IQN Routing: Integrating Quality and Novelty in P2P Querying and Ranking

Sebastian Michel[1], Matthias Bender[1], Peter Triantafillou[2], Gerhard Weikum[1]

[1]Max-Planck-Institut für Informatik
{smichel, mbender, weikum}@mpi-inf.mpg.de

[2]University of Patras
peter@ceid.upatras.gr

Abstract. We consider a collaboration of peers autonomously crawling the Web. A pivotal issue when designing a peer-to-peer (P2P) Web search engine in this environment is *query routing*: selecting a small subset of (a potentially very large number of relevant) peers to contact to satisfy a keyword query. Existing approaches for query routing work well on disjoint data sets. However, naturally, the peers' data collections often highly overlap, as popular documents are highly crawled. Techniques for estimating the cardinality of the overlap between sets, designed for and incorporated into information retrieval engines are very much lacking. In this paper we present a comprehensive evaluation of appropriate overlap estimators, showing how they can be incorporated into an efficient, iterative approach to query routing, coined *Integrated Quality Novelty (IQN)*. We propose to further enhance our approach using histograms, combining overlap estimation with the available score/ranking information. Finally, we conduct a performance evaluation in MINERVA, our prototype P2P Web search engine.

1 Introduction

1.1 Motivation

In recent years, the Peer-to-Peer (P2P) paradigm has been receiving increasing attention. While becoming popular in the context of file-sharing applications such as Gnutella or BitTorrent or IP telephony like 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 highly distributed, scalable, self-organizing way with resilience to failures and churn. Given the potentially very large set of peers storing relevant data, one of the key technical challenges of such a system is *query routing*, which is the process of efficiently selecting the most promising peers for a particular information need. For example, in a file-sharing or publish-subscribe setting, a peer may issue a structured query about MP3 files with operas by the Greek composer Mikis Theodorakis referring to attributes like file type, music genre, and composer; and the P2P network should quickly and efficiently identify other peers that offer many such files and can deliver them with short latency. Another example would be a Web search engine based on a P2P overlay, where a peer initiates a multi-keyword search, and the query routing mechanism should forward this request to the best peers that offer highly scoring documents for IR-style top-k results. In this paper, we will primarily address the ranked retrieval setting for P2P Web search, but our solutions are also applicable to and beneficial for DB-style structured queries without ranking.

Several techniques borrowed from the literature on distributed IR [20, 12, 24, 28] could be employed for query routing, based on statistics about term frequencies (tf) and inverse document frequencies (idf) that reflect the relevance of documents to a query term and thus can be aggregated into measures reflect the wealth and quality of a peer's corpus. However, these strategies typically ignore the fact that popular documents are replicated at a significant number of peers. 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 [22]. 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.

1.2 Contribution

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. We envision 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), 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 for query forwarding).

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.

In this paper we consider three kinds of synopses that each peer builds up and posts on a per-term basis, representing the global ids of documents (e.g., URLs or unique names of MP3 files) that a peer holds in its collection: Bloom filters [7], hash sketches [18], and min-wise permutations [9, 10]. These techniques have been invented for approximate, low-error representation of sets or multisets. In this paper we show how they can be adapted to a P2P setting and exploited for our highly effective IQN query routing. We assume that each peer locally

maintains inverted index lists with entries of the form $< term, docId, score >$, and posts for each term (or attribute value in a structured data setting) a set synopsis that captures the docIds that the peer has 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 paper are as follows:

- We have conducted 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.
- We have developed 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.
- We have carried 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.

Our previous paper [5] has addressed the same kind of problem, but used only Bloom filters and a fairly simple algorithm for aggregating synopses and making the actual routing decisions. The current paper identifies min-wise permutations as the most suitable type of synopsis, and develops the much more sophisticated IQN routing method.

The rest of the paper is organized as follows. Section 2 discusses related work and gives general background on P2P IR. Section 3 introduces the different types of synopses and presents our experimental comparison of the basic techniques. Section 4 introduces our P2P testbed, coined MINERVA [5, 6]. Section 5 develops the IQN routing method in detail. Section 6 discusses special techniques for handling multi-dimensional queries. Section 7 describes extensions to exploit histograms on score distributions. Section 8 presents our experimental evaluation of the IQN routing method versus the best previously published algorithms, namely, CORI [13] and our prior method from [5].

