respective collection sizes. Analogously, Figure 8.4 shows that a small fraction of peers contains a large fraction of the total accumulated PageRank authority score mass. We expect to see this kind of distribution in real-world settings, backing up our hypotheses that size alone does not matter and that the PageRank authority score masses are a discriminative factor even and in particular for similar-sized collections. So there is hope that a powerful query routing based on authority scores can have a remarkable impact on the result quality.

Results

We have created a hypothetical combined collection of the 10 movie peers that serves as a reference collection. As a query workload, we have chosen queries from Google's Zeitgeist archive that match the movies topic, at the time of and slightly prior to the Web crawl. Table 8.2 shows those queries. Preliminary experiments have shown that all query routing strategies managed to separate these 10 peers for movie-specific queries from the rest of the peers; so we focus on the hard task of ordering within the topic-specific peers. We report on *relative recall* as the average portion of documents from the reference collection's top-20 query results that could be retrieved as the number of peers selected by different query routing strategies increases. The selected peers locally deploy the same document scoring model that was used on the reference collection, based on standard *tf*idf* document scores.

We compare the following instances of our hybrid framework:

- $\beta = 1$: standard CORI
- $\beta = 0.5$, $\beta = 0.1$: hybrid strategies
- $\beta = 0$: term-specific PageRank masses only

We do not show the experimental results based on the non query-specific total PageRank authority score mass of a peer, as we have motivated that this can go arbitrarily wrong. Nevertheless we would like to point out that, for our specific setting, basing query routing solely on the total PageRank authority score mass worked remarkably well. This is due to the fact that we take a limited focus on peers with documents related to movies only, so that for our movie-related queries the total PageRank score mass is already a good indicator for the expected result quality. There are no peers in our setting that accumulate a high total PageRank score mass on topics other than movies, which would break the query routing solely based on the total PageRank authority score mass.

superbowl commercials	earthquake
harry potter	christopher reeve
julia roberts	angelina jolie
desperate housewives	golden globes
jennifer aniston	academy awards
blockbuster	

Table 8.2: Queries

Figure 8.5 plots the relative recall for an increasing number of peers selected by the different query routing strategies. The *optimal* curve shows a theoretical result where, for each query, we precomputed the relevant documents in each collection and query routing was based on an ascending order of relevant documents. Both the hybrid strategy and our strategy based on term-specific PageRank authority score masses outperform the baseline, CORI, in terms of relative recall, in particular for a small number of peers. This is crucial, because the ultimate goal of query routing is to achieve good recall with a very small number of peers. The fact that quality-unaware query routing based on PageRank authority scores only performs as good as our hybrid strategy is an artifact of our small-scale experimental setup. This gives evidence of proof for our hypothesis that authority score masses can be a helpful ingredient in discriminating peers for query routing.

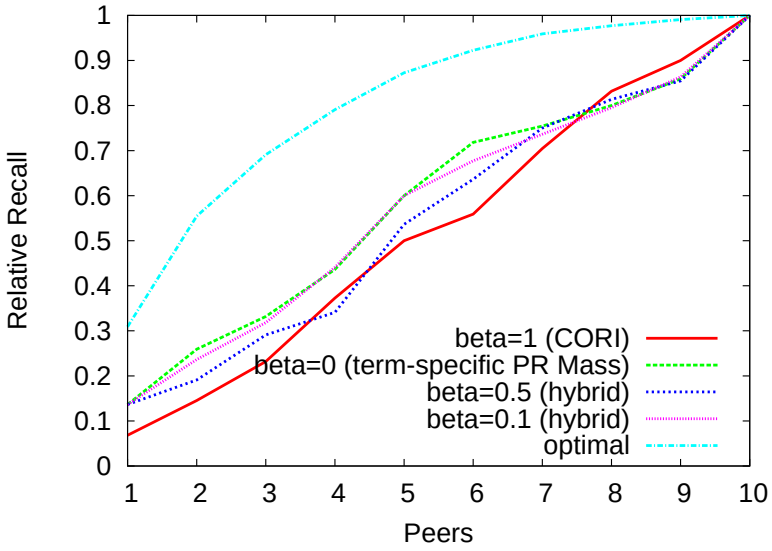

Figure 8.5: Relative recall performance

8.1.6 Discussion

This section has introduced an idea for a framework to integrate knowledge of PageRank authority scores into the query routing process, which is one of the key issues of P2P Web search. Experiments have shown the potential of this

approach, outperforming CORI as a popular baseline algorithm: the number of peers that have to be involved in order to achieve a certain level of relative recall is decreased substantially. This is an important step towards making P2P Web search scalable and feasible.

8.2 Global Document Occurrences (GDO)

Query routing experiments show that often promising peers are selected because they share the same high-quality documents, which already served as the motivation for overlap-aware query routing approaches earlier. As a short reminder, 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. Popular items 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) [PMBW05] that, when processing a query, penalizes frequent documents increasingly as more and more peers contribute their local results by adjusting their local relevance scores. The same approach can also be used for query routing. This is orthogonal to the overlap-aware query routing proposed earlier: while the previous methods use statistical synopses (e.g., Bloom filters) describing the local indexes to estimate the cardinality of mutual overlap between the peers, the GDO approach takes into account the frequency of individual documents and adjusts their relevance scores accordingly. These approaches can eventually be combined to form even more powerful systems. This section discusses the additional efforts necessary to create and maintain the GDO information and presents experiments indicating that the GDO approach can significantly decrease the number of peers that have to be involved in a query to reach a certain level of recall.

Subsection 8.2.1 introduces the notion of GDO, discusses its application at several stages of the querying process, and justifies the GDO approach mathematically. Subsections 8.2.2 and 8.2.3 discuss the application of GDO to query routing and query execution, respectively. Section 8.2.4 discusses how to carry out the maintenance of GDO values in a scalable manner, before Section 8.2.5 illustrates a number of experiments to show the potential of the approach. Subsection 8.2.6 concludes this approach.

8.2.1 Introduction of GDO

We define the *global document occurrence* of a document d (GDO_d) 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 a *global document frequency* of a term t (which is the number of documents that contain a *term* t) and from the notion of *collection frequency* (which is typically defined as the number of collections with at least one document that contains 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 that weights the relative importance of a query term t using the document frequency df_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 at many 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.

Suppose that we are running a single-keyword query and that each document d in the collection has a precomputed relevance to a term t (noted as $DocumentScore(d,t)$). When searching for the top-k documents, a Minerva-style P2P search engine forwards the query to selected peers, which determine the relevant documents locally, and merge the results. This individual and independent document selection has the disadvantage that it does neither consider overlapping results nor the position of the peer within the peer ranking, i.e., how many peers may have already conceptually been queried in advance. For example, one relevant document might be so common that every peer returns it as result. This can reduce the overall recall for a query if all peers return only a fixed maximum number of local results, as the document is redundantly contained in all local query results. In fact, massive document replication is common in real-world P2P systems of autonomous peers, 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. We ignore a slight inaccuracy at this point: we only use the probability that one of the previously asked peers *contains* a document, not the probability that it has actually *returned* that document. Thus, we are 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. We only underestimate (and, thus, effectively 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 most direct 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 experiments.

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

This formula reduces the scores for frequent documents, which hopefully helps to decrease the number of duplicate results. Note that $P_F^\lambda(document)$ decreases with λ, i.e., frequent documents are still returned by peers asked early, but discarded by the following peers.

8.2.2 Exploiting GDO for 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 take into account the number of documents that contain this term, or sum up document scores for certain documents (cf. Chapter 4). 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 the GDO 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 top-ranked peers in the peer ranking. 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. 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 subsection that the DocumentScore of a document d with regard to term t is calculated using the following formula:

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

where λ is the position in the peer ranking (i.e., the number of higher-ranked peers in the peer ranking), 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:

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

where D_p denotes the document collection of a peer p. Instead of including only one score in each term-specific statistical information, now a *list* of the term-specific scores $PeerScore(p,t,\lambda)$ is included in the statistics published to the distributed directory. Figure 8.6 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. Prior experiments have shown that typically involving only very few peers in a query already yields a reasonable recall; we only calculate $PeerScore(p,t,\lambda)$ for $\lambda \leq 10$ as we consider asking more than 10 peers very rare and not compatible with our goal of system scalability. The computational effort necessary for this magnitude of DocumentScores is negligible.

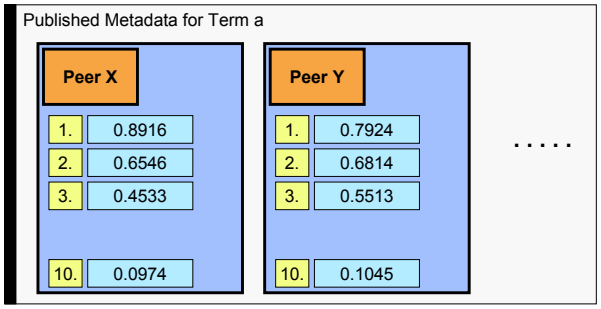

Figure 8.6: Extended term-specific scores for different ranking positions

Please also note that this process does not require the selected peers to locally execute the queries sequentially after the query routing phase, but it still 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.

8.2.3 Exploiting GDO for Query Execution

The peers that have been selected during query routing can additionally use GDO-dependent biases to penalize popular documents during their local query execution. The later a peer is involved in the processing of a query, the higher

the punishing impact that this GDO-dependent bias should have as popular documents are likely to be considered at prior peers. For this purpose, each peer re-weights the DocumentScores obtained by its local query execution with the GDO-values for the documents.

