Beyond Term Indexing: A P2P Framework for Web Information Retrieval

Ivana Podnar, Martin Rajman, Toan Luu, Fabius Klemm and Karl Aberer
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland

Keywords: information retrieval, peer-to-peer overlay networks

Received: December 05, 2005

Web search over peer-to-peer (P2P) networks shows promise to become an alternative to the state-of-the-art search engines since P2P overlays offer means for decentralized search across widely-distributed document collections. However, the design of effective techniques for P2P indexing and retrieval raises a number of technical challenges due to potentially unscalable resource (e.g. bandwidth, storage) consumption.
The paper presents a framework for full-text information retrieval in structured P2P networks and introduces a novel retrieval model based on highly discriminative keys*—terms and term sets appearing in a restricted number of documents—that ensure efficient and scalable retrieval. Our goal is to design scalable techniques for building a* global key index *in structured P2P overlays for large document collections. We present experimental results that show acceptable indexing and retrieval costs while the retrieval quality is comparable to standard centralized solutions with BM25 relevance computation scheme.*

Povzetek: Razvito je P2P ogrodje za internetne iskalnike.

1 Introduction

Web search over P2P overlay networks has the potential to become an alternative to current Web search engines due to its decentralized nature and favorable scalability properties. Contemporary Web search engines are in essence centralized and require a central coordination service, which, even when replicated, has been identified as a major system bottleneck [5]. P2P overlays are self-organizing networks for resource sharing that require no central coordination. Moreover, they require minimal infrastructure and maintenance, and enable higher diversity in contents and search methods, which makes P2P Web search appealing from a business perspective [4].

However, there is an ongoing debate on the feasibility of P2P Web search for scalability reasons. In [9] it is shown that the naïve use of P2P networks is practically infeasible for Web search, since the generated traffic exceeds the available capacity of the Internet. Thus different schemes have been devised to make P2P Web search feasible. Several approaches target at a term-to-peer indexing strategy, where the unit of indexing are peers rather than individual documents [7, 3]. Hierarchical solutions for resource discovery have also been proposed where a backbone P2P network maintains a directory service which routes queries to peers with relevant resources, i.e. documents [2, 11]. At this point little evidence is available whether these approaches can scale to very-large scale P2P Web search engines.

In this paper we introduce a general framework for information retrieval (IR) over structured P2P networks which keeps indexing at document granularity while ensuring scalable retrieval costs. Our assumption is that peers are cooperative and provide documents for indexing that will become searchable through a global index. At the same time, they offer computing and storage resources to build and maintain the global index and the underlying P2P network. Peers choose independently the indexing features and related posting lists for their local document collection and, for this task, combine their local knowledge and the global knowledge maintained by the P2P overlay. A peer basically advertises its local document collection and collaborates with other peers by taking the responsibility for a part of the global indexing space. Note that a single peer cannot impose the rules used at a global level, but a malicious peer might impede the search performance by providing fictive descriptions of its local collection.

The framework assumes an underlying distributed hash table (DHT) which maintains a global *key-to-document index*. The approach is characterized by the following central idea: We carefully select the indexing keys so that they consist of terms and term sets that are *rare* and thus *discriminative* with respect to a global document collection. As such, they appear in a limited number of documents, thus limiting the size of the posting lists associated with the keys in the global inverted index. This directly addresses the main problem identified in [9] for structured P2P Web search, namely the processing and transmission of extremely large posting lists. The achieved key-based indexing procedure results in an extremely efficient retrieval because the posting lists associated with the keys in the global P2P index are short and correspond to precomputed and therefore readily available results for potential multiple term queries.

Many possible techniques can be considered for creating

a rare key vocabulary, but the major issue to cope with is the fact that such a vocabulary can easily become unmanageable in size because it theoretically grows with $2^{|V|}$, where V is the single-term vocabulary. Therefore, we present a particular instantiation of our key-indexing which creates keys by combining terms appearing in well-defined contexts in a limited number of documents. We hereafter refer to such keys as highly discriminative keys (HDKs). Our experiments show the following: The growth of the HDK vocabulary per peer and the size of the global index per peer are logarithmic when increasing the global document collection by adding new peers that contribute with their local documents to the global collection. More importantly, the average posting list size remains constant when increasing the global document collection size, which bounds the generated traffic during retrieval. In addition, the observed retrieval performance is comparable to the one achieved with a single-term centralized model using BM25 (Okapi) as relevance computation scheme. Our indexing scheme therefore represents a contribution toward scalable P2P Web search engines that opens the opportunity to index and search virtually unlimited document collections, well beyond the capacity of today's best centralized solutions.