2 Related Work

Many approaches have been proposed for collection selection in distributed IR, most notably, CORI [13], the decision-theoretic framework by [28], the GlOSS method presented in [20], and methods based on statistical language models [32] In principle, these methods could be applied to a P2P setting, but they fall short of various critical aspects: they incur major overhead in their statistical models, they do not scale up to large numbers of peers with high dynamics, and they disregard the crucial issue of collection overlap.

The ample work on P2P networks, such as Chord Chord [33], CAN [29], Pastry [31], or P-Grid [1], has developed scalable routing protocols for single-dimensional key-based requests only. There is only little work on how to map multidimensional data onto distributed hash tables (DHTs) and other overlay networks [3, 35], and these approaches do not work for the very-high-dimensional data spaces formed by text keywords and they do not provide any support for ranked retrieval either. P2P Web search has emerged as a new topic only recently. A variety of ongoing research projects are pursuing this direction [2, 37,

4, 14, 15, 30, 36, 22], including our MINERVA project [5]. Query routing has been identified as a key issue, but none of the projects has a fully convincing solution so far.

Fundamentals for statistical synopses of sets and multisets have a rich literature, including work on Bloom filters [7, 17], hash sketches [18], and min-wise permutations [9, 10]. We will present the relevant background for these techniques in Section 3.

There is relatively little work on the specific issue of overlap and novelty estimation. [38] addresses redundancy detection in a centralized information filtering system; it is unclear how this approach could be made scalable in a highly distributed setting. [27, 21] present a technique 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.

Our own prior work [5] addressed overlap estimation for P2P collections, but was limited to Bloom filters and used only a simple decision model for query routing. The current paper shows how to utilize also more sophisticated and flexible kinds of synopses like min-wise permutations, analyzes their advantages, and develops the novel IQN routing method. IQN outperforms the method of [5] by a large margin in terms of the ratio of query result recall to execution cost.

3 Collection Synopses for Information Retrieval

3.1 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 and can be used for our problem setting [8].

$Containment(S_A, S_B) = \frac{|S_A \cap S_B|}{|S_B|}$ 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 easily calculate the other [11].

However, our experiments show that none of these notions fully captures the peculiarities of our system model. Specifically, we expect during P2P query routing to encounter peers with 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 $|S_A \cap S_C|$, resulting in 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 a notion of *novelty* of a set S_B with regard to S_A as $Novelty(S_B|S_A) = |S_B - (S_A \cap S_B)|$, i.e., the cardinality of the set difference.

3.2 Synopses

We want to be able to compute the novelty between data collections of different peers without explicitly transferring the peers' (potentially very large) index lists. Thus, methods for novelty estimators based on compact set synopses are needed. In the following, we briefly overview the three most relevant statistical synopses methods from the literature. Here we focus on estimating resemblance. In Section 5.2 we will show how to estimate our proposed novelty measure from the resemblance estimators; the details of this derivation are dependent on the specific synopsis technique.

Bloom Filters A Bloom filter (BF) [7] is a data structure that compactly represents a set as a bit vector in order to support membership queries. For a particular set, a Bloom filter is a bit map of length m initially set to zero and is created by applying k hash functions on each set member, each yielding a bit location in the vector. Exactly (and only) these yielding positions of the Bloom filter will be set to 1. To check if a given element is in the set, the element is hashed using the same hash function and the corresponding bits of the Bloom filter are examined. If there is at least one of these bits that is *not* set to 1, the element is not in the set; otherwise it is in the set with high probability. There is a chance that all bit positions that were examined had been set by other elements, thus creating a *false positive*. The probability of such a false positive can be calculated by $pfp \approx (1 - e^{-kn/m})^k$ where n is the number of items in the original set and can be tuned by selecting appropriate values for k and m (see, e.g., [17]). Improvements towards the creation of compressed Bloom filters have been introduced by [26].