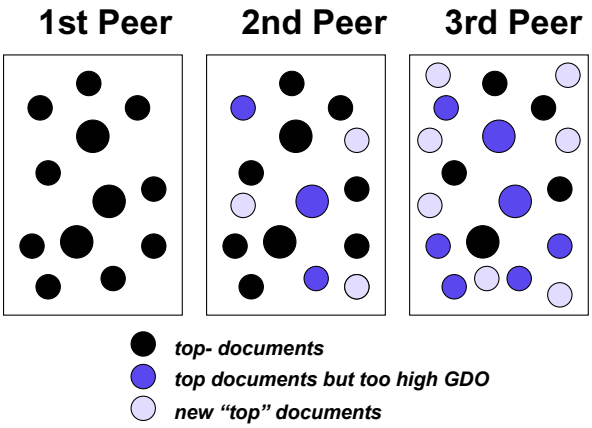

Figure 8.7: Impact of GDO-enhanced query execution

Figure 8.7 shows the impact of the GDO-based local query execution[3].

The additional local query execution cost caused by our approach is negligible. As the GDO values are cached locally, the DocumentScores can easily be adjusted on-line using a small number of basic arithmetic operations.

8.2.4 Building and Maintaining GDO

All the approaches introduced above build on top of a directory that globally counts the number of occurrences or each document. When a new peer joins the network, it increments the GDO values for all its documents by one and retrieves the GDO values for the its biased local DocumentScores at low extra cost.

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 would require 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.

The additional network resource consumption needed for our proposed approach is relatively small if 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

[3]In case you see a 79 in the right figure, please contact your local ophthalmologist immediately.

for a representative term within the document, (e.g., the first term or the most frequent term). Doing so, we can easily piggyback the GDO-related communication when publishing the statistical metadata and, in turn, 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 the Posts again. The statistical metadata become slightly larger because more than one score value is now included in each object; but this will typically still fit within the same network message, avoiding extra communication. In this advanced approach, piggybacking this information onto existing messages almost avoids additional messages completely. In fact, when a peer enters the network, no additional messages are required for the GDO maintenance, as all GDO-related information is piggybacked when publishing Post objects to the directory.

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 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 statistical metadata objects. This update process, in turn, again updates the local GDO values.

8.2.5 Experiments

Benchmarks

We have generated two synthetic benchmarks. The first benchmark includes 50 peers and 1,000 unique documents, while the second benchmark consists of 100 peers and 1,000 unique documents. We assign term-specific scores to the documents following a Zipf distribution (skewness $\alpha = 0.8$) [Zip49], as in practice we often find documents that were highly relevant with regard to one term, but practically irrelevant (with a very low score) with regard to the remaining terms. 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. 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 were following a Zipf distribution, the two distributions were not connected. This means that we do not expect a document with a very high importance for one term to be also highly replicated. 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.

Evaluated Strategies

The experimental evaluation compares six different strategies. All strategies consist of a query routing phase and a query execution phase. For query routing, our baseline algorithm for calculating the PeerScore of a peer p works as follows:

- $PeerScore(p,t) = \sum_{d \in D_p} DocumentScore(d,t)$, i.e., the (unbiased) score mass of all relevant documents in p's collection D_p
- $PeerScore(p,q) = \sum_{t \in q} PeerScore(p,t)$, 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 are derived by summing up the (synthetically assigned) term-specific scores described above. The local scores are used for result merging. At both stages, query routing and query processing, we can either choose a standard (non-GDO) approach or the GDO-enhanced approach, yielding a total of four strategies. The GDO values were provided to each strategy using global knowledge of our data.

In addition, we employ two other strategies that use a mod-k sampling-based query execution technique to return fresh documents: in the query execution process, the peers will return only documents with $(docID \ mod \ \kappa) = \lambda$ where κ is the total number of peers that are going to be queried (10 for the experiments), and λ is the number of peers that have already been queried (i.e., its position in the peer ranking).

Evaluation Methodology

We run several three-term queries using the six strategies introduced above. 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 as a baseline for the proposed strategies.

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

- the number of *distinct* retrieved documents, i.e., after eliminating duplicates
- the aggregated score mass of all distinct retrieved document
- the number of distinct retrieved top-100 documents
- the score mass of distinct retrieved top-100 documents

Results

The experiments are conducted on both benchmark collections. We present the results for the 50-peer setup; the results of the 100-peer setup are very similar.

The GDO-enhanced strategies show significant performance gains. Figure 8.8 shows the number of distinct retrieved documents, while Figure 8.9 shows the aggregated score masses for these documents. Figure 8.10 shows the number of distinct retrieved top-100 documents; Figure 8.11 shows the corresponding score masses. While all other strategies outperform the baseline strategy, it is interesting to notice that query execution (i.e., in essence the result merging) can obviously draw more benefit from the GDO-enhancement than query routing can; if applied to query routing only, our GDO-approach does not show significant performance improvements. This does not come as a surprise and is partly due to the nature of our benchmark. For larger peer populations showing significant mutual overlap, we expect the GDO-enhanced query routing to

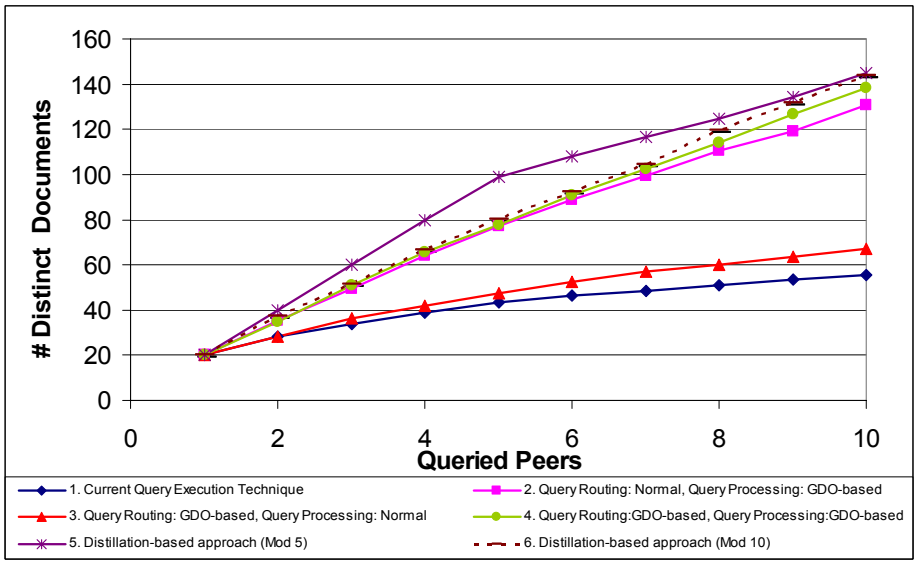

Figure 8.8: Distinct documents retrieved with regard to the number of queried peers

outperform the baseline strategy in a more impressive way. On the other hand, the query execution technique has a great impact on the number of distinct documents. Using GDO-enhancement, popular documents are discarded from the local query results, giving place to other (otherwise not considered) documents.

The naive mod-κ approaches are quite successful in retrieving distinct documents; however, they perform bad if we evaluate the quality of the returned documents by calculating score masses. On the other hand, using the two-way GDO-enhanced strategy (both GDO-routing and GDO-query processing) combines many fresh documents with high scores for our query, resulting in a significant recall improvement.

8.2.6 Discussion

This section has presented an approach to further improve 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 necessary efforts to build and maintain the required statistical information is small and our approach is expected to scale very well with a growing network. Experiments show the potential of the approach, significantly increasing the recall experienced.

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

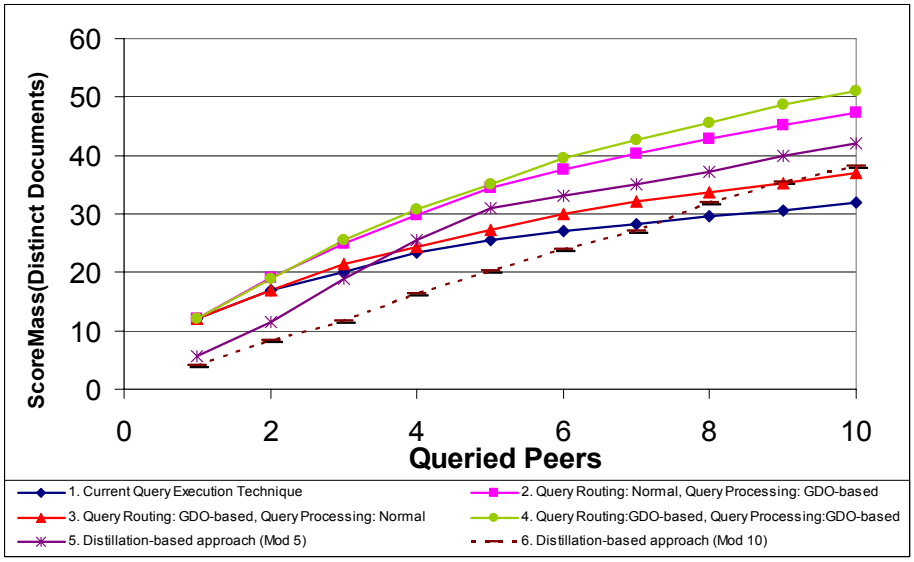

Figure 8.9: Score mass of the retrieved documents with regard to the number of queried peers

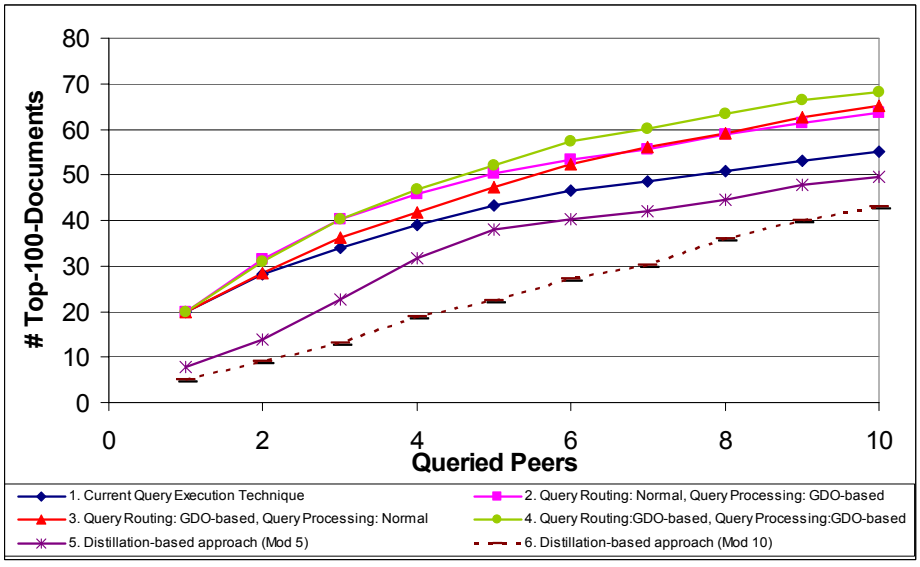

Figure 8.10: Distinct documents from global top-100 with regard to the number of queried peers