The rest of the paper is structured as follows: Section 2 gives an overview of existing strategies for P2P information retrieval and introduces our concept of indexing with HDKs. Section 3 presents the P2P framework for building a global full-text index in structured P2P overlays with scalable retrieval costs and provide details about the HDK-based retrieval model. In Section 4 we analyze its performance through experimental evaluations using our truly-distributed P2P-IR prototype. The related work in the area of P2P information retrieval is revisited in Section 5, and we conclude the paper in Section 6.

2 Web information retrieval over P2P overlays

There are two main strategies for designing P2P search engines in the area of IR: The first strategy uses unstructured/hierarchical P2P networks where peers maintain local indexes of their document collections, and the second strategy builds a global index in structured P2P networks.

In a P2P network with N peers and a global document collection D, the first strategy divides D over the peer network, and each peer P_i maintains the index of its local document collection D_i. Such indexes are in principle independent, and a naïve solution broadcasts a query to all the peers generating an unscalable number of messages. To limit the query traffic, more advanced approaches use e.g. random walks [12] or answer the query at two levels, the peer and document level. At the first level a group of peers with relevant document collections is located, while at the second level the query is submitted to relevant peers that produce answers by querying their local indexes. The answers are subsequently merged to produce a single ranked answer set. A number of different approaches for the initial peer selection process have been proposed in the context of unstructured [7], hierarchical [2, 10], and structured P2P overlays [3].

The second strategy splits the global document index over a structured P2P network that maintains a global DHT mapping of each indexing feature to the related posting list [15, 18]. Structured P2P overlays offer efficient data inserts and searches [1]: $insert(key, data)$ routes the data to the peer responsible for the given key, and $search(key)$ retrieves the data for the given key. Such networks typically guarantee the routing latency of $O(logN)$ number of hops, while the routing information maintained by each peer is bounded by $O(logN)$.

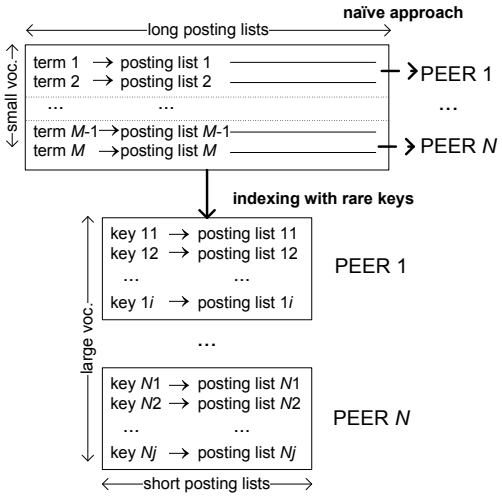

Figure 1: The basic idea of indexing using rare keys

The naïve approach splits the global index over a peer network as depicted in the upper part of Figure 1 and makes each peer responsible for a disjoint set of indexing terms from the global vocabulary. While solving the issue of storing very large indexes, such an approach is practically infeasible and unscalable because of extremely large posting lists for frequent terms. The *key-based indexing* approach directly addresses this issue by limiting the size of the posting lists as shown in the lower part of Figure 1. The key index only contains single terms and term sets that are *rare* and thus *discriminative* with respect to a document collection. Notice that the extension of the index entries to rare terms and term sets, while limiting the size of the posting lists, entails the expansion of the key vocabulary. However, the distributed nature of the global index can provide a virtually unlimited storage space if the number of peers is large enough, but only in case the growth of the key vocabulary size and the corresponding index size is scalable.

We define a key k from the set of keys K as a set of terms $\{t_1, t_2, \ldots, t_s\}$. These terms are elements of the document collection indexing vocabulary V appearing in a single document $d \in D$. The number of terms comprising a key is bounded, i.e. $1 \leq s \leq s_{max}$.

The quality of a key k for a given document d with respect to indexing adequacy is determined by its *discriminative power*. To be *discriminative*, a key k must be as specific as possible with respect to the document d it is associated with and the corresponding document collection D [17]. In this perspective, we categorize a key on the basis of its *global document frequency* (DF), and define a threshold DF_{max} to divide the set of keys K into two disjoint classes, rare and frequent keys. If a key k appears in more than DF_{max} documents, i.e. $DF(k) > DF_{max}$, the key is *frequent*, and has low discriminative power. Otherwise, k is *rare* and specific with respect to the documents it is associated with in the document collection.

The rareness property is in line with the capacity constraints of P2P networks that are capable of storing and transmitting posting lists of limited size. However, the set of rare keys, although bounded for a limited document collection size, might be extremely large. It is therefore important to select among the potential key candidates those that have favorable properties for the retrieval performance of the P2P search engine. Additionally, it is important to understand that keys can be interpreted as discriminative user queries that can be answered very efficiently because the global index already stores precomputed answers associated with such queries.