Bloom filters can easily approximate intersections and unions by bit-wise AND and ORing of two filters. The cardinality of a set with a given Bloom-filter approximation can be estimated from the following combinatorial computation. The probability that a bit is set is $p := 1 - (\frac{m-k}{m})^n$, and the expected number of set bits is $E := mp$. There are various high-accuracy approximations of E (e.g., using the first few terms of a Taylor series expansion) which yield analytic formulas that can be easily solved for n, given the value of E. Finally, the resemblance between two sets is derived from the cardinalities of their union and intersection.

Min-Wise Independent Permutations (MIPs) Min-Wise Independent Permutations, or MIPs for short, have been introduced in [9, 10]. 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. The technique is illustrated with an example in Figure 1. Its fundamental rationale is that each element has the same probability of becoming the minimum element under a random permutation. By using sufficiently many different permutations, we can approximate the set cardinality.

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 [11]. 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 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 these are no longer statistically sound unbiased estimators.

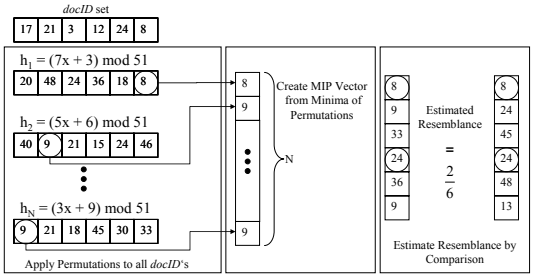

Fig. 1. Example of Min-Wise Permutations

Hash Sketches Hash sketches were first proposed by Flajolet and Martin in [18], to probabilistically estimate the cardinality of a multiset S. [19] proposes a hash-based synopsis data structure and algorithms to support low-error and high-confidence estimates for general set expressions. Hash sketches rely on the existence of a pseudo-uniform hash function $h() : S \rightarrow [0, 1, \ldots, 2^L)$. Durand and Flajolet presented a similar algorithm in [16] (*super-LogLog counting*) which reduced the space complexity and relaxed the required statistical properties of the hash function.

Hash sketches work as follows. Let $\rho(y) : [0, 2^L) \rightarrow [0, L)$ be the position of the least significant (leftmost) 1-bit in the binary representation of y; that is, $\rho(y) = min_{k \geq 0}\, bit(y, k) \neq 0,\ y > 0$, and $\rho(0) = L$. $bit(y, k)$ denotes the kth bit in the binary representation of y (bit position 0 corresponds to the least significant bit). In order to estimate the number n of distinct elements in a multiset S we apply $\rho(h(d))$ to all $d \in S$ and record the results in a bitmap vector $B[0 \ldots L-1]$. Since $h()$ distributes values uniformly over $[0, 2^L)$, it follows that $P(\rho(h(d)) = k) = 2^{-k-1}$.

Thus, when counting elements in an n-item multiset, $B[0]$ will be set to 1 approximately $\frac{n}{2}$ times, $B[1]$ approximately $\frac{n}{4}$ times, etc. Then, the quantity $R(S) = max_{d \in S}\rho(d)$ provides an estimation of the value of $\log n$. [18, 16] presents analyses and techniques to bound from above the error introduced, relying basically on using multiple bit vectors and averaging over their corresponding R positions.

3.3 Experimental Characterization

We evaluated the three different kinds of 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]

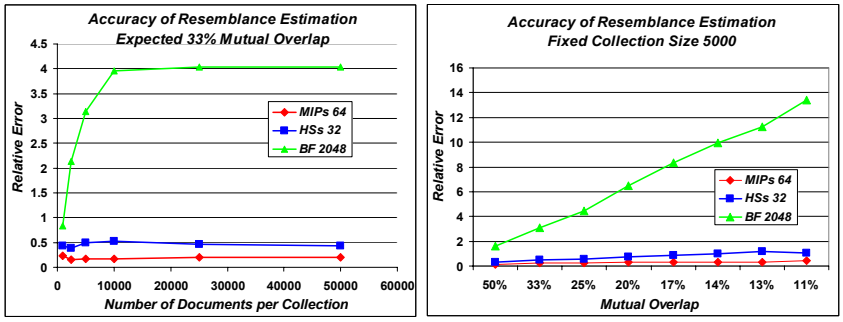

Fig. 2. Relative Error of Resemblance Estimation

Figure 2 shows, on the left side, 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 16,384 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 2 on the right side, are similar to the observations above: Bloom Filters suffer again from their overload; MIPs and hash sketches offer accurate estimates with a low variance for all degrees of overlap.

