Discovering and Exploiting Keyword and Attribute-Value Co-occurrences to Improve P2P Routing Indices

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

[†] Max-Planck-Institut für Informatik
66123 Saarbrücken, Germany

{smichel, mbender, weikum, czimmer}@mpi-inf.mpg.de

[‡] RACTI and University of Patras
Rio, 26500, Greece

{ntarmos,peter}@ceid.upatras.gr

ABSTRACT

Peer-to-Peer (P2P) search requires intelligent decisions for *query routing*: selecting the best peers to which a given query, initiated at some peer, should be forwarded for retrieving additional search results. These decisions are based on statistical summaries for each peer, which are usually organized on a per-keyword basis and managed in a distributed directory of routing indices. Such architectures disregard the possible correlations among keywords. Together with the coarse granularity of per-peer summaries, which are mandated for scalability, this limitation may lead to poor search result quality.

This paper develops and evaluates two solutions to this problem, *sk-STAT* based on single-key statistics only, and *mk-STAT* based on additional multi-key statistics. For both cases, hash sketch synopses are used to compactly represent a peer's data items and are efficiently disseminated in the P2P network to form a decentralized directory. Experimental studies with Gnutella and Web data demonstrate the viability and the trade-offs of the approaches.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*selection process, information filtering*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Distributed Systems*

General Terms

Algorithms, Design, Experimentation

Keywords

Peer-to-Peer information systems, distributed IR, query routing, key co-occurrences

CIKM'06, November 5–11, 2006, Arlington, Virginia, USA.

1. INTRODUCTION

1.1 Motivation

Decentralized search on top of peer-to-peer overlay architectures is an intriguing research direction which aims to interconnect many peers, each with its own local data collection, and to utilize the aggregated resources of the underlying computers for decentralized large-scale search. The goal is to improve search result quality with unlimited scalability and better capabilities for exploiting user behavior and recommendations (e.g., click streams, bookmarks, etc.) [12, 43, 27, 24]. Applications for such a scenario are manifold. While early research was often driven by file sharing applications, where the search space typically consists of keywords contained in file names or manually given annotations, today a huge amount of more challenging application classes are emerging. Consider a photo sharing community that - free of commercial interests - wants to pool and search personal snapshots, e.g., taken at holiday sites. The search space in this scenario includes the examples given before, but also high-dimensional space image features, user annotations, or automatically generated annotations, e.g. GPS coordinates.

As another scenario, we envision P2P Web search engines with thousands or millions of peers. The data shared by the *data peers* may be web pages harvested from Web crawls or specialized data collections. Here, the search space typically consists of keywords (or terms in IR jargon). Each peer autonomously builds and maintains its own topic-specific or personalized document collection that reflects the user's thematic interests; this could be done, for example, by using a focused crawler with a specifically trained classifier.

When a peer issues a query, it should first be executed locally on the peer's own collection. Only when this local search does not return satisfactory or sufficiently many results, the system can contact *directory peers* that - by fully decentralized means, such as distributed hash tables - maintain summaries describing the local data collections of the data peers. Eventually, the query should be forwarded to a small number of judiciously chosen data peers that would have high likelihood of providing highly relevant additional results. This *query routing decision* (aka. dynamic peer or database or resource selection) is the very core of a P2P search engine.

The selection of the most promising peers for a query with multiple keywords or attribute-values is driven by statistical summaries that the P2P system keeps about the peers' local data collections. These summaries are themselves distributed across the P2P network and can be managed in a variety of ways, e.g., in the form of *routing indices* stored at each peer [16, 27, 42] or in the form of a P2P *directory* built

on top of a distributed hash table (DHT) (e.g., [1, 17, 37, 34]) or other kind of overlay network. For scalability, the summaries 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, for 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 [4, 29], 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.

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 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.

1.2 Contribution

In this paper we develop and evaluate two conceptually diverse approaches to address the above stated problem: *sk-STAT*, using the already existing single-keyword statistics to estimate a peer's quality for key sets, and *mk-STAT*, enhancing the distributed directory to explicitly include also statistical information about judiciously chosen sets of multiple keys.