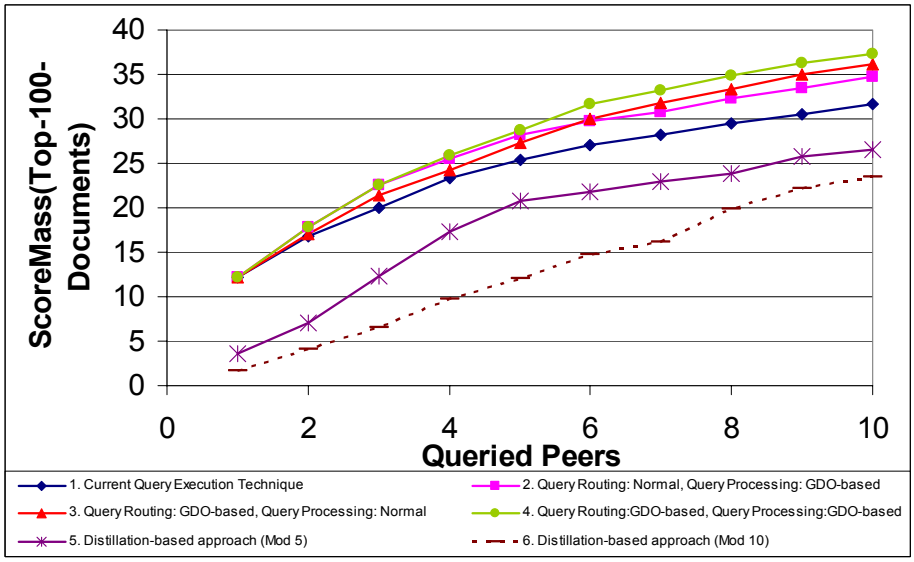

Figure 8.11: Score mass of distinct retrieved documents from global top-100 with regard to the number of queried peers

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, this might allow us to even more efficiently estimate the GDO values.

8.3 Global Document Frequencies

Given the large-scale data distribution of a P2P system, one of the key technical challenges is *result merging*, i.e., the process of effectively combining local query results from different sources (cf. Subsection 3.1.2). 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 *tf*idf* or other popular approaches use collection-specific statistical information for this purpose (cf. Section 2.2). Most prominently, they often use *document frequencies (df)*, i.e., the number of documents in the collection that contain a query term[4]. The local usage of collection-specific *df* values in these scoring models result in document scores that are incompatible across collections and, thus, make result merging difficult. On the other hand, if *global df (gdf)* values could be applied, the document scoring and ranking would be ideal in the sense that it would be identical to the document scoring that would be produced by a hypothetical combined collection.

[4] Again 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.

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 peers' local *df* values for a term. Instead, 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) [LCC00], as popular documents are repeatedly accounted for. Additionally, thematically focused collections show a high variance of *df* values for the same term (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.

We present a robust and scalable approach towards estimating *global df* values using hash sketches [FM85]. We show the superiority of our *global df* estimation technique to other techniques and present experimental evidence of the effectiveness improvements in result merging stemming from this improved knowledge.

The section summarizes the work published in [BMTW06] and is organized as follows. Subsection 8.3.1 discusses the extensions necessary to support an overlap-aware global *df* estimation in the presence of peers entering and leaving the system without prior notice. Subsection 8.3.2 illustrates the additional costs to build and maintain the necessary directory metadata. Subsection 8.3.3 presents an experimental evaluation of the accuracy of our approach from different angles. Finally, Subsection 8.3.4 concludes this section and points at future research directions.

8.3.1 Overlap-Aware Global DF Estimation

Given the system design introduced in Chapter 3 with a hash-based assignment of terms to responsible directory peers, it is very natural for these directory peers to maintain additional data that supports the global *df* estimation for the terms they are responsible for. When publishing the term-specific statistical metadata about its local collection, we propose that each peer includes a *hash sketch* representing its index list for the respective term in its (term-specific) statistical metadata[5], so that each directory peer can compute an estimate for the global *df* values for the terms it is responsible for, using the combination method introduced in Section 6.2. Thus, the hash sketch synopses representing the index lists of all peers for a particular term are *all* sent to the same directory peer responsible for this term. This peer can, by means of inexpensive bit-wise operations, calculate an estimate for the *global df*, for the terms it is responsible for, from these synopses. Note the importance of utilizing compact synopses, such as hash sketches, which introduce small bandwidth and storage requirements.

The query initiator collects the *df* estimates at query time as piggybacked information when retrieving the PeerLists from the directory peers during the query routing phase. Remember that the *df* estimate for a particular term is

[5] if it is not included anyway, e.g., to support overlap-aware and correlation-aware query routing

maintained at the same peer that maintains the respective PeerList, so that the peers that hold the *gdf* estimated for the query terms are the very same peers that are contacted anyway in order to retrieve the respective PeerLists. The query initiator can then attach the current *gdf* estimates to the query message when sending the query to the selected peers. These remote peers can use the estimates on-the-fly as weights during their index scans to compute their local query results.

Note that it is *not* a design choice to let the remote peers simply return unnormalized ("objective") scores (e.g., based on *tf* values only) and then let the query initiator do the re-calibration using *gdf* estimates. In that case, the local query execution at the remote peers may already miss some of the desired (i.e., globally best) results. For example, high-scoring documents for the terms with low *gdf* (i.e., high *idf*) may not be returned at all in that case.

Note that the performance of the local query execution itself is not significantly affected by the necessary online score recalibration: if index lists are created on *(docID, score)*-tuples (where now the score item does no longer include a locally biased *df* component, but some possibly normalized derivate of the *tf* value), index lists can easily be sorted by these scores and index list scanning can be performed as usual. One extra computational operation is required for each list item to compute the final (term-) score for this item (normalization using the *global df* estimate). In this case, the order of items in an index list does *not* change, as all scores in a list are re-weighted by the same *df* value (monotonicity applies). Thus, all index structures and performance acceleration techniques for local query execution work without special adaptation.

8.3.2 Cost Analysis

Most of the **network cost** is caused during the posting process, i.e., when a peer publishes its per-term metadata. Conceptually, each statistical metadata object 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 and maximum term frequency). In our prototype, such an object on average accounts for approximately 50 bytes. Our experiments have shown that a hash sketch with a reasonably small number of 8-byte bitmaps, e.g., 64 bitmaps, allows a good estimation for our purposes (cf. Section 5.2.3). Such a hash sketch requires $64 * 8 = 512$ bytes, i.e., it fits easily in the same TCP packet that is needed anyway to send the metadata itself to the responsible directory peer. Thus, the number of messages necessary to disseminate the statistical metadata does not increase.

Where applicable, we use batching of metadata (for terms that have the same directory peer) to further decrease the number of messages. For all messages, we can additionally apply *gzip* compression to additionally decrease the message payload size. Obviously, the network cost caused by the metadata publication additionally depends on the Time-to-live interval of the metadata, i.e., the time span after which the metadata has to be refreshed. We report on actual traffic measured in the course of our experimental evaluation in Subsection 8.3.3.

After the dissemination of the statistical metadata, peers initiating a query perform PeerList requests to retrieve a list of peers that have published statistics about the specific query terms. Note that the cost of this PeerLists retrieval does *not* change significantly, as the hash sketches themselves do not need to

be transferred back to the PeerList requester[6]. Instead, as the *df* estimation is conducted at the directory peer, only one additional value representing the current *df* estimate has to be included in the answer to a PeerList request. The same holds true for the actual query execution; when sending the query to the selected peers, just one additional *df* value per query term has to be transferred.

The **storage cost** at the directory peers storing the statistical metadata objects is also directly dependent on the number of objects, the size of a Post, and the size of a hash sketch. In a network with n peers storing statistical metadata of m distinct terms, each peer is responsible for an expected number of 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 previously unseen 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 28.1 MB (including all hash sketches) for each peer list, which we consider a reasonably small storage effort. Remember from the previous subsection that, alternatively, the directory peer does not have to store all hash sketches sent together with the Posts, but can aggregate them immediately.

The additional **computational cost** incurred by adding hash sketches to the posting process is also negligible. For nearly no additional cost, the peer that receives the hash sketches for a particular term can combine these in an iterative manner by simple bit-wise OR operations of bit vectors.

8.3.3 Experiments

For a general study on the accuracy of hash sketches please refer to Section 5.2.3.

DF Estimation in the Presence of Churn

We want to evaluate the accuracy of our approach in the presence of network dynamics. For this purpose, we consider a model of node arrivals and departures as outlined in [LNBK02], where nodes arrive according to a Poisson process with rate λ, while a node in the system departs according to an exponential distribution with rate parameter μ. Resulting in a system of about $1,000$ peers at a time, we assume time units of 10 min and choose $\lambda = 3$ and $\mu = 0.002$ and fix the interval at which peers refresh their statistics to 6 time units (60 min). For simplicity, we further set the Time-to-live for all statistics also to 6 time units. In a real world scenario, one could argue to increase the TTL to cope with network latencies and network failures, such that statistical metadata in the directory survive one failed refresh attempt. Each peer randomly picks 1,000 documents from a domain of 2,000,000 documents. We use 256-bitmap hash sketches for our evaluation.

