

Global Document Frequency Estimation in Peer-to-Peer Web Search

Matthias Bender ^{*}, Sebastian Michel ^{*}, Peter Triantafillou [◇], Gerhard Weikum ^{*}

^{*} Max-Planck-Institut für Informatik

66123 Saarbrücken, Germany

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

[◇] RACTI and University of Patras

26500 Rio, Greece

peter@ceid.upatras.gr

ABSTRACT

Information retrieval (IR) in peer-to-peer (P2P) networks, where the corpus is spread across many loosely coupled peers, has recently gained importance. In contrast to IR systems on a centralized server or server farm, P2P IR faces the additional challenge of either being oblivious to global corpus statistics or having to compute the global measures from local statistics at the individual peers in an efficient, distributed manner. One specific measure of interest is the global document frequency for different terms, which would be very beneficial as term-specific weights in the scoring and ranking of merged search results that have been obtained from different peers.

This paper presents an efficient solution for the problem of estimating global document frequencies in a large-scale P2P network with very high dynamics where peers can join and leave the network on short notice. In particular, the developed method takes into account the fact that the local document collections of autonomous peers may arbitrarily overlap, so that global counting needs to be duplicate-insensitive. The method is based on hash sketches as a technique for compact data synopses. Experimental studies demonstrate the estimator's accuracy, scalability, and ability to cope with high dynamics. Moreover, the benefit for ranking P2P search results is shown by experiments with real-world Web data and queries.

1. INTRODUCTION

In recent years, distributed information retrieval systems based on Peer-to-Peer (P2P) architectures are increasingly receiving attention [27, 29, 22, 1, 21, 31, 4, 13, 39]. The P2P approach offers the ability to handle huge amounts of data in a highly distributed, self-organizing way and, thus, offer enormous potential for search engines powerful in terms of scalability, efficiency, and resilience to failures and dynamics. Additionally, such a search engine can potentially benefit from the intellectual input (e.g., bookmarks, query logs, etc.) of a large user community. Finally, but perhaps even more importantly, a P2P web search engine can also facilitate pluralism in informing users about internet content, which is crucial in order to preclude the formation of information-resource monopolies and the biased visibility of

content from economically powerful sources.

1.1 Problem

Given the large-scale data distribution, one of the key technical challenges is *result merging*, i.e., the process of effectively combining local query results from different sources. While document scoring and ranking is a challenging problem already in centralized systems, additional difficulty in a distributed environment stems from the fact that most of the popular document scoring models, such as $tf*idf$ or [35], use collection-specific statistical information for this purpose. Most prominently, both use *document frequencies* (df), i.e. the number of documents in the collection that contain a query term¹. The local usage of collection-specific df values in these scoring models result in document scores that are incompatible across collections and, thus, make result merging difficult. On the other hand, if *global df* values could be applied, the document scoring and ranking would be ideal in the sense that it would be identical to the document ranking that would be produced by a hypothetical combined collection.

Early research on distributed information retrieval systems typically assumed disjointly partitioned collections. In such a setting, the *global df* value is simply the sum over all local df values. Instead, we envision autonomous peers that independently gather thematically focused collections through web crawls or similar techniques. In such a setting, studies show a skewed distribution of documents across the collections, with popular documents contained in a large fraction of collections. Thus, summing up the df values across collections would inevitably lead to biased df values (and, thus, document scores) [24], as popular documents are repeatedly accounted for. Additionally, thematically focused collections show a high variance of df values for the same term (whereas randomly partitioned collections show a rather uniform distribution of df values for the same term). This further increases the necessity of a score normalization across peers.

1.2 Contribution

We present a robust and scalable approach towards estimating *global df* values using hash sketches [16]. We study the general accuracy of hash sketches when used as synopses to estimate document frequencies and we develop an efficient strategy to combine these hash sketch synopses across collections in a way that does not incur any additional error from combining them. We show the superiority of our *global df* estimation technique compared to other techniques

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

and present experimental evidence of the effectiveness improvements in result merging stemming from this improved knowledge. The experiments are conducted on real-world web data using our fully operational P2P Web search engine prototype.

