Building a peer-to-peer full-text Web search engine with highly discriminative keys

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

Abstract

Web search engines designed on top of peer-to-peer (P2P) overlay networks show promise to enable attractive search scenarios operating at a large scale. However the design of effective indexing techniques for extremely large document collections still raises a number of open technical challenges. Resource sharing, self-organization, and low maintenance costs are favorable properties of P2P overlays in the perspective of large-scale search, but we also face new problems due to potentially huge bandwidth consumption during both indexing and querying, as well as the unavailability of global document collection statistics. Since a straightforward application of P2P solutions for Web search generates unscalable indexing and search traffic, we propose a novel indexing technique which maintains a *global key index* in structured P2P overlays. Keys are highly-discriminative terms and term sets that appear in a restricted number of collection documents, thus limiting the size of the global index, while ensuring scalable search cost. Our experimental results show reasonable indexing costs while the retrieval quality is comparable to standard centralized solutions with TF-IDF ranking. Our indexing scheme represents a contribution toward realistic P2P Web search engines that opens the opportunity to virtually unlimited resources, well beyond the capacity of today's best centralized Web search engines.

Keywords: peer-to-peer information systems, information retrieval, distributed indexing

1 Introduction

Web search over P2P overlay networks has recently become an intensive field of study as this approach bears the potential to become an attractive alternative to current Web search engines, both for technical and economic reasons. Contemporary Web search engines based on large computer clusters are expected to reach soon scalability limits. Recently it has been argued that the required centralized coordination service for handling incoming queries, even if replicated, is a major system bottleneck [1]. On the other hand, P2P overlay networks have no central coordination service, and, as such, are promising candidates for next-generation search engines. Moreover, P2P Web search is appealing from an economic perspective [2]. It allows higher diversity in contents and search methods, and supports community-oriented publishing and search of Web content. P2P networks require minimal infrastructure and maintenance, and potentially provide unlimited resources provided that enough peers join the network.

However, there is an ongoing debate on the feasibility of P2P Web search for scalability reasons. In [3] it is shown that the naïve use of unstructured or structured overlay networks is practically infeasible for the Web, since the bandwidth consumption required for indexing and search exceeds the available bandwidth in 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 [4, 5]. Hierarchical solutions have also been proposed where a backbone P2P network maintains a directory service which routes queries to peers with relevant content [6, 7]. At this point little evidence is available whether these approaches can scale to very-large scale P2P Web search engines.

We argue that the design of an effective indexing technique for P2P Web search is still an open challenge. A solution to the problem should be characterized by at least the following three properties: Firstly, it should support general retrieval queries, in particular multi-term queries. Secondly, it should provide retrieval performance comparable to centralized search engines. And thirdly, it should scale to very large numbers of peers, i.e. millions of peers as in contemporary content sharing networks. Our goal is to develop a solution that achieves these properties.

In this paper we introduce a novel indexing approach based on building a *global key-to-document index* that maintains indexing at document granularity. Our approach is characterized by the following key idea: We carefully select the keys used for indexing so that they consist of terms or of multiple terms that are *rare* and thus *discriminative* with respect to a global document collection. More precisely, keys are precomputed term joins that may be considered as selective queries for the global document collection. As such, they appear in a limited number of documents limiting thus the size of a key's posting list stored in the global index. This directly addresses the main problem identified in [3] for structured P2P Web search, namely the processing and transmission of extremely large posting lists. We use a standard structured overlay network for maintaining a key-to-document index of rare keys, as well as the vocabulary of globally frequent keys.

In this paper we will focus on the scalability aspects of our approach, and compare it to existing indexing strategies. Since scalability issues only come into play with large peer populations, we perform it theoretically to point out the salient properties of our indexing scheme that makes it superior to the alternative solution with respect to search cost scalability. We then present the algorithm for index construction over a structured P2P overlay network. Since an increase in the indexing cost is usually expected for search-efficient solutions, the efficiency of the indexing algorithm is crucial to the feasibility of our approach. In this perspective we carried out an experimental study with our prototype system to gather evidence that our method scales. The experiments show that when the index keys are carefully chosen, the growth of the key vocabulary remains bounded, as well as the size of the global index. In fact, the average posting list size even shows a slight decay with the number of peers and the document collection size increase, which is an important advantage compared to other existing solutions. The observed retrieval performance is comparable to the one achieved with a centralized TF-IDF ranking model. We have also observed that when distributing the computational load for building the key index over a P2P network, the indexing time remains almost constant. Our indexing scheme therefore represents a contribution toward realistic P2P Web search engines that opens the opportunity to virtually unlimited resources, well beyond the capacity of today's best centralized solutions.

In summary, the main contributions presented in the paper are the following:

- we introduce a novel retrieval model using a global key-to-document index that overcomes scalability problems of existing solutions with respect to search costs;
- we present a comparative scalability analysis of existing indexing strategies for P2P information retrieval;

- we design an algorithm for building a global key-to-document P2P index, and present an efficient implementation for maintaining global document collection statistics, namely global document frequencies;
- we report experimental results obtained with our prototype system that demonstrate the practical feasibility of building a global key index.

The paper is structured as follows: Section 2 analyzes existing indexing techniques. In Section 3 we present our retrieval model based on global key indexing, and introduce the concepts used for both indexing and search. A comparative scalability analysis of search costs caused by the index design decisions is given in Section 4. The process of building a P2P index with the algorithms for computing rare keys, and maintaining the global index and vocabulary are defined in Section 5. Section 6 presents experimental setup, and points out performance results of our indexing technique obtained with our prototype system. The related work in the area of P2P information retrieval is revisited in Section 7, and we conclude the paper in Section 8.

2 Existing indexing strategies for P2P search engines

In this section we revise the well-established design solutions for centrally coordinated distributed search engines, namely, federated search engines that independently index local document collections, and search engines maintaining a global inverted index of the entire document collection. The first strategy splits the document collection over a set of hosts that maintain *local inverted indexes*, while the second strategy assigns peers with the responsibility for disjunct parts of the vocabulary thus building a *global inverted index* [8]. We discuss these strategies in the context of P2P, specifically for unstructured and structured P2P networks.

Let us assume a P2P overlay network $P_1, P_2, \ldots, P_N$ sharing a global document collection $D = \{d_1, d_2, \ldots, d_n\}$ with a vocabulary of m distinct terms $T = \{t_1, t_2, \ldots, t_m\}$.

