A Peer-to-Peer Architecture for Information Retrieval Across Digital Library Collections ⋆

Ivana Podnar, Toan Luu, Martin Rajman, Fabius Klemm, Karl Aberer

School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
[ivana.podnar, vinhtoan.luu, martin.rajman,
fabius.klemm, karl.aberer]@epfl.ch

Abstract. Peer-to-peer networks have been identified as promising architectural concepts for developing attractive search scenarios across digital library collections. Digital libraries typically offer sophisticated search over their local content, however, search methods involving a network of such stand-alone components are currently quite limited. We present an architecture for highly-efficient search over digital library collections based on structured P2P networks. As the standard single-term indexing strategy faces significant scalability limitations in distributed environments, we propose a novel indexing strategy–*key-based indexing*. The keys are term sets that appear in a restricted number of collection documents. Thus, they are discriminative with respect to the global document collection, and ensure scalable search costs. Moreover, key-based indexing computes posting list joins during indexing time, which significantly improves query performance. As search efficient solutions usually imply costly indexing procedures, we present experimental results that show acceptable indexing costs while the retrieval effectiveness is comparable to the standard centralized solutions with TF-IDF ranking.

1 Introduction

Research in the area of information retrieval has largely been motivated by the growth of digital content provided by digital libraries (DLs). Today DLs offer sophisticated retrieval features, however, search methods are typically bound to a single stand-alone library. Recently, peer-to-peer (P2P) networks have been identified as promising architectural concepts for integrating search facilities across DL collections [1, 2]. P2P overlays are self-organizing systems for decentralized data management in distributed environments. They can be seen as a common media for 'advertising' DL contents e.g. to specialists in a particular area, or to the broader public. We argue that a wide range of topic and genre specific P2P search engines can facilitate larger visibility of existing DLs while providing guaranties for objective search and ranking performance. Note that P2P networks cannot be centrally controlled: Peers are located in various domains requiring minimal in place infrastructure and maintenance.

⋆ 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).

Full-text P2P search is currently an active research area as existing P2P solutions still do not meet the requirements of relevance-based retrieval. It is a challenging problem since search engines traditionally rely on central coordination, while P2P is inherently decentralized. For example, global document collection statistics are not readily available in P2P environments, and naïve broadcast solutions for acquiring such statistics induce huge network traffic. In fact, scalability issues and potentially high bandwidth consumption are one of the major obstacles for large-scale full-text P2P search [3].

In this paper we present an integrated architecture for information retrieval over textual DL collections. We assume DLs are cooperative and provide an index of a representative sample of their collections, or supply documents they want to make searchable through a P2P engine. In this way DLs can choose the content that becomes globally available, which naturally resolves the problems related to restricted crawler access. The architecture accommodates distributed indexing, search, retrieval, and ranking over structured P2P networks by means of a common global document index, and serves as a blueprint for our prototype system ALVIS PEERS, a full-text search engine designed to offer highly-efficient search with retrieval quality comparable to centralized solutions. It is the result of our research efforts within the project ALVIS [1] that aims at building an open-source semantic search engine with P2P and topic specific technology at its core [4].

We propose a novel indexing scheme and design a distributed algorithm for maintaining the global index in structured P2P networks. Our engine indexes *keys*—term sets appearing in a restricted number of global collection documents—while keeping indexing at document granularity. Indexed keys are rare and discriminative with respect to a global document collection. They represent selective queries readily retrievable from the global P2P index, while search costs are significantly reduced due to limited posting list size. As our engine provides highly-efficient search over a global P2P network, the indexing procedure is costly. However, since DL collections are rather static, it is appropriate to invest resources into the indexing procedure and benefit largely from the search performance. We will show experimentally that, as we carefully choose keys, the key indexing costs remain acceptable. The number of indexed keys per peer is nearly constant for large document collections, as well as the average posting list size when we keep the number of documents per peer constant and increase the global collection by adding new peers. The querying bandwidth consumption is substantially smaller compared to single-term indexing, while the observed retrieval quality (top-k precision) is comparable to the standard centralized solutions with TF-IDF ranking. In contrast to the majority of published experimental results that rely on simulations, our experiments have been performed using a fully fledged prototype system built on top of the P-Grid P2P platform [2].

