Overlap-Aware Global *df* Estimation in Distributed Information Retrieval Systems

Matthias Bender, Sebastian Michel,
Peter Triantafillou, Gerhard
Weikum

MPI–I–2006–5-001 January 2006

Authors' Addresses

Matthias Bender, Sebastian Michel, Gerhard Weikum
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany
{mbender, smichel, weikum}@mpi-inf.mpg.de

Peter Triantafillou
University of Patras
26500 Rio-Patra
Greece
peter@ceid.upatras.gr

Abstract

Peer-to-Peer (P2P) search engines and other forms of distributed information retrieval (IR) are gaining momentum. Unlike in centralized IR, it is difficult and expensive to compute statistical measures about the entire document collection as it is widely distributed across many computers in a highly dynamic network. On the other hand, such network-wide statistics, most notably, global document frequencies of the individual terms, would be highly beneficial for ranking global search results that are compiled from different peers. This paper develops an efficient and scalable method for estimating global document frequencies in a large-scale, highly dynamic P2P network with autonomous peers. The main difficulty that is addressed in this paper is that the local collections of different peers may arbitrarily overlap, as many peers may choose to gather popular documents that fall into their specific interest profile. Our method is based on hash sketches as an underlying technique for compact data synopses, and exploits specific properties of hash sketches for duplicate elimination in the counting process. We report on experiments with real Web data that demonstrate the accuracy of our estimation method and also the benefit for better search result ranking.

Keywords

peer-to-peer, distributed information retrieval, global document frequency estimation

Contents

1 Introduction

In recent years, distributed information retrieval systems based on Peer-to-Peer (P2P) architectures are increasingly receiving attention [23, 25, 19, 1, 18, 26, 27, 3, 11, 35]. 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.

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 BM25 [31], 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 term[1]. 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 as-

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

sumed 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) [21], as popular documents are repeatedly accounted for. Additionally, thematically focused collections show a high variance of *df* values for the same term (whereas randomly partitioned collections show a rather uniform distribution of *df* values for the same term). This further increases the necessity of a score normalization across peers.

We present a robust and scalable approach towards estimating *global df* values using hash sketches[14]. 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 and present experimental evidence of the recall improvements in result merging stemming from this improved knowledge. The experiments are conducted on real-word web data using our fully operational P2P Web search engine prototype.

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. [5] describes a permutation-based technique for efficiently estimating set similarities for informed content delivery. [16] proposes a hash-based synopsis data structure and algorithms to support low-error and high-confident estimates for general set expressions. Bloom [4] describes a data structure for succinctly representing a set in order to support membership queries; [9] present an extension for dealing with multisets, but still focuses on membership queries rather than cardinality estimation. [19] proposes a gossip-based protocol for computing aggregate values in a fully decentralized fashion. [24] addresses communication topology issues for distributed aggregation and identifying frequent items in a network. [10] develops a sketch-based framework for distributed estimation of query result cardinalities, but does not consider duplicates. None of [19, 24, 10] addresses the elimination of overlap.

2.2 Peer-to-Peer Architectures

Recent research on P2P systems, such as Chord [34], CAN [30], Pastry [32], 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. However, the approaches are limited to exact-match, single keyword queries on keys.

This is insufficient when queries should return a ranked result list of the most relevant approximate matches in the spirit of IR models.

2.3 Distributed IR and Web Search

Many approaches have been proposed for distributed IR, most notably, CORI [7], the decision-theoretic framework by [28], the GlOSS method presented in [17], and methods based on statistical language models [33]. In principle, these methods could be applied to a P2P setting, but they fall short in 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.

Galanx [36] 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 [11] is a publish-subscribe service for P2P communities, supporting content ranking search. The global index is replicated using a gossiping algorithm. Minerva [3] assumes peers independently crawling the web, generating collections tailored to their interest profiles. It uses a Chord style DHT to build a distributed meta data directory that provides a mapping from terms to published per-peer per-term statistics to identify promising peers for a query. Odissea [35] 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. GridVine [1] addresses the problem of building scalable semantic overlay networks and identifies strategies for their traversal using P-Grid [2]. P2P-Diet [18] consists of super-peers and client-peers and aims to support both ad-hoc and continues queries. Pepper [27] is a hierarchical peer-to-peer system that supports searching and browsing. Super-peers use the decision-theoretic framework [15] for resource selection. 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 [20] proposes to collect local statistics from the selected databases to normalize document scores. [22, 25] uses a centralized database of collection samples, which is incompatible with our architectural vision and seems infeasible in the presence of high network

dynamics. [6] 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 [21]. Experiments showed that global *idf* scores is the most desirable method, but they considered neither real-world Web pages nor overlap between collections. [29] 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. [8] uses statistical language models and query expansion to improve the result merging process.

