Web Text Retrieval with a P2P Query-Driven Index

Gleb Skobeltsyn[†], Toan Luu[†], Ivana Podnar Žarko[‡], Martin Rajman[†], Karl Aberer[†]

[†]Ecole Polytechnique Fédérale de Lausanne (EPFL)
School of Computer and Communication Sciences
Lausanne, Switzerland

[‡]University of Zagreb
Faculty of Electrical Engineering and Computing
Zagreb, Croatia

{gleb.skobeltsyn, vinhtoan.luu, martin.rajman, karl.aberer}@epfl.ch, ivana.podnar@fer.hr

ABSTRACT

In this paper, we present a query-driven indexing/retrieval strategy for efficient full text retrieval from large document collections distributed within a structured P2P network. Our indexing strategy is based on two important properties: (1) the generated distributed index stores posting lists for *carefully chosen indexing term combinations*, and (2) the posting lists containing too many document references are truncated to a *bounded number of their top-ranked elements*. These two properties guarantee acceptable storage and bandwidth requirements, essentially because the number of indexing term combinations remains scalable and the transmitted posting lists never exceed a constant size. However, as the number of generated term combinations can still become quite large, we also use term statistics extracted from available query logs to index only such combinations that are frequently present in user queries. Thus, by avoiding the generation of superfluous indexing term combinations, we achieve an additional substantial reduction in bandwidth and storage consumption. As a result, the generated distributed index corresponds to a constantly evolving *query-driven indexing structure* that efficiently follows current information needs of the users.

More precisely, our theoretical analysis and experimental results indicate that, at the price of a marginal loss in retrieval quality for rare queries, the generated index size and network traffic remain manageable even for web-size document collections. Furthermore, our experiments show that at the same time the achieved retrieval quality is fully comparable to the one obtained with a state-of-the-art centralized query engine.

Categories and Subject Descriptors: H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing – *Indexing Methods*; E.1 [Data Structures]: Distributed Data Structures

General Terms: Algorithms

Keywords: P2P DHT IR Text Retrieval Query-Driven Indexing TREC Precision

1. INTRODUCTION

In reaction to the scalability problems encountered with centralized information retrieval engines, P2P networks that distribute a global index over a large number of interconnected peers have been recently considered as a promising solution to cope with web-scale document retrieval. However, while P2P networks containing very large number of peers indeed provide virtually unlimited storage capacities, there still is an ongoing debate about the true scalability of P2P web search. Particularly, in [9], the authors show that a naïve use of structured or unstructured P2P networks for web retrieval leads to unscalable network traffic, and, even for more sophisticated schemes, such as term-to-peer indexing [6, 4] or hierarchical federated architectures [3, 11], only little evidence of their scalability is available. In fact, in [23], it has been shown that, even when carefully optimized, distributed algorithms using traditional single-term indices in structured P2P networks generate unscalable network traffic during retrieval, mainly because of the bandwidth consumption resulting from the large posting list intersections required to process queries with frequent terms.

In the approach described in [14], instead of indexing with single terms, which leads to potentially very large posting lists, we showed a significant reduction of the *retrieval* traffic by systematically truncating large posting lists to a constant size, while compensating the resulting loss of information by also indexing with carefully selected combinations of indexing terms. Consequently, the index contains a larger number of entries (terms *and* term combinations), but each entry is associated with a shorter posting list. The central property (shown in [14]) of such an indexing strategy is that the size of the index grows linearly with the number of documents, which is acceptable under the reasonable assumption that the ratio between the total number of documents and the total number of peers in the network remains bounded.

However, as the number of generated term combinations might still remain practically unmanageable for large document collections, in [19], we suggested that filtering indexing keys using query statistics might lead to a further reduction of the *indexing* traffic without compromising retrieval quality. In this paper we will provide a detailed description of our query-driven indexing technique and carefully evaluate its impact on retrieval quality using extensive evaluations based on realistic web-scale document collections and query logs, as well as the TREC collection. In a companion paper [20], we provide a detailed description of architectural and algorithmic issues of this approach which we cannot include here due to space limitations.

Thus, our query-driven indexing strategy relies on the use of statistics about query popularity and an on-demand indexing mechanism to ensure that only popular and non-redundant term combinations are maintained in the index. With this new indexing strategy, the requirements in terms of storage and bandwidth consumption are further reduced, due to a more compact and efficient indexing structure, while the bounded size of the associated posting lists still guarantees a manageable bandwidth consumption during retrieval. These two properties make our approach a promising solution for web-scale P2P information retrieval.

The rest of this paper is organized as follows: We describe the indexing and retrieval models and mechanisms in Section 2. We briefly outline our scalability analysis in Section 3 and provide our new experimental results in Section 4. Finally, we position our approach with respect to related work in Section 5, give some more general perspectives for P2P information retrieval in Section 6 and conclude the paper in Section 7.

2. DISTRIBUTED INDEXING/RETRIEVAL

2.1 P2P global index

Let us consider a structured P2P network with N peers P_i, $1 \leq i \leq N$, and a possibly very large document collection $\mathcal{D}$, consisting of $|\mathcal{D}|$ documents d_j, $1 \leq j \leq |\mathcal{D}|$. $T_{\mathcal{D}}$ is the set of all indexing single terms in $\mathcal{D}$ and $|T_{\mathcal{D}}|$ denotes the number of terms in $T_{\mathcal{D}}$.

In addition, we assume that a large query log L is available, where each query $q \in L$ is a set of terms. The set of all terms present in the query log is denoted by T_L, and the intersection $T_L \cap T_{\mathcal{D}}$ thus corresponds to the query terms producing non-empty results.