Federated local indexes. This strategy relies on *document-based partitioning* of D over the P2P network: Each peer P_i hosts a part of the document collection $D_i \subset D$ which may span over the whole vocabulary T, and maintains a local index of the form

$$t_k, df(t_k), D_i(t_k) = \{d_{ij} \mid t_k \in d_{ij}, \forall d_{ij} \in D_i\},$$

where $df(t_k)$ is the local document frequency of term t_k in D_i, and $D_i(t_k)$ is the set of documents in the local collection containing t_k. $D_i(t_k)$ is t_k's *local posting list*, and the number of documents in $D_i(t_k)$ is t_k's document frequency, i.e., $|D_i(t_k)| = df(t_k)$. The document partitioning can be done through a document clustering which creates subject-specific document clusters, by dividing the document identifier space, or by feeding each peer with documents provided by independent crawlers. When a user issues a query Q, the original query is broadcasted to all the peers, since each peer is an independent search engine that may host documents relevant to the query. Next, the results are merged, and re-ranked to produce a single hit list which is presented to the user.

The cost of adding a new document in this environment is low because only one of the peers needs to update its local index, although it might be difficult to decide which peer should be responsible for a particular document. On the other hand, the cost of search is substantial because a query needs to be broadcasted to all peers. In addition, the peers have to cope with large numbers of queries since each peer processes all original incoming queries. However the approach reduces the size of the posting lists transmitted in the network since local document collections are of reasonable size.

The strategy is pertinent to unstructured P2P networks [9] where peers typically flood the network with queries to locate the data stored by the neighboring peers. The approach is effective for finding a highly replicated item, but is highly ineffective when searching for a rare item since it generates an enormous number of messages without guarantees of finding the item.

Global single-term index. The second strategy uses *term-based partitioning* of the global index where each peer P_i is responsible for a set of terms $T_i \subset T$. The global index is of the form

$$t_k, DF(t_k), D(t_k) = \{d_j \mid t_k \in d_j, \forall d_j \in D\},$$

where $DF(t_k)$ is the global document frequency of term t_k, and $D(t_k)$ is the set of all documents in the entire document collection that contain t_k. $D(t_k)$ represents t_k's *global posting list*, and $|D(t_k)| = DF(t_k)$. To process a query $Q = \{t_1, t_2, \ldots, t_q\}$, peers maintaining posting lists for $t_1, t_2, \ldots, t_q$ are contacted, and the retrieved posting lists are joined to create a single ranked document hit list.

This approach minimizes the cost of search in terms of the number of queried peers since at most q peers hold the posting lists relevant to a query with q terms. The update cost is high because when inserting a new document with k terms for indexing, at most k peers have to be contacted to update the relevant posting lists. However in a realistic usage scenario, collection updates are less frequent than queries, and therefore the global index offers better query performance than the federated local indexes.

This strategy fits the design of structured P2P networks [10] building a global distributed hash-table. Structured P2P overlays offer key-based routing: $insert(key, data)$ routes the data to a peer responsible for a given key, and $search(key)$ retrieves the data for a given key. Such networks limit the routing latency by $O(logN)$ number of hops, while the routing information maintained by each peer is also limited to $O(logN)$. A naïve approach to distributing the global index, and transforming it to a $(key, data)$ pair, hashes a term to produce a key, and encodes a posting list into a data field. The analysis presented in [3] shows that the naïve approach is practically infeasible due to unacceptable storage and traffic requirements caused by extremely large posting lists for the Internet-scale document collection.

From the given argumentation it is obvious that both strategies face significant scalability limitations. It is therefore necessary to design special algorithms and techniques to implement, and deploy a workable solution for large-scale document collections comparable in performance to centralized systems.

Federated local indexes with a global peer index. A number of existing P2P search engines deals with the limitations of the two strategies by combining them [4, 5, 7, 6]. Peers are independent search engines holding local indexes, while the global information maintained by either peers themselves or special directory nodes enables the *peer selection process*, i.e., routing of queries to peers that are likely to hold documents relevant to a query. The reasoning behind the idea is that the global information about peer collections is smaller than the global document index, and therefore manageable in large-scale systems. Additionally, the querying cost is significantly reduced compared to federated local indexes because each query is sent to a carefully chosen group of peers.

This approach faces a tradeoff between a simplified indexing procedure, and querying efficiency and effectiveness. The querying efficiency is affected by a three step process; firstly, a query is routed to relevant peers, next the peers search their local collections, and subsequently peer answers are merged to produce a single document hit list. The querying effectiveness deteriorates in case of random document distribution over a peer set, and can be improved by document clustering.

3 Global key index

Our indexing approach uses a structured P2P overlay network for maintaining a *global key index* with a limited size of the index posting lists. The main principle is quite intuitive as depicted in Figure 1. We index not only single terms, but also sets of terms that create *keys* occurring simultaneously in documents, and distribute posting lists for such keys over the peers in a structured P2P overlay. The *keys* occur only in a limited number of documents restricting thus the size of the posting list, but also cause the expansion of the *key vocabulary*. Such keys are *rare* and therefore highly discriminative with respect to a document collection. We call such keys *Highly Discriminative Keys* (HDKs), and the corresponding index is an HDK index. Let us now formally define HDKs.

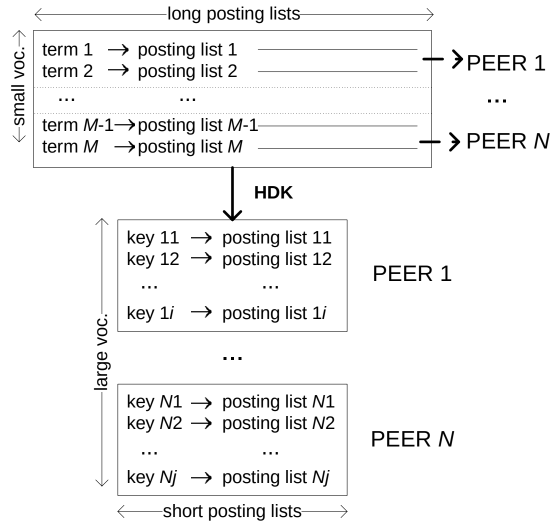

Figure 1: The basic idea of indexing using HDKs

Definition 1. Let K be the set of keys. A key $k \in K$ is a set of terms $\{t_{k_1}, t_{k_2}, \ldots, t_{k_s}\}$ from the vocabulary T appearing in a document $d \in D$. The number of terms comprising a key is bounded, $1 \leq s \leq s_{max}$.

The document collection vocabulary T is a subset of K since elements of T build single term keys.

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 [11]. We categorize a key on the basis of its *global document frequency*, and define a threshold DF_{max} to divide the set of keys K into two disjoint classes, a set of rare and non-rare keys.

Definition 2. $K_{non-rare} = \{k \in K | DF(k) > DF_{max}\}$ is a set of $non-rare$ keys that appear in many documents, and have a low discriminative power.

Definition 3. $K_{rare} = \{k \in K | DF(k) \leq DF_{max}\}$ is set of *rare* keys that appear in few documents and are therefore highly discriminative.

As far as the *discriminative power* is concerned, the interesting keys are in K_{rare}. To reduce the size of indexed keys from the set of rare keys, we introduce in Section 3.1 two filtering methods, the *proximity* *filter*, and *redundancy filter*.