Figure 8.12 plots the document frequency estimates obtained by our approach together with the true document frequency. While intuitively, the approach should tend to overestimate the number of documents in the system, because metadata of peers that have recently left the system hang around for some time before they time out, in practice our experiments don't clearly show

[6] or are transferred anyway, to support overlap-aware query routing

this behavior. This is due to the fact that the hash sketches themselves show a certain degree of variance that overrules the (usually small) conceptual errors of the approach. Nevertheless, the approach has been shown to be robust against churn.

Regarding the network traffic caused by the experiment under the above assumptions with only one term per peer, we can report an average bandwidth consumption of less than 11 kilobytes per peer per hour (no gzip compression applied). Even for typical numbers of terms per peer (50,000-100,000), this is well within today's bandwidth capabilities.

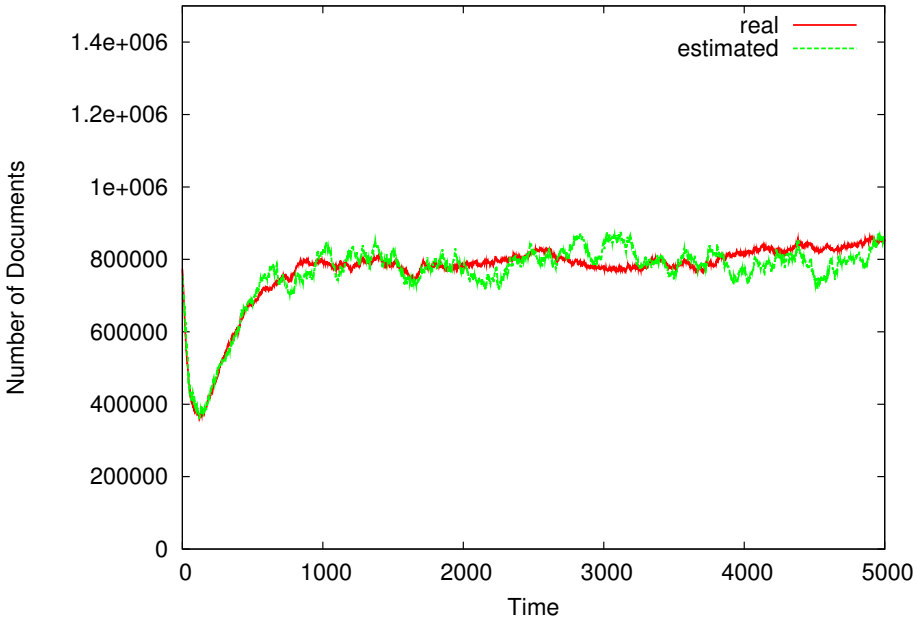

Figure 8.12: *df* estimation accuracy (256-bitmap hash sketches)

Improving Result Merging

For this experiment we re-use the real-world Web data of 10 topically focused collections from Table 8.1. In order to create our benchmark, we have split each topical collection into 4 fragments. We create 40 peers such that each peer hosts 3 out of 4 fragments from the same topic, thus creating high overlap among same-topic peers. As query load, we re-use the queries taken from Zeitgeist (cf. Table 7.1). We use CORI as our baseline query routing strategy and compare four different result merging strategies, according to the matrix given in Table 8.3. The document scores are based on collection-specific (i.e., "local") *df* values or our *global df* values and are normalized using their respective weighted CORI peer score (from the query routing phase) or not. For this normalization, more specifically, we use the norm-dbs method used by the INQUERY framework [CCH92], that re-computes document scores as $score = (D + 0.4 \times C_{norm} \times D)/1.4$.

	Global *df* values values used for document scoring	Document Scores normalized using CORI peer scores
Document score merger (global)	yes	no
Document score merger (local)	no	no
CORI merger (global)	yes	yes
CORI merger (local)	no	yes

Table 8.3: Result merging strategies

As a rank distance, we use Spearman's footrule distance [DG77], defined as $F(\sigma_1, \sigma_2) = \sum_i |\sigma_{ref}(i) - \sigma_{peers}(i)|$ where $\sigma_{ref}(i)$ is the rank of document i in the reference ranking and $\sigma_{peers}(i)$ is the position of document i in the peers' document ranking. If a document from σ_{peers} is not in σ_{ref} we assign a fictitious rank $(k+1)$. We normalize all distances by $1 - \frac{distance}{maxdistance}$ to obtain a normalized quality measure.

Figure 8.13 shows the results for the 40-peers benchmark. With local query execution based on *global df* values, the ranking quality is remarkably above the quality obtained by the CORI-based merging methods. In particular, three out of the four methods do not even come close to the optimal document ranking at all, even if all 40 peers are involved in the query. This is due to the fact that the document scores based on *local df* values are incomparable across the peers and, thus, documents that are not in the reference top-20 document ranking are pushed in by skewed local *df* scores at the peers. To better understand the document frequencies' effect on the ranking, Table 8.4 shows the local *df* values for all query terms. For terms that are likely to occur in some topics but not in others, we observe highly skewed distribution. For example, the term *cup* occurs in 2,015 documents in collection 9 (Sports) and less than 350 documents for each of the other collections.

8.3.4 Discussion

This section has developed and evaluated a novel and efficient algorithm to estimate global document frequencies in large-scale dynamic P2P networks. The algorithm utilizes compact synopses based on hash sketches, which can be combined from an arbitrary number of autonomous distributed sources without incurring additional error. To our knowledge, this is the first approach to this problem that can cope with arbitrarily overlapping collections without significant additional effort. We study the network and storage requirements.

The main focus of this section is *not* to quantify the effect that the knowledge of global *df* values can have on result merging. The corresponding experiment is only preliminary, but nevertheless indicates the potential for improvements. While this effect has already been observed in the literature [LCC00], more comprehensive experiments on result merging in P2P networks are subject of future work.

The approach can easily be generalized to all forms of distributed systems that can benefit from global counting with duplicate elimination, e.g., cardinality estimations in distributed database systems.

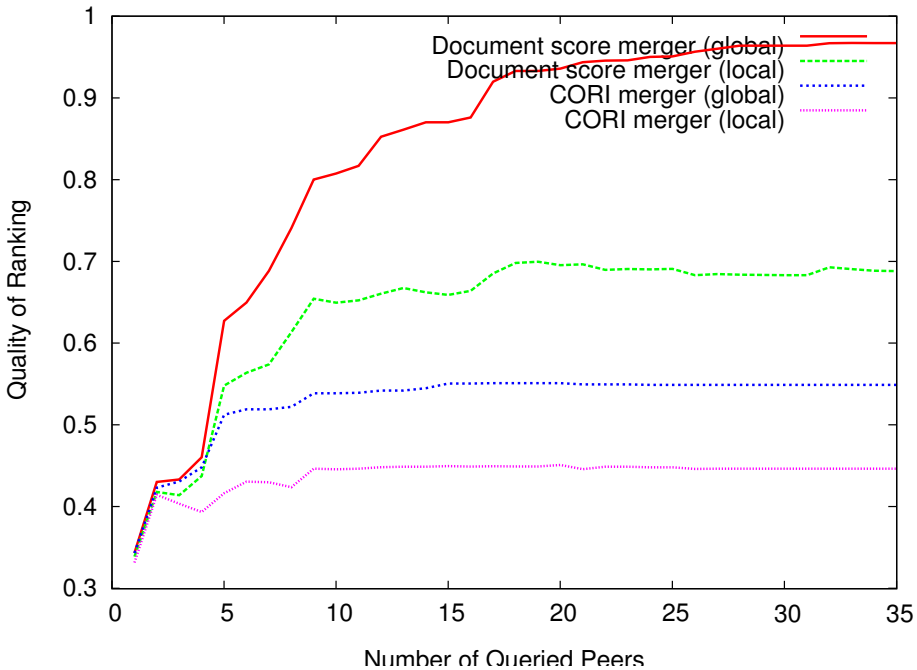

Figure 8.13: Quality of query results (40 Peers)

8.4 Influence of Directory Pruning on Query Routing

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 published 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 have high overlap with the local index of our peer p_0 — a problem we have addressed in Chapter 5 — 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; only these features should be published to the P2P directory. As most peers are distinguished authorities only for a few topics (characterized, e.g., by a set of terms), such a 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.

This section is based on our own work in [BMW06] and develops 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 metadata publishing efforts on its specific abilities and strengths. This classification and the resulting publishing 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.

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; a possible estimation technique for these values has been described in a previous section. 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 section 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-world Web data.

8.4.1 Peer Strategies for P2P Directory Posting

This subsection 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 statistical metadata objects 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:

- to decrease the network traffic, as fewer Posts are sent around at publishing time,
- to decrease the storage cost at the directory peers, and

- to decrease the traffic when retrieving the PeerLists for a particular query.

We argue that, if selecting the metadata carefully, 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.

Threshold Strategies

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

1. **Absolute Threshold.** Peers only publish statistical metadata 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 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 artifacts 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 before). 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*[7], 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 statistical metadata 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 two 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 receiving 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

[7] Google as of Nov 18 2005: 1,760,000,000 vs. 24,800,000

peers are publishing metadata for a term or, vice-versa, increase the threshold if there is enough metadata describing peers of reasonable quality.

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* [CG95], 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.

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:

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

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

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

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

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

For computing the MI values, a peer looks up the estimated values for *gdf* and 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 publishing metadata to the P2P directory.

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 *tf*idf*-style scoring function with the collection-wide *local df* values, as in many of today's P2P Web search approaches. As a document's *tf*idf* 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 8.14 shows a refined rank distance [BMWZ05] between the query results of the Minerva query execution (without any threshold) versus a reference result of a hypothetical 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 *tf*idf*-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.