3 P2P framework for information retrieval with HDKs

In this section we present a general framework for full-text retrieval in a P2P environment using key-based indexing. The framework is designed for building a global key-based index in structured P2P networks and takes into account capacity constraints of P2P networks, such as available bandwidth and storage. Resource management is performed at two levels, *locally* as a peer builds an index of its local document collection to be subsequently inserted into the global index, and *globally* as the peer maintains a part of the global index. Note that a single peer can simultaneously perform tasks at both levels.

3.1 Indexing

Indexing can be divided into the following three steps:

1. local selection of the vocabulary,
2. local selection of postings, and
3. global selection of postings.

The two initial steps represent local efforts of a single peer to choose suitable indexing features for its local document collection since each peer is competing with others for a part of the global index space. The third step is performed by all the peers maintaining the global index.

As the example in Figure 2 shows, P_i builds the key vocabulary K_i for its local document collection D_i. In the next step it chooses a set of postings associated with each key according to the global resource constraints it is aware of, and subsequently inserts the $(key, posting\ list)$ pairs into the P2P overlay. In the given example, P_j is responsible for three such pairs from P_i. Concurrently, two other peers P_k and P_l have built a vocabulary for their document collections, and have started inserting $(key, posting\ list)$ pairs into the P2P overlay. In the third step P_j chooses postings to be stored in the global index, e.g., it builds a posting list for k_x using the postings received from P_i, P_k and P_l.

We will now describe in more details each of the three indexing steps.

3.1.1 Vocabulary selection

A peer chooses locally rare keys because it assumes they are good indexing candidates that are potentially globally rare. A peer can also put frequent keys into the key vocabulary if they are strongly representative of the documents they are associated with (as measured by standard IR weighting schemes). The goal of this phase is to select discriminative keys that meet the following criteria: the selected keys can achieve good retrieval quality, they are likely to appear in user queries, and the key vocabulary does not contain redundant keys.

Possible key filtering mechanisms for selecting a 'good' key vocabulary are the following:

Proximity. The *proximity filter* uses textual context to reduce the size of the rare key vocabulary, and retains keys only composed of terms appearing in the same textual context, e.g., a phrase, a paragraph, or a document window of a given maximal size. The main argumentation behind this approach is that words appearing close to each other in documents are good candidates to appear together in a query. The analysis presented in [16] reports the importance of text passages that are more responsive to particular user needs than full documents. A similar reasoning is used in a recently proposed method for static index pruning [8] which indexes 'significant sentences', i.e. phrases appearing in similar contexts.

Redundancy. A key k is *semantically adequate* for a document d if there is a high probability that a user produces k when searching for d. As the user is more likely to generate generic terms in a query, a semantically adequate key contains a specific combination of quite generic and frequent terms. In other words, the redundancy filter ensures that all term subsets of a key are frequent keys. Note that all supersets of rare keys are redundant, as they only increase the vocabulary without improving retrieval performance.

Top-k keys. As it is possible to compute relevance between a document and a key, we can choose for each document the top-k most relevant keys. The number of chosen keys can, for example, depend on document

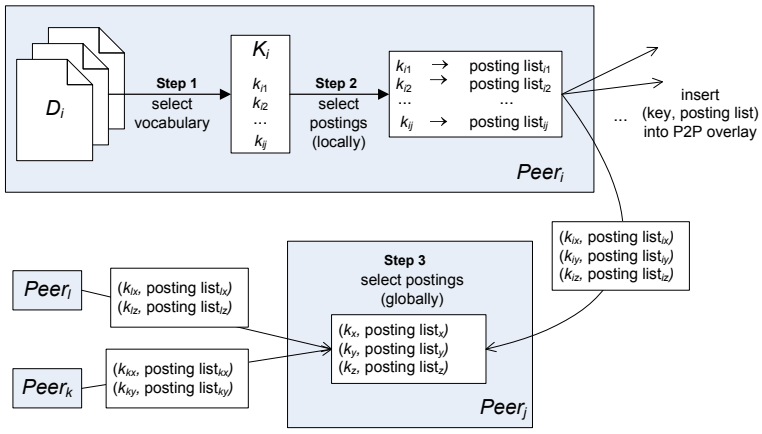

Figure 2: The framework for P2P indexing

length and be controlled by a relevance threshold. The key vocabulary size can also be limited to a predefined value.

Query-adaptive indexing. Keys are coined by combining terms appearing in past user queries.