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 in the rest of the paper.

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 the paper's novel key contributions is how to make this 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) [20] 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.

The complete procedure is fully implemented in the P2P search prototype system Minerva [7], and our experimental studies demonstrate the viability of the method and its performance improvements over the prior state of the art.

The rest of the paper is organized as follows. We conclude the introduction with a brief overview of related work. Section 2 introduces background information on distributed hash tables and hash sketches, major building blocks in both of our approaches. Section 3 presents the system architecture for our correlation-aware P2P search network. Section 4 discusses measures of key correlations. Sections 5 and 6 present our algorithms, sk-STAT and mk-STAT, for utilizing statistics on key sets to improve the query routing process. Section 7 discusses the scalability properties of our methods. Section 8 presents experimental results with two major setups: one based on Gnutella-style file sharing data, one based on Web data. Section 9 concludes this paper.

1.3 Related Work

In contrast to the early forms of unstructured P2P networks based on message flooding and other forms of epidemic dissemination, all structured P2P systems, e.g., for file sharing or sensor data management, build on some form of routing indices or directories that are either kept locally at each peer or at designated peers [42, 24].

The latter can be chosen randomly, hash-based, or based on attribute range partitioning, and can be replicated if needed. The indices or directories contain peer lists with the best peers for individual keywords or attribute values, known to the local peer or directory peer. If the network topology can be influenced, some of these best peers may be chosen as the local peer's neighbors. A variant of this predominant approach is that peers are classified into topics (e.g., Jazz music or Alpine climbing) and thematic related peers form neighbors of a "semantic overlay network" [16, 39, 14, 40, 36, 2]. Recent work has addressed performance trade-offs and optimization issues for building and maintaining routing indices [42, 24, 27, 16].

Recent work on P2P IR is related to earlier research on distributed IR and metasearch engines [12, 32], but older work assumed a small and static set of collections among

which a query had to choose. In contrast, P2P systems consider a much larger scale and also face high dynamics, which rules out comprehensive and complex statistical models for query routing. The most important routing methods in the literature are CORI [13], simple forms of statistical language models [35, 30], the decision-theoretic framework (DTF) [21], and the overlap-aware method [6, 33].

All of the above methods organize the statistics about peers, which drive the query routing decisions, on a per-term basis and thus disregard term correlations. The only recent works that consider term correlations in the context of P2P search are [8] and [3]. [8] 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. [3] 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.

Synopses for compact approximation of sets, multisets, and their statistical properties have recently received attention in the context of sensor networks, data streams, content delivery, and estimation issues in structured databases [15]. The methods under consideration include Bloom filters [9], hash sketches [20], and min-wise independent permutations [11], all of which are hash-based but differ in their strengths and limitations of representing various kinds of properties. The latter include set membership testing, set cardinality estimation, estimating the number of distinct elements in a multiset, forming intersections, unions, differences, etc. Overlap-awareness in P2P IR has long been overlooked, with the exceptions of [23] and the recent work in [33], which focuses on utilizing such synopses for estimating the overlap of the data collections at different peers.

2. BACKGROUND

2.1 Distributed Hash Tables

Distributed Hash Tables, DHTs, (such as Chord [37], CAN [34], Pastry [17], etc.) have emerged as the preferred family of structured architectures for overlay P2P networks. The main advantage of DHTs compared to unstructured P2P networks stems from the performance guarantees that they can offer regarding the routing efficiency and ultimately the network scalability, even in the presence of high network dynamics (such as high rates of node arrivals/departures and failures/recoveries). DHTs offer two basic primitives: *insert(key, value)* and *lookup(key)*. DHT nodes and data items are assigned unique IDs. The node IDs dictate the place occupied by the node in the overlay topology. Each node is responsible for a well-defined subspace of the items' ID space. Each item is stored at the node responsible for the subspace containing the item's ID. To facilitate efficient and scalable routing, each node in an N-node DHT maintains the IP addresses (aka fingers) to $O(\log(N))$[1] other nodes in appropriate positions in the overlay. With this $O(\log(N))$ routing state at each node, DHTs guarantee that routing between any two nodes requires an expected $O(\log(N))$ messages.