3 Multiset 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 allows 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 [14], 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 mulitset synopses. We will show in this section how this duplicate-sensitive multiset-counting problem is elegantly solved by our approach based on hash sketches, and we demonstrate the low approximation error in experiments with real data.

3.1 Hash Sketches

Hash sketches were first proposed by Flajolet and Martin in [14] to probabilistically estimate the cardinality of a multiset S. [16] 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 $h() : S \rightarrow [0, 1, \ldots, 2^L)$. Du-

rand and Flajolet presented a similar algorithm in [13] (*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, $\rho(y) = min_{k \geq 0}\, bit(y, k) \neq 0,\ y > 0$, and $\rho(0) = L$. $bit(y, k)$ 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 $B[0 \ldots L-1]$. 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 $\log_2 n$. The authors in [14, 13] present analyses and techniques to bound from above the error introduced.Techniques which provably reduce the statistical estimation error typically rely on employing multiple bitmap 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. Combining an arbitrary number of hash sketches to form a hash sketch for the combined collection is easy by design: given the hash sketch representations of two index lists for the same term from different peers (and relying on a globally unique way of constructing docIDs, e.g., from URLs), a simple bit-wise OR-operation yields a hash sketch for the combined collection that instantly allows us to estimate the number of (distinct) documents of the combined index list for that term.

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 documents in one of the original collections had set this

bit. Particularly notice that, if both original collections carry this document, the document will conceptually be counted only once, effectively removing duplicates.

3.3 Experiments

To evaluate the accuracy and the robustness of hash sketches as set cardinality estimators, we have made series of 100 runs each for different sized hash sketches (i.e., different numbers of 8-byte bitvectors per sketch) and different set sizes, randomly created for each run. Remember from the previous section that it is sufficient to evaluate the performance of hash sketches on *one* set, as there is no additional error incurred from the distribution. As shown in Figure 3.1, the median error is below 5% and the error becomes lower for larger sets. This makes us believe that hash sketches will work even better in a large-scale system. The plotted quartiles show the robustness of the approach which, as expected, becomes better as more bitmaps are used per hash sketch.

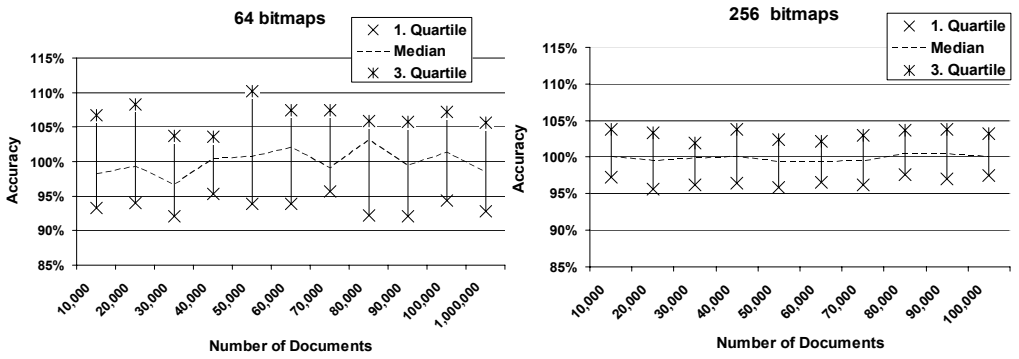

Figure 3.1: Hash Sketch Accuracy

4 System Issues

4.1 Design Fundamentals

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 determines the peer currently responsible for this term, which 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 be enormously benefit from the knowledge of global df values.

We have implemented a fully operational P2P Web search engine building on these design fundamentals.

4.2 Extensions for *df* estimation

Given the system design introduced above with a hash-based assignment of terms to responsible directory peers, it is very natural for these 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 a moving-window estimate for the *global df* for the terms it is responsible for from these synopses.