Note that the above mentioned techniques can be combined to reduce the key vocabulary size. Our current prototype uses the proximity and redundancy filter, and computes HDKs by combining terms appearing within a document window of maximal size w while ensuring that all key subsets are globally frequent. We also extend the key vocabulary with *frequent keys* to improve retrieval performance. The top-k indexing of frequent keys is feasible because our analysis shows that the size of the frequent key vocabulary is less than 1% of the HDK vocabulary size.

Computing HDKs. We compute HDKs of increasing size because a peer needs to know s-level globally frequent keys to compute $(s+1)$-level keys since these are candidates for expansion with additional keys. An indexing peer first categorizes terms from V as locally rare and locally frequent single-term keys, and publishes their local DFs in the P2P network. In our current implementation the indexing peer needs to be informed if keys from its local document collection are *globally* frequent. Locally frequent keys are necessarily globally frequent, but locally rare keys may still be globally frequent. As the indexing peer has no local knowledge about the global key DFs, it relies on the P2P network which maintains the global statistics and notifies the indexing peer if a key considered as locally rare is globally frequent. With such a notification mechanism in place, each indexing peer will have a consistent knowledge about globally frequent keys for its local document collection. A globally frequent key is expended by another frequent key with a restriction that their terms occur within a predefined window as defined by the proximity filter. The indexing peer counts the number of documents in which this property holds to determine local DF of a newly created key, and categorizes it as locally rare or frequent. Then it publishes the key and its DF into the P2P network. The process continues until s_{max}-term keys are created.

3.1.2 Local selection of postings

In this phase, a peer decides which documents are indexed for its key vocabulary. Since there are capacity constrains that restrict the number of postings associated with a key, each peer is allowed to insert only a limited number of postings, e.g. controlled by a globally defined value DF_{max}. Firstly, a peer can insert all postings for locally rare keys, and the P2P overlay then decides globally about the set of postings that get stored. Secondly, a peer can be notified when its already published locally rare keys become globally frequent, and using this global knowledge provided by the P2P network, insert postings only for globally rare keys, which will reduce the indexing traffic compared to the first solution. Thirdly, for locally and globally frequent keys, a peer can publish top-DF_{max} representative postings by ranking documents based on their relevance with respect to a given key.

Our current prototype implements the second and third option: Indexing peers publish all local postings for globally rare keys. To improve retrieval performance and efficiently answer queries composed of frequent terms and term combinations, we have decided to index also globally frequent keys and maintain only DF_{max} postings in the global index. Indexing peers therefore also insert top-DF_{max} postings associated with their local keys that are globally frequent.

3.1.3 Global selection of postings

This phase takes place at the peer responsible for storing and maintaining the posting list associated with a given key. The total index size maintained by the peer is has to comply with its available storage space, which in turn also influences the maximum number of postings associated with a key. Furthermore, the posting list size is restricted to meet capacity constraints imposed by the retrieval phase.

For globally rare keys peers store all received posting lists because rare key are by definition associated with short posting lists. If there are more postings than the constraints allow, the responsible peer has to select a set of postings to keep as it is already done locally for frequent keys. The mechanisms for choosing DF_{max} postings can range from a simple random selection procedure, to sophisticated ranking of postings using available techniques for content and link-based ranking. Our prototype currently stores all received posting associated with globally rare keys and top-DF_{max} ranked postings associated with frequent keys, i.e. documents with highest relevance to a frequent keys according to TF-IDF scores.

3.2 Retrieval

The retrieval part of our framework deals with the problem of finding, given a query $Q = \{t_1, t_2, \ldots, t_q\}$, the corresponding relevant keys in the global P2P index, retrieving the postings associated with those keys, and ranking of the result set. It is performed in two steps:

1. mapping of a query to keys and subsequent retrieval of postings from the P2P overlay, and

2. merging of the received answers with a global ranking scheme in order to produce a single hit list.

Figure 3 shows the querying process when peer P_i receives the query Q. We assume Q is a simple set of terms, and it must be mapped to a set of keys present in the P2P global index. All term subsets of a query are possible key candidates that may be stored in the P2P index, and P_i needs to explore the lattice of term combinations and check which key candidates are indeed keys from the global index.