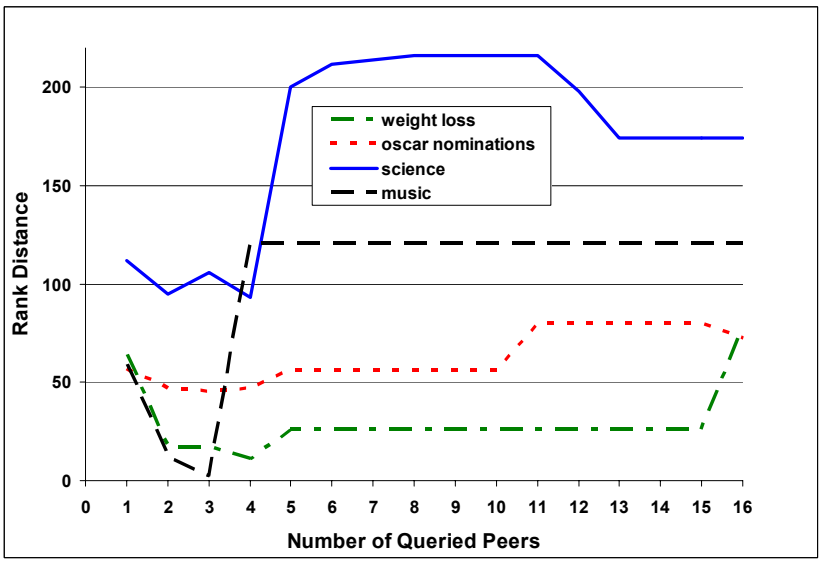

Figure 8.14: Noise reduction

8.4.2 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 re-use the 40 collection from Subsection 8.3.3. For the query workload we took 17 queries from Google Zeitgeist 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. We report on *relative recall* with respect to executing the query on this combined collection.

Performance Gains

We measure the total number of metadata objects in the system as a function of the *threshold* value. As can be seen in Tables 8.5 and 8.6, even small *threshold* values result in a significantly decreased number of metadata objects for both relative and absolute threshold strategies. The effect is even stronger for absolute thresholds, as a substantial fraction of terms contained in a collection only appear once or twice and are artifact 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 metadata objects 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.

Recall

Figures 8.15 and 8.16 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 (cf. Table 8.5)- is due to the two facts that

1. it reduces largely typing errors and other artifacts 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. Table 8.6). Note at this point that a threshold of 20% for a particular term means that a peer publishes 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.

8.4.3 Discussion

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.

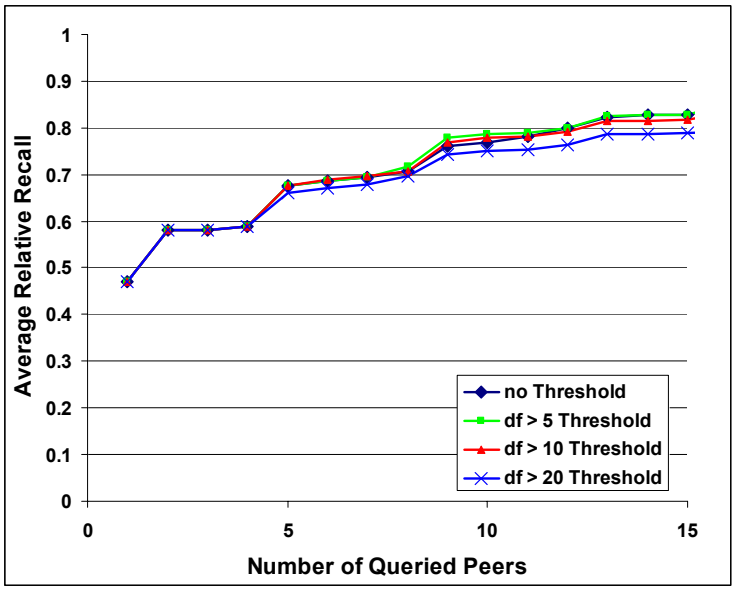

Figure 8.15: Absolute threshold

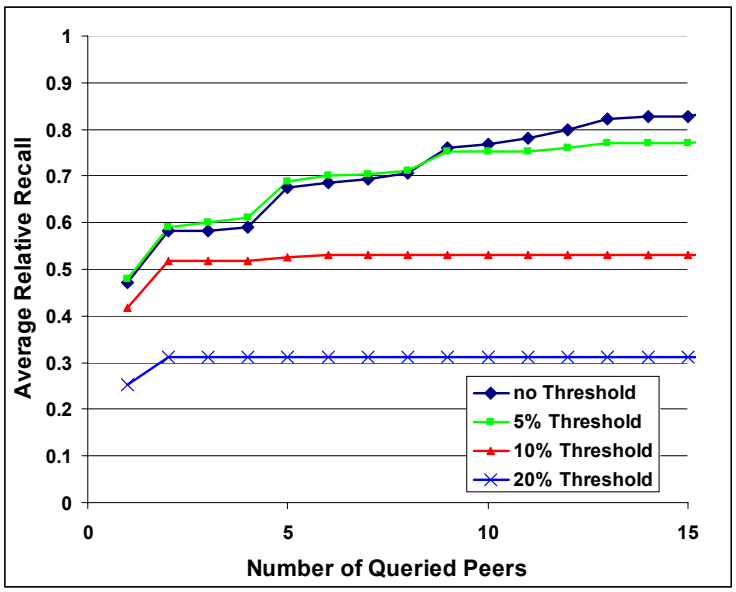

Figure 8.16: Relative threshold

	Peer 1	Peer 2	Peer 3	Peer 4	Peer 5	Peer 6	Peer 7	Peer 8	Peer 9	Peer 10
american	3527	5056	2784	4446	1655	3503	3544	1721	2362	8247
andy	89	446	55	23	54	269	701	41	288	254
award	1791	1846	1389	1792	1328	2048	10525	847	1443	1760
baseball	135	235	114	74	57	85	301	116	3553	485
berlin	1217	327	111	33	253	287	390	87	245	218
carmen	128	55	27	20	14	81	272	10	27	63
charley	10	7	6	10	7	26	77	13	6	29
chicago	1806	803	410	441	388	723	1182	231	612	1206
city	11080	3398	3381	1527	704	1914	4600	838	3001	6339
columbus	729	215	55	63	85	82	138	43	72	356
day	7468	4114	6725	4415	2793	4846	6425	4072	4637	7905
diane	34	190	46	79	27	94	341	43	13	291
eclipse	214	31	31	9	354	34	94	23	51	43
electra	42	20	16	2	2	6	104	3	9	13
emmy	10	25	4	5	10	36	297	7	13	52
father	211	474	187	210	120	498	1746	186	219	1194
fifa	32	12	43	11	1	11	55	0	848	5
gibson	93	118	38	26	104	217	354	37	68	325
gregori	52	162	57	72	339	91	328	32	69	300
haiti	449	210	58	51	79	115	59	73	60	256
hine	4	35	4	8	34	39	34	27	3	30
hurrican	118	27	69	51	140	19	146	108	116	322
illustrated	494	2350	307	419	837	523	416	777	541	689
iraq	487	363	205	76	100	96	212	115	301	2269
jame	927	1259	513	388	787	1034	2597	407	582	2674
klingerman	0	0	0	0	0	0	0	0	0	0
lake	1926	552	229	347	304	229	737	977	703	1493
lane	455	956	223	292	134	223	455	624	302	495
lebron	1	3	2	0	1	1	0	0	18	1
madrid	1159	182	60	32	121	51	103	10	196	120
marathon	111	22	23	37	14	43	100	10	229	65
marilyn	31	289	18	47	11	116	321	20	15	140
matrix	64	93	107	88	353	69	930	107	224	47
mel	27	65	26	32	11	805	428	19	59	347
monro	44	89	38	40	16	55	235	25	9	454
music	3121	4628	1899	824	275	18516	8920	434	3505	1646
nfl	59	31	92	14	3	19	127	5	1818	90
oscar	65	141	66	20	12	197	2287	21	146	129
poker	703	171	626	32	17	75	464	54	2150	64
real	2370	1818	3761	777	1187	2148	3110	636	2257	2338
reload	32	129	71	8	94	104	318	20	107	43
roddick	1	2	3	0	0	1	0	0	81	6
salt	298	191	61	339	147	70	336	459	394	395
series	619	1662	659	922	2080	1952	6585	876	1503	1484
solar	98	68	70	24	4420	55	165	114	98	140
sport	8206	3202	3415	2484	210	1420	2707	553	14310	6440
star	3825	991	818	427	2479	2056	8311	904	2280	1276
thailand	3107	356	155	100	93	242	255	223	166	242
virus	156	131	361	1504	58	287	231	235	194	159
war	126	213	96	32	46	622	2048	37	665	490
world	11280	6254	4841	2835	2975	6313	7466	3255	7961	11242

Table 8.4: Number of documents per peer for all query terms

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 8.5: 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 8.6: Relative threshold

Chapter 9

Conclusion

This thesis has addressed the problem of query routing: effectively and efficiently identifying peers that can return high-quality results for a given query, with a focus on P2P Web search applications. While prior state-of-the-art methods from the areas of distributed information retrieval and metasearch engines had not adequately addressed the peculiarities of a peer-to-peer network, this thesis has developed and evaluated a number of building blocks to tackle the shortcomings of those methods.

Our novel method for overlap-aware query routing addresses the fact that, in a network of autonomous peers, popular content is often indexed by a large fraction of peers. In this case, it is not a sufficiently smart strategy to identify peers with the highest expected result quality, but it is crucial to additionally incorporate the mutual overlap between the local collections of the peers — a peer might be able to provide high-quality results for a query, but if all its local results have already previously been contributed by other peers in the network, asking that peer is a waste of resources. In order to support overlap-awareness, we have extended the statistical metadata that are used for the query routing process with compact statistical synopses that allow an estimation of the degree of the pair-wise overlap between two peers' collections. Our IQN method combines this estimate in an iterative process with the expected result quality of a peer to increase the recall experienced by the query initiator, even and in particular for a small number of remote peers that are involved in the executing of a query. Our experiments on several real-world datasets have confirmed our claim that this is a great improvement to query routing.