If we examine keys in K_{rare} from the search perspective, term sets building a key may appear in user queries. Actually, such queries are favorable because they represent good discriminative queries since the global index already stores precomputed joined posting lists for such queries. Nevertheless, as the HDK index does not necessarily contain all terms appearing in user queries, we expand queries using *distributional semantics* to improve retrieval performance. Query expansion is presented in Section 3.2.

3.1 Indexing with HDKs

Here we explain the process of building the HDK vocabulary which significantly reduces the number of keys.

Theorem. The growth of the key set size $|K|$ is bounded by $O(|D|)$ if at most s_{max} terms comprise a key, and all documents $d_j \in D$ are bounded in size.

Proof. Since at most s_{max} terms from a document $d_j \in D$ can compose a key, the total number of generated keys from the entire document collection is $|K| = \sum_{j=1}^{|D|} \sum_{i=1}^{s_{max}} \begin{pmatrix} |d_j| \\ i \end{pmatrix}$. Since $|d_j|$ is of bounded size $\forall d_j \in D$, $|K| \leq C\,|D|$, where C is a constant.

However, as the constant C is quite large, it is practically infeasible to build a key index without additional restrictions, and we define methods to limit the key vocabulary size by choosing keys adequate for indexing. Here we introduce two filtering methods, the *proximity filter* and *redundancy filter.* The proximity filter eliminates keys from K_{rare} that do not appear in the same textual context, while the redundancy filter removes keys on the basis of their semantic adequacy.

Proximity filter. The proximity filter uses textual context to reduce the size of K_{rare}, and retains keys built of terms appearing in the same textual context, e.g., a phrase, a paragraph, or a document window of a certain size. The main argumentation is that words appearing close in documents are good candidates to appear together in a query. The analysis presented in [12] reports the importance of text passages that are more responsive to particular user needs than the full document. Similar reasoning is used in a recently proposed method for static index pruning [13] which indexes 'significant sentences', i.e. phrases appearing in similar contexts. Therefore HDKs are keys composed of terms appearing within a document window of size w.

Definition 4. K_{rw} is a set of *rare keys* appearing within a document window of size w.

Redundancy filter. An important quality of a key is its *semantic adequacy.* 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 a user is more likely to generate generic terms in a query, a semantically adequate key contains a specific combination of quite generic terms. In other words, a semantically adequate key is by hypothesis a key, for which all its subsets are non-discriminative and therefore non-rare. Note that all supersets of semantically adequate keys are redundant, as they only increase the vocabulary without improving the retrieval performance. Therefore the filter that removes redundant keys containing other discriminative keys is called the redundancy filter.

Semantically adequate keys that are also rare are good indexing candidates, and we call such keys *intrinsically rare keys.*

Definition 5. K_{i-rare} is a set of intrinsically rare (*i-rare*) keys. A key k is i-rare iff all its constitutive terms and term sets are non-rare.

By applying both the proximity and redundancy filter to rare keys, we obtain a significantly smaller set of HDKs.

Definition 6. K_{irw} is a set of *irw keys*, i.e. intrinsically rare keys appearing within a document window of size w.

Our hypothesis is that all the keys in $K_{irw} \subseteq K_{i-rare} \subseteq K_{rare}$ are both semantically and contextually adequate for indexing a document collection, and these are used when building our HDK index.

Computing HDKs. The relationship between non-rare, rare and i-rare keys within a document window w is depicted in Figure 2. The whole key vocabulary built from terms appearing within a window w is denoted by K_w. It comprises non-rare keys in K_{non-rw} and rare keys in K_{rw}. The i-rare keys in K_{irw} are a subset of K_{rw}. A key k is non-i-rare iff $k \in K_{non-rw}$, or $k \in (K_{rw} \backslash K_{irw})$, i.e., any of its sub-keys is i-rare. Single-term keys are a special case because the proximity filter does not apply to them, and the whole document vocabulary V is a subset of K_w. Moreover, the rare and i-rare sets are equal for single-term keys.

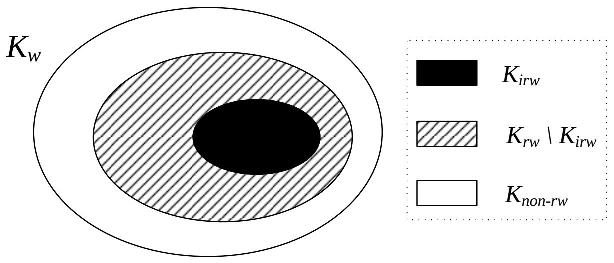

Figure 2: Non-rare, rare, and i-rare keys

We compute HDKs by initially applying the proximity filter to each document in the collection: First all combinations of terms appearing in a window of size w are determined building thus K_w. Next, the redundancy filter is applied which removes all rare keys that are non-i-rare by ensuring that only keys composed of non-rare sub-keys remain in the vocabulary. The redundancy filter significantly reduces the size of the remaining key set since our experiments show that approximately 83% of 2-term keys and 99% of 3-term keys in K_w are in $K_{rw} \backslash K_{irw}$. After redundancy filtering, the remaining keys are from $K_{irw} \cup K_{non-rw}$. To keep only discriminative keys, we need to check whether a key's document frequency is smaller or equal to DF_{max} removing thus keys from K_{non-rw}. And finally, only K_{irw} will remain in the HDK vocabulary. The formal algorithm for computing HDKs in a distributed environment is defined in Section 5.1.

3.2 Retrieval

We will now discuss the problem of finding, given a query $Q = \{t_1, t_2, \ldots, t_q\}, t_i \in T$, the corresponding most relevant discriminative keys $k \in K_{irw}$ that are part of the HDK index. This process is preformed in two phases. In the first phase we extract all the subsets of $s_{max}, s_{max} - 1, \ldots, 1$ terms from the query Q and retrieve the posting lists (if any) associated with the corresponding key. If no document, or not enough documents are retrieved during the first phase, we perform query expansion using distributional semantics. Keys produced during the first phase are potentially discriminative and as such can be found in the HDK index. However, some keys may not be in the HDK index, and relevant documents will not be returned degrading thus retrieval quality. Therefore, we apply query expansion in the second phase in case the first produces unsatisfactory answers to improve the recall performance of our approach.

Distributional semantics. We use a method based on *distributional semantics* which uses a co-occurrence matrix to make a probabilistic connection between the full vocabulary T and T_{irw} (terms building the keys in K_{irw}). With such a co-occurrence matrix, query terms can be associated with terms in T_{irw}, and the associated terms can then be used to build new HDKs increasing the chance of finding relevant documents.