Having calculated such *df* estimates, we propose two methods for the dissemination of these values and their usage in the query process:

- Each time a peer contacts a remote directory node, e.g., during the posting process, it retrieves the current *df* estimates from the directory peer and uses these values to re-compute its local scores. In that way, all document scores locally calculated at the peers are directly comparable, as they share the same statistical information. The disadvantage is the necessity to repeatedly re-compute a large number of local scores, which poses a high computation burden on the peers.

- The query initiator collects the estimates as piggybacked information when retrieving the PeerLists from the directory peers. [1] The query initiator can then include the *df* values when sending the query to the selected peers in the queryrouting phase. These remote peers can use

[1]Remember that the *df* estimate for a particular term is maintained at the same peer that maintains the respective PeerList

the *df* estimates on-the-fly (as weights during index scans) to compute their local query results.

We advocate for the second option and present a detailed study of its low overhead cost in Subsection 4.4.

Also note that it is *not* a design choice to let the remote peers simply return unnormalized scores (e.g., based on local tf values only) and then let the query initiator do the re-calibration using global *df* estimates. In that case, the local query execution at the remote peers may already miss some of the globally best results. For example, a document that has high scores for the terms with low global *df* (i.e., high *idf*) may not be returned at all if these terms have high local *df* values. So especially the remote peers with many good results for the important (low global *df*) terms are handicapped by the lack of global *df* statistics.

4.3 Dealing with Churn

To cope with the presence of churn[2], we propose a *time-sliding window* approach that (in line with the timeout policy for the Posts) combines only those hash sketches that are not older than a certain threshold and shows a negligible storage overhead. A directory peer keeps per-term $n+1$ instances of hash sketches. At time x, every $\frac{TTL}{n}$ time units, each hash sketch is shifted to the right, and the leftmost hash sketch is reset to all-zeros. The i-th hash sketch ($i > 1$) represents the time interval $[x - \frac{(i-2)*TTL}{n}, x - \frac{(i-1)*TTL}{n})$. All incoming hash sketches are OR'ed with the leftmost hash sketch. The *df* estimate is a simple bitwise OR of the $n+1$ hash sketches. This technique is depicted in Figure 4.1.

Since the sliding window keeps information about more than one TTL interval and each peer is updating its hash sketches at least once in this time span, its contribution will at every time be part of the *df* estimation. On the other hand, if a peer fails or its hash sketch changes, the false hash sketch will contribute to the estimation for a maximum timespan $\frac{n+1}{n}TTL$, because by then it will move out of the window.

[2]The term *churn* is popularly used to describe the high dynamics of a P2P system, as peers enter and leave the system at a high rate, at unpredictable intervals, and without prior notice.

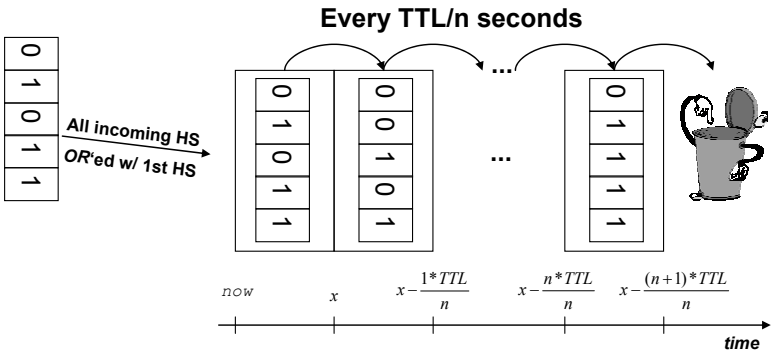

Figure 4.1: Sliding Window for *df* estimation

4.4 Cost Analysis

Most of the **network cost** is caused during the posting process, i.e., when a peer publishes its per-term statistics. 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). Typically, such a Post accounts for approximately 50 bytes. As we have shown earlier, 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 Post itself to the responsible directory peer. Thus, the number of messages when disseminating 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 also apply *gzip* compression to additionally decrease the message payload size.

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 requestor. 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 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 new 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 2.5 MB, which we consider a negligible storage effort.

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 and the cost for combining two hash sketches is a simple simple bit-wise *or* of 8-byte bitvectors.

Note that the local query execution is not affected at all by adding hash sketches: index lists can be sorted by tf scores or whatever criteria the local query processor prefers. and can be scanned as usual. One extra computational operation is required for each list item to compute the final (term-) score for this item. In this case, the processing of items 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.