The rest of the paper is organized as follows. Section 2 discusses related work and gives general background on P2P IR. Section 3 introduces hash sketches as our technique for multiset cardinality estimation and discusses their general application in our distributed environment. Section 4 introduces the design fundamentals that serve as a basis for our approach and discusses the extensions necessary to support an overlap-aware global df estimation in the presence of peers entering and leaving the system without prior notice. Section 5 presents an experimental evaluation of the general accuracy of hash sketches as cardinality estimators and of the accuracy of our approach from different angles. Finally, Section 6 concludes this work and points at future research directions.

2. RELATED WORK

2.1 Estimating Set Cardinalities

Estimating overlap of sets has been receiving increasing attention for modern emerging applications, such as data streams, internet content delivery, etc. [6] describes a permutation-based technique for efficiently estimating set similarities for informed content delivery. [18] proposes a hash-based synopsis data structure and algorithms to support low-error and high-confident estimates for general set expressions. Bloom [5] describes a data structure for succinctly representing a set in order to support membership queries; [10] present an extension for dealing with multisets, but still focuses on membership queries rather than cardinality estimation. [22] proposes a gossip-based protocol for computing aggregate values in a fully decentralized fashion. [28] addresses communication topology issues for distributed aggregation and identifying frequent items in a network. [11] develops a sketch-based framework for distributed estimation of query result cardinalities, but does not consider duplicates. None of [22, 28, 11] addresses the elimination of overlap.

Recently and independently, [12] proposed an approach similar to ours which aims at duplicate-insensitive counting in sensor networks. However, they have not published any results regarding the resilience to churn of their method, nor has it been applied to an IR scenario.

2.2 Peer-to-Peer Architectures

Recent research on P2P systems, such as Chord [38], CAN [34], Pastry [36], or P-Grid [2] is typically based on various forms of distributed hash tables (DHTs) and supports mappings from keys, e.g., titles or authors, to locations in a decentralized manner such that routing scales well with n , the number of peers in the system. Typically, an exact-match key lookup can be routed to the proper peer(s) in at most $O(\log n)$ hops, and no peer needs to maintain more than $O(\log n)$ routing information. These architectures can also cope well with failures and the high dynamics of a P2P system as peers join or leave the system at a high rate and in an unpredictable manner.

2.3 Distributed IR and Web Search

Many approaches have been proposed for distributed IR, most notably, CORI [9], the decision-theoretic framework [32], the GIOSS method [19], and methods based on statistical language models [37]. Recently, there has been research towards overlap aware resource selection methods [20, 3, 30] that consider the mutual overlap between peers during the

selection process but do not consider global df estimation.

Galanx [40] is a P2P search engine implemented using the Apache HTTP server and BerkeleyDB. The Web site servers are the peers of this architecture; pages are stored only where they originate from, thus forming an overlap free network. PlanetP [13] is a publish-subscribe service for P2P communities, supporting content ranking search. The global index is replicated using a gossiping algorithm. Odissea [39] assumes a two-layered search engine architecture with a global index structure distributed over the nodes in the system. It actually advocates using a limited number of nodes, in the spirit of a server farm. None of this prior work consider the problem of estimating the global df value, for peers with overlapping local contents.

2.4 Result Merging

For cooperative environments Kirsch’s algorithm [23] proposes to collect local statistics from the selected databases to normalize document scores. [26, 29] uses a centralized database of collection samples, which is incompatible with our architectural vision and seems infeasible in the presence of high network dynamics. [7] gives an overview of algorithms for distributed IR style result merging and database content discovery. None of the presented techniques incorporates overlap detection between the peers into the merging process.

Result merging techniques for topically organized collections were studied in [24]. Experiments showed that global idf scores is the most desirable method, but they considered neither real-world Web pages nor overlap between collections. [33] incorporates an estimated number of global occurrences of the same document into the result merging process, but does not estimate the global number of documents that contain a specific term.

3. MULTISSET CARDINALITY ESTIMATION USING HASH SKETCHES

Estimating the global document frequency for a given term would be straightforward if peers had pair-wise disjoint local collections. The global collection is the union of all local collections, and the disjointness would allow us to simply sum up all local document frequencies for the same term. We will discuss the resulting communication and system aspects in Section 4. However, with non-disjoint local collections, computing their union essentially produces a multiset (bag) with duplicates. If we had the full document ids of all items in the multiset, we could eliminate duplicates by sorting or hashing and subsequently count the distinct items. But this approach is expensive on large multisets with all documents explicitly represented. We would rather prefer an approach where each local collection is represented by a compact synopsis, with a small and controllable approximation error.