2.2 Hash Sketches

Flajolet and Martin in [20] 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 [18] (*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,

$$\rho(y) = \min_{k \geq 0} bit(y, k) \neq 0,\ y > 0 \quad (1)$$

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

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

With the above process, note that in the bit vector B hosting the hash sketch, $B[0]$ is expected to be set to 1 $n/2$ times, $B[1]$ $n/4$ 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[18].

2.3 Combining Hash Sketches

A key property of hash sketches 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

$$|S_A \cap S_B| = |S_A| + |S_B| - |S_A \cup S_B| \quad (3)$$

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:

$$|\bigcup_{i=1}^{n} S_i| = \sum_{k=1}^{n} (-1)^{k+1} \sum_{\substack{I \subseteq \{1,\ldots,n\}, i \in I \\ |I|=k}} |\bigcap S_i| \quad (4)$$

3. SYSTEM ARCHITECTURE

Table 1 summarizes the notation we will be using throughout the rest of this paper.

[1] All $\log(\cdot)$ notation refers to base-2 logarithms.

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 1: Summary of Notation

The operational environment is a P2P-based search engine. Every peer builds local collections and local per-key indexes, (and perhaps also combining data from external sources), according to the user's thematic profile. We shall refer to such peers as the *data peers*. The local per-key indexes consist of inverted lists of data items relevant to the corresponding key. Because the peers are functioning independently, it is expected that there will be considerable overlap in the data collections (e.g. the set of data items known to individual peers) and (consequently) in the indexes maintained by different peers.

Directory peers are responsible for maintaining very compact, aggregated meta-information about the peers' local indexes (to the extent that the individual data peers are willing to share). Naturally, each data peer can potentially also be a directory peer. More specifically, each directory peer is responsible for the meta-information from all over the network regarding a random subset of all keys. For performance and fault tolerance reasons, the directory metadata for a key may be replicated across multiple directory peers. A DHT is used to connect all (directory and data) peers. The DHT's *lookup* method can be used to determine the peer responsible for a particular key, in order to send information or retrieve information about this keyword.

3.1 Publishing Statistical Summaries

Every data peer publishes per-key summaries describing its local index to the distributed directory; these summaries will be referred to as *Posts*. The DHT's lookup method determines the directory peer currently responsible for a key, which maintains a PeerList, i.e., a list of all data peers' Posts regarding this key. Posts contain contact information about the publishing data peer, together with statistics to calculate quality measures for a key (e.g., frequencies or IR-style scores). Note again that this information does only capture the overall quality of a peer's index for a particular key, but does not represent individual data items. Typically, such statistics include the length of the inverted list for the key, the maximum and average score among the key's inverted list entries, etc. These statistics are used to support the *query routing* process, i.e., determining the most promising peers for a particular query. Specifically, the size of the inverted list for a key – that is, the frequency for key a at peer i, or $df_i(a)$ – can be maintained in the form of a hash sketch; for each data item in p_i's collection regarding key a, the peer inserts an identifier into a local hash sketch for this key, denoted $HS_i(a)$.

Since there is a well-defined directory peer responsible for each key, the hash sketch synopses representing the index lists of all peers for a particular key a are *all* sent to the same directory peer $p(a)$. Thus, $p(a)$ can compute a moving-window estimate for the global df value of a – $df(a)$ – by performing an inexpensive bitwise-OR of all individual hash sketches $HS_i(a)$ sent by every peer i for term a.

To deal with the high dynamics in a P2P network, each Post is assigned a Time-to-Live (TTL) value. If the originator peer has not updated (refreshed) its Post after this time interval, it is discarded. On the other hand, if a peer's metadata has not changed in the meantime, it can refresh its metadata by simple means, without resending it completely.