To find HDKs relevant to a query we use probabilistic associations between T and T_{irw}. The basic idea behind distributional semantics [14] is that two terms are semantically similar if their textual contexts are similar. For any term, its textual context can be approximated by the vector of its co-occurrence frequencies with other terms in the vocabulary. In the $N_T \times N_{T_{irw}}$ *co-occurrence matrix* CO, each row, also called co-occurrence profile, represents the normalized co-occurrence frequencies of the associated terms in T_{irw} with a term in T:

$$CO = \begin{pmatrix} co_1 \\ co_2 \\ \vdots \\ co_{N_T} \end{pmatrix} = \begin{pmatrix} co_{11} & co_{12} & \dots & co_{1N_{T_{irw}}} \\ co_{21} & co_{22} & \dots & co_{2N_{T_{irw}}} \\ \vdots & \vdots & \ddots & \vdots \\ co_{N_T 1} & co_{N_T 2} & \dots & co_{N_T N_{T_{irw}}} \end{pmatrix}$$

Usually, we restrict to co-occurrences in a given textual segment, e.g. a window of given size, and the co-occurrence frequencies are computed on a reference corpus that is representative of the domain for which the semantic model is defined. The co-occurrence profile of a term $t_i \in T$ is interpreted as an estimate of the probability distribution that measures the association between t_i and the terms $t' \in T_{irw}$. More precisely, co_{ij} is an estimate for $p(t'_j|t_i)$, the probability that a topic triggered by the term $t_i \in T$ is also triggered by the term $t'_j \in T_{irw}$.

$$p(t'_j|t_i) \simeq co_{ij} = \frac{f(t'_j, t_i)}{\sum_k f(t'_k, t_i)},$$

where $f(t'_j, t_i)$ is the co-occurrence frequency between the two terms t'_j and t_i. In our case, we consider contexts in the form of a window of size w_{co} terms.

Query expansion. The expansion process proceeds as follows: if a query $Q = \{t_1, t_2, \dots, t_q\}, t_i \in T$ does not contain any *irw* keys, we consider the co-occurrence profiles of $t_1, t_2, ...t_q$ to derive the n best terms $t'_1, t'_2, \dots, t'_n \in T_{irw}$ that maximize the following score:

$$score(t'_j, Q) = p(t'_j|t_1, t_2, \dots, t_q)$$

$$= p(t_1, t_2, \dots, t_q|t'_j) \cdot \frac{p(t'_j)}{p(t_1, t_2, \dots, t_q)} = \prod_{i=1}^{q}(p(t'_j|t_i)) \cdot \frac{1}{p(t'_j)^{q-1}},$$

where $p(t'_j) = \frac{\sum_{i=1}^{N_T} f(t'_j, t_i)}{\sum_{j=1}^{N_{T_{irw}}} \sum_{i=1}^{N} f(t'_j, t_i)}$ under the hypothesis that $t_1, t_2, ...t_q$ are independent.

The set of the selected n best terms t' is then considered as an expansion of the original query Q. The expended query $Q \cup \{t'_1, t'_2, \dots, t'_n\}$ is processed as explained earlier on Q itself with a restriction that the explored sets of terms must contain at least one of t'_i.

4 Comparative scalability analysis

To assess the scalability of the presented indexing strategies in terms of retrieval costs, we analyze the average traffic load in the peer network caused by the submitted queries and subsequent answers to them. In other words, we are interested in the average total size of messages submitted to the network at any given time once it has reached a steady state. For this, we assume that in the steady state at any time a fraction r of the N peers are producing a query. We analyze the asymptotic behavior of the system with an extremely large number of peers N joining the network and hosting a large-scale document collection D. We assume each peer contributes at most d_{max} documents to the global document collection, i.e., $|D| = d_{max}N$, and that all documents are of bounded size.

To simplify the analysis, we do not presume any specific routing architecture, and do not analyze the traffic on links interconnecting the peers. We are merely analyzing the quantity of information the network needs to absorb and transmit.

Federated local indexes. If we assume a steady state, there are on average rN query messages, each of them being sent to $(N-1)$ peers, thus generating $rN(N-1)$ messages. The size of each query message is reasonably small, as it is usually just a set of terms. We assume this size is limited by $e\ q_{max}$, where e is a uniform term size, and q_{max} is the maximum number of terms in a submitted query. All $(N-1)$ peers receiving a query will generate an answer to rN peers, and therefore also generate $rN(N-1)$ answer messages. The size of the answer messages is substantially larger than the size of queries, but since each peer hosts a local document collection of at most d_{max} documents, the retrieved hit lists are also of size limited by d_{max}. The total traffic thus grows with $(e\ q_{max} + d_{max})rN(N-1)$ which is not scalable since it is $O(N^2)$.

Global single-term index. Under the same assumption of a fraction of r peers generating query messages at any moment, the maximal number of query messages is $q_{max}rN$, and the generated traffic is bounded by $e\ q_{max}rN$. At most q_{max} contacted peers will generate an answer that is sent to rN peers. However, the size of an answer message can be extremely large because the size of the global collection posting lists can be huge. The average answer message size equals $u\ S/|V|$, where u is the size of a term's single posting, S is the size of the index, and $|V|$ is the vocabulary size. Note that $S/|V|$ is the average posting list size. If we assume that each peer brings a bounded amount of documents in the system, the total number of documents is $O(N)$ and therefore a lower bound for u is $O(log(N))$. The global index size S is $O(|D|) = O(d_{max}N) = O(N)$, which leads to an average posting list size of $O(\sqrt{N})$, if we assume that the vocabulary grows as $O(\sqrt{N})$ (Heaps law). The total answer traffic is therefore $u\ q_{max}\ S/|V|\ r\ N$. Therefore the total generated traffic is

$$(e\ q_{max} + u\ q_{max}\ \frac{S}{|V|})rN, \tag{1}$$

which grows with $O(N\sqrt{N}\log(N))$.

Federated local indexes with a global peer index. In this analysis we assume that a structured P2P network maintains a peer index relating terms to a ranked list of peers, as it is done in [5]. The generated traffic for this approach has to be calculated in two steps. In the first step a list of peers related to a query is retrieved from the P2P overlay generating traffic in the same way as for the global single-term index, except that a posting element is a peer reference instead of a document reference. An average answer message size is still $u\ S/|V|$, where u is a peer reference again bounded by $O(logN)$ because the total number of peers is N. $S/|V|$ is $O(\sqrt{N})$ because the size of the index grows with $O(N)$ with a vocabulary size $O(\sqrt{N})$. Therefore we have the same traffic growth as in the global single-term

Federated local indexes	$O(N^2)$
Global single-term index	$O(N\sqrt{N}\log(N))$
Local indexes with a global peer index	$O(N\sqrt{N}\log(N))$
Global key index	$O(N\log(N))$

Table 1: Retrieval costs for various indexing strategies

index $O(N\sqrt{N}\log(N))$. In the second step rNp_{max} queries are sent to a limited number of peers p_{max}, generating the answer traffic bounded by $p_{max}d_{max}rN$. The second step does indeed grow with $O(N)$, but the first grows with $O(N\sqrt{N}\log(N))$.