In the P2P network, each peer P_i stores a fraction of the global document collection $\mathcal{D}$, denoted by $\mathcal{D}_i$, and builds a local index for $\mathcal{D}_i$. At the same time, P_i contributes to store and maintain a fraction of the global distributed index $\mathcal{I}$ that associates indexing terms and term combinations with references to documents in $\mathcal{D}$.

Definition 1. A *key* k refers to an indexing term or a combination of indexing terms. We apply standard stop-word elimination and stemming procedures [15] while generating a key.

A posting list $\rho(k)$ associated with a key k is the list of references to documents that contain k: $\rho(k) = \{d_j \in \mathcal{D} \mid k \in d_j\}$. In addition, each pair (d_j, k), $d_j \in \rho(k)$ is associated with a relevance score $r(d_j, k)$. Various models can be used to compute the relevance scores. Currently, we are using the top performing BM25 relevance computation scheme [17]. Notice, however, that any other relevance computation scheme could be used instead, provided that the required global statistics are stored in the P2P network.

Definition 2. A *truncated posting list* (*TPL*) $\tau(k)$ associated with a key k refers to the DF_{max} best-ranked document references in the posting list $\rho(k)$, where DF_{max} is a parameter of our model, corresponding to the maximal size of a TPL.

By construction, $|\tau(k)| \leq DF_{max}$, and, obviously, $\tau(k) = \rho(k)$ when $|\rho(k)| \leq DF_{max}$.

Definition 3. The *usage frequency* $qf(k)$ of a key k refers to a value that indicates the global popularity of the key k in the query log L. In the simplest case, $qf(k)$ can be the query frequency of k, which is incremented each time a new query that contains k is processed.

Currently, we use oblivious frequency counters to maintain usage frequencies only within a time interval of predefined length and therefore enable a timely reaction to changes in the query popularity distribution.

Definition 4. A *candidate index item* for a key k is the pair $(k,\ qf(k))$ associating k with its usage frequency $qf(k)$. A candidate index item is created for the key k and inserted in the global index $\mathcal{I}$ iff:

- k contains from 2 to s_{max} terms: $2 \leq |k| \leq s_{max}$, where s_{max} is a parameter of our model (*size filter*).
- for all immediate sub-keys of k, their posting lists contain more than DF_{max} elements and the corresponding TPLs are stored in the global index (see Definition 5). $\forall\ k' \subset k,\ |k'| = |k| - 1 : |\rho(k')| > DF_{max}\ \&\ \tau(k') \in \mathcal{I}$ (*redundancy filter*).

Definition 5. An *active index item* for a key k is the triple $(k,\ qf(k),\ \tau(k))$ associating k with its usage frequency $qf(k)$ *and* its TPL $\tau(k)$. An existing candidate index item for a key k is *activated* (i.e., its status is changed from candidate to active) iff:

- k is *popular*: $qf(k) \geq QF_{min}$, where QF_{min} is a parameter of our model (*popularity filter*).

The global distributed index $\mathcal{I}$ maintains the large set of candidate and active index items generated for all the keys inserted in the index. Notice that any key k can be, either 1) not present in the index, or 2) associated with a candidate index item, or 3) associated with an active index item.

We call a key k a candidate key if the global index contains a *candidate* index item for k. Similarly, we say that a key k is indexed if an *active* index item is stored for it in the global index. Notice that a key can be indexed only if it passed all three filers (size, redundancy and popularity) defined above.

The global index is distributed over the peers such that the fraction of the index under the responsibility of a peer P_i is exactly the set of index items associated with the keys that are allocated to P_i by the Distributed Hash Table (DHT) built on top of the P2P network. Note that, in such a DHT, the peer responsible for a given key can be uniquely determined by applying a globally known hash function, thus, ensuring balanced placement of the indexing information. An efficient and self-organizing communication protocol enables any peer to route a message to the peer responsible for a given key in $O(\log N)$ overlay hops, where N is the total number of peers in the network. It is out of scope of this paper to explain the details of such a protocol, which can be found in [1].

Within such a setup, each peer P_i is responsible for the following complementary tasks:

- P_i takes care that its local document collection $\mathcal{D}_i$ is properly represented in the global distributed index $\mathcal{I}$.
- P_i maintains its fraction of the global index $\mathcal{I}_i$[1]. In particular, it takes care that the usage frequencies are

[1]Notice that the fraction of the global index maintained at P_i is has no a priori reason to be related to the local document collection stored at P_i.

updated during query processing. Based on this information, P_i can decide to activate candidate index items or to deactivate active index items.

- While processing a query q, P_i interacts with the global P2P index in order to retrieve relevant TPLs stored within active index items. If necessary, P_i requests the corresponding peers to create new candidate index items for one or more keys contained in q.

A more detailed description of the above-mentioned indexing and retrieval tasks is given below. The formal description of the corresponding algorithms can be found in [20].

2.2 Indexing/Retrieval mechanisms

The goal of distributed indexing is to generate and maintain a suitable set of active index items, associated with the corresponding TPLs, for any given global document collection $\mathcal{D}$ distributed over N peers and the current query popularity distribution. Since the indexing process is computationally intensive, peers share the indexing load, and collaboratively build the required distributed index.