3.2 Execution of Multi-Key Queries

Given the above, in general, query routing for a single-key query a proceeds as follows. First, the query initiator issues a request for the Posts 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 summary 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 published statistics for *all* queried keys. However, this approach may fail miserably. A peer appearing in both PeerLists for a and b and, thus, in the intersection of Posts for the keys a and b, is only guaranteed to have data items regarding a or b separately, but *not* necessarily data items regarding both a and b simultaneously. To illustrate this point, 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 posted 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!

Sections 5 and 6 present two different approaches to overcome this problem, striking different compromises: while 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, mk-STAT enhances the distributed directory by adding explicit statistics for judiciously selected key sets, namely, those that exhibit particularly high correlation or other forms of strong association among the individual keys in the key set. The choice of an appropriate correlation measure is the topic of the next section.

4. 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 the subsequent sections. We will restrict ourselves to key pairs, as we expect the major benefit when we move from single keys to correlated pairs.

The obvious choice for standard measures like 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), 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

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 2: Selected Conditional Probabilities

estimator:

$$\hat{P}(A|B) = \frac{df(ab)/|D|}{df(b)/|D|} = \frac{df(ab)}{df(b)} \quad (5)$$

where

$$df(ab) = df(a) + df(b) - df(a \cup b) \quad (6)$$

and $df(a \cup b)$ can be estimated by taking the bitwise OR of $HS(a)$ and $HS(b)$ (see Section 2.3). 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 [5, 19] 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 that such techniques are feasible within our P2P architecture without incurring extra communication costs. For data items, a similar approach is conceivable but it 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 some 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 5, 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 when a key pair, identified in the discovery step, deserves special action for posting 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 Posts; 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 have negative correlation close to minus one, there are only few data items in the network that contain both a and b, and 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 (e.g., $\beta = 0.1$) within the data items. Table 2 shows some conditional probability estimates for popular Google queries[2], 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 Post 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.

5. SK-STAT: SINGLE-KEY STATISTICS

Recall that the Post for 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 3 from Section 2.3, 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,
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 to 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 Posts for a and b, a peer probably has to retrieve and inspect the PeerLists for a and b at much higher depth, compared 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 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

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

peer in each list. This effect leads to much higher network load on the directory peers and the query initiator. If, like in mk-STAT, there readily exist Posts for combined key sets like 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 Posts (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. 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. 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 Posts, 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.

6.1 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. Thus, 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 can limit our efforts for discovering key sets with strongly related keys to the key sets 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 many users with manageable effort.

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

- When retrieving the Post for each query key q_i from $p(q_i)$, have the query initiator also send the actual query, i.e., *all* query keys, to $p(q_i)$.
- Have 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 [5, 19, 22] and truncating the logs.

For example, a request for all Posts 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 (e.g., by frequent itemset mining), 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.

6.2 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 we discussed in Section 4, it is exactly the uncorrelated or negatively correlated keys that mandate multi-key statistics, whereas the keys with high positive correlation also in the data do not really need these additional Posts.

Assume the 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, the directory peer $p(b)$ locally computes the union of these hash sketches by bit-wise OR and transfers only *one combined* hash sketch representing df_b.
2. Then, $p(a)$ can compute the hash sketch 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 using equation 6.
4. From this, $p(a)$ can finally compute the conditional probabilities $P(A|B)$ and $P(B|A)$ using equation 5.

The conditional probabilities (cf. Section 4) provide us with a means to quantify the relatedness between two keys in the data and assess the utility of explicit key-set statistics. As discussed in Section 4, the query routing process benefits mostly from explicit knowledge of statistical summaries for uncorrelated or negatively correlated keys. 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.

6.3 Creating and Disseminating Summaries

When a key set has been identified to be a valuable candidate to produce multi-key statistics, 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 Post 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 Post, a data peer retrieves information about such valuable multi-key sets containing a. Remember from Section 6.1 that all applicable key sets containing a have been identified at $p(a)$ and, thus, are available there. The data peer can subsequently start to produce multi-key statistics for those key sets and publish it during the next round of updating the Post. 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 postings 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 Post for the key set ab should be stored at one of the directory peers 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, for any data peer publishing a summary for ab would also post a summary for a alone and $p(a)$ would be contacted anyway. The approach also simplifies the retrieval of the summaries for query routing purposes, as we will see in the next subsection.