This section introduces such a synopsis, namely, hash sketches [16], and shows how to employ them for our goal. When we form the union of several synopses, originating from different peers, we face again the problem of how to discount duplicates in the multiset synopses. We will show in this section how this duplicate-aware multiset-counting problem is elegantly solved by our approach based on hash sketches, and we demonstrate the low approximation error.

3.1 Hash Sketches

Hash sketches were first proposed by Flajolet and Martin in [16] to probabilistically estimate the cardinality of a multiset S . [18] proposes a hash-based synopsis data structure and algorithms to support low-error and high-confident estimates for general set expressions. Hash sketches rely on the existence of a pseudo-uniform hash function $h() : S \rightarrow$

$[0, 1, \dots, 2^L]$. Durand and Flajolet presented a similar algorithm in [15] (*super-LogLog counting*) which reduced the space complexity and relaxed the required statistical properties of the hash function.

Briefly, 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,

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

, and $\rho(0) = L$. $\text{bit}(y, k)$ denotes the k -th 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 least-significant 1-bits in a bitmap vector $B[0 \dots L - 1]$. Since $h(\cdot)$ 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 $\log_2 n$. The authors in [16, 15] present analyses and techniques to bound from above the error introduced. Techniques which probably reduce the statistical estimation error typically rely on employing multiple bitmaps for each hash sketch, instead of only one. The overall estimation then is an averaging over the individual estimations produced using each bitmap.

3.2 Combining Hash Sketches

Hash sketches offer duplicate elimination "for free", or in other words, they allow counting distinct elements in multisets. Estimating the number of distinct elements (e.g., documents) of the *union* of an arbitrary number of multisets (e.g., distributed and autonomous collections) - each represented by a hash sketch synopsis - is easy by design: a simple bit-wise *OR*-operation over all synopses yields a hash sketch for the combined collection that instantly allows us to estimate the number of distinct documents of the combined collection.

More formally, we can derive the following *distributivity theorem*:

THEOREM 1. *Let $\beta(S)$ be the set of bit positions $\rho(h(d))$ for all $d \in S$. Then $\beta(S_1 \cup S_2) = \beta(S_1) \cup \beta(S_2)$.*

The proof follows directly from the definitions of ρ and β . The corresponding bit in the resulting combined hash sketch will be set if and only if at least one of the elements in one of the original sets had set this bit. Particularly notice that, if more than one set holds this element, the element will conceptually be counted only once, effectively removing duplicates.

3.3 Application to Global DF Estimation

The above methods can be employed for the purpose of global *df* estimation as follows. Assume that each peer, given its collection, prepares hash sketches, one for each set of documents that contain a specific term (i.e., its index list for that term). The network-wide combination of all hash sketches for a specific term, thus, yields an estimate for the number of distinct elements in the union of the sets represented by their synopses, i.e., for the number of distinct documents that contain the given term. This is the *global document frequency* for that term.

4. OVERLAP-AWARE DF ESTIMATION

4.1 Design Fundamentals

We have implemented MINERVA², a fully operational P2P Web search engine building on the following design fundamentals [3, 30, 4]. We consider a P2P network in which every peer is autonomous and has a local index that can be built from the peer's own crawls or imported from external sources and tailored to the user's thematic interest profile. The index contains inverted lists with URLs for Web pages that contain terms. A conceptually global but physically distributed directory, which is layered on top of a distributed hash table (DHT), holds only very compact, aggregated meta-information about the peers' local indexes and only to the extent that the individual peers are willing to disclose. As part of the DHT, every peer is responsible for the meta-information of a randomized subset of terms within the global directory. For failure resilience and availability, the entry for a term may be replicated across multiple peers. The DHT offers a *lookup* method to determine the peer responsible for a particular term.

Every peer publishes per-term summaries (*Posts*) of its local index to the directory. The DHT hash function determines the directory peer currently responsible for this term. This peer maintains a *PeerList* of all *Posts* for this term from across the network. *Posts* contain contact information about the peer who posted this summary together with statistics to calculate IR-style measures for a term (e.g., the size of the inverted list for the term, the maximum and average score among the term's inverted list entries, or some other statistical measure). These statistics are used to support the *query routing* process, i.e., determining the most promising peers for a particular query. 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.