First, the peers build a so-called *basic single-term index* that contains *active* index items for all indexing single terms in $\mathcal{D}$. Each peer P_i performs indexing of its local document collection $\mathcal{D}_i$ and inserts all found single-term keys, associated with their local TPLs, into the P2P network. As a result, an active index item is generated for each single term $t \in \mathcal{D}$. It contains the corresponding global TPL and is maintained by the peer responsible for t. Recall that all TPLs are of bounded size, i.e., as soon as more than DF_{max} document references are collected, entries with the lowest scores are discarded such that only the DF_{max}-best ranked elements remain in the TPL. By default, all (even unpopular) index items associated with single-terms are active, and hence, the basic index enables the processing of any query, possibly with a degraded retrieval performance due to the loss of information caused by truncation.

The subsequent indexing process is fully driven by the query statistics, and is performed in parallel with retrieval. More precisely, as soon as a peer P receives a new query q, it starts to explore the lattice of the query term subsets (hereafter called the *query lattice*), in decreasing subset size order starting with the query itself (an example of a query lattice is given in Figure 1, which shows 3 potential scenarios for the processing of a query). For each of the explored lattice nodes q' (hereafter called the *query keys*), the querying peer P requests from the peer P' responsible for q' the posting list associated with q'.

Each peer P' that receives such a request attempts to increment the usage frequency of q' in the corresponding index item. If no index item exists for q', no extra action is taken. Additionally, as soon as a candidate key q' becomes popular, P' initiates the key activation process by triggering an on-demand indexing mechanism for q' (see Section 2.3). As a result of the on-demand key indexing, the candidate index item associated with q' will acquire a new global TPL and thus become an active index item that can be used for subsequent query processing. For example, in Figure 1-b, a candidate index item is stored for the key bc, and, as soon as its popularity reaches the QF_{min} threshold, it is activated as shown in Figure 1-c.

If an active index item for q' is located, the peer P' sends back to the querying peer P the content of the TPL associated with q'. When P receives the TPL, it stores it locally, and the part of the query lattice dominated by q', i.e., all the query keys contained in q', are excluded from the subsequent lattice exploration. For example, for the query lattice shown in Figure 1-c, if a TPL associated with the key bc is retrieved from the P2P index, the query keys b and c are not further explored. Thus, the top-down query lattice exploration can lead to two mutually exclusive terminal situations:

- At least one query key associated with a *non-truncated* posting list is reached. For example the key a in Figure 1-a is associated with an exhaustive posting list that can be used to answer any query containing a, e.g., abc.

- A cut[2] of query keys associated with *truncated* posting lists is reached. If none of the query keys is popular enough to be associated with an active index item, the cut will consist of all the single-term keys contained in the query (see Figure 1-b). Otherwise, the cut can consist of keys of different sizes (see Figure 1-c where keys a and bc form the cut for the query abc).

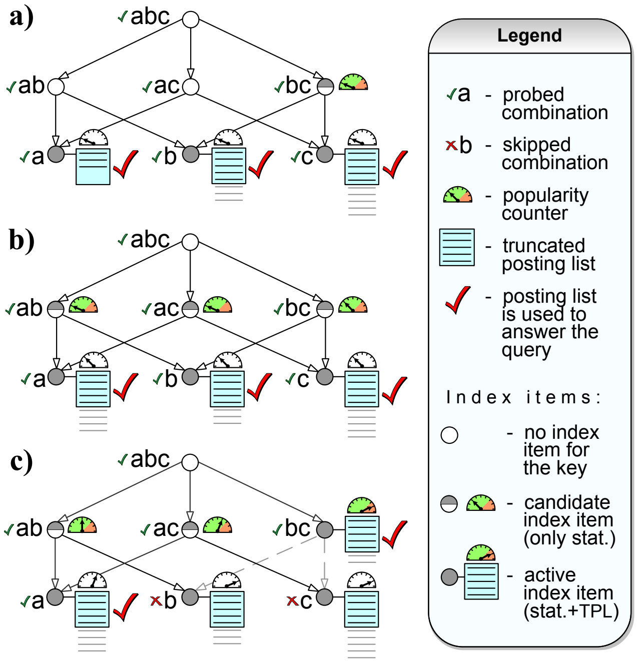

Figure 1: Examples for processing the query abc if: a) a posting list for a is truncated while posting lists for b and c are not, b) the posting list for a is also truncated, c) additionally the key bc is indexed.

When either of the two terminal situations is reached, the query lattice exploration stops and all the retrieved (possibly truncated) posting lists are used by the querying peer for postprocessing. More precisely, the peer produces their union, re-ranks all the resulting documents d_j with respect

[2]In a subset lattice a *cut* is a set of nodes N_i, s.t.: 1) the union of all the nodes dominated by N_i is equal to the top node of the lattice; and 2) none of the nodes in the cut dominates any other node from the cut. For example the set of nodes a and bc in Figure 1-c is a cut because: 1) $a \cup bc = abc$ and 2) $a \nsubseteq bc,\ bc \nsubseteq a$.

to the original query q (i.e., it computes the relevance scores $r(d_j, q)$), and presents the top-ranked document references to the user as the result for the submitted query q.

For example, Figure 1 shows the processing of the query *abc* with three different states for the global index. Notice that the tick sign highlights the TPLs that are collected by the querying peer and used to produce the final result for *abc* in each case. The top-k results obtained for the query *abc* in the case 1-c can potentially be of better quality than the ones in 1-b due to the fact that an extra TPL for the key *bc* is available. In the case 1-a, the quality of the result is already maximal because the size of the posting list for a is below DF_{max} and hence it contains all possible document references relevant for a, and therefore for *abc*.

Finally, as a consequence of the query lattice exploration, the querying peer can also discover new candidate keys that have to be created. These candidate keys belong to the set of *immediate ancestors* of all identified query keys that are: 1) indexed and 2) associated with posting lists that contain more than DF_{max} elements[3]. The peer then simply requests the creation of all such candidate index items, provided that they are not already stored in the global index. Recall that the created candidate index items will only maintain their usage frequencies, but can be activated later in case they become popular.