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

6.4 Enhanced Query Routing

Now that the summaries for ab are contained in the distributed directory 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 summaries to $p(a)$ and $p(b)$. Recall that these requests carry the full query 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 requestor. Note that, if no summaries for any multi-key subset regarding have been published, every directory peer ships the single keyword summaries, so that baseline query routing can proceed as usual. So falling back to the single-key case in those situations when no multi-key posts exist does not need additional messages either.

For queries with more than two keys, there is an additional complication: it could happen that there is no Post 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 Posts. 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 Post 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 Posts. 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 Posts 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 [31], but our setting has the additional complexity of very-high-dimensional key space (e.g., keywords over text documents).

6.5 Adding Overlap-Awareness

The recently proposed methods for making query routing overlap-aware [33, 6] can be easily applied to mk-STAT, if the statistical summaries published by the data peers contain appropriate data set synopses, e.g., hash sketches or min-wise independent permutations [11]. In our approach, the multi-key Posts include per-peer hash sketches for the interesting key sets anyway. These are kept at the corresponding directory peers, and these directory peers also precompute the union of all peers' hash sketches for the given key set.

Query routing decisions based on mk-STAT usually only need to fetch the PeerList for a key set, but not the hash sketches for the individual peers in the list (recall that this is one of the advantages that mk-STAT has over sk-STAT, for sk-STAT does indeed need the per-peer hash sketches). However, it we also want to estimate the overlap in the result sets that we would obtain from different peers in the same PeerList, or equivalently estimate the novelty of results obtained by adding a particular peer to the targets for forwarding the query, then it is necessary to fetch the individual peers' hash sketches, too. 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 substantially. Other than this, it is straightforward to incorporate the overlap-aware techniques from [33, 6] in order to combine it with our new methods for correlation awareness.

7. SCALABILITY

Among the two suggested methods, sk-STAT is more lightweight when maintaining the P2P directory, but pays higher cost at query run-time, whereas mk-STAT has little overhead at query run-time but appears to be more costly at posting time. In this section we briefly discuss 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 exactly 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 post 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 posting until it needs to contact the same directory peer for a query-routing lookup anyway (for the same or a different key). So all posting messages can effectively be piggybacked on messages that are sent on behalf of queries anyway. The messages become slighly bigger, but the added information is compact so that the message size stays 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 posting time.

The only situation where the posting cost may become critical is when a data peer wants to post statistics at a high rate, but has a much lower query-generation rate. In this situation, piggybacking postings 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 postings that mk-STAT newly introduces in addition to the data-update postings capture query key correlations and are thus triggered by locally issued queries; therefore, a peer with few queries will not issue many postings of this kind either.

8. 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[3]. As an IR text retrieval scenario, we use real-world web data from a TREC[38] benchmark.

For both datasets, we compare the query result quality obtained by using a state-of-the-art CORI-style query routing approach [13] based on single-key frequencies[4] versus both of our approaches.

CORI is a probabilistic IR model that ranks peers for a given key based on the key frequency at a peer (i.e., the number of files that contain this key) and the key's inverse peer frequency (i.e., the number of peers that have at least one file with this key). 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.

8.1 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).

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

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

Each user was viewed as a separate peer, so our P2P network contained about 4,000 peers, with the original assignment of files to peers (including many duplicates 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 1 shows the results averaged over all remaining *distinct* queries for standard CORI-based query routing vs. sk-STAT and mk-STAT. Figure 2 considers only those queries that use at least one key pair or triplet from the chart-based "training queries". Figure 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, all results for a query are distributed across 135 peers on average (i.e., averaged over all queries). These include many duplicates, however. 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 makes 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.