5 Experiments

To show the accuracy of our global *df* estimation method we have first conducted several experiments with 100 peers (each with 500 documents) and a set of 100,000 synthetically generated single-term documents where the degree of document replicas is controlled by a Zipf distribution with parameter θ. According to Zipf's law the frequency of occurrence of the document at rank n is $P_n \sim 1/n^\theta$, so we create 100,000 document ids corresponding to ranks 1 to 100,000. Subsequently, each peer chooses 500 distinct documents (given by ranks) out of the 100,000 where the probability of choosing a document depends on the rank as described above. Figure 5.1 shows the accuracy of the global df estimation for this controlled setup. As expected from the experiments in Section 3.3, our estimation is very exact. Note that the naive approach of summing up the local document frequencies estimates the number of distinct documents as $500 \times 100 = 50,000$ and is way off, we do not show it in Figure 5.1. Remember at this point (cf. Section 3.2) that our method does *not* incur any additional error from the distribution over 100 peers, but the accuracy is the same as if we had done the estimation on a combined collection of all 100 peers.

For experiments on real web data we have created 10 topically focused collections using a focused crawler; each collection is assigned to a separate peer (see Table 5.1). For an additional test with a larger number of peers we have created 40 peers out of the 10 collections by splitting each collection into 4 fragments. Each of the 40 peers hosts 3 out of 4 fragments from the same topic, thus forming high overlap among same-topic peers. To evaluate our approach we have used 18 popular Google queries taken from Zeitgeist (www.google.com/press/zeitgeist.html); they are shown in Table 5.2. For all experiments, we use CORI [6] scores to find an appropriate order in which to query the peers. We vary the parameters of the local tf*idf scoring function: as a state-of-the-art baseline, every peer applies local *df* values for document scoring; we compare this to our approach of using estimated global df values for local query execution. For comparing the top-k ranked documents (with

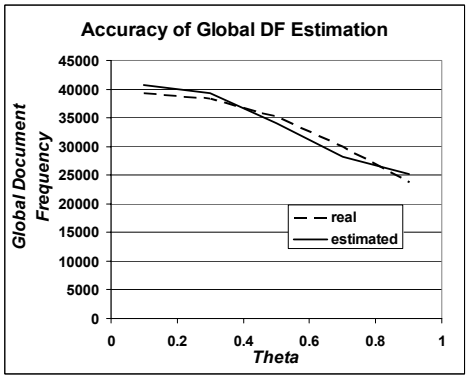

Figure 5.1: Accuracy of DF Estimation for Different Values of θ

$k = 20$) delivered by an increasing number of peers with the ideal top-k ranking (from querying the centralized union of collections) we apply Spearman's footrule distance [12], 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 $(k + 1)$. Figure 5.2(left) shows the results for the 10-peers benchmark, and Figure 5.2(right) the results for the 40-peers benchmark. One can see that the global df-based ranking has a higher quality w.r.t. the reference collection (i.e. the union of all collections). In Figure 5.2(left) one can see that local df based ranking never reaches the ideal ranking as the scores coming from different peers are incomparable. To understand the document frequencies' effect on the ranking, Table 5.3 shows the local document frequency values for the query terms. For terms that are likely to occur in some topics but not in others, we observe highly skewed distribution. For example, the term *cup* occurs in $2,015$ documens at Peer 9 (the sports peer) and less than 350 documents for each of the other peers.

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

Table 5.1: Collection attributes

Query 1	world series
Query 2	pamela anderson
Query 3	national hurrican center
Query 4	anna kournikova
Query 5	world aids day
Query 6	pearl harbor
Query 7	weight loss
Query 8	emmy awards
Query 9	ryder cup
Query 10	national hurricane center
Query 11	serena williams
Query 12	beach volleyball
Query 13	arafat
Query 14	lunar eclipse
Query 15	daylight savings
Query 16	westminster dog show
Query 17	southwest airlines
Query 18	oscar nominations