A second key shortcoming of prior state-of-the-art methods for query routing has stemmed from the fact that, for scalability reasons, the metadata granularity has typically been limited to term-specific metadata, i.e., the metadata could capture the expected result quality of a peer for a particular query only with regard to a single term. Moreover, the metadata does not reflect individual documents, so that query routing for queries containing multiple terms may yield poor results. Our family of correlation-aware query routing methods addresses this issue by using the term-specific synopses to derive the expected result quality of a peer for a term *set* by means of well-founded mathematical operations on the synopses. Query routing decisions for multi-term queries can now actually be based on a sound estimation of the expected result quality of a peer for the query. Additionally, we have described a scalable method to identify

a judiciously chosen small number of term sets with certain characteristics for which an explicit handling promises even higher benefits for the query routing process.

This thesis has also combined overlap-aware and correlation-aware query routing to an even more sophisticated approach and has evaluated its performance. Not surprisingly, the actual benefit is highly dependent on the distribution of data in the network and also on the nature of the query load. Nevertheless, our experiments on real-world data have provided strong evidence that the degree of improvement that our methods can achieve over prior state-of-the-art methods is significant. This is a big leap towards making P2P Web search scalable, which has been questioned by parts of the prior research literature.

We have also presented a number of additional building blocks to enhance query routing, namely methods based on authority scores and global document occurrences, which future work should pursue further. We have also studied approaches to limit the load of the distributed directory by carefully selecting those parts of the metadata describing a peer that are likely to be important for query routing. We have presented an approach to estimate global document frequencies in the context of overlapping collections, which is an important building block for result merging. This estimation procedure can be generalized to different application classes that require distributed counting of distinct elements.

Finally, the P2P software prototype Minerva has been developed as part of this thesis. While originally being implemented in order to serve as a testbed for the experimental evaluation of our query routing methods, we have released Minerva to the public as open source software in the meantime.

Some aspects and research directions that were out of the immediate scope of this thesis are subject to future work. First, while this thesis has clearly focused on the retrieval effectiveness, a sound definition of an execution cost metric and its integration into the query routing framework is an open issue. Ultimately, the decision of whether or not to involve a remote peer in the query execution should not only consider its expected result quality, but also take into account cost-related factors (computational load, expected response time, available bandwidth). Second, the statistical synopses used in our query routing framework suffer from certain weaknesses. While Bloom filters tend to need a large number of bits to sufficiently represent a collection, hash sketches and min-wise independent permutations suffer from the fact that they cannot be natively intersected. Even though our experiments have shown that this is not a major issue, having more expressive synopses that are sufficiently compact and natively support intersection could further increase the performance improvement potential of our query routing approaches. Such synopses could readily be plugged into our framework.

Third, the interconnection of query routing with result merging is an interesting aspect that was largely put aside in this thesis. However, the presented approach to compute global document frequencies and the JXP approach to compute globally valid authority scores may seem to solve this issue as scores become globally comparable; this is only true if the effectiveness measure is relative recall. For complex evaluations with manual relevance assessments the situation could be quite different. Fourth (and closely related to the third point), personalization as a key ingredient to powerful P2P information systems is far from being studied exhaustively. As each peer is driven by a human user with a

personal interest profile, it is intriguing to further benefit from this knowledge, by means of explicit user feedback or by more implicit sources like query logs and click streams.

Finally, regarding the Minerva prototype, it is highly desirable to ease the public deployment of the software by running some stable bootstrap nodes inside our institute. In order to make the usage of Minerva as transparent and hassle-free as possible for the user, we have recently started to develop a light-weight version of Minerva that gathers local data by means of a Web proxy automatically indexing the content that a user views in the browser. This design choice will also prove useful for collecting the user feedback (query logs, click streams) mentioned before.

Bibliography

[AAS05] Divyakant Agrawal, Amr El Abbadi, and Subhash Suri. Attribute-based access to distributed data over p2p networks. In Subhash Bhalla, editor, *Databases in Networked Information Systems, 4th International Workshop, DNIS 2005, Aizu-Wakamatsu, Japan, March 28-30, 2005, Proceedings*, volume 3433 of *Lecture Notes in Computer Science*, pages 244–263. Springer, 2005.

[Abe01] Karl Aberer. P-grid: A self-organizing access structure for p2p information systems. In Carlo Batini, Fausto Giunchiglia, Paolo Giorgini, and Massimo Mecella, editors, *Cooperative Information Systems, 9th International Conference, CoopIS 2001, Trento, Italy, September 5-7, 2001, Proceedings*, volume 2172 of *Lecture Notes in Computer Science*, pages 179–194. Springer, 2001.

[ACMHP04] Karl Aberer, Philippe Cudré-Mauroux, Manfred Hauswirth, and Tim Van Pelt. Gridvine: Building internet-scale semantic overlay networks. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2004.

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993.*, pages 207–216. ACM Press, 1993.

[BCFM00] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *J. Comput. Syst. Sci.*, 60(3):630–659, 2000.

[BCMR04] John W. Byers, Jeffrey Considine, Michael Mitzenmacher, and Stanislav Rost. Informed content delivery across adaptive overlay networks. *IEEE/ACM Trans. Netw.*, 12(5):767–780, 2004.

[BKK+03] Hari Balakrishnan, M. Frans Kaashoek, David R. Karger, Robert Morris, and Ion Stoica. Looking up data in p2p systems. *Commun. ACM*, 46(2):43–48, 2003.

[Blo70] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[BM05] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2005.

[BMPC07] Matthias Bender, Sebastian Michel, Josiane Xavier Parreira, and Tom Crecelius. P2p web search: Make it light, make it fly (demo). In *3rd Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, CA, USA, January 7-10, 2007*, pages 164–168, 2007.

[BMS+06] Holger Bast, Debapriyo Majumdar, Ralf Schenkel, Martin Theobald, and Gerhard Weikum. Io-top-k: Index-access optimized top-k query processing. In Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim, editors, *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, pages 475–486. ACM, 2006.

[BMT+05a] Matthias Bender, Sebastian Michel, Peter Triantafillou, Gerhard Weikum, and Christian Zimmer. Improving collection selection with overlap awareness in p2p search engines. In Ricardo A. Baeza-Yates, Nivio Ziviani, Gary Marchionini, Alistair Moffat, and John Tait, editors, *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*, pages 67–74. ACM, 2005.

[BMT+05b] Matthias Bender, Sebastian Michel, Peter Triantafillou, Gerhard Weikum, and Christian Zimmer. Minerva: Collaborative p2p search. In Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi, editors, *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 1263–1266. ACM, 2005.

[BMT+06] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. P2p content search: Give the web back to the people. In *Fifth International Workshop on Peer-to-Peer systems, IPTPS 2006, Santa Barbara, CA, USA, February 26-28*, 2006.

[BMTW06] M. Bender, S. Michel, P. Triantafillou, and G. Weikum. Global document frequency estimation in peer-to-peer web search. In *Proceedings of the Ninth International Workshop on the Web & Databases (WebDB 2006), Chicago, Illinois, USA, Co-located with ACM CIKM 2006, November 11, 2006*, 2006.

[BMW06] Matthias Bender, Sebastian Michel, and Gerhard Weikum. P2p directories for distributed web search: From each according to his ability, to each according to his needs. In Roger S. Barga and Xiaofang Zhou, editors, *Proceedings of the 22nd International Conference on Data Engineering Workshops, ICDE 2006, 3-7 April 2006, Atlanta, GA, USA*, page 51. IEEE Computer Society, 2006.

[BMWZ04] Matthias Bender, Sebastian Michel, Gerhard Weikum, and Christian Zimmer. Bookmark-driven query routing in peer-to-peer web search. In Jamie Callan, Norbert Fuhr, and Wolfgang Nejdl, editors, *Peer-to-Peer Information Retrieval*, 2004.

[BMWZ05] Matthias Bender, Sebastian Michel, Gerhard Weikum, and Christian Zimmer. The minerva project: Database selection in the context of p2p search. In Gottfried Vossen, Frank Leymann, Peter C. Lockemann, and Wolffried Stucky, editors, *Datenbanksysteme in Business, Technologie und Web, 11. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme"(DBIS), Karlsruhe, 2.-4. März 2005*, volume 65 of *LNI*, pages 125–144. GI, 2005.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[CAPMN03] Francisco Matias Cuenca-Acuna, Christopher Peery, Richard P. Martin, and Thu D. Nguyen. Planetp: Using gossiping to build content addressable peer-to-peer information sharing communities. In *12th International Symposium on High-Performance Distributed Computing (HPDC-12 2003), 22-24 June 2003, Seattle, WA, USA*, pages 236–249. IEEE Computer Society, 2003.

[CCH92] James P. Callan, W. Bruce Croft, and Stephen M. Harding. The inquery retrieval system. In Isidro Ramos A. Min Tjoa, editor, *Proceedings of the Third Database and Expert Systems Applications, DEXA1992, Valencia, Spain*, pages 78–83. Springer, 1992.

[CG95] K. Church and W. Gale. Poisson mixtures. *Natural Language Engineering*, 1(2):163–190, 1995.

[CGM02] Arturo Crespo and Hector Garcia-Molina. Routing indices for peer-to-peer systems. In *22rd International Conference on Distributed Computing Systems (ICDCS 2002), 2-5 July 2002, Vienna, Austria*, pages 23–. IEEE Computer Society, 2002.

[Cha02] Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from HyperText Data*. Science & Technology Books, 2002.

[CLC95] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching distributed collections with inference networks. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *SIGIR'95, Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Seattle, Washington, USA, July 9-13, 1995 (Special Issue of the SIGIR Forum)*, pages 21–28. ACM Press, 1995.

[CMH+02] Ian Clarke, Scott G. Miller, Theodore W. Hong, Oskar Sandberg, and Brandon Wiley. Protecting free expression online with freenet. *IEEE Internet Computing*, 6(1):40–49, 2002.

[CW04] Pei Cao and Zhe Wang. Efficient top-k query calculation in distributed networks. In Soma Chaudhuri and Shay Kutten, editors, *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*, pages 206–215. ACM, 2004.

[DF03] Marianne Durand and Philippe Flajolet. Loglog counting of large cardinalities (extended abstract). In Giuseppe Di Battista and Uri Zwick, editors, *ESA*, volume 2832 of *Lecture Notes in Computer Science*, pages 605–617. Springer, 2003.