The optimistic strategy is to start with the largest possible term set (marked as level-1), limited either by the query size q (as it is assumed in Figure 3) or the maximal number of terms forming a key (s_{max}). P_i will try to retrieve the keys sequentially by exploring initially level-1 keys, and then depending on the result of the previous step continue with level-2 keys, ..., level-q keys. The perfect situation occurs when $k_{11} = \{t_1, t_2, \ldots, t_q\}$ is an HDK, in other words, a user has posed a good discriminative query for the indexed document collection: The posting list is readily available and is simply retrieved from the global index by issuing a single request $search(k_{11})$ to the P2P network. Indeed, this will not happen with all user queries. Therefore, level-2 set of potential key candidates of size $q - 1$ is explored. If the P2P network stores postings associated with level-2 keys that cover all terms in Q, the hit list is the union of the retrieved postings. However, it is possible that either (a) no level-2 keys exist in the index, or (b) that some $t_i \in Q$ are not covered by the retrieved keys. In case of (a), P_i explores all level-3 keys and subsequently higher level keys until finally reaching level-q. In case of (b), P_i explores level-3 keys but only those that contain non-covered terms from the previous levels. Depending on the retrieved set of keys, the procedure is extended to higher-level keys until all $t_i \in Q$ are covered by the retrieved key set. The resulting answer set is the union of postings associated with the retrieved keys.

However, in some cases the answer set may be empty because all key candidates are frequent. A valid option is to notify a user that the query is non-discriminative with respect to the document collection, and offer support for query refinement. Another possibility is to apply additional measures and, for example, index frequent keys or perform query expansion that maps terms to semantically related terms which enables the creation of many key-candidates and increases the probability of finding relevant keys in the global index. As our prototype currently indexes frequent keys, all single-term keys from V can eventually be retrieved, which solves the problem of queries that map to frequent keys.

Let us again consider the example in Figure 3. The P2P network does not contain k_{11}, and P_i searches for level-2 keys. Out of q possible level-2 keys, only k_{21} is an HDK and its posting list is retrieved from P_l. However, t_q is still not covered, and P_i unsuccessfully explores keys on higher levels containing t_q, until finally reaching level-q where a posting list is associated with k_{qq}.

Both posting lists are merged by a simple union procedure in the second step. Note that there is a possibility that t_q appears in some documents associated with k_{21}, but as it does not appear within the predefined window, k_{11} was not produced during the indexing procedure. Note also that documents associated with k_{qq} are those with highest relevance to t_q, but they may also contain other terms from Q, also outside the window. The ranking function produces the final ranked result set giving higher scores to documents with higher relevance to Q. The peer uses available ranking methods, either content based, and/or link-based, to produce the final ranking of the merged result set. The ranking procedure must be implemented in a distributed fashion using global document collection statistics such as global DFs available from the P2P index.

4 Experimental evaluation

Experiments have been performed using our prototype P2P search engine which is built on top of the P-Grid P2P layer [19]. Our implementation is a fully-functional P2P search engine that integrates a solution for distributed maintenance of global key vocabulary with associated document frequencies, calculates HDKs used for indexing in a completely distributed fashion, and stores posting lists in a global index for computed keys. The presented experiments analyze both rare and frequent keys associated with top-DF_{max} postings, and the corresponding index sizes.

Experimental setup. The experiments were carried out using a subset of news articles from the Reuters corpus[1].

[1] http://about.reuters.com/researchandstandards/corpus/

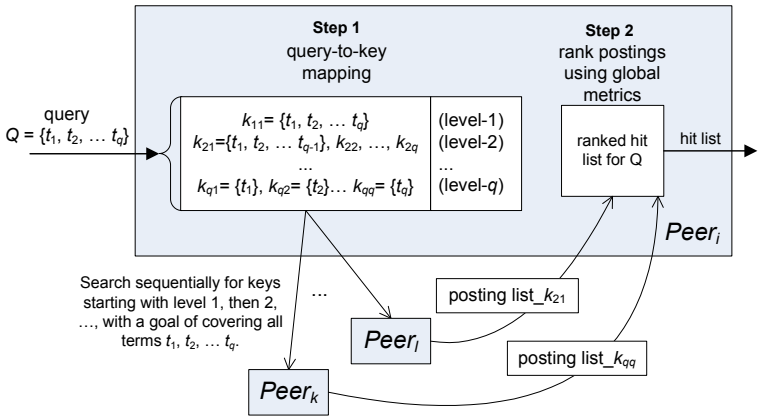

Figure 3: The framework for P2P retrieval

The documents in our test collection contain between 70 and 3000 words, while the average number of terms in a document is 170, and the average number of unique terms is 102. To simulate the evolution of a P2P system, i.e. peers joining the network and increasing the document collection, we started the experiment with 4 peers, and added additional 4 peers at each new experimental run. Each peer contributes with 5000 documents to the global collection, and computes HDKs for its local documents. Therefore, the initial global document collection for 4 peers is 20,000 documents, and it is augmented by the new 20,000 documents for each experimental run. The maximum number of peers is 24, hosting in total the global collection of 120,000 documents. The experiments were performed on our campus intranet. Each peer runs on a Linux RedHat PC with 1GB of main memory connected via 100 Mbit Ethernet. The prototype system is implemented in Java.