Global key index. The reasoning is basically the same as for the global single-term index, except that the global key index limits the average posting list size by DF_{max}, and therefore $S/|V|$ is $O(1)$. We are limiting the average posting list size, accepting, at the same time, an increase of the vocabulary size which according to Theorem 1 is bounded by $O(|D|) = O(d_{max}N) = O(N)$.

Due to query expansion, we have to take into account the increased number of keys associated with a query of size q. Let us denote this number of keys by $n(q)$. The search mechanism defined for the HDK approach is performed in two steps as defined in Section 3.2: Firstly all the term subsets of size at most s_{max} are produced, and secondly, for each of the q terms of the query, the corresponding row of the co-occurrence matrix is retrieved, and the M best key candidates are generated. Therefore, $n(q) = \sum_{i=1}^{s_{max}} (\ q\ )$
$i + M$, and is $O(1)$. The total generated traffic is analogous to eq. 1 $(e\ n(q) + u\ n(q)\ S/|V|)rN$ which is $O(N\log(N))$.

To summarize, the results of our scalability analysis are presented in Table 1. The results show that the global key indexing strategy is superior to other strategies in terms of retrieval costs, and that it has the potential to scale up with a large number on peers and huge document collections. A detailed explanation of the scalability analysis is available in [15].

5 Building the P2P index with HDKs

Our P2P search engine is built of peers extended by the information retrieval (IR) functionality. The peers are part of a structured P2P network as depicted in Figure 3 which is responsible for maintaining a global key index of document collections provided by Web servers, and can accept queries from Web browsers. Web servers and Web browsers are clients of the P2P-IR network, and interact with peers using their remote interface to do the following: to *insert documents* for incorporating them into a distributed index, and to *send queries*. Note that an IR peer is in our case a remotely accessible component completely independent of the document collection source. However, it could be integrated within a Web server, enabling thus the Web server to become part of a Web search engine.

The system has a layered architecture, and layers participating in distributed indexing are magnified in Figure 3. On top of the transport layer is the P2P layer offering services to the HDK layer. The remote interface builds the top layer which is responsible for accepting incoming client requests and for dispatching them to the underlaying HDK layer. The HDK layer computes HDKs for the received set of documents. Since the indexing process is computationally intensive, the peers share computational load and build the HDK vocabulary in parallel. Each peer creates keys for the local document collection it receives from an external source. It inserts local document frequencies for keys it considers locally i-rare

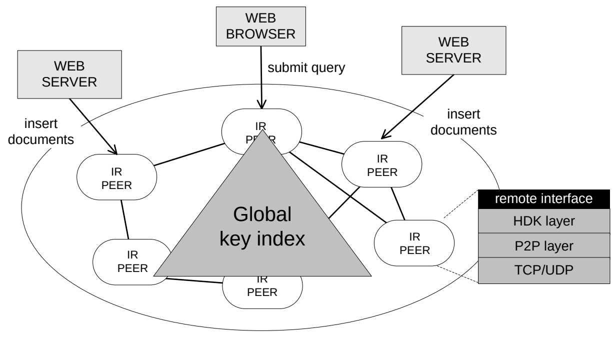

Figure 3: System architecture

and non-rare[1], and inserts posting lists for i-rare keys into the P2P overlay. The P2P layer performs the following tasks: It stores posting lists for globally i-rare keys, maintains global key vocabulary for i-rare and non-rare keys together with their global document frequencies, and notifies the HDK layer when i-rare keys become non-rare due to addition of new documents. The P2P layer is an extended version of the P-Grid system [16] which is an implementation of a trie-structured P2P overlay [17]. We define the algorithm for computing i-rare keys which is performed by the HDK layer, and the algorithm for maintaining the global document index implemented by the peer layer in the sequel.

5.1 Computing HDKs

Each peer computes a set of locally i-rare keys that are potentially also globally i-rare, and maintains locally a set of globally non-rare keys. The peer needs non-rare keys for computing the i-rare keys. Since according to our experiments only a small fraction of keys are non-rare ($< 1\%$ of all keys stored in the global index), this global knowledge does not represent a big overhead for a peer. Certain number of keys are locally non-rare, and therefore also automatically globally non-rare. However, some keys that are locally rare, are globally non-rare, and we use an event-based mechanism which enables a peer to be notified by the P2P layer when keys become globally non-rare. The P2P overlay stores posting lists only for i-rare keys, and maintains global document frequencies for non-rare keys.

Algorithm 1 defines the process for computing i-rare keys locally by peer P_i. The peer stores a set of *potentially i-rare* keys in $K_{irw} = \bigcup_{s=1}^{s_{max}} K_{irw}(s)$, where s_{max} is the maximal number of terms comprising a key. Non-rare keys that are *unquestionably globally non-rare* are maintained in the set $K_{non-rw} = \bigcup_{s=1}^{s_{max}} K_{non-rw}(s)$. Note that a peer can independently decide that a key is globally non-rare only if the key is locally non-rare. The peer cannot decide without global knowledge whether a key is definitely globally i-rare because a locally i-rare key may be globally non-rare. Therefore a peer registers a listener to receive messages from the P2P overlay notifying it when a key regarded as i-rare becomes non-rare.

The algorithm starts by inserting local document frequencies for P_i's document vocabulary V_i into the P2P overlay (lines 6 and 7). Then it classifies terms from the local document vocabulary V_i into the non-rare and i-rare set of single-term keys (lines 8 to 14). We assume the peer has previously created or

[1]To be precise, these are *irw* and non-rare keys in K_w. In the remaining part of the paper we assume i-rare and non-rare keys are in K_w to simplify the explanation.

received a single term index of its local document collection. For all terms $t_k \in V_i$, the algorithm checks $df(t_k)$. If $df(t_k) \leq DF_{max}$, t_k is locally rare and also i-rare, and as such is stored in $K_{irw}(1)$. Otherwise, it is definitely globally non-rare and is added to $K_{non-r}(1)$. If t_k is locally rare, it is potentially globally non-rare. P_i relies on the P2P overlay to notify it if t_k is globally non-rare. Note that these notifications are asynchronous messages, and depend on the same initial classification process and insertion of local document frequencies performed by other peers. We assume this process is performed in parallel, and that notifications will lead peers into a steady state when most peers will have a fairly realistic knowledge about globally non-rare keys. The algorithm proceeds by inserting posting lists for i-rare keys in $K_{irw}(1)$ into the P2P overlay (lines 39 to 41). The approach is tolerant to erroneous insertions of posting lists for non-rare keys: The P2P overlay disregards the received posting list, updates the global document frequency of a key, and notifies a peer that the key is non-rare.