The querying process for a multi-term query proceeds as follows: the query initiator retrieves a list of potentially useful peers by issuing a *PeerList request* for each query term to the underlying overlay network. A number of promising peers for the complete query is computed from these *PeerLists*. Subsequently, the query is forwarded to these peers and executed based on their local indexes. Finally, the results from the various peers are combined at the querying peer into a single result list; this step is referred to as *result merging* and would enormously benefit from the knowledge of global *df* values.

Note that this design is DHT agnostic, since it utilizes only a DHT's lookup/routing function and thus enjoys wide applicability.

4.2 Accommodating DF Metadata

Given the system design introduced above with a hash-based assignment of terms to responsible directory peers, it is very natural for these directory peers to maintain additional data that supports the global *df* estimation for the terms they are responsible for. When publishing the term-specific *Posts* about the local collection, we propose that every peer includes a *hash sketch* representing its index list for the respective term in its (term-specific) *Post*, so that each directory peer can compute an estimate for the global *df* values for the terms it is responsible for using the combination method introduced in Section 3.2. Thus, the hash sketch synopses representing the index lists of all peers for a particular term are *all* sent to the same directory peer responsible for this term. This peer can, by means of inexpensive bit-wise operations, calculate an estimate for the *global df*, for the terms it is responsible for, from these synopses. Note the importance of utilizing compact synopses for this goal, such as hash sketches, which introduce small bandwidth and storage requirements.

²<http://www.minerva-project.org>

The query initiator collects the df estimates at query time as piggybacked information when retrieving the PeerLists from the directory peers during the query routing phase. Remember that the df estimate for a particular term is maintained at the same peer that maintains the respective PeerList, so that the peers that hold the gdf estimated for the query terms are the very same peers that are contacted anyway in order to retrieve the respective PeerLists. The query initiator can then attach the current gdf estimates to the query message when sending the query to the selected peers. These remote peers can use the estimates on-the-fly as weights during their index scans to compute their local query results.

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

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

4.3 Cost Analysis

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

Where applicable, we use batching of Posts (for terms that have the same directory peer) to further decrease the number of messages. For all messages, we can additionally apply *gzip* compression to additionally decrease the message payload size. Obviously, the network cost caused by the metadata publication additionally depends on the Time-to-Live interval of the metadata, i.e., the time span after which the metadata has to be refreshed. We report on actual traffic measured in the course of our experimental evaluation in Subsection 5.2.

After the dissemination of the Posts, peers executing a query perform PeerList requests to retrieve a list of peers that have published statistics about the specific query terms. Note that the cost of this PeerLists retrieval does *not* change significantly, as the hash sketches themselves are not transferred back to the PeerList requester. Instead, as the df estimation is conducted at the directory peer, only one additional value representing the current df estimate has to

be included in the answer to a PeerList request. The same holds true for the actual query execution; when sending the query to the selected peers, just one additional df value per query term has to be transferred.

The **storage cost** at the directory peers storing the Posts is also directly dependent on the number of Posts, the size of a Post, and the size of a hash sketch. In a network with n peers storing Posts of m distinct terms, each peer is responsible for an expected number of m/n PeerLists. For example, in a system with 50,000 terms and 10,000 peers, each peer is responsible for the maintenance of an average of 5 PeerLists. This number decreases even further as more and more peers join the system, because they typically do not add a significant number of previously unseen terms. In a worst case scenario (every peer has posted information for all terms), a directory peer would thus be responsible for 50,000 Posts or 28.1 MB (including all hash sketches) for each peer list, which we consider a reasonably small storage effort. Remember from the previous subsection that, alternatively, the directory peer does not have to store all hash sketches sent together with the Posts, but can aggregate them immediately using our sliding-window approach - at the cost of increased network traffic.