Performance analysis. Experiments investigate indexing and retrieval costs in terms of bandwidth and storage consumption, and compare retrieval performance of our P2P search engine to a centralized engine. All documents are pre-processed: First we removed 250 common English stop words and applied the Porter stemmer, and then we removed 100 extremely frequent terms (e.g. the term 'reuters' appears in all the news). DF_{max} is set to 250 and 500, and s_{max} is 3. We have decided to build keys with a maximum of 3 terms having observed that an increase of s_{max} substantially increases the computational load without significant improvement of retrieval performance. In particular this is beneficial for P2P Web search engines because short queries are the ones typically used for Web retrieval. We set the window size to 20 words for the proximity filter as this is the average length of phrases in the collection.

Figure 4 shows the growth of the HDK vocabulary per peer maintained in the global index when increasing the number of documents by adding new peers. Note that each peer contributes an equal number of documents into the global collection. As expected, an increased value of DF_{max} reduces the key vocabulary size because there are

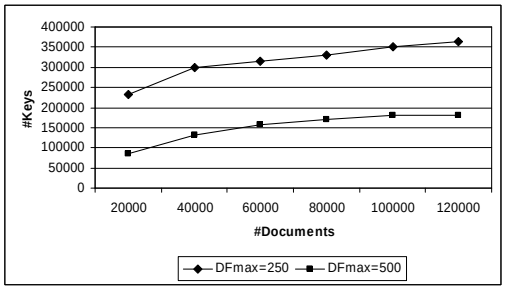

Figure 4: HDK vocabulary per peer

less frequent single-term keys to build 2 and 3-term rare keys. Both experimentally obtained result sequences show logarithmic growth and are expected to converge to a constant value for large document collection sizes. We can conclude that a peer will maintain a negligibly increasing number of HDKs when increasing the document collection size by adding new peers. Therefore, our experiments show that the size of the total key vocabulary maintained in the global P2P index will increase almost linearly with the number of documents for large document collections. The number of keys is significantly larger compared to the single-term vocabulary that grows with $\sqrt{x}$ (Heaps law), but we expect benefits in terms of retrieval efficiency and bandwidth consumption.

Figure 5 compares the average posting list size in case of HDK and single-term indexing. Surprisingly, the average posting list size of the HDK index remains constant and much smaller than DF_{max}, although we only limit the maximum size of posting lists. It is superior to single-term indexing that exhibits a linear increase of the average posting list size. This finding has a major impact on the bandwidth consumption during retrieval as it is shown in Figure 6 that depicts the average number of retrieved postings per query and compares single-term to the HDK approach with $DF_{max} = 250$ and $DF_{max} = 500$.

Since a query log is not available for the Reuters corpus,

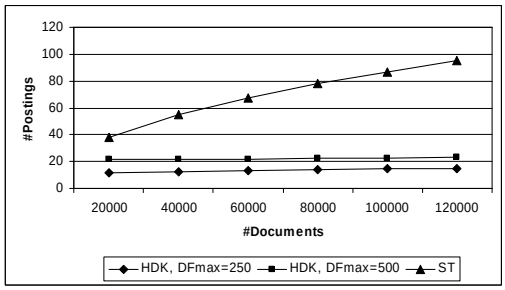

Figure 5: Average posting list size

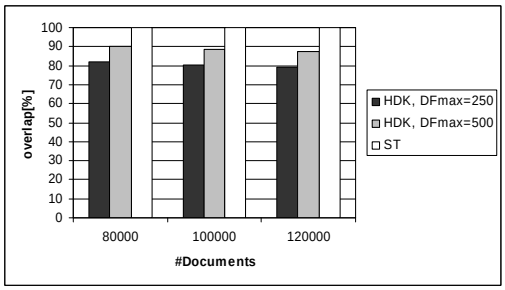

Figure 7: Overlap on top 20

we have preprocessed a Wikipedia query log[2] to generate 1,000 queries. From all available queries we have initially extracted 1,000,000 queries that have more than 20 hits from the Reuters corpus. The resulting 1,000 queries were selected randomly from this set. The generated queries contain on average 3.14 terms, with a minimum of 2 and maximum of 8 terms. Note that longer queries are favorable for our search engine because the probability of finding precomputed rare keys in the P2P index increases. Single term queries would generate bounded traffic since only DF_{max} postings are retrieved and are therefore not considered.

Figure 6 shows an enormous reduction of bandwidth consumption per query of the HDK-based approach compared to the naïve single term indexing: 92% for $DF_{max} = 500$ and even 96% for $DF_{max} = 250$. In addition, the reduction increases when increasing the number of documents in the global collection which practically demonstrates the effect of the bounded number of index postings to bandwidth consumption during retrieval.