[DG77] P Diaconis and R Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society*, pages 262–268, 1977.

[Dia88] Persi Diaconis. *Group Representations in Probability and Statistics.* Institute of Mathematical Statistics, 1988.

[Fag02] Ronald Fagin. Combining fuzzy information: an overview. *SIGMOD Record*, 31(2):109–118, 2002.

[FCAB00] Li Fan, Pei Cao, Jussara M. Almeida, and Andrei Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Trans. Netw.*, 8(3):281–293, 2000.

[FKS03] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top k lists. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2003, Baltimore, Maryland, USA, January 12-14, 2003*, pages 28–36, 2003.

[FLN01] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 21-23, 2001, Santa Barbara, California, USA*. ACM, 2001.

[FM85] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[FSGM+98] Min Fang, Narayanan Shivakumar, Hector Garcia-Molina, Rajeev Motwani, and Jeffrey D. Ullman. Computing iceberg queries efficiently. In Ashish Gupta, Oded Shmueli, and Jennifer Widom, editors, *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, pages 299–310. Morgan Kaufmann, 1998.

[Fuh99] Norbert Fuhr. A decision-theoretic approach to database selection in networked ir. *ACM Trans. Inf. Syst.*, 17(3):229–249, 1999.

[GBK00] Ulrich Güntzer, Wolf-Tilo Balke, and Werner Kießling. Optimizing multi-feature queries for image databases. In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal,

Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang, editors, *Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 419–428. Morgan Kaufmann, 2000.

[GGG+03] P. Krishna Gummadi, Ramakrishna Gummadi, Steven D. Gribble, Sylvia Ratnasamy, Scott Shenker, and Ion Stoica. The impact of dht routing geometry on resilience and proximity. In Anja Feldmann, Martina Zitterbart, Jon Crowcroft, and David Wetherall, editors, *Proceedings of the ACM SIGCOMM 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 25-29, 2003, Karlsruhe, Germany*, pages 381–394. ACM, 2003.

[GGM95] Luis Gravano and Hector Garcia-Molina. Generalizing gloss to vector-space databases and broker hierarchies. In Umeshwar Dayal, Peter M. D. Gray, and Shojiro Nishio, editors, *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland.*, pages 78–89. Morgan Kaufmann, 1995.

[GGMT94] Luis Gravano, Hector Garcia-Molina, and Anthony Tomasic. The effectiveness of gloss for the text database discovery problem. In Richard T. Snodgrass and Marianne Winslett, editors, *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 24-27, 1994*, pages 126–137. ACM Press, 1994.

[GGMT99] Luis Gravano, Hector Garcia-Molina, and Anthony Tomasic. Gloss: Text-source discovery over the internet. *ACM Trans. Database Syst.*, 24(2):229–264, 1999.

[GM05] Hector Garcia-Molina. P2P Search. Panel discussion. In *2nd Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, CA, USA, January 4-7, 2005*, 2005.

[gnu00] gnutella.wego.com. Gnutella: Distributed information sharing., 2000.

[GWJD03] Leonidas Galanis, Yuan Wang, Shawn R. Jeffery, and David J. DeWitt. Locating data sources in large distributed systems. In Johann Christoph Freytag, Peter C. Lockemann, Serge Abiteboul, Michael J. Carey, Patricia G. Selinger, and Andreas Heuer, editors, *VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany*, pages 874–885. Morgan Kaufmann, 2003.

[HHL+03] Ryan Huebsch, Joseph M. Hellerstein, Nick Lanham, Boon Thau Loo, Scott Shenker, and Ion Stoica. Querying the internet with pier. In Johann Christoph Freytag, Peter C. Lockemann, Serge Abiteboul, Michael J. Carey, Patricia G. Selinger, and Andreas Heuer, editors, *VLDB 2003, Proceedings of 29th International*

Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany, pages 321–332. Morgan Kaufmann, 2003.

[HK00] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[HK05] Thomas Hernandez and Subbarao Kambhampati. Improving text collection selection with coverage and overlap statistics. In Allan Ellis and Tatsuya Hagino, editors, *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005 - Special interest tracks and posters*, pages 1128–1129. ACM, 2005.

[HT99] David Hawking and Paul B. Thistlewaite. Methods for information server selection. *ACM Trans. Inf. Syst.*, 17(1):40–76, 1999.

[KG90] M. Kendall and J. D. Gibbons. *Rank Correlation Methods.* Charles Griffin, 1990.

[KHMG03] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Exploiting the block structure of the web for computing pagerank. Technical report, Stanford University, 2003.

[Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[Kle00] Jon M. Kleinberg. The small-world phenomenon: an algorithm perspective. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA. ACM, 2000*, pages 163–170, 2000.

[KLFW06] Nils Kammenhuber, Julia Luxenburger, Anja Feldmann, and Gerhard Weikum. Web search clickstreams. In Jussara M. Almeida, Virgílio A. F. Almeida, and Paul Barford, editors, *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement 2006, Rio de Janeriro, Brazil, October 25-27, 2006*, pages 245–250. ACM, 2006.

[LAE+04] Kristen LeFevre, Rakesh Agrawal, Vuk Ercegovac, Raghu Ramakrishnan, Yirong Xu, and David J. DeWitt. Limiting disclosure in hippocratic databases. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 108–119. Morgan Kaufmann, 2004.

[LC03] Jie Lu and James P. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management, New Orleans, Louisiana, USA, November 2-8, 2003*, pages 199–206. ACM, 2003.

[LCC00] Leah S. Larkey, Margaret E. Connell, and James P. Callan. Collection selection and results merging with topically organized u.s. patents and trec data. In *Proceedings of the 2000 ACM CIKM International Conference on Information and Knowledge Management, McLean, VA, USA, November 6-11, 2000*, pages 282–289. ACM, 2000.

[LKP+06] Toan Luu, Fabius Klemm, Ivana Podnar, Martin Rajman, and Karl Aberer. Alvis peers: a scalable full-text peer-to-peer retrieval engine. In *P2PIR '06: Proceedings of the international workshop on Information retrieval in peer-to-peer networks*, pages 41–48, New York, NY, USA, 2006. ACM Press.

[LLH+03] Jinyang Li, Boon Thau Loo, Joseph M. Hellerstein, M. Frans Kaashoek, David R. Karger, and Robert Morris. On the feasibility of peer-to-peer web indexing and search. In M. Frans Kaashoek and Ion Stoica, editors, *Peer-to-Peer Systems II, Second International Workshop, IPTPS 2003, Berkeley, CA, USA, February 21-22,2003, Revised Papers*, volume 2735 of *Lecture Notes in Computer Science*, pages 207–215. Springer, 2003.

[LNBK02] David Liben-Nowell, Hari Balakrishnan, and David R. Karger. Analysis of the evolution of peer-to-peer systems. In *Proceedings of the Twenty-First Annual ACM Symposium on Principles of Distributed Computing, PODC 2002, Monterey, CA, USA, July 21-24, 2002*, pages 233–242. ACM, 2002.

[LNS96] Witold Litwin, Marie-Anne Neimat, and Donovan A. Schneider. Lh* - a scalable, distributed data structure. *ACM Trans. Database Syst.*, 21(4):480–525, 1996.

[MBN+06] Sebastian Michel, Matthias Bender, Nikos Ntarmos, Peter Triantafillou, Gerhard Weikum, and Christian Zimmer. Discovering and exploiting keyword and attribute-value co-occurrences to improve p2p routing indices. In Philip S. Yu, Vassilis J. Tsotras, Edward A. Fox, and Bing Liu, editors, *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management, Arlington, Virginia, USA, November 6-11, 2006*, pages 172–181. ACM, 2006.

[MBT+05] Sebastian Michel, Matthias Bender, Peter Triantaffilou, Gerhard Weikum, and Christian Zimmer. P2p web search with minerva: How do you want to search tomorrow? (demonstration). In Gustavo Alonso, editor, *Middleware*, volume 3790 of *Lecture Notes in Computer Science*. Springer, 2005.

[MBTW06] Sebastian Michel, Matthias Bender, Peter Triantafillou, and Gerhard Weikum. Iqn routing: Integrating quality and novelty in p2p querying and ranking. In Yannis E. Ioannidis, Marc H. Scholl, Joachim W. Schmidt, Florian Matthes, Michael Hatzopoulos, Klemens Böhm, Alfons Kemper, Torsten Grust, and Christian Böhm,

editors, *Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology, Munich, Germany, March 26-31, 2006, Proceedings*, volume 3896 of *Lecture Notes in Computer Science*, pages 149–166. Springer, 2006.

[MEH05] Wolfgang Müller, Martin Eisenhardt, and Andreas Henrich. Scalable summary based retrieval in p2p networks. In Otthein Herzog, Hans-Jörg Schek, Norbert Fuhr, Abdur Chowdhury, and Wilfried Teiken, editors, *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 - November 5, 2005*, pages 586–593. ACM, 2005.

[Mic07] S. Michel. *Top-k Aggregation Queries in Large-Scale Distributed Systems*. PhD thesis, Universität des Saarlandes, 2007.

[MLS99] David R. H. Miller, Tim Leek, and Richard M. Schwartz. A hidden markov model information retrieval system. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15-19, 1999, Berkeley, CA, USA*, pages 214–221. ACM, 1999.

[MMK+05] Volker Markl, Nimrod Megiddo, Marcel Kutsch, Tam Minh Tran, Peter J. Haas, and Utkarsh Srivastava. Consistently estimating the selectivity of conjuncts of predicates. In Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi, editors, *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 373–384. ACM, 2005.

[MNR02] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy: a scalable and dynamic emulation of the butterfly. In *Proceedings of the Twenty-First Annual ACM Symposium on Principles of Distributed Computing, PODC 2002, Monterey, CA, USA, July 21-24, 2002*, pages 183–192. ACM, 2002.

[MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT, Cambridge, Massachusetts, 1999.