The paper is structured as follows. Section 2 reviews the characteristics of P2P networks in the context of full-text search, while Sect. 3 presents our novel key-based indexing strategy. Section 4 specifies the integrated architecture for P2P full-text search and defines a distributed algorithm for building the key index. Experimental results

[1] http://www.alvis.info/
[2] http://www.p-grid.org/

investigating indexing costs and retrieval performance are presented in Sect. 5. Section 6 briefly covers related work, and we conclude the paper in Sect. 7.

2 Unstructured vs. Structured P2P

There are two main categories of P2P systems, unstructured and structured. In unstructured systems peers broadcast search requests in the network, which works well if used to search for popular highly-replicated content. However, broadcast performs poorly if used to search for rare items as many messages are sent through the network. More advanced approaches restrict the amount of query messages by using random walks [5] or special routing indexes, which maintain content models of neighboring peers in order to determine routing paths for a query [6]. In structured P2P, each peer is responsible for a subset of identifiers id in a common identifier space. Multiple peers may be responsible for the same identifier space to achieve higher reliability. All peers use an overlay routing protocol to forward messages for which they are not responsible. To allow efficient routing, most DHTs maintain routing tables of size $O(log(N))$. Starting at any peer in the network, a message with any destination id can be routed in $O(log(N))$ overlay hops to the peer responsible for id. Structured P2P overlay networks therefore exhibit much lower bandwidth consumption for search compared to unstructured networks. However, they are limited to exact-match key search. Please refer to [7] for a comprehensive analysis of generic P2P properties.

There are two architectural concepts for designing P2P search engines in the area of information retrieval: a) local indexes in unstructured/hierarchical P2P networks, and b) global index in structured P2P networks. The first strategy divides documents over the peer network, and each peer maintains the index of its local document collection. Such indexes are in principle independent, and a query is broadcasted to all the peers in unstructured networks generating an enormous number of messages. To limit the query traffic, the query can be answered at two levels, the peer and document level: The first step locates a group of peers with potentially relevant document collections, while in the second step the query is submitted to the peers, which then return answers by querying their local indexes. The answers are subsequently merged to produce a single ranked hit list. The second strategy distributes the global document index over a structured P2P network. Each peer is responsible for a part of the global vocabulary and their associated posting lists. Queries are processed by retrieving posting lists of the query terms from the P2P network.

3 Our Approach: Indexing Rare Keys

The key idea of our indexing strategy is to limit the posting list size of the global P2P index to a constant predefined value, and extend the index vocabulary to improve retrieval effectiveness. Fig. 1 compares our *rare-key indexing strategy* to the standard single-term indexing approach. It is visible that we trade in an increased index vocabulary for the limited posting list size. As posting lists are extremely large for a single-term index, the process of joining them at query time consumes unacceptable network bandwidth, which makes this approach practically unfeasible. On the other hand, rare-key indexing

offers highly-efficient query performance as we limit the posting list size according to network characteristics and perform posting list joins at indexing time.

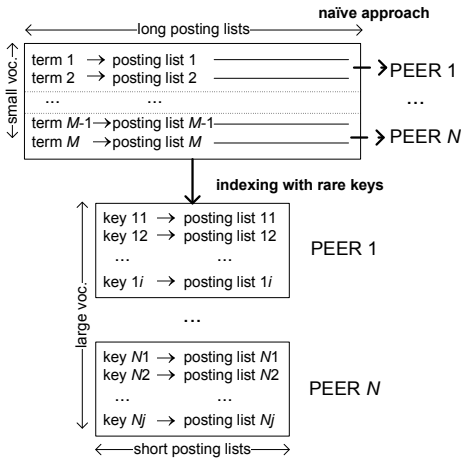

Fig. 1. The basic idea of indexing with rare keys

Let D be a global document collection, and T its single-term vocabulary. A key $k \in K$ consists of a set of terms $\{t_{k1}, t_{k2}, \ldots, t_{ks}\}$, $t_{ki} \in T$, appearing within the same 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 d, and the corresponding document collection D [8]. 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, a set of 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 document collection (it's relevance *idf* score is high).