3.4 Discussion

A qualitative comparison for selecting the most appropriate synopsis representation of the peer collections in our framework should be based on the following criteria: 1) low estimation error, 2) small space requirements for low storage and communication costs, 3) 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, and 4) 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

[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.

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 size filters, i.e., the size of the Bloom filters has to be a global system parameter. This leads either 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 accurate (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. Our experiments also indicate that hash sketches tend to become more accurate in the presence of increasing collection sizes, but produce some unreliable estimates for very small collections.

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 AND). This somewhat limits their flexibility in some application classes with conjunctive multi-dimensional queries (cf. Section 6). Moreover, they share with Bloom filters the disadvantage that all hash sketches need to have the same bit lengths in order to be comparable. In a large-scale loosely coupled P2P environment, this is a penalty.

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 8 show that the accuracy is typically still good enough. Note that this flexibility in handling synopses of heterogeneous lengths is very important in a P2P setting with high autonomy and dynamics of peers. For example, one peer may choose to encode the DocIDs for a term with a very long index list into a long MIPs synopsis, while another peer is generally much more space constrained (e.g., because of other, even longer index lists) and uses a shorter MIPs synopsis for the same term. Due to the very good performance of MIPs in our experiments and the ability to handle different lengths, we use MIPs as the synopses of choice in the following sections.

4 MINERVA Prototype for P2P Web Search

MINERVA is a fully operational distributed search engine that we have implemented and that serves as a testbed for our work. A conceptually global but physically distributed directory, which is layered on top of Chord [33], holds compact, aggregated information about the peers' local indexes, to the extent that the individual peers are willing to disclose. Unlike [23], we use the Chord DHT to partition the term space, such that every peer is responsible for the statistics and metadata of a randomized subset of terms within the directory.

For failure resilience and availability, the responsibility for a term can be replicated across multiple peers. We do *not* distribute the actual index lists or even documents across the directory.

Directory maintenance, query routing, and query processing work as follows. Every peer publishes statistics, denoted as *Posts*, about every term in its local index to the directory. The peer onto which the term is hashed maintains a *PeerList* of all postings for this term from all peers across the network. Posts contain contact information about the peer who posted the summary together with statistics to calculate IR-style relevance measures for a term, e.g., the length of the inverted index list for the term, the maximum or average score among the term's inverted list entries, etc. A peer that initiates a multi-keyword query first retrieves the PeerLists for all query terms from the distributed directory. It combines this information to identify the most promising peers for the current query. For efficiency reasons, the query initiator can decide to not retrieve the complete PeerLists, but only a subset, say the top-k peers from each list based on IR relevance measures, or more appropriately the top-k peers over all lists, calculated by a distributed top-k algorithm like [25].

5 Enhancing Query Execution using Novelty Estimation

5.1 The IQN Query Routing Method

A good query routing method should be 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 in a particularly cost-efficient way for making routing decisions.

The Integrated Quality Novelty (IQN) method that we have developed based on this rationale 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 measured. 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 in Section 6 how to handle multi-keyword or multi-attributed queries.

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 [13, 12]: CORI computes the collection score s_i of the i-th peer with regard to a query $Q = \{t_1, t_2, ..., t_n\}$ as $s_i = \sum_{t \in Q} \frac{s_{i,t}}{|Q|}$ where $s_{i,t} = \alpha + (1 - \alpha) \cdot T_{i,t} \cdot I_{i,t}$.

The computations of $T_{i,t}$ and $I_{i,t}$ use the *number of peers* in the system, denoted np, the *document frequency* (cdf) of term t in collection i, and the *maximum document frequency* (cdf^{max}) for any term t in collection i:

$$T_{i,t} = \frac{cdf_{i,t}}{cdf_{i,t} + 50 + 150 \cdot \frac{|V_i|}{|V^{avg}|}} \qquad I_{i,t} = \frac{\frac{log(np+0.5)}{cf_t}}{log(np+1)}$$

where the *collection frequency* cf_t is the number of peers that contain the term t. The value α is chosen as $\alpha = 0.4$ [13].