[MS05] Peter Mahlmann and Christian Schindelhauer. Peer-to-peer networks based on random transformations of connected regular undirected graphs. In Phillip B. Gibbons and Paul G. Spirakis, editors, *SPAA 2005: Proceedings of the 17th Annual ACM Symposium on Parallel Algorithms, July 18-20, 2005, Las Vegas, Nevada, USA*, pages 155–164. ACM, 2005.

[MTW05a] Sebastian Michel, Peter Triantafillou, and Gerhard Weikum. Klee: A framework for distributed top-k query algorithms. In Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi, editors, *Proceedings of the*

31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005, pages 637–648. ACM, 2005.

[MTW05b] Sebastian Michel, Peter Triantafillou, and Gerhard Weikum. Minerva$_{\text{infinity}}$: A scalable efficient peer-to-peer search engine. In Gustavo Alonso, editor, *Middleware*, volume 3790 of *Lecture Notes in Computer Science*, pages 60–81. Springer, 2005.

[MYL02] Weiyi Meng, Clement T. Yu, and King-Lup Liu. Building efficient and effective metasearch engines. *ACM Comput. Surv.*, 34(1):48–89, 2002.

[NF03] Henrik Nottelmann and Norbert Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. In *SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 - August 1, 2003, Toronto, Canada*, pages 290–297. ACM, 2003.

[NF06] Henrik Nottelmann and Norbert Fuhr. Comparing different architectures for query routing in peer-to-peer networks. In Mounia Lalmas, Andy MacFarlane, Stefan M. Rüger, Anastasios Tombros, Theodora Tsikrika, and Alexei Yavlinsky, editors, *Advances in Information Retrieval, 28th European Conference on IR Research, ECIR 2006, London, UK, April 10-12, 2006, Proceedings*, volume 3936 of *Lecture Notes in Computer Science*, pages 253–264. Springer, 2006.

[NKH03] Zaiqing Nie, Subbarao Kambhampati, and Thomas Hernandez. Bibfinder/statminer: Effectively mining and using coverage and overlap statistics in data integration. In Johann Christoph Freytag, Peter C. Lockemann, Serge Abiteboul, Michael J. Carey, Patricia G. Selinger, and Andreas Heuer, editors, *VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany*, pages 1097–1100. Morgan Kaufmann, 2003.

[NOT02] Wee Siong Ng, Beng Chin Ooi, and Kian-Lee Tan. Bestpeer: A self-configurable peer-to-peer system. In *Proceedings of the 18th International Conference on Data Engineering, ICDE, 26 February - 1 March 2002, San Jose, CA*, page 272. IEEE Computer Society, 2002.

[NR99] Surya Nepal and M. V. Ramakrishna. Query processing issues in image (multimedia) databases. In *Proceedings of the 15th International Conference on Data Engineering, 23-26 March 1999, Sydney, Austrialia*, pages 22–29. IEEE Computer Society, 1999.

[Ora01] Andy Oram, editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.

[PC98] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia*, pages 275–281. ACM, 1998.

[PDMW06] Josiane Xavier Parreira, Debora Donato, Sebastian Michel, and Gerhard Weikum. Efficient and decentralized pagerank approximation in a peer-to-peer web search network. In Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim, editors, *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, pages 415–426. ACM, 2006.

[PF03] Allison L. Powell and James C. French. Comparing the performance of collection selection algorithms. *ACM Trans. Inf. Syst.*, 21(4):412–456, 2003.

[PMB06] Josiane Xavier Parreira, Sebastian Michel, and Matthias Bender. Size doesn't always matter: Exploiting pagerank for query routing in distributed ir. In *Proceedings of the Workshop on Information Retrieval in Peer-to-Peer Networks (P2PIR 2006), Arlington, Virginia, USA, Co-located with ACM SIGMOD/PODS 2006, June 30, 2006*, 2006.

[PMBW05] Odysseas Papapetrou, Sebastian Michel, Matthias Bender, and Gerhard Weikum. On the usage of global document occurrences in peer-to-peer information systems. In *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, OTM Confederated International Conferences, Agia Napa, Cyprus, October 31-November 4 20045*, 2005.

[PRL+07] I. Podnar, M. Rajman, T. Luu, F. Klemm, and K. Aberer. Scalable peer-to-peer web retrieval with highly discriminative keys. In *Proceedings of the 23nd International Conference on Data Engineering, ICDE, 15-20 April 2007, Istanbul, Turkey*, 2007.

[PRR97] C. Greg Plaxton, Rajmohan Rajaraman, and Andréa W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *SPAA*, pages 311–320. ACM, 1997.

[RD01] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In Rachid Guerraoui, editor, *Middleware*, volume 2218 of *Lecture Notes in Computer Science*, pages 329–350. Springer, 2001.

[RFH+01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard M. Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of the ACM SIGCOMM Conference on Applications,*

Technologies, Architectures, and Protocols for Computer Communication, August 27-31, San Diego, CA, USA., pages 161–172. ACM, 2001.

[RV03] Patrick Reynolds and Amin Vahdat. Efficient peer-to-peer keyword searching. In Markus Endler and Douglas C. Schmidt, editors, *Middleware 2003, ACM/IFIP/USENIX International Middleware Conference, Rio de Janeiro, Brazil, June 16-20, 2003, Proceedings*, volume 2672 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2003.

[RW94] Stephen E. Robertson and Steve Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, pages 232–241. ACM/Springer, 1994.

[SC99] Fei Song and W. Bruce Croft. A general language model for information retrieval (poster abstract). In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15-19, 1999, Berkeley, CA, USA*, pages 279–280. ACM, 1999.

[SE00] Atsushi Sugiura and Oren Etzioni. Query routing for web search engines: architecture and experiments. *Computer Networks*, 33(1-6):417–429, 2000.

[SJCO02] Luo Si, Rong Jin, James P. Callan, and Paul Ogilvie. A language modeling framework for resource selection and results merging. In *Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management, McLean, VA, USA, November 4-9, 2002*, pages 391–397. ACM, 2002.

[SMK+01] Ion Stoica, Robert Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 27-31, San Diego, CA, USA.*, pages 149–160. ACM, 2001.

[SMwW+03] Torsten Suel, Chandan Mathur, Jo wen Wu, Jiangong Zhang, Alex Delis, Mehdi Kharrazi, Xiaohui Long, and Kulesh Shanmugasundaram. Odissea: A peer-to-peer architecture for scalable web search and information retrieval. In Vassilis Christophides and Juliana Freire, editors, *International Workshop on Web and Databases, San Diego, California, June 12-13, 2003*, pages 67–72, 2003.

[SSB03] Karthikeyan Sankaralingam, Simha Sethumadhavan, and James C. Browne. Distributed pagerank for p2p systems. In *12th International Symposium on High-Performance Distributed*

Computing (HPDC-12 2003), 22-24 June 2003, Seattle, WA, USA, pages 58–69. IEEE Computer Society, 2003.

[STS+03] Sergej Sizov, Martin Theobald, Stefan Siersdorfer, Gerhard Weikum, Jens Graupmann, Michael Biwer, and Patrick Zimmer. The bingo! system for information portal generation and expert web search. In *1st Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, CA, USA, January 5-8, 2003*, 2003.

[SW05] Ralf Steinmetz and Klaus Wehrle, editors. *Peer-to-Peer Systems and Applications*, volume 3485 of *Lecture Notes in Computer Science*. Springer, 2005.

[SYYW03] Shuming Shi, Jin Yu, Guangwen Yang, and Dingxing Wang. Distributed page ranking in structured p2p networks. In *32nd International Conference on Parallel Processing (ICPP 2003), 6-9 October 2003, Kaohsiung, Taiwan*, pages 179–186. IEEE Computer Society, 2003.

[TC91] Howard R. Turtle and W. Bruce Croft. Evaluation of an inference network-based retrieval model. *ACM Trans. Inf. Syst.*, 9(3):187–222, 1991.

[TD04] Chunqiang Tang and Sandhya Dwarkadas. Hybrid global-local indexing for efficient peer-to-peer information retrieval. In *1st Symposium on Networked Systems Design and Implementation (NSDI 2004), March 29-31, 2004, San Francisco, California, USA, Proceedings*, pages 211–224. USENIX, 2004.

[Tex] Text REtrieval Conference (TREC). http://trec.nist.gov/.

[TSW05] Martin Theobald, Ralf Schenkel, and Gerhard Weikum. An efficient and versatile query engine for topx search. In Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi, editors, *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 625–636. ACM, 2005.

[WA05] Jie Wu and Karl Aberer. Using a layered markov model for distributed web ranking computation. In *25th International Conference on Distributed Computing Systems (ICDCS 2005), 6-10 June 2005, Columbus, OH, USA*, pages 533–542. IEEE Computer Society, 2005.

[WGD03] Yuan Wang, Leonidas Galanis, and David J. DeWitt. Galanx: An efficient peer-to-peer search engine system. Technical report, University of Wisconsin - Madison, 2003.

[WMYL01] Zonghuan Wu, Weiyi Meng, Clement T. Yu, and Zhuogang Li. Towards a highly-scalable and effective metasearch engine. In *Proceedings of the Tenth International World Wide Web Conference,*

WWW 10, Hong Kong, China, May 1-5, 2001, pages 386–395. ACM, 2001.

[XC99] Jinxi Xu and W. Bruce Croft. Cluster-based language models for distributed retrieval. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15-19, 1999, Berkeley, CA, USA*, pages 254–261. ACM, 1999.

[ZCM02] Yi Zhang, James P. Callan, and Thomas P. Minka. Novelty and redundancy detection in adaptive filtering. In *SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 11-15, 2002, Tampere, Finland*, pages 81–88. ACM, 2002.

[Zip49] George Kingsley Zipf. *Human Behaviour and the Principle of Least Effort: an Introduction to Human Ecology.* Addison-Wesley, 1949.

[ZL01] ChengXiang Zhai and John D. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In W. Bruce Croft, David J. Harper, Donald H. Kraft, and Justin Zobel, editors, *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA*, pages 334–342. ACM, 2001.

[ZM06] Justin Zobel and Alistair Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2), 2006.