Table 5.2: Google Zeitgeist Queries

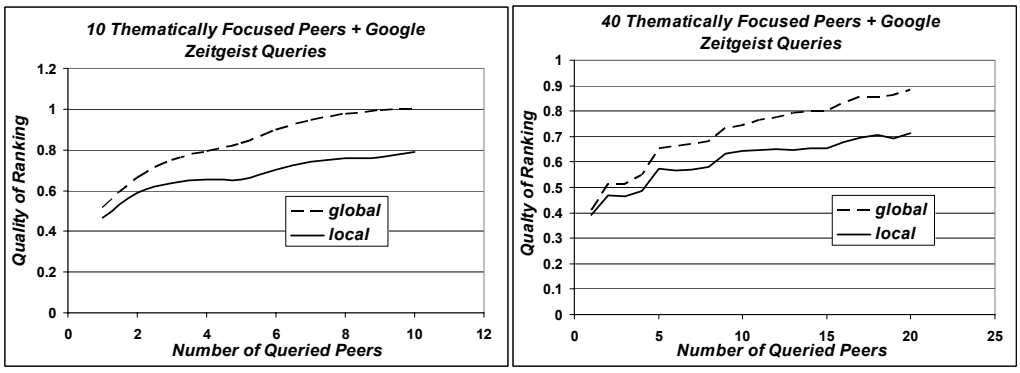

Figure 5.2: Quality of the Document Ranking

	Peer 1	Peer 2	Peer 3	Peer 4	Peer 5	Peer 6	Peer 7	Peer 8	Peer 9	Peer 10
anderson	76	274	85	161	346	257	678	127	100	480
anna	86	502	70	57	52	165	1348	62	104	209
arafat	25	15	16	2	1	4	7	0	9	157
beach	3865	812	302	187	132	397	1290	542	1081	766
center	3209	3987	2845	8416	5124	2411	2120	2206	1301	7496
cup	321	254	206	179	72	103	312	341	2015	172
day	7468	4114	6725	4415	2793	4846	6425	4072	4637	7905
daylight	53	32	13	14	81	21	132	67	45	57
dog	743	748	449	451	81	774	1757	691	888	676
harbor	237	164	57	96	112	37	286	181	54	369
hurrican	118	27	69	51	140	19	146	108	116	322
kournikova	4	10	5	1	1	1	142	0	16	1
loss	865	308	1023	2421	647	437	345	711	565	932
lunar	40	16	12	4	602	10	58	17	106	47
national	20	57	7	0	18	8	10	5	2	16
oscar	65	141	66	20	12	197	2287	21	146	129
pamela	34	117	24	48	16	67	234	18	24	129
pearl	266	267	95	37	54	279	537	197	114	155
ryder	23	31	19	4	10	37	110	2	69	18
serena	11	12	30	7	9	4	55	4	27	5
show	3757	4318	3317	2512	3708	3899	11797	3326	2888	4434
southwest	605	275	80	124	159	193	94	436	65	262
weight	1924	1111	1728	3224	609	1083	3432	754	1573	1388
world	11280	6254	4841	2835	2975	6313	7466	3255	7961	11242

Table 5.3: Number of Documents per Peer for the 18 Query Terms

6 Conclusion

This paper has adressed the efficient estimation of global document frequencies in P2P networks. We have shown that globally comparable document scores can be enjoyed with only a small overhead in network resource consumption and that these scores improve the quality of the final document ranking in P2P Web Search. Our approach is not limited to P2P Web Search but can be applied in distributed systems where global counting with duplicate elimination is an issue. To our knowledge this is the first paper to provide a general and efficient solution to the problem of estimating global *df* with high accuracy and efficiency in distributed IR environments.

Bibliography

[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. Punceva, M. Hauswirth, and R. Schmidt. Improving data access in p2p systems. *IEEE Internet Computing*, 6(1), 2002.

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

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

[5] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed content delivery across adaptive overlay networks. In SIGCOMM 2002.

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

[7] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR1995*, 1995.

[8] S. Chernov, P. Serdyukov, M. Bender, S. Michel, G. Weikum, and C. Zimmer. Database selection and result merging in P2P web search. In *DBISP2P 2005*.

[9] S. Cohen and Y. Matias. Spectral bloom filters. In *SIGMOD 2003*.

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

[11] 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.

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

[13] 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.

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

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

[16] S. Ganguly, M. Garofalakis, and R. Rastogi. Processing set expressions over continuous update streams. In *SIGMOD2003*, 2003.

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

[18] 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.

[19] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(1), 2005.

[20] 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.

[21] 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*.

[22] J. Lu and J. Callan. Merging retrieval results in hierarchical peer-to-peer networks. In *SIGIR 2004*.

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

[24] A. Manjhi, S. Nath, and P. B. Gibbons. Tributaries and deltas: Efficient and robust aggregation in sensor network streams. In *SIGMOD 2005*.