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

5. EXPERIMENTS

5.1 Cardinality Estimation using Hash Sketches

To study the accuracy and the robustness of hash sketches as set cardinality estimators, we have performed a series of 100 runs, each for 256 8-byte bitvectors per sketch and different set sizes. The document sets are randomly created for each run from a sufficiently large domain. We report on the accuracy of the cardinality estimation using *accuracy*, i.e., the ratio $\frac{\text{estimated cardinality}}{\text{true cardinality}}$. As shown in Figure 1, the estimation works very well. On average (over 100 runs each), the accuracy is close to 100%, the standard deviation is sufficiently small with quartile errors around 5%. The plotted quartiles show the robustness of the approach which, as expected [16, 15], becomes better as more bitmaps are used per hash sketch. Notice also that the errors in both figures tend to become smaller for larger sets; this indicates that hash sketches will work very well in our intended environment (i.e., a large-scale system with a high number of documents).

5.2 DF Estimation in the Presence of Churn

While the previous experiments assumed a static setup of peers and their hash sketches, we now want to evaluate the accuracy of our approach in the presence of network dynamics. For this purpose, we consider a model of arrivals and departures as outlined in [25], where nodes arrive according to a Poisson process with rate λ , while a node in the system departs according to an exponential distribution with rate parameter μ . Resulting in a system of about 1,000 peers at a time, we assume time units of 10 min and choose $\lambda = 3$ and $\mu = 0.002$ and fix the interval at which peers refresh their statistics at 6 time units (60 min). For simplicity, we further set the Time-to-live for all statistics also to 6 time units. In a real world scenario, one could argue to increase the TTL to cope with network latencies and network failures, such that Posts in the directory survive one failed refresh attempt. Each peer randomly picks 1,000 documents from a domain of 2,000,000 documents. We use 256-bitmap hash sketches for our evaluation.

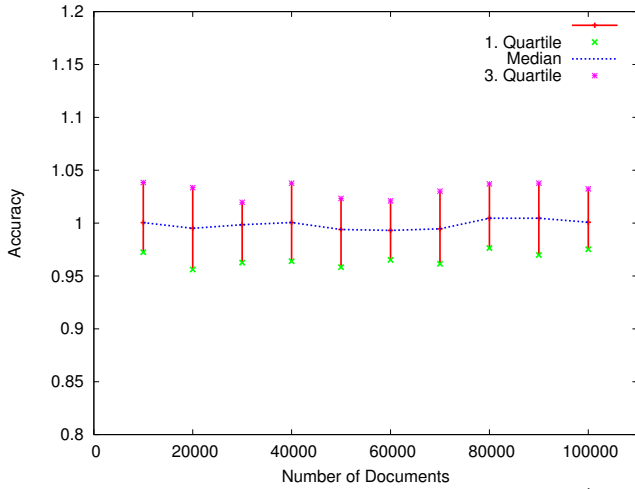


Figure 1: Cardinality Estimation Accuracy (256-bitmap Hash Sketches)

Figure 2 plots the document frequency estimates obtained by our approach together with the true document frequency. While intuitively, the approach should tend to overestimate the number of documents in the system, because metadata of peers that have recently left the system hang around for some time before they time out, in practice our experiments don’t clearly show this behavior. This is due to the fact that the hash sketches themselves show a certain degree of variance that overrules the (usually small) conceptual errors of the approach. Nevertheless, the approach has been shown to be robust against churn.

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

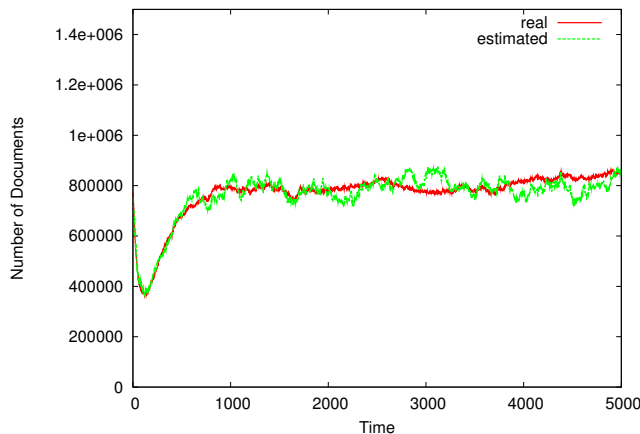


Figure 2: df Estimation Accuracy (256-bitmap Hash Sketches)