8.2 Web Data

As Web data, we consider the GOV document collection [38] with roughly 1.2 million documents compiled by a Web crawl of the .gov internet domain and used in TREC benchmarks. To evaluate the distributed behavior, we created 750

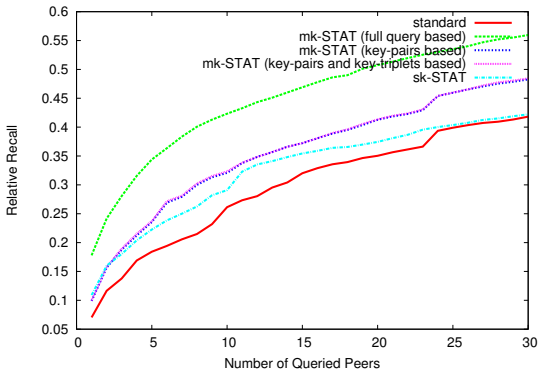

Figure 1: All queries

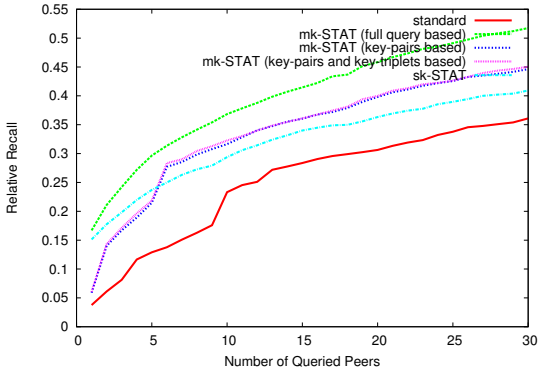

Figure 2: Queries with applicable triplet or pair

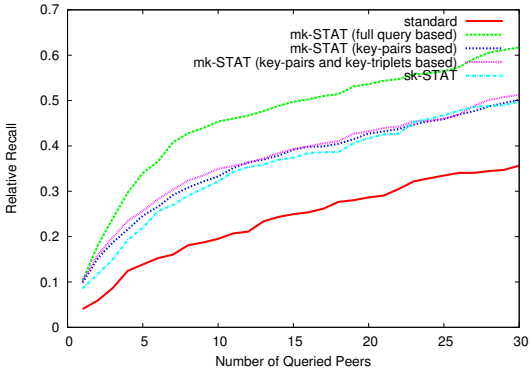

Figure 3: Queries with applicable triplet

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

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 3: Extended queries

remaining queries, expanded to increase the document recall, are shown in Table 3.

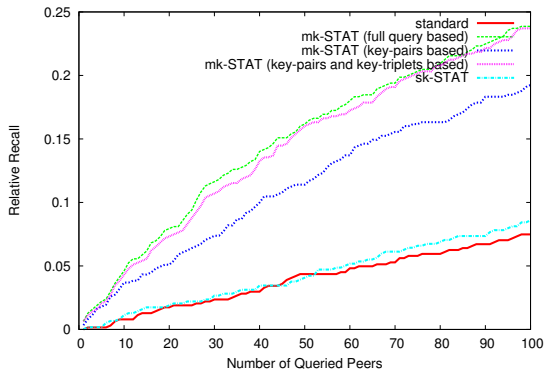

Figure 4: Relative recall for extended queries

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 (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 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 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.

9. CONCLUSIONS

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.

10. REFERENCES

[1] K. Aberer. P-Grid: A self-organizing access structure for P2P information systems. In *CoopIS*, 2001.

[2] K. Aberer and P. Cudré-Mauroux. Semantic overlay networks. In *VLDB*, 2005.

[3] K. Aberer, F. Klemm, T. Luu, I. Podnar, and M. Rajman. Building a peer-to-peer full-text Web search engine with highly discriminative keys. Technical report, EPFL (LSIR), 2005.

[4] D. Agrawal, A. E. Abbadi, and S. Suri. Attribute-based access to distributed data over p2p networks. In *DNIS*, 2005.

[5] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, 1993.

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

[7] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Minerva: collaborative p2p search. In *VLDB*, 2005.