For determining multi-term i-rare keys, the algorithm starts by building an inverted index of multi-term keys since it is not available a priori (lines 19 to 29). Each document must be processed sequentially by creating s-term keys in a window of size w. Next, we filter out redundant keys: In case all sub-keys of a key are non-rare, the key is added to $K_{irw}(s)$, and we update its local document frequency and posting list (lines 23 to 27). In the following step (lines 30 to 37), local document frequencies for keys in $K_{irw}(s)$ are inserted into the P2P overlay, and subsequently, we check whether a key is locally and thus globally non-rare. If the key is globally non-rare, $K_{irw}(s)$ and $K_{non-r}(s)$ are updated accordingly (lines 32 to 35). The algorithms proceeds by inserting posting lists for i-rare keys from $K_{irw}(s)$ into the P2P overlay analogously to the same process for single-term keys. This process is repeated until s equals s_{max}.

5.2 Building the global key vocabulary and rare-key index

The P2P network maintains the following entries

$$key, DF(key), PeerList(key), Posting(key)$$

to store a global key vocabulary and key index. $PeerList(key)$ is the list of peers that have reported $df(key)$, and $Posting(key)$ is the key's global posting list. The $Posting(key)$ equals $null$ in case key is non-rare. Note that the P2P layer distinguishes only rare and non-rare keys based on their global document frequencies.

Algorithm 2 defines the steps a peer P_j takes when maintaining the global vocabulary. Let us assume P_j is responsible for key. When P_i inserts $df(key)$ into the P2P overlay, P_j will eventually receive the following message "(key):$(df(key), P_i)$". P_j first checks whether it already contains a data value for (key). In case such an entry is not available, it initializes a new entry; otherwise it reads an existing local entry (lines 2 to 6). If the global value for $DF(key) > DF_{max}$, key is already non-rare, and P_j updates the value of $DF(key)$. If P_i considers key being rare, P_j sends a message "$key : non{-}rare$" to the originating peer P_i (lines 10 to 13). In case the global value $DF(key) \leq DF_{max}$, key is still considered rare by the peers in the P2P network. If by adding the new local document frequency $df(key)$, key becomes non-rare, P_j needs to notify other concerned peers that key is no longer rare. P_j will do so using the list of peers $PeerList(key)$ (lines 22 to 24). P_i is previously added to the list of peers that need the notification only if it believes key is rare (lines 17 to 19). Next, P_j removes the list of peers in $PeerList(key)$ and the key's posting list to decrease the storage load (lines 25 and 26).

Algorithm 3 defines the steps P_j performs to maintain the global index. When P_j receives a message key:$(Posting_i(key), P_i)$, it first checks whether key is globally rare or not. If key is globally non-rare, P_j needs to notify P_i because P_i still considers key to be i-rare. In case key is still rare, $Posting(key)$ is augmented by the reported posting list $Posting_i(key)$.

Algorithm 1 Computing HDKs at peer P_i

```
for $s = 1$ to $s_{max}$ do
  $K_{irw}(s) \leftarrow \emptyset$
  $K_{non-rw}(s) \leftarrow \emptyset$
  if $s = 1$ then
    /* process single-term keys */
    for all $t_k \in V_i$ do
      P2P.updateDF($key$)
      if $df(t_k) \leq DF_{max}$ then
        /* $t_k$ is locally rare */
        $K_{irw}(s) \leftarrow K_{irw}(s) \cup t_k$
      else
        /* $t_k$ is locally and also globally non-rare */
        $K_{non-rw}(s) \leftarrow K_{non-rw}(s) \cup t_k$
      end if
    end for
  else
    /* create an inverted index for multi-term keys */
    /* process each document to create a set of potential term combinations */
    for all $d_{ij} \in D_i$ do
      /* generate keys by concatenating terms appearing in a window of size $w$ */
      for all $tuple$ of $s$ terms $t_{k_1}, \ldots, t_{k_s}$ in a window of size $w$ do
        $key$ = concat($t_{k_1}, \ldots, t_{k_s}$)
        if checkRedundancy.($key$) then
          $K_{irw}(s) \leftarrow K_{irw}(s) \cup key$
          updateLocalDF($key$)
          updateLocalPostingList($key$, $d_{ij}$)
        end if
      end for
    end for
    for all $key \in K_{irw}(s)$ do
      P2P.updateDF($key$)
      if $df(key) > DF_{max}$ then
        /* $key$ is locally and globally non-rare */
        $K_{irw}(s) \leftarrow K_{irw}(s) \backslash key$
        $K_{non-rw}(s) \leftarrow K_{non-rw}(s) \cup t_k$
      end if
    end for
  end if
  for all $key \in K_{irw}(s)$ do
    P2P.insertPostingList($key$)
  end for
end for
```

Algorithm 2 Maintaining global vocabulary at peer P_j

```
1: upon receiving a message "$key$:$(df(key), P_i)$"
2: if $data(key) = null$ then
3:     $data(key) \leftarrow (DF(key) = 0, PeerList(key) = null, Posting(key) = null)$
4: else
5:     get $data(key)$
6: end if
7: if $DF(key) > DF_{max}$ then
8:     /* $key$ is already non-rare, update it's global DF */
9:     $DF(key) += df(key)$
10:    if $df(key) \leq DF_{max}$ then
11:        /* notify $P_i$ $key$ is globally non-rare */
12:        send message "$key : non-rare$" to $P_i$
13:    end if
14: else
15:    /* $key$ is still rare */
16:    $DF(key) += df(key)$
17:    if $df(key) \leq DF_{max}$ then
18:        $PeerList(key) \leftarrow P_i$
19:    end if
20:    if $DF(key) > DF_{max}$ then
21:        /* $key$ becomes non-rare, notify concerned peers */
22:        for all $P_k \in PeerList(key)$ do
23:            send message "$key : non-rare$" to $P_k$
24:        end for
25:        $PeerList(key) \leftarrow null$
26:        $Posting(key) \leftarrow null$
27:    end if
28: end if
```

Algorithm 3 Maintaining global index at peer P_j

```
1: upon receiving a message "$key$:$(Posting_i(key), P_i)$"
2: get $data(key)$
3: if $DF(key) > DF_{max}$ then
4:     /* $key$ is already non-rare, notify $P_i$ */
5:     send message "$key : non-rare$" to $P_i$
6: else
7:     /* $key$ is still rare */
8:     $Posting(key) \leftarrow Posting(key) \cup Posting_i(key)$
9: end if
```

Note that the algorithms support document additions, but as such are not designed for document deletions. Also $DF(key)$ should be regarded as an approximate global value due to race conditions in a concurrent distributed environment. Our implementation of the global vocabulary with the maintenance of global document frequencies can be adapted for aggregating other global collection-wide statistics, and further details are available in [18].