5.3 Improving Result Merging

For this experiment we use real-world Web data from 10 topically focused collections harvested by a focused web crawler. In order to create a benchmark, we have split each topical collection into 4 fragments. We create 40 peers such that each peer hosts 3 out of 4 fragments from the same topic, thus creating high overlap among same-topic peers. As query load, we 30 popular Google queries taken

from Zeitgeist³. We use CORI [7] as a common query routing strategy and compare 4 different result merging strategies. The document scores are based on collection-specific (i.e., “local”) df values or our *global* df values and are normalized using their respective weighted CORI peer score (from the query routing phase) or not. For this normalization, more specifically, we use the norm-dbs method used by the INQUERY framework [8], that re-computes document scores as $score = (D + 0.4 \times C_{norm} \times D)/1.4$. As a rank distance, we use Spearman’s footrule distance [14], defined as $F(\sigma_1, \sigma_2) = \sum_i |\sigma_{ref}(i) - \sigma_{peers}(i)|$ where $\sigma_{ref}(i)$ is the rank of document i in the reference ranking and $\sigma_{peers}(i)$ is the position of document i in the peers’ document ranking. If a document from σ_{peers} is not in σ_{ref} we assign a fictitious rank $(k + 1)$. We normalize all distances by $1 - \frac{distance}{max\ distance}$ to obtain a normalized quality measure.

Figure 3 shows the results for the 40-peers benchmark. With local query execution based on *global* df values, the ranking quality is remarkably above the quality obtained by the CORI-based merging methods. In particular, three out of the four methods do not even come close to the optimal document ranking at all, even if all 40 peers are involved in the query. This is due to the fact that the document scores based on *local* df values are incomparable across the peers and, thus, documents that are not in the reference top-20 document ranking are pushed in by skewed local df scores at the peers.

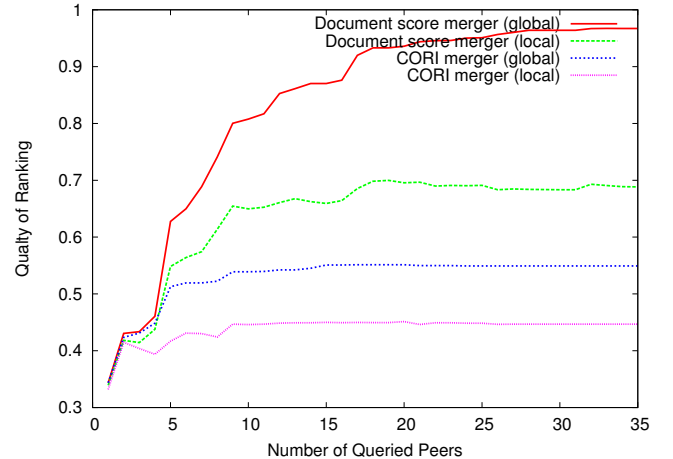


Figure 3: Quality of Query Results (40 Peers)

6. CONCLUSION AND OUTLOOK

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

We point out that the main focus of this paper is *not* to quantify the effect that the knowledge of global df values can have on result merging. The corresponding experiment is only preliminary, but nevertheless indicates the potential for improvements. While this effect has already been observed

³www.google.com/press/zeitgeist.html

in the literature [24], more comprehensive experiments on result merging in P2P networks are subject of future work.