To summarize, the processing of new queries leads to key activations and hence to the generation of new TPLs, which, in turn, increases the retrieval quality for subsequent queries. Obsolete active keys that become unpopular over time due to changes in the user query distribution can also be deactivated, thus constantly adapting the set of active index items stored in the global index to the current user information needs.

2.3 On-demand indexing mechanism

When a key k is activated, the on-demand indexing mechanism is executed by the peer P responsible for k to generate the global TPL $\tau(k)$. As all peers could potentially hold documents containing k, a naïve approach would be to broadcast an indexing request containing k to the whole network. P would then collect the answers and generate the global TPL. Such a naïve approach is obviously quite expensive in terms of bandwidth consumption and can also lead to load balancing problems. Nevertheless, our initial solution described in [20] was based on a carefully optimized version of broadcast, called *opportunistic notification mechanism*, which uses a special P2P-level multicast and various piggybacking techniques. However, despite of many possible improvements that can decrease bandwidth consumption, the broadcast-based solution might not scale well with the network size.

A more "bandwidth-friendly" solution is to store full posting lists for all single term keys together with their truncated TPLs. This of course costs extra storage but does not entail any additional bandwidth overhead, because the construction of the basic single term index already implies all necessary communications that are required to generate the full posting lists. If such full posting lists are available, on-demand indexing can be carried out in a conventional way by intersecting the full posting lists of all single terms contained in the key that is being activated. Any distributed set intersection algorithm, such as the Threshold Algorithm [7], can be used.

The substantial difference with the standard single-term indexing approach is that the intersection operation is not performed on a per-query basis (i.e. frequently), but is executed only once when a new key is activated. Moreover, as the indexing latency is much less crucial than the retrieval latency, we can tolerate a certain delay for the key activation. Notice that full posting lists are used only for key activation, whereas the bounded TPLs are used for query processing, which guarantees a scalable bandwidth consumption during retrieval.

3. SCALABILITY

In this section we briefly discuss the scalability of the query-driven approach in terms of bandwidth consumption. The detailed study and proofs can be found in [14, 20]. In short, the main reasons for scalability are the following: 1) the retrieval traffic generated while processing a query is low, since all transmitted TPLs are of bounded size, and 2) the indexing traffic needed to generate and maintain the global distributed index is manageable, as it depends on the number of indexing keys, which can be adjusted with the QF_{min} parameter.

Retrieval traffic: As answering a query leads, in the worst case, to the exploration of all the nodes in the query lattice that corresponds to query term sets of at most s_{max} terms, the processing of a query requires the transmission of at most $(\log N + 1)\sum_{i=1}^{s_{max}}\binom{|q|}{i}$ messages. Additionally, the re-ranking of the final result can lead to the transmission of at most $DF_{max}\sum_{i=1}^{s_{max}}\binom{|q|}{i}$ extra messages. Thus, as the size of all the transmitted messages is bounded, and, if we assume a bounded query rate for each peer in the network, the total number of transmitted messages grows with $O(N \log N)$. It corresponds to $O(\log N)$ retrieval traffic per peer, which is scalable.

Indexing traffic: The traffic generated to produce and maintain the global distributed index consists of: 1) the *single-term indexing traffic* required to populate the basic single-term index, and 2) the *query-driven indexing traffic* required to generate TPLs for newly activated keys.

Single-term indexing traffic: If we assume that the number of documents published by a peer is bounded, the number of messages that have to be transmitted in the network in order to generate the basic single-term index grows with $O(N)$. As the routing cost to deliver a message to the corresponding peer is $O(\log N)$, the total number of messages to generate the single-term index is $O(N \log N)$, which corresponds to $O(\log N)$ messages a peer has to transfer during the single-term index generation.

Query-driven indexing traffic: Each key activation triggers the on-demand indexing mechanism, which performs the distributed intersection of the corresponding single-term posting lists to generate a new TPL. While the bandwidth consumption of finding top-DF_{max} postings stored in the TPL depends on the chosen algorithm (e.g., see [7, 5]), we can take the worst case complexity as $O(N)$ (messages). The overall query-driven indexing traffic depends on the number of actual activations. In [20] we derived an upper bound for the number of activated keys for a given query log and showed that it scales linearly with the query log size, provided that the popularity of term combinations in the log follows a Zipf law. Thus, the activation rate linearly depends

[3]In other words these candidate keys belong to the set of immediate ancestors of all the nodes in the cut.