6 Experimental evaluation

Experimental setup. The experiments presented here were carried out using a subset of news articles from the Reuters corpus[2]. The documents in our test collection contain between 70 and 3000 words, while the average number of terms in a document is 190, 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 20 peers, and added additional 10 peers at each new experimental run. Each peer contributes with 500 documents to the global collection, and computes HDKs for its local documents. Therefore the initial global document collection for 20 peers is 10,000 documents, and it is augmented by the new 5,000 documents at each experimental run. The maximum number of peers is 60 peers hosting in total the global collection of 30,000 documents (90.4 MB, XML encoded). The experiments were performed on our campus intranet. Each peer runs on a SUN Ultra 10 station with 256Mb of main memory connected by a 100 megabit network. Note that computers are located in a common laboratory, and were used by others during the experiments. Thus the experiments were performed in a fairly uncontrolled environment that, to a certain degree, mimics a real-world P2P environment. The prototype system implementing our retrieval model is written in Java.

Performance analysis. The experiments investigate the number of keys generated using our HDK algorithm, and the resulting average posting list size for the postings maintained in the P2P network. The documents were pre-processed: First we removed 250 common English stop words and applied the Porter stemmer, and then we removed extremely frequent terms (e.g. the term ‘reuters’ appears in all the news). The DF_{max} is set to 90 in our experiments, and s_{max} is set to 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 improving the retrieval performance. In particular this is beneficial for the P2P Web search engines because short queries are the ones typically used on the Web. We have set the window size to 20 words because it is the average length of phrases appearing in our test collection.

Figure 4 shows the total number HDKs stored together with their posting lists in the P2P network. The HDK vocabulary grows linearly when we increase the number of peers in the network and thus the global document collection size. Consequently, each peer stores on average a fairly constant number of HDKs, as can be observed in Figure 5. The size of the non-rare key vocabulary maintained by the P2P overlay for the purpose of computing HDKs is less than 1 % of the maintained vocabulary as Figure 6 shows, and indeed represents a modes storage overhead.

Figure 7 shows the average posting list size for the HDK and single-term indexing. The average posting list size of the HDK index exhibits a slight decrease when increasing the number of peers, and thus the global collection size, while the average posting list size of the single-term index increases linearly. This experimentally shows that $S/|V|$ is $O(1)$ for the global key indexing, as we assumed theoretically. With respect to time costs for indexing, we have observed a negligible increase of the total indexing time when we increase the number of peers in the network and keep the document collection size per peer constant.

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

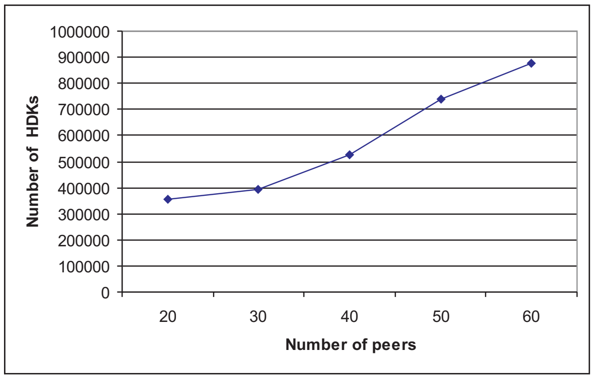

Figure 4: Number of HDKs with respect to the number of peers

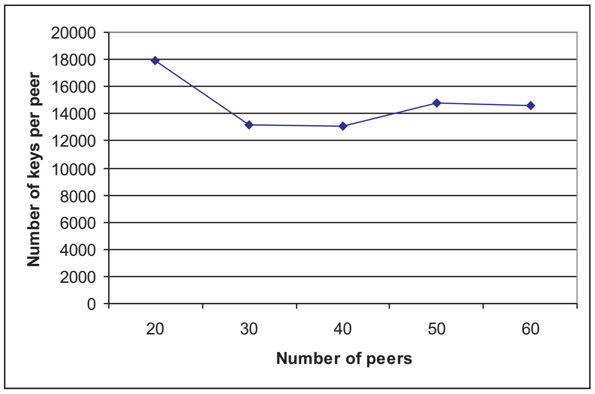

Figure 5: Average number of HDKs per peer with respect to the number of peers

For the retrieval performance evaluations, we have created the total of 200 queries by randomly choosing 2 to 3 terms from news titles. Because of the lack of the relevant judgment for the query set, we compared the retrieval performance of our P2P search engine to a centralized baseline. In particular, we use the Terrier search engine[3], a modular platform for the rapid development of text search engines. To measure the retrieval quality, for each query we take top 20 documents retrieved by our prototype and ranked using TF-IDF, and compare them to the top 20 documents retrieved from the Terrier also using TF-IDF ranking. We compare the Terrier result set $\mathcal{R}^{20}_{ST-TFIDF}$ to the two sets retrieved from our engine: top 20 documents retrieved without query expansion $\mathcal{R}^{20}_{HDK-NoDS}$, and top 20 documents retrieved with query expansion applying distributional semantics $\mathcal{R}^{20}_{HDK-DS}$, which expends the original query set to the maximum of 15 terms. We are interested in the high-end ranking as typical users are often interested only in the top 20 results. Two metrics are used to compare the top 20 result sets: the

[3]http://ir.dcs.gla.ac.uk/terrier/

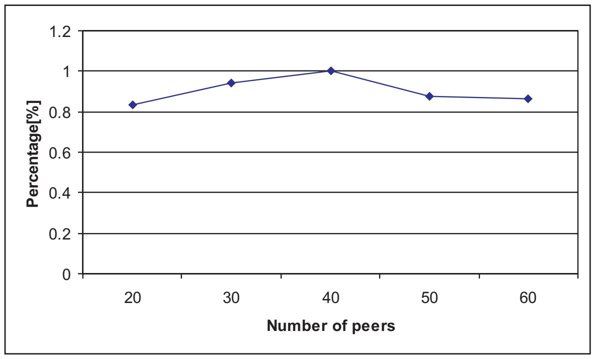

Figure 6: Percentage of non-rare keys in the vocabulary

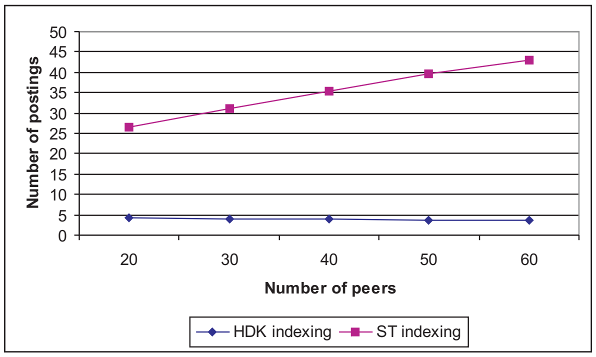

Figure 7: The average posting list size with respect to the number of peers

first one is the overlap between the two sets, and the second one is the average ranking value of the documents in the top hit list sets.