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

7. REFERENCES

- [1] K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. V. Pelt. Gridvine: Building internet-scale semantic overlay networks. In *ISWC 2004*.
- [2] K. Aberer, M. Puceva, M. Hauswirth, and R. Schmidt. Improving data access in p2p systems. *IEEE Internet Computing*, 6(1):58–67, 2002.
- [3] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Improving collection selection with overlap awareness in p2p search engines. In *SIGIR05*, Salvador, Brasil, 2005. ACM.
- [4] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Minerva: Collaborative p2p search. In *VLDB*, 2005.
- [5] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7), 1970.
- [6] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed content delivery across adaptive overlay networks. *SIGCOMM*, 2002.
- [7] J. Callan. Distributed information retrieval. *Advances in information retrieval*, Kluwer Academic Publishers., 2000.
- [8] J. P. Callan, W. B. Croft, and J. Broglio. Trec and tipster experiments with inquiry. *Inf. Process. Manage.*, 31(3), 1995.
- [9] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR*, 1995.
- [10] S. Cohen and Y. Matias. Spectral bloom filters. In *SIGMOD 2003*.
- [11] G. Cormode and M. N. Garofalakis. Sketching streams through the net: Distributed approximate query tracking. In *VLDB*, 2005.
- [12] G. Cormode, S. Muthukrishnan, and W. Zhuang. What’s different: Distributed, continuous monitoring of duplicate-resilient aggregates on data streams. In *ICDE*, 2006.
- [13] F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen. Planetp: Using gossiping to build content addressable peer-to-peer information sharing communities. In *HPDC*, 2003.
- [14] P. Diaconis and R. Graham. Spearman’s footrule as a measure of disarray. *Journal of the Royal Statistical Society*, 1977.
- [15] M. Durand and P. Flajolet. Loglog counting of large cardinalities. In G. Di Battista and U. Zwick, editors, *ESA03*, volume 2832 of *LNCS*, Sept. 2003.
- [16] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2), 1985.
- [17] N. Fuhr. A decision-theoretic approach to database selection in networked IR. *ACM Transactions on Information Systems*, 17(3), 1999.
- [18] S. Ganguly, M. Garofalakis, and R. Rastogi. Processing set expressions over continuous update streams. In *SIGMOD*, 2003.
- [19] L. Gravano, H. Garcia-Molina, and A. Tomasic. Gloss: text-source discovery over the internet. *ACM Trans. Database Syst.*, 24(2), 1999.
- [20] T. Hernandez and S. Kambhampati. Improving text collection selection with coverage and overlap statistics. pc-recommended poster. WWW 2005. Full version available at <http://rakaposhi.eas.asu.edu/thomas-www05-long.pdf>.
- [21] S. Idreos, M. Koubarakis, and C. Tryfonopoulos. P2p-diet: An extensible p2p service that unifies ad-hoc and continuous querying in super-peer networks. In *SIGMOD*, 2004.
- [22] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(1):219–252, 2005.
- [23] S. Kirsch. Document retrieval over networks wherein ranking and relevance scores are computed at the client for multiple database documents. US patent 5,659,732, 1997.
- [24] L. S. Larkey, M. E. Connell, and J. P. Callan. Collection selection and results merging with topically organized u.s. patents and TREC data. In *CIKM 2000*.
- [25] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the evolution of peer-to-peer systems, 2002.
- [26] J. Lu and J. Callan. Merging retrieval results in hierarchical peer-to-peer networks. In *SIGIR*, 2004.
- [27] J. Lu and J. P. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *CIKM*, pages 199–206, 2003.
- [28] A. Manjhi, S. Nath, and P. B. Gibbons. Tributaries and deltas: Efficient and robust aggregation in sensor network streams. In *SIGMOD 2005*.
- [29] W. Meng, C. Yu, and K.-L. Liu. Building efficient and effective metasearch engines. *ACM Comput. Surv.*, 34(1):48–89, 2002.
- [30] S. Michel, M. Bender, P. Triantafillou, and G. Weikum. Iqn routing: Integrating quality and novelty in p2p querying and ranking. In *EDBT*, pages 149–166, 2006.
- [31] H. Nottelmann, G. Fischer, A. Titarenko, and A. Nurzenski. An integrated approach for searching and browsing in heterogeneous peer-to-peer networks. In *HDIR 2005*.
- [32] H. Nottelmann and N. Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. In *SIGIR*, 2003.
- [33] O. Papapetrou, S. Michel, M. Bender, and G. Weikum. On the usage of global document occurrences in peer-to-peer information systems. In *COOPIS 2005*.
- [34] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *SIGCOMM 2001*.
- [35] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR*, 1994.
- [36] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 2001*.
- [37] L. Si, R. Jin, J. Callan, and P. Ogilvie. A language modeling framework for resource selection and results merging. In *CIKM 2002*.
- [38] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM 2001*.
- [39] T. Suel, C. Mathur, J. wen Wu, J. Zhang, A. Delis, M. Kharrazi, X. Long, and K. Shanmugasundaram. Odissea: A peer-to-peer architecture for scalable web search and information retrieval. In *WebDB*, 2003.
- [40] Y. Wang, L. Galanis, and D. J. de Witt. Galanx: An efficient peer-to-peer search engine system. Available at <http://www.cs.wisc.edu/yuanwang>.