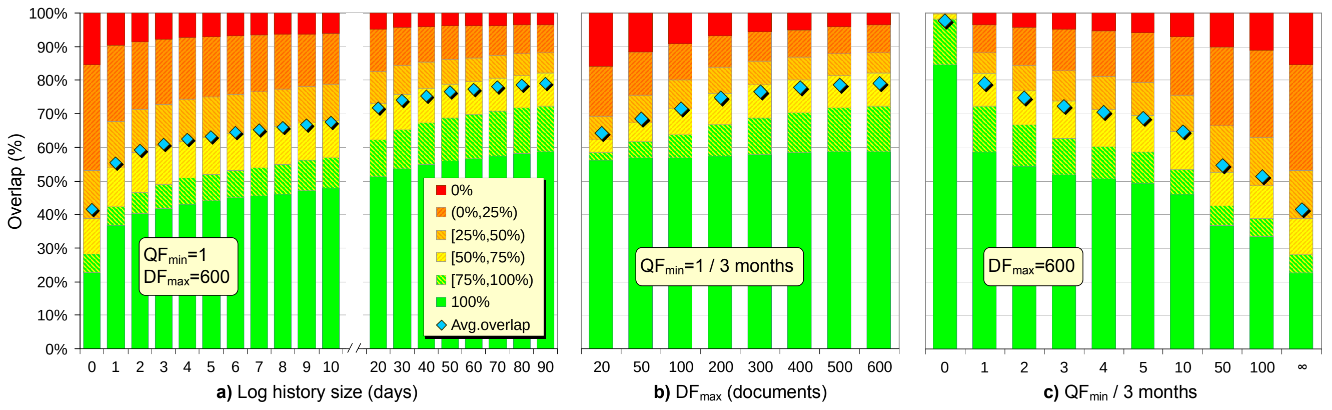

Figure 2: Results of the Google experiment: a) overlap achieved for different sizes of the query log measured in days, b) overlap achieved for different values of DF_{max}, c) overlap achieved for different values of QF_{min}.

on the global query rate. But as the latter grows with $O(N)$ this still corresponds to a scalable traffic in the P2P setup. In addition, the traffic can be further controlled at the price of retrieval quality degradation by tuning the QF_{min} parameter for a given time interval. Our extensive experimental evaluation (see Section 4) shows that the retrieval quality remains acceptable even for web-scale document collections with reasonably chosen time interval and QF_{min} values.

Furthermore, in practice, the bandwidth consumption is further reduced by using the DHT-level congestion mechanism [8] developed by our group. This module takes care of the on-the-fly aggregation of the messages with the same next-hop destination and insures efficient P2P network utilization at peak loads.

4. EXPERIMENTS

In this section, we investigate the retrieval performance of our query-driven approach. Notice that additional experiments focusing on the scalability are presented in [20].

To analyze retrieval performance, we conducted several experiments using query logs from the AOL search engine. These logs contain more than $17M$ web queries collected from $650K$ users during 3 months, from March to May 2006. We discarded the information about user sessions and obtained a large list of queries sorted by timestamps. In addition, we considered only unique entries in each user session, so the repetition of a query by the same user does not affect the query popularity distribution. Finally, we filtered out queries corresponding to web site URLs, which represent a large fraction (about a third) of all queries, but are not interesting for our experiments[4].

4.1 Google experiment

The aim of this set of experiments was to evaluate the impact of query-driven indexing on retrieval quality in the context of real life web-scale document retrieval. To do so, we randomly generated a test set of $2K$ queries taken from the last day of the AOL log. We then crawled Google's top-20 results for each query in the test set. In the following, we will refer to these top-20 results for a query q as the *reference result* for q .

For each test query, we removed the common stop words, applied the stemmer [15], sorted the terms in alphabetical order[5] and generated all possible term combinations. Then, for each of these combinations, we computed their query frequencies using the AOL query log for the previous 3 months and crawled Google's top-DF_{max} results for all generated term combinations. In our query-driven index, these results would be contained in the TPLs associated with indexed term combinations, provided that each peer uses the same ranking mechanism as Google.

To evaluate the retrieval quality for a query q, we measure the *overlap* between the reference top-20 results for q and the union of the TPLs associated with the term combinations (keys) that are: 1) contained in q and 2) indexed. The overlap is expressed as the fraction of the reference top-20 result that appear in the generated union. In other words, the overlap corresponds to the recall when Google considered as the reference.

In all experiments, we categorize the overlap values into the following categories: 1) $overlap = 0\%$, i.e., no result from the reference top-20 appears in the union, 2) $overlap = (0\% - 25\%)$, 3) $overlap = [25\% - 50\%)$, 4) $overlap = [50\% - 75\%)$, 5) $overlap = [75\% - 100\%)$ and 6) $overlap = 100\%$, i.e., all results from the reference top-20 appear in the union.

Impact of the log size: Figure 2-a shows the achieved overlap as the function of the size of the query log, measured in days. QF_{min} was set to 1, i.e., a term combination should be encountered at least once in the previous query history to be activated and indexed. As our measurements have shown that considering all possible combinations of size at most 3 is already sufficient to achieve $> 97\%$ maximum overlap, we set s_{max} to 3. The DF_{max} parameter was set to 600.

With this setting, one can see that, starting from the poor performance of the single term index[6], the overlap rapidly grows with 100-200K queries being processed per day. The three months query log yields an overlap of about 80%. Notice that our approach could use larger query logs, which would further improve the retrieval quality. It is also important to mention that for more than 80% of the test queries at least 10 out of 20 reference results were found, whereas a very low fraction (about 3-4% of the queries) performed poorly returning no reference result. We analyzed the rea-

[4]Such queries can be easily resolved in our framework by treating URLs as single terms.

[5]The sorting is performed to diminish the effect of term ordering on Google's ranking.

[6]Recall that if no query log is available, only single term keys are indexed by default.