Although the size of the key vocabulary is bounded for a bounded collection size of limited size documents, there are many term combinations that form potential rare keys, and special filtering methods are needed to reduce the key vocabulary to a practically manageable size. We currently use the proximity and redundancy filter to produce *highly-discriminative keys* (HDKs) indexed by our search engine. *Proximity filter* uses textual context to reduce the size of the rare key vocabulary, and retains keys built of terms appearing in the same textual context—a document window of predefined size w—because words appearing close in documents are good candidates to appear together in a query. The analysis presented in [9] reports the importance of text passages that are more responsive to particular user needs than the full document. *Redundancy filter* removes supersets of rare keys from the vocabulary as such keys are redundant

and only increase the vocabulary size without improving retrieval performance. Therefore, all properly contained term subsets in rare keys are frequent, and we call such keys *intrinsically rare* (i-rare) keys. Proximity filtering strongly depends on the window size and document characteristics. Although it seems intuitive that it would remove most keys, our experiments show the great importance of the redundancy filter which removes many keys after proximity filtering (e.g. 83% of 2-term and 99% of 3-term keys). By applying both the proximity and redundancy filter to rare keys, we obtain a significantly smaller set of HDKs, as reported in Section 5.

As our engine indexes keys, it is essential to map queries to keys for an effective retrieval performance. 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 relevant keys in the HDK index. A perfect situation occurs when $\{t_1, t_2, \ldots, t_q\}$ 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. However, this may not happen with all user queries. Therefore, we use terms and term sets from Q to form potential HDKs. We extract all the subsets of $s_{max}, (s_{max} - 1), \ldots, 1$ terms from the query Q to retrieve the posting lists associated with the corresponding keys, and provide a union of retrieved posting lists as an answer to Q. In fact, we first check s_{max}-term combinations, and if all of them retrieve posting list, we stop the procedure because there will be no $(s_{max} - 1)$-term HDKs. For example, for a query $Q = \{t_1, t_2, t_3\}$ and $s_{max} = 2$, possible 2-term keys are $\{t_1, t_2\}$, $\{t_1, t_3\}$, and $\{t_2, t_3\}$. If we retrieve postings for $\{t_1, t_2\}$ and $\{t_1, t_3\}$, there is no need to check whether $\{t_1\}$, $\{t_2\}$, or $\{t_3\}$ are indexed because rare keys cannot be subsets of other rare keys. If we retrieve a posting only for $\{t_1, t_2\}$, we still need to check $\{t_3\}$, as it may be an HDK. The same query mapping principle has recently been proposed for structuring user queries into smaller maximal term sets [10].

However, users may still pose queries containing only frequent keys, or some query terms may not be covered by HDKs. A valid option is to notify a user that his/her query in non-discriminative with respect to the document collection, and provide support for refining the query. We have also devised two other possible strategies to improve the retrieval performance in such cases: The first strategy uses *distributional semantics* [11] to find semantically similar terms to query terms (further details can be found in [12]), while the second strategy indexes k-best documents for frequent keys, as the size of the frequent key vocabulary is less then 1% of the HDK size. We leave further analysis of the two strategies for future work.

4 Architecture

We assume an environment comprising a set of M independent DLs hosting local document collections, and willing to make a part of their collections searchable through a global distributed index. Each DL is a standalone component that can index and search its local document collection, and therefore provide (a part of) its *local single-term index* as a contribution to the global index. On the other hand, a structured P2P network with N peers is available to share a *global index*, and offer efficient search over the global collection composed of documents contributed by M DLs.

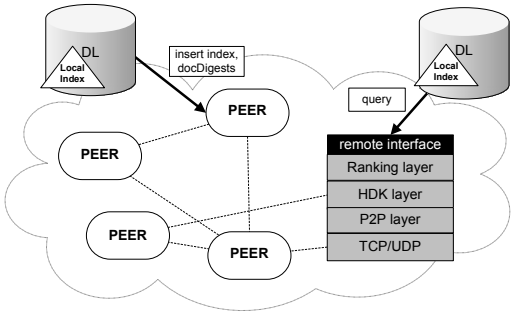

Fig. 2. An overview of the P2P architecture for digital libraries

The high-level architecture of our P2P search engine is presented in Fig. 2. DLs interact with peers to *submit an index* and to *send a query* to the engine. A peer can be regarded as an entry point to a distributed index, and a P2P network as a scalable and efficient media for sharing information about DL content. The architecture is layered to enable clean separation of different concepts related to P2P networks, document and content modeling, and the applied retrieval model [13]. As the global index is key-based, the system is decomposed into the following four layers: 1) transport layer (TCP/UDP) providing the means for host communication; 2) P2P layer building a distributed hash table and storing global index entries; 3) HDK layer for building a key vocabulary and corresponding posting lists, and mapping queries to keys; and 4) Ranking layer that implements distributed document ranking.