[25] W. Meng, C. Yu, and K.-L. Liu. Building efficient and effective metasearch engines. *ACM Comput. Surv.*, 34(1), 2002.

[26] W. Müller and A. Henrich. Fast retrieval of high-dimensional feature vectors in p2p networks using compact peer data summaries. In *MIR 2003*.

[27] 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*.

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

[29] 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*.

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

[31] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR*, 1994.

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

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

[34] 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*.

[35] 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.

[36] Y. Wang, L. Galanis, and D. J. de Witt. Galanx: An efficient peer-to-peer search engine system. *http://www.cs.wisc.edu/ yuanwang*.

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available by anonymous ftp from `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Anja Becker
Stuhlsatzenhausweg 85
66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-2006-5-001	M. Bender, S. Michel, G. Weikum, P. Triantafilou	Overlap-Aware Global df Estimation in Distributed Information Retrieval Systems
MPI-I-2005-5-002	S. Siersdorfer, G. Weikum	Automated Retraining Methods for Document Classification and their Parameter Tuning
MPI-I-2005-4-006	C. Fuchs, M. Goesele, T. Chen, H. Seidel	An Emperical Model for Heterogeneous Translucent Objects
MPI-I-2005-4-005	G. Krawczyk, M. Goesele, H. Seidel	Photometric Calibration of High Dynamic Range Cameras
MPI-I-2005-4-004	C. Theobalt, N. Ahmed, E. De Aguiar, G. Ziegler, H. Lensch, M.A.,. Magnor, H. Seidel	Joint Motion and Reflectance Capture for Creating Relightable 3D Videos
MPI-I-2005-4-003	T. Langer, A.G. Belyaev, H. Seidel	Analysis and Design of Discrete Normals and Curvatures
MPI-I-2005-4-002	O. Schall, A. Belyaev, H. Seidel	Sparse Meshing of Uncertain and Noisy Surface Scattered Data
MPI-I-2005-4-001	M. Fuchs, V. Blanz, H. Lensch, H. Seidel	Reflectance from Images: A Model-Based Approach for Human Faces
MPI-I-2005-2-004	Y. Kazakov	A Framework of Refutational Theorem Proving for Saturation-Based Decision Procedures
MPI-I-2005-2-003	H.d. Nivelle	Using Resolution as a Decision Procedure
MPI-I-2005-2-002	P. Maier, W. Charatonik, L. Georgieva	Bounded Model Checking of Pointer Programs
MPI-I-2005-2-001	J. Hoffmann, C. Gomes, B. Selman	Bottleneck Behavior in CNF Formulas
MPI-I-2005-1-008	D. Michail	?
MPI-I-2005-1-007	I. Katriel, M. Kutz	A Faster Algorithm for Computing a Longest Common Increasing Subsequence
MPI-I-2005-1-003	S. Baswana, K. Telikepalli	Improved Algorithms for All-Pairs Approximate Shortest Paths in Weighted Graphs
MPI-I-2005-1-002	I. Katriel, M. Kutz, M. Skutella	Reachability Substitutes for Planar Digraphs
MPI-I-2005-1-001	D. Michail	Rank-Maximal through Maximum Weight Matchings
MPI-I-2004-NWG3-001	M. Magnor	Axisymmetric Reconstruction and 3D Visualization of Bipolar Planetary Nebulae
MPI-I-2004-NWG1-001	B. Blanchet	Automatic Proof of Strong Secrecy for Security Protocols
MPI-I-2004-5-001	S. Siersdorfer, S. Sizov, G. Weikum	Goal-oriented Methods and Meta Methods for Document Classification and their Parameter Tuning
MPI-I-2004-4-006	K. Dmitriev, V. Havran, H. Seidel	Faster Ray Tracing with SIMD Shaft Culling
MPI-I-2004-4-005	I.P. Ivrissimtzis, W.-. Jeong, S. Lee, Y.a. Lee, H.-. Seidel	Neural Meshes: Surface Reconstruction with a Learning Algorithm
MPI-I-2004-4-004	R. Zayer, C. Rössl, H. Seidel	r-Adaptive Parameterization of Surfaces
MPI-I-2004-4-003	Y. Ohtake, A. Belyaev, H. Seidel	3D Scattered Data Interpolation and Approximation with Multilevel Compactly Supported RBFs