Precision	$DF_{max} = 100$				$DF_{max} = 500$				ST-BM25
	$QF_{min}=\infty$	$QF_{min}=5$	$QF_{min}=3$	$QF_{min}=1$	$QF_{min}=\infty$	$QF_{min}=5$	$QF_{min}=3$	$QF_{min}=1$	
P@5	0.306	0.345	**0.347**	0.341	0.345	0.343	0.343	0.343	0.337
P@10	0.266	0.299	0.295	0.294	**0.307**	0.302	0.303	0.302	0.298
P@15	0.237	0.267	0.267	0.267	0.276	0.279	**0.280**	0.278	0.278
P@20	0.212	0.243	0.243	0.246	0.254	**0.259**	**0.259**	**0.259**	0.257
P@30	0.174	0.206	0.209	0.212	0.214	0.221	0.221	0.224	**0.226**
P@50	0.139	0.169	0.171	0.174	0.175	0.181	0.181	0.183	**0.186**
P@100	0.097	0.126	0.127	0.130	0.128	0.135	0.135	0.136	**0.140**
#docRef	236.7	184.6	179.1	173.3	1148.2	880.9	846.2	813.2	193652.4

Table 1: Precision at top-K ranked pages

sons for poor overlap for such queries and identified that in 30-40% of the cases the queries were misspelled. Therefore, if misspells are treated properly, the overlap values would be higher.

From Figure 2-a, we can conclude that taking popular combinations: 1) significantly increases retrieval quality when compared to the single term index (a twice higher overlap with the 90-days query log), and 2) yields an overall satisfactory retrieval quality.

Impact of DF_{max}: We used the 3-month query log, set QF_{min} to 1 and investigated the impact of DF_{max} on retrieval quality. Figure 2-b shows the achieved overlap for different values of DF_{max}.

It is interesting to observe that changing DF_{max} hardly affects the fraction of queries with a 100% overlap, with a growth from 56% to 59% for DF_{max} changing from 20 to 600 respectively. The DF_{max} value however affects the average overlap.

Impact of QF_{min}: Finally, Figure 2-c shows the decrease in retrieval quality when increasing the QF_{min} from 0 (all possible combinations are indexed) up to infinity (basic single term index). As before, s_{max} was set to 3 and DF_{max} to 600.

Based on these results, we can assume that, in practice, the QF_{min} parameter should be chosen in the 5-20 range resulting in a 60%-70% overlap with our settings. In this case, the period during which we keep the query statistics should be increased accordingly.

4.2 TREC experiment

To further evaluate the retrieval quality of our approach, we also used the WT10G collection[7] that contains 1'692'096 documents. 100 test queries were selected from the Ad hoc topics of the Web Track in TREC-9 and 10[8]. We processed title-only queries because queries with additional fields were not used in the real Web search. The standard TREC assessments supplied by the U.S. National Institute of Standards and Technology were used. We used the Terrier[9] engine with the BM25 weighting scheme to compute top-DF_{max} documents stored in the TPLs, and also to compute the final ranked results.

After processing 17M queries from the AOL query log to generate the query-driven index, we submitted the 100 TREC queries to the system. Then we compared our results to the ones returned by the centralized Terrier engine (we denote this by ST-BM25, i.e., single term indexing using the BM25 weighting scheme). DF_{max} was set to 100 and 500. Notice that the DF_{max} parameter is useful to control the trade-off between the retrieval cost and the retrieval quality (the smaller the DF_{max}, the lower the bandwidth consumption during retrieval). QF_{min} was set to 1, 3, 5 and ∞, where $QF_{min} = \infty$ means that no key is activated and only the basic single term index is used to process the queries.

Table 1 shows the achieved precisions at K (P@K). The highest value in each line of the table is highlighted in bold. In general, the results achieved by our system (excluding $QF_{min} = \infty$, $DF_{max} = 100$) are slightly better than ST-BM25 for K≤20. For K>20, our system starts loosing some relevant documents, when compared to ST-BM25, because we only store at most top DF_{max} document references per key. However, we believe this should not be a problem in the context of Web search where users are usually only interested in the top 10-20 documents.

In addition, for K>20, Table 1 also shows that, with a higher value for DF_{max}, our system is becoming similar to ST-BM25 (in fact, if $DF_{max} = |\mathcal{D}|$, our system is equivalent to ST-BM25). In the worst case, when $DF_{max} = 100$ (we only keep top-100 documents in the posting lists) and $QF_{min} = \infty$ (the query driven mechanism is not applied), our system retrieves 75% of the relevant documents retrieved by ST-BM25 at top-50 (0.139/0.186) and 89% at top-10 (0.266/0.298). Notice that these values are already quite high due to the relatively small size of the WT10G collection. In general, the query-driven technique with reasonable values of QF_{min} performs similarly to ST-BM25.

The last line in the table shows the average number of transmitted document references during the processing of the 100 TREC queries, which indicates the bandwidth consumption during retrieval. For ST-BM25, we simulate the naïve approach where the full posting lists are transmitted to the querying peer for each of the terms in the query. Obviously, with smaller values of DF_{max}, we achieve lower bandwidth consumption. Since our posting lists are truncated to a constant size, the bandwidth consumption will remain constant when the size of the collection increases, as shown in [14].

Finally, the TREC experiment confirms the conclusion that the query driven indexing approach indeed delivers a retrieval quality that is fully comparable to the one of a centralized single-term index, and, at the same time, guarantees a scalable traffic during retrieval.

5. RELATED WORK

A number of solutions for text-based retrieval in decentralized environments have been proposed in the literature. They are based on either unstructured [6], hierarchical [11, 2], or structured P2P networks [4, 13] populated with *peer-level* collection descriptions to facilitate the peer-selection process followed by document-level retrieval from the se-

[7]http://ir.dcs.gla.ac.uk/test_collections/wt10g.html
[8]TREC Web Track, http://trec.nist.gov/data/webmain.html
[9]Terrier search engine, http://ir.dcs.gla.ac.uk/terrier/

lected peers. Such solutions depend on the quality of peer-level descriptors and in general perform well for clustered content when a small subset of peers holds documents relevant to a given query. Additionally, it has been recognized in [13] that when taking into account term co-occurrences to identify promising peer-level index entries associated with term combinations, the peer selection process and corresponding retrieval performance are largely improved. However, while performing well in a small network, it is unclear whether the retrieval quality will remain acceptable in a large-scale setting due to the peer granularity of the index.