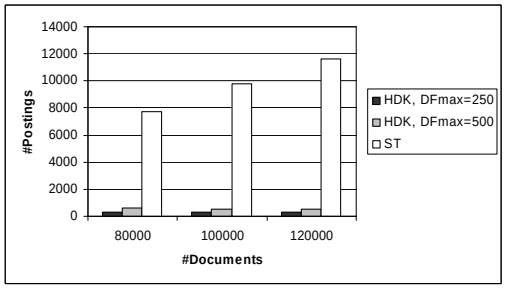

Figure 6: Number of retrieved postings per query

The essential question remains whether the retrieval performance of the HDK approach is satisfactory and comparable to centralized counterparts. Due to the lack of relevant judgment for the generated query set, we have compared the retrieval performance to a centralized engine[3] with BM25 relevance computation scheme which is currently considered as one of the top performing relevance schemes [13].

[2]www.wikipedia.org, query log 08/2004 and 09/2004

[3]Terrier search engine, http://ir.dcs.gla.ac.uk/terrier/

Figure 7 presents the overlap on top-20 retrieved documents retrieved by our system and the Terrier search engine. We are interested in the high-end ranking as typical users are often interested only in the top 20 results. The comparison shows significant and satisfactory overlap between the retrieved result sets. As expected, the retrieval performance is better for larger DF_{max} as we are approaching single-term indexing (when DF_{max} equals vocabulary size, all HDKs are single-term). There is obviously a trade-off between retrieval quality and bandwidth consumption of our indexing strategy because an increased value of DF_{max} results in an increased bandwidth consumption during retrieval, while on the other hand, offers retrieval performance that better mimics centralized engines. However, as required bandwidth is the major obstacle for retrieval, it is vital to choose an adequate value for DF_{max} taking into account available network capacity. There is another limiting factor influenced by this parameter: Note that the process of building the key vocabulary is less expensive in terms of computational costs and bandwidth consumption for an increased DF_{max}. This is intuitively clear because for small values of DF_{max}, the number of potential terms for building rare keys increases, and the whole process creates more precomputed keys, which is obviously very efficient during the retrieval phase if such term sets, i.e. keys, appear in user queries.

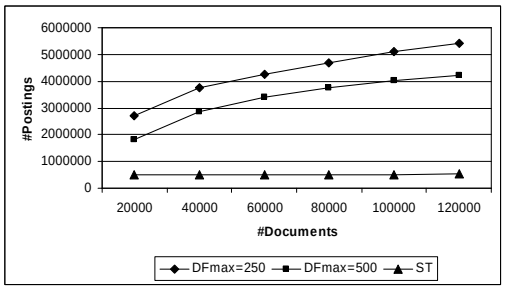

Figure 8: Number of postings per peer (index size)

To quantify indexing costs and the influence of DF_{max} on required storage and bandwidth consumption, Figure 8 shows the average number of postings stored per peer for $DF_{max} = 250$, $DF_{max} = 500$, and the single-term index.

This is the actual index size per peer. The curve is logarithmic for HDKs as it is mostly influenced by the key vocabulary size, and is linear for the single-term index (the decrease of the term vocabulary size per peer compensates for the increased posting list size). It is visible that a peer hosts significantly more postings for the HDK approach, and the index size increases when decreasing DF_{max}. The indexing costs are still feasible and profitable because the gains in terms of bandwidth consumption during the query phase are huge, and compensate for the increased indexing costs. Nevertheless, we plan to investigate further techniques to reduce the index size, and study the influence of DF_{max} on bandwidth consumption during indexing and retrieval, and on the resulting retrieval performance.

5 Related Work

To the best of our knowledge, [6] is the only published P2P framework for information retrieval that is based on distributed semantic indexing on top of structured P2P networks, and therefore highly related to our work. However, it uses a different indexing approach which relies on semantic locality to cluster documents with similar semantics, and to store them on nearby peers.

A number of solutions have been proposed to cope with the scalability problem of P2P information retrieval. Recent approaches combine global knowledge with local indexes: For example, PlanetP [7] gossips compressed information about peers' collections in an unstructured P2P network, while MINERVA [3] maintains a global index with peer collection statistics in a structured P2P overlay to facilitate the peer selection process, and implements a method which penalizes peers holding overlapping document collections. A similar approach for resource selection is presented in [10, 11] in the context of hierarchical P2P networks where special directory nodes route queries to appropriate peers having high chances of answering a query. A recent solution builds an index dynamically following user queries [2]. A super-peer backbone network maintains the information about good candidates for answering a query, while peers answer the queries based on their local document collections. Since large posting lists are the major concern for global single-term indexing, both [15] and [18] have proposed top-k posting list joins, Bloom filters, and caching as promising techniques to reduce search costs for multi-term queries.