Each peer incorporates a *local and global system view*. The HDK layer focuses on the local view and builds the key index from a received single-term index for a DL's local collection. The received single-term index must contain a positional index needed for key computation, and may provide DL's relevance scores for (term, document) pairs. The P2P layer provides a global system view by maintaining the global key index with information about rare and frequent keys. Global index entries have the following structure $\{key, DF(key), PeerList(key), Posting(key)\}$, where $DF(key)$ is key's global document frequency, $PeerList(key)$ is the list of peers that have reported local document frequencies $df(key)$, and $Posting(key)$ is the key's global posting list. The $Posting(key)$ is $null$ in case key is frequent.

4.1 Distributed Indexing

The indexing process is triggered when a DL inserts an index into the P2P search engine. Since the indexing process is computationally intensive, peers share computational load and build the HDK vocabulary in parallel. Each peer creates HDKs from the received index, inserts local document frequencies for HDKs it considers locally i-rare or frequent, and subsequently inserts posting lists for globally i-rare keys into the P2P overlay. The P2P layer stores posting lists for globally i-rare keys, maintains the global key vocabulary with global DFs, and notifies the HDK layer when i-rare keys become frequent due to addition of new documents.

Algorithm 1 defines the process of computing HDKs locally by peer P_i at its HDK layer. It is performed in levels by computing single-term, 2-term, ... s_{max}-term keys. The peer stores a set of potentially i-rare keys in K_{ir}, and globally frequent keys in K_{freq}. Note that a locally frequent key is also globally frequent, but each locally rare key may become globally frequent. The P2P overlay is aware when a key becomes frequent, and notifies interested peers from the $PeerList(key)$.

The algorithm starts by inserting local document frequencies for the single-term vocabulary T_i and classifying terms as frequent or rare. Note that a peer is notified if locally rare keys are globally frequent, which depends on the HDK computation process performed by other peers. We assume this process is performed in parallel which leads peers into a steady state with a fairly realistic global knowledge. Next, P_i re-checks single-term DFs, and inserts posting lists for the rare ones into the P2P overlay. The approach is tolerant to erroneous insertions of posting lists for frequent 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 frequent.

For determining multi-term i-rare keys, the algorithm uses term locations from the received single-term index. A potential term combination needs to appear within a predefined window, next the redundancy property is checked, and if a key passes both filters, it is an HDK candidate. It's global frequency is updated in the P2P overlay, but the HDK layer at this point updates its posting list only locally. The global posting list will be updated only in case the key was not reported globally frequent by the P2P layer.

4.2 Distributed Retrieval

The query and retrieval scenario involves all four architectural layers. A query is submitted through a peer's remote interface to the HDK layer which maps query terms to HDKs as discussed in Section 3. The HDK layer retrieves posting lists associated with relevant HDKs from the global P2P index. The received posting lists are merged, and submitted to the ranking layer. The ranking layer ranks documents, and must be designed to provide relevance scores with the minimal network usage. There are a number of ranking techniques the proposed architecture can accommodate, but here we only sketch an approach using the vector space model since distributed ranking is outside the scope of this paper.

As the P2P global index maintains global DFs for all frequent and rare terms, DFs for the vocabulary T are readily available in the index, and may be retrieved together with HDK postings. Term frequencies are local document-related values that can be computed from a single-term index during key computation and stored together with document identifiers in the global posting lists. Therefore, both term and key relevance scores may be precomputed and available for calculating document scores at query time. We have devised another strategy: A DL provides document digests with e.g. document URL, term statistics, document title and abstract, together with a single-term index when initiating the indexing process. Such digests are stored on the peer computing DL's HDKs. Document digests are retrieved at query time to obtain term frequencies for ranking computation, and the corresponding abstract and title are presented to the user. Otherwise, DLs themselves may be responsible for storing document digests. Consequently, we can rank the result set using both global document frequencies and