Query-driven solutions have been used to improve the peer-selection process in hierarchical P2P networks [2, 12]. In [2] a super-peer backbone network maintains the information about good candidates for answering a query based on past queries, while [12] improves resource selection by modeling past user behavior to direct the search into the adequate part of the network. In general, the main problem of hierarchical P2P solutions is the increase of the number of generated messages during retrieval with the size of the network in order to maintain acceptable retrieval quality.

In contrast to peer-level solutions, *document-level* indexing has mainly been applied in structured P2P networks [16, 21, 22]. Since large posting lists are the major concern for such solutions, both [16] and [21] have proposed top-k posting list joins, Bloom filters, and caching as promising techniques to reduce search costs for multi-term queries. However, a recent study [23] shows that single-term indexing is practically unscalable for web sizes even when sophisticated protocols are combined to reduce retrieval costs. Therefore, the pSearch system [22] proposes another approach that places documents onto a DHT network according to their semantic vectors produced by Latent Semantic Indexing (LSI) in order to reduce document dimensionality and guarantee solution scalability. However, as semantic vectors have to be defined a priori, the method cannot efficiently handle dynamic scenarios and adapt to changing collections. A query-driven indexing method at document granularity has recently been proposed in [10]. However, the solution is based on single-term indexing and does not consider indexing with term combinations. Finally, the Distributed Cache Table approach [18] also uses the idea of dynamic indexing with term combinations tailored to the query distribution, but is rather suitable for middle-size P2P text retrieval systems as it relies on local postprocessing of possibly large posting lists.

Contrary to existing approaches, we index documents with popular term combinations extracted from user queries and continuously update the index adapting it to user interests. Our solution assumes a random distribution of documents over peers and performs document-level indexing in a structured P2P network. It is the continuation of our efforts to design scalable solutions for full-text P2P search. While in [14] we have concentrated on minimizing the generated retrieval traffic, our query-driven solution initially outlined in [19], additionally reduces the corresponding indexing traffic. We have shown in [20] that the proposed solution is scalable, while the present paper is essentially focused on the evaluation of retrieval performance.

6. PERSPECTIVES FOR P2P-IR

As Internet based document access is becoming widespread, the field of textual information retrieval is also undergoing a strong change related to the progressive emergence of a novel distributed and decentralized retrieval paradigm based on P2P architectures. In our view, this new paradigm is not only concerned with the well defined scalability problem, but also with more general issues such as the efficient management of heterogenous information sources and user-centric approaches to information retrieval. In this perspective P2P retrieval offers a viable alternative to existing Web search engines, and supports interesting novel usage scenarios. In such scenarios, we assume that each member of the P2P network contributes documents to a global document collection and invests a part of its local computing resources (storage, CPU, bandwidth) to maintain a fraction of a global P2P index. This investment is rewarded by the network-wide accessibility that the global P2P search engine provides for the local documents by making them globally searchable. In addition, as the retrieval of a document always remains under the control of the peer that owns it, the approach opens interesting perspectives for business models based on direct rewarding of the original content providers instead of "information brokers", such as the current major centralized search engines.

More precisely, we identified several distinguishing features for P2P-based information retrieval:

Scalability: A P2P network of 0.5M peers (which we believe to roughly corresponds to the number of servers used by a large scale centralized search engine), within which each peer would handle only 50K documents, should be able to index a document collection of around 25 billions pages. This number, according to unofficial estimates, approximately corresponds the document collection size that major search engines currently index. Notice that 0.5M peers is fairly small when compared to the 4 millions peers in the eMule P2P network[10] or to the 100 millions Skype users[11], revealing a great potential of a large scale P2P network in terms of scalability.

Heterogeneity: Standard centralized search engines use pull mechanisms to populate their indices: they periodically crawl the Internet, extract textual elements from the acquired Web pages and build their indexes from these elements. Such uniform and centralized processing implies that some specific indexing features[12] might get lost unless they are specifically supported by the search engine, which requires a substantial centralized effort. In contrast, P2P-IR systems use push mechanisms: peers decide themselves which documents they want to make globally searchable and, more importantly, how these documents should be indexed. Thus, the effort of handling heterogenous data is distributed in the network and can be managed more efficiently. Such a scenario is therefore appropriate for the management of heterogeneous, frequently changing document collections. For instance, a specialized digital library could continue to use its own sophisticated means to index/query local documents, while using a P2P-based IR infrastructure as a common search framework that makes its specialized indexing/retrieval means available to the whole P2P network.

Provider-centric vs. broker-centric approach: Major search engines play a central role as information brokers,

[10]http://en.wikipedia.org/wiki/Emule

[11]http://about.skype.com

[12]E.g., complex gene names in bioinformatic collections, or formulas in math or chemistry related sites.

but not so much as information creators. Therefore, IR infrastructures allowing novel business models based on the direct rewarding of original content providers might be considered. Within this perspective, P2P-IR systems exhibit interesting characteristics: namely, as already mentioned, in such systems only the document indices are published in the network, while the documents (and thus the associated added value) always remain under the control of the original provider (peer). The peers can therefore decide about the conditions upon which the document access will be granted (e.g., free access, micro-payment, subscription, remunerated advertizement, etc.).