CORI considers the size $|V_i|$ of the term space of a peer (i.e., the total number of distinct terms that the peer holds in its local index) and the average term space size $|V^{avg}|$ over all peers that contain term t.

In practice, it is difficult to compute the average term space size over all *peers in the system* (regardless of whether they contain query term t or not). We approximate this value by the average over all *collections found in the PeerLists* (cf. Section 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 those 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.

5.2 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 3.2. 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 can 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: $bf[i] := bf_p[i] \wedge \not bf_r ef[i]$. 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 length). Finally, we estimate the cardinality of the set difference from the number of set bits in bf by the combinatorial formula given in the discussion of Bloom filters in Section 3.2.

5.3 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 follows $MIPs_{A \cup B}[i] = min\{MIPs_A[i], MIPs_B[i]\}\ \forall i : 1 \leq i \leq n$.

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 an always use the smaller number of permutations as a common denominator. This loses accuracy in the result 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 5.2.

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: $BF_{A \cup B}[i] = BF_A[i]\ OR\ BF_B[i]\ \forall i : 1 \leq i \leq n$.

6 Multi-Dimensional Queries

As the synopses posted by the peers are per term or attribute value, 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.

6.1 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 for 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., bit vectors) 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 prior literature does not offer any solutions either. For hash sketches a very crude approach would be use unions also for conjunctive queries; this would at least give a valid synopsis as unions are superset 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 mininum 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 minumum 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 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 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.

6.2 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. For each candidate peer under consideration by IQN, this combined synopsis is used to estimate the novelty with respect to the aggregated reference synopsis of the previously covered result space, using the techniques described before. After selecting the most promising peer, its combined synopsis is aggregated with the reference synopsis of the current IQN iteration.

6.3 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 simply 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 preserves the relative ranking of peers, so it does lead 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 believe that this aggregation technique can be further extended, e.g., for exploiting term correlation measures mined from the P2P system. Our MINERVA testbed has implemented both of the two presented aggregation techniques, for all three kinds of synopses.

7 Extensions

7.1 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. We observe that we are more interested in the higher-scoring portions of an index list and the mutual overlap that different peers have in these portions. We employ histograms to put documents of each index list into cells, where each cell represents a certain score range of an index list.

Synopses as introduced before (e.g., MIPs) are 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).

7.2 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 together. This is primarily 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 is a critical issue.

In this framework, a peer with a total budget B has the freedom to choose specific a 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.

8 Experiments

8.1 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 have taken the complete GOV document collection, a crawl of the .gov Internet domain used in the TREC 2003 Web Track benchmark (http://trec.nist.gov). 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 partitioned 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 can 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 [34]. These were relatively short multi-keyword queries, typical examples being "forest fire" or "pest safety control".

All experiments were conducted on the MINERVA testbed described in Section 4, with peers running on a PC cluster. We compared query routing based on the CORI method which is merely quality-driven (see Section 5.1) against the quality- and novelty-conscious IQN method. Recall that CORI is among the very best database selection methods for distributed IR. We measured 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.

8.2 Experimental Results

Figure 3 shows the recall results (micro-averaged over all our benchmark queries), using the $\binom{f}{s}$ technique in the chart on the left side and the sliding-window technique on the right side. 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. We discarded hash sketches from these experiments because of the insights from Section 3. The shorter synopsis length was 1024 bits or equivalently 32 min-wise permutations; the longer one was 2048 bits or 64 min-wise permutations.

Figure 3 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. Another way of reading the charts is by fixing a certain recall target that the query should reach, say 50 percent, and determining how many peers each of the methods needed in order to reach this target. In the more challenging sliding-window scenario, the IQN methods needed about 5 peers for this target, whereas CORI required more than 20 peers (not even visible in the chart).

In the comparison of the two different synopses techniques, our expectation, from the stand-alone experiments in Section 3, 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. Here it is important to notice that response times are a highly superlinear function of load when peers or network components such as routers are heavily utilized.