Table 2 compares the retrieval performance for the single-term and HDK indexing, with and without distributional semantics, for the collection of 15,000 documents and 30 peers. The results show that when using distributional semantics for query expansion, the retrieval quality of our retrieval model is comparable to traditional single-term (ST) indexing with TF-IDF ranking. Without query expansion only 157 queries (78.5%) have been answered, and the retrieval performance compared to ST indexing is still acceptable. In addition, Table 2 shows the average size of the longest posting list transmitted per query. As the transmission of posting lists per query are performed in parallel, the longest posting list will determine the overall query response time. The average maximum posting list size for the HDK index is substantially lower compared to single term indexing ($\approx 5\%$), and thus the expected corresponding bandwidth consumption is substantially lower.

	Overlap on top20	Avr. ranking of top20	Max. posting list size
ST-TFIDF	20	10.5	1262.59
HDK-DS	17.47	12.44	71.72
HDK-NoDS	13.98	16.70	31.37

Table 2: Retrieval quality of HDK indexing compared to the centralized TF-IDF system

7 Related Work

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 [4] gossips compressed information about peers' collections in an unstructured P2P network, while MINERVA [5] 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 is presented in [19, 7] 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 [6]. 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 [20] and [21] have proposed top-k posting list joins, Bloom filters, and caching as promising techniques to reduce search costs for multi-term queries. Hybrid global-local indexing proposed in [22] uses a global index of top-k terms extended by a complete list of terms appearing in each top-k posting. The size of posting lists is significantly larger compared to single-term index which makes this solution unscalable, although it is highly efficient for multi-term queries since each query can be answered by contacting a single peer.

The listed solutions are orthogonal to our approach since they use different assumptions to reduce network traffic. As our scalability analysis shows, our method is superior to existing approaches in terms of search cost scalability. Highly related to our approach is a method for static index pruning used in the centralized context to reduce the size of positional index [13]. Although it has a different objective, the method uses similar reasoning to extract 'significant terms' from top-k documents, and uses them to determine the set of 'significant sentences'. From the set of significant sentences, a list of 'highly ranked' sentences is extracted to further reduce the size of the index. A term is indexed only if it appears in highly ranked significant sentences which significantly reduces the single-term index while offering good retrieval performance.

8 Conclusion

We have presented a novel retrieval model for P2P Web search based on global key 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 TF-IDF approach while providing scalability to a scale that centralized systems cannot reach. The scalability analysis shows that our approach is superior to other existing solutions with respect to search costs. The experimental evaluation 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, and even decreases, when increasing the global document collection size. Thus the average maximal posting list size retrieved during the querying phase is substantially lower compared to single-term indexing, as well as the expected overall bandwidth consumption. The main strength of our retrieval model in an almost unlimited resource capacity provided that enough peers join the network.

Our future work remains both theoretical and experimental. We need to perform an in-depth time complexity analysis of the HDK algorithm to explain experimental observations related to indexing time. Next, since our scalability analysis for global indexing uses an assumption that the distribution of terms, i.e. keys, is uniform over queries which is not realistic, we need to perform further analysis based on existing query sets. With respect to experimental evaluation, we will perform experiments with larger and various document collections increasing also the size of the peer network, to confirm existing positive results concerning both the indexing costs and retrieval performance.

9 Acknowledgments

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

[2] Buntine, W., Aberer, K., Podnar, I., Rajman, M.: Opportunities from open source search. In: The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2005). (2005)

[3] Li, J., Loo, T., Hellerstein, J., Kaashoek, F., Karger, D., Morris, R.: On the Feasibility of Peer-to-Peer Web Indexing and Search. IPTPS03 (2003)

[4] Cuenca-Acuna, F.M., Peery, C., Martin, R.P., Nguyen, T.D.: 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), IEEE Press (2003)

[5] Bender, M., Michel, S., Triantafillou, P., Weikum, G., Zimmer, C.: Improving collection selection with overlap awareness in p2p search engines. In: SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM Press (2005) 67–74

[6] Balke, W., Nejdl, W., Siberski, W., Thaden, U.: Progressive distributed top-k retrieval in peer-to-peer networks. In: Proceedings of the 21st International Conference on Data Engineering (ICDE 2005). (2005)

[7] Lu, J., Callan, J.: 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). (2005) 52–66

[8] Ribeiro-Neto, B.A., Barbosa, R.A.: Query performance for tightly coupled distributed digital libraries. In: DL '98: Proceedings of the third ACM conference on Digital libraries, New York, NY, USA, ACM Press (1998) 182–190

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

[10] Aberer, K., Hauswirth, M., Punceva, M., Schmidt, R.: Improving Data Access in P2P Systems. IEEE Internet Computing **6** (2002)

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

[12] Salton, G., Allan, J., Buckley, C.: 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. (1993) 49–58

[13] de Moura, E.S., dos Santos, C.F., Fernandes, D.R., Silva, A.S., Calado, P., Nascimento, M.A.: Improving web search efficiency via a locality based static pruning method. In: WWW '05: Proceedings of the 14th international conference on World Wide Web, New York, NY, USA, ACM Press (2005) 235–244

[14] Rajman, M., Bonnet, A.: Corpora-Base Linguistics: New Tools for Natural Language Processing. 1st Annual Conference of Association for Global Strategic Information (1992)

[15] Rajman, M., Podnar, I., Aberer, K.: Further results on the scalability of P2P information retrieval. Technical report, School of Computer and Communication Sciences, Ecole Polytechnique Fédérale de Lausanne (EPFL) (2005)

[16] The P-Grid Consortium: The P-Grid project (2005) http://www.p-grid.org/.

[17] Aberer, K.: P-Grid: A self-organizing access structure for P2P information systems. Sixth International Conference on Cooperative Information Systems (2001)

[18] Klemm, F., Aberer, K.: Aggregation of a Term Vocabulary for Peer-to-Peer Information Retrieval: a DHT Stress Test. In: Third International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P 2005). (2005)

[19] Lu, J., Callan, J.: Content-based retrieval in hybrid peer-to-peer networks. In: Proceedings of the twelfth international conference on Information and knowledge management. (2003)

[20] Reynolds, P., Vahdat, A.: Efficient Peer-to-Peer Keyword Searching. Middleware03 (2003)

[21] Suel, T., Mathur, C., Wu, J., Zhang, J., Delis, A., Kharrazi, M., Long, X., Shanmugasunderam, K.: Odissea: A peer-to-peer architecture for scalable web search and information retrieval (2003)

[22] Tang, C., Dwarkadas, S.: Hybrid global-local indexing for efficient peer-to-peer information retrieval. In: Symposium on Networked Systems Design and Implementation (NSDI), University of Rochester (2004)