Algorithm 1 Computing HDKs at peer P_i

1: **for** $s = 1$ to s_{max} **do**
2: $K_{ir}^s \leftarrow \emptyset$
3: $K_{freq}^s(s) \leftarrow \emptyset$
4: **if** $s = 1$ **then**
5: /* process single-term keys */
6: **for all** $t_k \in T_i$ **do**
7: P2P.**updateDF**(key)
8: **if** $df(t_k) \leq DF_{max}$ **then**
9: $K_{ir}^s \leftarrow K_{ir}^s(s) \cup t_k$
10: **else**
11: $K_{freq}^s \leftarrow K_{freq}^s \cup t_k$
12: **end if**
13: **end for**
14: **else**
15: /* generate new keys from frequent keys*/
16: **for all** $key = (t_{k_1}, \ldots, t_{k_{s-1}}) \in K_{freq}^{s-1}$ **do**
17: /* process each document in the key posting list to create a set of potential term combinations */
18: **for all** $d_j \in$ **localPostingList**(key) **do**
19: **for all** $t_{k_s} \in$ **windowOf**(key) **do**
20: $newKey$ = **concat**(key, t_{k_s})
21: **if checkRedundancy**($newKey$) **then**
22: $K_{ir}^s \leftarrow K_{ir}^s \cup newKey$
23: P2P.**updateDF**($newKey$)
24: **updateLocalPostingList**($newKey$, d_j)
25: **end if**
26: **end for**
27: **end for**
28: **end for**
29: **end if**
30: /* update global key frequency and insert posting list for i-rare*/
31: **for all** $key \in (K_{ir}^s \cup K_{freq}^s)$ **do**
32: **if** $DF(key) > DF_{max}$ **then**
33: /* key is globally frequent */
34: $K_{ir}^s \leftarrow K_{ir}^s \backslash key$
35: $K_{freq}^s(s) \leftarrow K_{freq}^s \cup key$
36: **else**
37: P2P.**insertPostingList**(key)
38: **end if**
39: **end for**
40: **end for**

term frequencies, without knowing the total global document size, as this parameter is typically used to normalize the scores.

5 Experimental Evaluation

Experimental setup. The experiments were carried out using a subset of news articles from the Reuters corpus[3]. 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 2 peers, and added additional 2 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 2 peers is 10,000 documents, and it is augmented by the new 10,000 documents at each experimental run. The maximum number of peers is 16, hosting in total the global collection of 80,000 documents. The experiments were performed on our campus intranet. Each peer runs on a Linux RedHat PC with 1024Mb of main memory connected by a 100 megabit network. The prototype system is implemented in Java.

Performance analysis. Experiments investigate the number of keys generated by our HDK algorithm, and the resulting average posting list size maintained by the P2P network. All documents were 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). The DF_{max} is set to 250 and 500, s_{max} is 3, and $w = 20$ for the proximity filter.

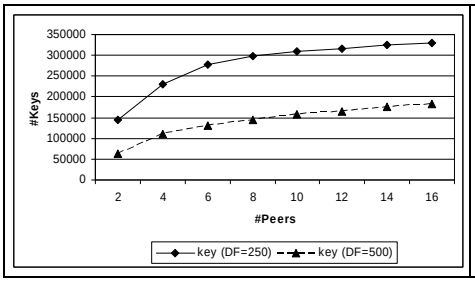

Fig. 3. Average HDK vocabulary per peer

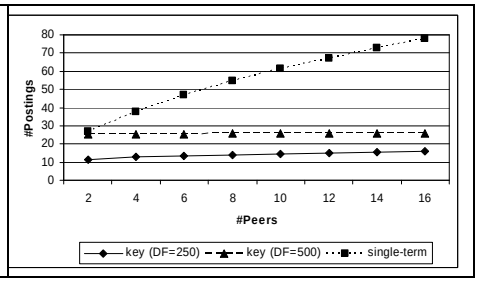

Fig. 4. Average posting list size

Figure 3 shows the total number HDKs stored per peer for $DF_{max} = 250$, and $DF_{max} = 500$. As expected, an increased value of DF_{max} results in decreased key vocabulary. Both experimentally obtained result sequences have a good fit to the following function $y(x) = a \cdot log(b\ x)$, x being the number of collection documents, and y modeling the key vocabulary size. As x is quite large, and will further increase for larger document collections, we can conclude that a peer will maintain a negligibly

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

increasing number of HDKs when increasing the document collection size by adding new peers. Therefore, our experiments show the total key vocabulary size increases linearly with the number of documents. The number of keys is quite large compared to the single-term vocabulary, but we expect to benefit from the query performance.