The listed solutions are orthogonal to our approach since they use different assumptions to reduce network traffic. However, our approach is not the only indexing strategy that uses term sets as indexing features. The set-based model [14] indexes term sets occurring in queries, and exploits term correlations to reduce the number of indexed term sets. The authors report significant gains in terms of retrieval precision and average query processing time, while the increased index processing time is acceptable. In contrast to our indexing scheme, the set-based model has been used to index frequent term sets, and has been designed for a centralized setting.

6 Conclusion

We have presented a P2P framework for information retrieval that uses a novel retrieval model based on global indexing of rare keys in structured P2P overlay networks. Rare keys are terms and term sets occurring in a limited number of documents, limiting thus the size of posting lists stored in the global index. This approach directly address the unscalable network traffic cased by large posting lists for single-term indexing. The experimental results have shown the potential of our approach in preserving a retrieval quality (top-k precision) comparable to the standard single term with BM25 scoring scheme with enormous reduction of generated traffic during retrieval. Next, it gives evidence that the indexing process is feasible because it produces the key vocabulary and global index of manageable size. More importantly, the average posting list size remains constant when increasing the global document collection size. Therefore, the generated traffic during retrieval remains bounded, which solves one of the major obstacles for scalable IR in P2P networks.

Acknowledgements

The work presented in this paper was carried out in the framework of the EPFL Center for Global Computing and supported by the Swiss National Funding Agency OFES as part of the European FP 6 STREP project ALVIS (002068).

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 *Fifth IEEE International Conference on Peer-to-Peer Computing*, pages 11–20, 2005.

[2] W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden. Dl meets p2p - distributed document retrieval based on classification and content. In *9th European Conference on Research and Advanced Technology for Digital Libraries, (ECDL)*, pages 379–390, 2005.

[3] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Improving collection selection with overlap awareness in P2P search engines. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '05)*, pages 67–74, 2005.

[4] W. Buntine, K. Aberer, I. Podnar, and M. Rajman. Opportunities from open source search. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 2–8, 2005.

[5] F. Cacheda, V. Plachouras, and I. Ounis. A case study of distributed information retrieval architectures to index one terabyte of text. *Inf. Process. Manage.*, 41(5):1141–1161, 2005.

[6] Y. Chen, Z. Xu, and C. Zhai. A scalable semantic indexing framework for peer-to-peer information retrieval. In *SIGIR 2005 workshop: Heterogeneous and Distributed Information Retrieval*, 2005.

[7] 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 *12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12)*, June 2003.

[8] E. S. de Moura, C. F. dos Santos, D. R. Fernandes, A. S. Silva, P. Calado, and M. A. Nascimento. Improving Web search efficiency via a locality based static pruning method. In *WWW '05: Proceedings of the 14th International Conference on World Wide Web*, pages 235–244, 2005.

[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 *Peer-to-Peer Systems II: 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 207Ű–215, 2003.

[10] J. Lu and J. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM '03)*, pages 199–206, 2003.

[11] J. Lu and J. Callan. Federated search of text-based digital libraries in hierarchical peer-to-peer networks. In *Advances in Information Retrieval, 27th European Conference on IR Research (ECIR)*, pages 52–66, 2005.

[12] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *16th International Conference on Supercomputing*, June 2002.

[13] V. Plachouras, B. He, and I. Ounis. University of Glasgow at TREC2004: Experiments in Web, Robust and Terabyte tracks with Terrier. In *Proceeddings of the 13th Text REtrieval Conference (TREC 2004)*, 2004.

[14] B. Pôssas, N. Ziviani, J. Wagner Meira, and B. Ribeiro-Neto. Set-based vector model: An efficient approach for correlation-based ranking. *ACM Trans. Inf. Syst.*, 23(4):397–429, 2005.

[15] P. Reynolds and A. Vahdat. Efficient Peer-to-Peer Keyword Searching. *MiddlewareŠ03*, 2003.

[16] G. Salton, J. Allan, and C. Buckley. Approaches to Passage Retrieval in Full Text Information Systems. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 49–58, 1993.

[17] G. Salton and C. Yang. On the specification of term values in automatic indexing. *Journal of Documentation*, 4(29):351–372, 1973.

[18] 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. *Proc. Int'l Workshop Web and Databases (WebDB '03)*, pages 67Ű–73, 2003.

[19] The P-Grid Consortium. The P-Grid project, 2005. http://www.p-grid.org/.