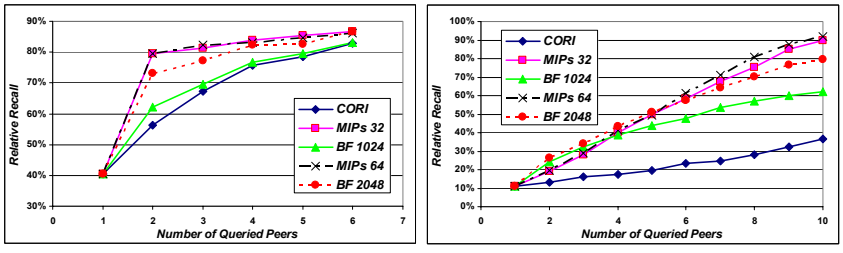

Fig. 3. Recall as a function of the number of peers involved per query

9 Conclusion and Future Work

This paper has developed 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.

Our future work will aim at further refinements and improvements of distributed statistics management in a highly dynamic P2P environment. Two issues that we will particularly look into are 1) strategies for adaptively choosing the synopses types and lengths depending on the P2P usage scenario and with dynamic and automatic adaptation to evolving data and system characteristics, and 2) incorporating statistics about correlations between different index lists on the same peer or correlations across peers into the synopses management.

References

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

[2] K. Aberer and J. Wu. Towards a common framework for peer-to-peer web retrieval. In *From Integrated Publication and Information Systems to Virtual Information and Knowledge Environments*, pages 138–151, 2005.

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

[4] W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden. Dl meets p2p - distributed document retrieval based on classification and content. In *ECDL*, 2005.

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

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

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

[8] Broder. On the resemblance and containment of documents. In *SEQS: Sequences '91*, 1998.

[9] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations (extended abstract). In *STOC*, pages 327–336, 1998.

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

[11] J. W. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed content delivery across adaptive overlay networks. *IEEE/ACM Trans. Netw.*, 12(5):767–780, 2004.

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

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

[14] P. Cao and Z. Wang. Efficient top-k query calculation in distributed networks. In *PODC*, pages 206–215, 2004.

[15] A. Crainiceanu, P. Linga, A. Machanavajjhala, J. Gehrke, and J. Shanmugasundaram. An indexing framework for peer-to-peer systems. In *SIGMOD*, 2004.

[16] M. Durand and P. Flajolet. Loglog counting of large cardinalities. In G. Di Battista and U. Zwick, editors, *ESA03*, volume 2832 of *LNCS*, pages 605–617, 2003.

[17] L. Fan, P. Cao, J. M. Almeida, and A. Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Trans. Netw.*, 8(3), 2000.

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

[19] S. Ganguly, M. Garofalakis, and R. Rastogi. Processing set expressions over continuous update streams. In *SIGMOD2003*, pages 265–276. ACM Press, 2003.

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

[21] T. Hernandez and S. Kambhampati. Improving text collection selection with coverage and overlap statistics. pc-recommended poster. WWW 2005.

[22] R. Huebsch, J. M. Hellerstein, N. L. Boon, T. Loo, S. Shenker, and I. Stoica. Querying the internet with pier, Sept. 2003.

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

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

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

[26] M. Mitzenmacher. Compressed bloom filters. *IEEE/ACM Trans. Netw.*, 10(5):604–612, 2002.

[27] Z. Nie, S. Kambhampati, and T. Hernandez. Bibfinder/statminer: Effectively mining and using coverage and overlap statistics in data integration. In *VLDB*, pages 1097–1100, 2003.

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

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

[30] P. Reynolds and A. Vahdat. Efficient peer-to-peer keyword searching. In *Proceedings of International Middleware Conference*, pages 21–40, June 2003.

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

[32] L. Si, R. Jin, J. Callan, and P. Ogilvie. A language modeling framework for resource selection and results merging. In *Proceedings of CIKM02*, 2002.

[33] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM2001*, pages 149–160. ACM Press, 2001.

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

[35] P. Triantafillou and T. Pitoura. Towards a unifying framework for complex query processing over structured peer-to-peer data networks. In *DBISP2P*, 2003.

[36] Y. Wang and D. J. DeWitt. Computing pagerank in a distributed internet search engine system. In *VLDB*, pages 420–431, 2004.

[37] J. Zhang and T. Suel. Efficient query evaluation on large textual collections in a peer-to-peer environment. In *P2P2005*.

[38] Y. Zhang, J. Callan, and T. Minka. Novelty and redundancy detection in adaptive filtering. In *SIGIR*, 2002.