Community-based search: A P2P-IR client is an easy to install software that requires only limited resources from the hosting system. A P2P-IR network is therefore quite simple to deploy, as it does not require anyone taking the responsibility of setting up a centralized server (or network of servers). Consequently, as soon as P2P-IR clients become widely available, building topical communities sharing a document collection within a given domain should become very simple. Thus, as the emergence of such topical communities in fact corresponds to an implicit structuring of the global Web-scale document collection, this opens an interesting possibility to fight the unavoidable precision drop associated with the growth of any document collection.

7. CONCLUSION

Using a structured P2P network for distributing the load among a large number of interconnected nodes represents a promising approach for indexing very large document collections, but poses serious challenges on the design of the distributed index in order to remain scalable with respect to bandwidth consumption, storage space and load balancing at indexing and retrieval.

This paper makes the following contributions to the P2PIR area: 1) it describes a novel query-driven indexing strategy based on indexing of popular term combinations that guarantees scalable storage and bandwidth requirements; 2) it provides the scalability analysis, based on both theoretical results and experimental evaluations, that shows the viability of our approach for web-scale document collections; 3) it reports the experimental evaluation of the P2P information retrieval performance for real web-size document collections and query logs.

The provided theoretical analysis and experimental results indicate, that at the price of a marginal loss in recall for rare queries, the generated index and network traffic remain manageable even for web-size document collections. Furthermore, our experimental evaluation shows that the retrieval precision achieved for a random set of real queries is fully comparable to the one obtained with a state-of-the-art centralized query engine.

As a future work, we plan to continue to optimize the prototype used for our experiments in order to process even larger document collections allowing us a more detailed analysis of the characteristics specific to peer-to-peer information retrieval. In addition, a computationally more efficient prototype will also allow us to better explore the parameter space of our model in order to better understand the impact of each of the parameters, and to achieve a more optimal tradeoff between the different constraints that these parameters are associated with. The most recent version of our P2P prototype is available at http://globalcomputing.epfl.ch/alvis.

Acknowledgement: The work presented in this paper was (partly) carried out in the framework of the EPFL Center for Global Computing and supported by the Swiss National Funding Agency FER as part of the European projects BRICKS (507457) and ALVIS (002068).

8. REFERENCES

[1] K. Aberer, L. O. Alima, A. Ghodsi, S. Girdzijauskas, S. Haridi, and M. Hauswirth. The Essence of P2P: A Reference Architecture for Overlay Networks. In *P2P*, 2005.

[2] W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden. DL Meets P2P - Distributed Document Retrieval Based on Classification and Content. In *ECDL*, 2005.

[3] W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden. Progressive Distributed Top-K Retrieval in Peer-to-Peer Networks. In *ICDE*, 2005.

[4] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Improving Collection Selection With Overlap Awareness in P2P Search Engines. In *SIGIR*, 2005.

[5] P. Cao and Z. Wang. Efficient Top-K Query Calculation in Distributed Networks. In *PODC*, 2004.

[6] 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.

[7] R. Fagin, A. Lotem, and M. Naor. Optimal Aggregation Algorithms for Middleware. In *PODS*, 2001.

[8] F. Klemm, J.-Y. L. Boudec, and K. Aberer. Congestion Control for Distributed Hash Tables. In *NCA*, 2006.

[9] J. Li, B. Loo, J. Hellerstein, F. Kaashoek, D. Karger, and R. Morris. The Feasibility of Peer-to-Peer Web Indexing and Search. In *Workshop on Peer-to-Peer Systems*, 2003.

[10] Y. Li, H. V. Jagadish, and K.-L. Tan. Sprite: A Learning-Based Text Retrieval System in DHT Networks. In *ICDE*, 2007.

[11] J. Lu and J. Callan. Federated Search of Text-Based Digital Libraries in Hierarchical Peer-to-Peer Networks. In *ECIR*, 2005.

[12] J. Lu and J. Callan. User Modeling for Full-Text Federated Search in Peer-to-Peer Networks. In *SIGIR*, 2006.

[13] S. Michel, M. Bender, N. Ntarmos, P. Triantafillou, G. Weikum, and C. Zimmer. Discovering and Exploiting Keyword and Attribute-Value Co-occurrences to Improve P2P Routing Indices. In *CIKM*, 2006.

[14] I. Podnar, M. Rajman, T. Luu, F. Klemm, and K. Aberer. Scalable Peer-to-Peer Web Retrieval with Highly Discriminative Keys. In *ICDE*, 2007.

[15] M. F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980.

[16] P. Reynolds and A. Vahdat. Efficient Peer-to-Peer Keyword Searching. In *Middleware*, 2003.

[17] S. E. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, and M. Lau. Okapi at TREC. In *TREC*, 1992.

[18] G. Skobeltsyn and K. Aberer. Distributed Cache Table: Efficient Query-Driven Processing of Multi-Term Queries in P2P Networks. In *P2PIR*, 2006.

[19] G. Skobeltsyn, T. Luu, I. Podnar Žarko, M. Rajman, and K. Aberer. Query-Driven Indexing for Peer-to-Peer Text Retrieval (poster). In *WWW*, 2007.

[20] G. Skobeltsyn, T. Luu, I. Podnar Žarko, M. Rajman, and K. Aberer. Query-Driven Indexing for Scalable Peer-to-Peer Text Retrieval. In *Infoscale*, 2007.

[21] T. Suel, C. Mathur, J.-W. 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.

[22] C. Tang, S. Dwarkadas, and Z. Xu. On Scaling Latent Semantic Indexing for Large Peer-to-Peer Systems. In *SIGIR*, 2004.

[23] J. Zhang and T. Suel. Efficient Query Evaluation on Large Textual Collections in a Peer-to-Peer Environment. In *P2P*, 2005.