MPI-I-2004-4-002	Y. Ohtake, A. Belyaev, H. Seidel	Quadric-Based Mesh Reconstruction from Scattered Data
MPI-I-2004-4-001	J. Haber, C. Schmitt, M. Koster, H. Seidel	Modeling Hair using a Wisp Hair Model
MPI-I-2004-2-007	S. Wagner	Summaries for While Programs with Recursion
MPI-I-2004-2-002	P. Maier	Intuitionistic LTL and a New Characterization of Safety and Liveness
MPI-I-2004-2-001	H. de Nivelle, Y. Kazakov	Resolution Decision Procedures for the Guarded Fragment with Transitive Guards
MPI-I-2004-1-006	L.S. Chandran, N. Sivadasan	On the Hadwiger's Conjecture for Graph Products
MPI-I-2004-1-005	S. Schmitt, L. Fousse	A comparison of polynomial evaluation schemes
MPI-I-2004-1-004	N. Sivadasan, P. Sanders, M. Skutella	Online Scheduling with Bounded Migration
MPI-I-2004-1-003	I. Katriel	On Algorithms for Online Topological Ordering and Sorting
MPI-I-2004-1-002	P. Sanders, S. Pettie	A Simpler Linear Time 2/3 - epsilon Approximation for Maximum Weight Matching
MPI-I-2004-1-001	N. Beldiceanu, I. Katriel, S. Thiel	Filtering algorithms for the Same and UsedBy constraints
MPI-I-2003-NWG2-002	F. Eisenbrand	Fast integer programming in fixed dimension
MPI-I-2003-NWG2-001	L.S. Chandran, C.R. Subramanian	Girth and Treewidth
MPI-I-2003-4-009	N. Zakaria	FaceSketch: An Interface for Sketching and Coloring Cartoon Faces
MPI-I-2003-4-008	C. Roessl, I. Ivrissimtzis, H. Seidel	Tree-based triangle mesh connectivity encoding
MPI-I-2003-4-007	I. Ivrissimtzis, W. Jeong, H. Seidel	Neural Meshes: Statistical Learning Methods in Surface Reconstruction
MPI-I-2003-4-006	C. Roessl, F. Zeilfelder, G. Nürnberger, H. Seidel	Visualization of Volume Data with Quadratic Super Splines
MPI-I-2003-4-005	T. Hangelbroek, G. Nürnberger, C. Roessl, H.S. Seidel, F. Zeilfelder	The Dimension of C^1 Splines of Arbitrary Degree on a Tetrahedral Partition
MPI-I-2003-4-004	P. Bekaert, P. Slusallek, R. Cools, V. Havran, H. Seidel	A custom designed density estimation method for light transport
MPI-I-2003-4-003	R. Zayer, C. Roessl, H. Seidel	Convex Boundary Angle Based Flattening
MPI-I-2003-4-002	C. Theobalt, M. Li, M. Magnor, H. Seidel	A Flexible and Versatile Studio for Synchronized Multi-view Video Recording
MPI-I-2003-4-001	M. Tarini, H.P.A. Lensch, M. Goesele, H. Seidel	3D Acquisition of Mirroring Objects
MPI-I-2003-2-004	A. Podelski, A. Rybalchenko	Software Model Checking of Liveness Properties via Transition Invariants
MPI-I-2003-2-003	Y. Kazakov, H. de Nivelle	Subsumption of concepts in DL $\mathcal{FL}_0$ for (cyclic) terminologies with respect to descriptive semantics is PSPACE-complete
MPI-I-2003-2-002	M. Jaeger	A Representation Theorem and Applications to Measure Selection and Noninformative Priors
MPI-I-2003-2-001	P. Maier	Compositional Circular Assume-Guarantee Rules Cannot Be Sound And Complete
MPI-I-2003-1-018	G. Schaefer	A Note on the Smoothed Complexity of the Single-Source Shortest Path Problem
MPI-I-2003-1-017	G. Schäfer, S. Leonardi	Cross-Monotonic Cost Sharing Methods for Connected Facility Location Games
MPI-I-2003-1-016	G. Schäfer, N. Sivadasan	Topology Matters: Smoothed Competitive Analysis of Metrical Task Systems
MPI-I-2003-1-015	A. Kovács	Sum-Multicoloring on Paths
MPI-I-2003-1-014	G. Schäfer, L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, T. Vredeveld	Average Case and Smoothed Competitive Analysis of the Multi-Level Feedback Algorithm