[8] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. P2p content search: Give the web back to the people. In *IPTPS*, 2006.

[9] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7), 1970.

[10] A. Broder. On the resemblance and containment of documents. In *SEQS*, 1997.

[11] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3), 2000.

[12] J. Callan. Distributed information retrieval. *Advances in information retrieval, Kluwer Academic Publishers.*, 2000.

[13] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR*, 1995.

[14] E. Cohen, A. Fiat, and H. Kaplan. Associative search in peer to peer networks: Harnessing latent semantics. In *INFOCOM*, 2003.

[15] G. Cormode and M. N. Garofalakis. Sketching streams through the net: Distributed approximate query tracking. In *VLDB*, 2005.

[16] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems. In *AP2PC*, 2004.

[17] P. Druschel and A. Rowstron. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Middleware*, 2001.

[18] M. Durand and P. Flajolet. Loglog counting of large cardinalities. In *ESA*, 2003.

[19] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D. Ullman. Computing iceberg queries efficiently. In *VLDB*, 1998.

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

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

[22] J. Han and M. Kamber. *Data Mining Concepts and Techniques*. Morgan Kaufman, 2001.

[23] T. Hernandez and S. Kambhampati. Improving text collection selection with coverage and overlap statistics. poster at WWW 2005.

[24] R. Huebsch, B. N. Chun, J. M. Hellerstein, B. T. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. R. Yumerefendi. The architecture of pier: an internet-scale query processor. In *CIDR*, 2005.

[25] M. Jelasity, W. Kowalczyk, and M. van Steen. An approach to massively distributed aggregate computing on peer-to-peer networks. In *PDP*, 2004.

[26] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252, 2005.

[27] P. Kalnis, W. S. Ng, B. C. Ooi, and K.-L. Tan. Answering similarity queries in peer-to-peer networks. *Inf. Syst.*, 31(1), 2006.

[28] R. M. Karp, C. Schindelhauer, S. Shenker, and B. Voecking. Randomized rumor spreading. In *FOCS*, 2000.

[29] W. Litwin and M.-A. Neimat. k-rp*s: A scalable distributed data structure for high-performance multi-attribute access. In *PDIS*, 1996.

[30] J. Lu and J. P. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *CIKM*, 2003.

[31] V. Markl, N. Megiddo, M. Kutsch, T. M. Tran, P. J. Haas, and U. Srivastava. Consistently estimating the selectivity of conjuncts of predicates. In *VLDB*, 2005.

[32] W. Meng, C. T. Yu, and K.-L. Liu. Building efficient and effective metasearch engines. *ACM Computing Surveys*, 34(1):48–89, 2002.

[33] S. Michel, M. Bender, P. Triantafillou, and G. Weikum. IQN Routing: Integrating quality and novelty for web search. In *EDBT*, 2006.

[34] S. Ratnasamy, et al. A scalable content-addressable network. In *SIGCOMM*, 2001.

[35] L. Si, R. Jin, J. P. Callan, and P. Ogilvie. A language modeling framework for resource selection and results merging. In *CIKM*, 2002.

[36] K. Sripanidkulchai, B. M. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *INFOCOM*, 2003.

[37] I. Stoica, et al. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *SIGCOMM*, 2001.

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

[39] P. Triantafillou, C. Xiruhaki, M. Koubarakis, and N. Ntarmos. Towards high performance peer-to-peer content and resource sharing systems. In *CIDR*, 2003.

[40] S. Voulgaris, A.-M. Kermarrec, L. Massoulié, and M. van Steen. Exploiting semantic proximity in peer-to-peer content searching. In *FTDCS*, 2004.

[41] L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer, Feb. 2004.

[42] B. Yang, P. Vinograd, and H. Garcia-Molina. Evaluating guess and non-forwarding peer-to-peer search. In *ICDCS*, 2004.

[43] J. Zhang and T. Suel. Efficient query evaluation on large textual collections in a peer-to-peer environment. In *Peer-to-Peer Computing*, 2005.