Figure 4 shows the average posting list size for the HDK and single-term indexing. As the average posting list size for HDK indexing remains constant, the expected bandwidth consumption is significantly smaller than for the single-term index exhibiting a linear increase. With respect to time costs, we have observed a moderate 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. These results prove the practical feasibility and scalability of our approach: The values that have impact on the indexing and querying process remain constant or grow negligibly with respect to the growth of collection size when new peers join the network.

For the retrieval performance evaluations, we have created the total of 200 queries by randomly choosing 2 to 3 terms from the news titles. Because of the lack of relevance judgments for our query set, we compared the retrieval performance to a centralized baseline [4]. We have indexed the collection using both single-term and HDK indexing with deferent DF_{max} values (200, 250, 500). Then for each query we compared the top 20 documents retrieved by our prototype and by the baseline, both hit lists have been ranked using TF-IDF. 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 result sets: the first one is the overlap between our system and the centralized baseline, and the second one is maximal size of posting list transmitted during query processing.

Table 1. Retrieval quality of HDK indexing compared to the centralized TF-IDF system

	Overlap ratio on top20	Max. posting list size
single-term (TF-IDF)	100 %	2437.47
HDK, $DF_{max} = 500$	94.30%	196.80
HDK, $DF_{max} = 250$	85.84%	82.65
HDK, $DF_{max} = 200$	83.02%	66.68

Table 1 presents our findings related to retrieval performance for the collection of 30,000 documents over 6 peers. The results show acceptable retrieval performance even for short queries without any additional techniques for dealing with frequent keys. As expected, the retrieval performance is better for larger DF_{max} as we are getting closer to the single-term indexing, but the maximum size of the retrieved posting list also increases, although it is still significantly smaller compared to the single-term case. Therefore, the total bandwidth consumption is expected to be much smaller for the HDK index, and further experimentation is needed to quantify our expectations. There is a trade-off between retrieval quality and bandwidth consumption in our indexing strategy.

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

6 Related Work

Full-text P2P search is investigated in two overlapping domains: DLs and the Web. There is an ongoing debate on the feasibility of P2P Web search for scalability reasons. In [14] 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: PlanetP [15] gossips compressed information about peers' collections in an unstructured P2P network, while MINERVA [16] maintains a global index with peer collection statistics in a structured P2P overlay to facilitate the peer selection process.

As DLs represent only a small fraction of the entire Web space, the feasibility of full-text P2P search across DL collections is not in question. Hierarchical solutions have been investigated for federated search where a backbone P2P network maintains a directory service to route queries to peers with relevant content [6, 1]. A recently proposed solution uses collection-wide statistics to update routing indexes dynamically at query time, and reports low traffic overheads for the Zipf-distribution queries after the initial 'learning phase' [17]. These solutions are orthogonal to our approach since they are designed for unstructured P2P networks with the low-cost indexing schemes, while the processing and major network traffic is generated during the query phase. Our technique is costly in terms of indexing, however, it offers highly-efficient and responsive querying performance. It is comparable to solutions for distributed top-k retrieval that aim at minimizing query costs by transmitting a limited number of postings [17, 18]. However, the major difference is our novel indexing strategy. On the other hand, our approach is not the only indexing strategy that uses term sets as indexing features. The set-based model [19] 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.

7 Conclusion

We have presented a P2P architecture for information retrieval across digital library collections. It relies on a novel indexing strategy that indexes rare term sets to limit the bandwidth consumption during querying, and enable scalable and highly-efficient search performance. As a proof of concept, we have implemented a prototype system following the presented architectural design, and performed experiments to investigate query effectiveness and indexing costs. Our experiments have demonstrated the feasibility of the proposed indexing strategy for P2P environments. Our future work will further investigate techniques for reducing the cost of the proposed indexing strategy, e.g., by using query statistics, or query-driven indexing. We will perform experiments with larger and various document collections, and increased size of the peer network to

confirm existing positive results concerning both the indexing costs and retrieval performance.

References

1. 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
2. Balke, W.T., Nejdl, W., Siberski, W., Thaden, U.: DL Meets P2P - Distributed Document Retrieval Based on Classification and Content. In: 9th European Conference on Research and Advanced Technology for Digital Libraries, (ECDL). (2005) 379–390
3. Li, J., Loo, B., Hellerstein, J., Kaashoek, F., Karger, D., Morris, R.: The Feasibility of Peer-to-Peer Web Indexing and Search (2003)
4. Buntine, W., Aberer, K., Podnar, I., Rajman, M.: Opportunities from open source search. In: Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence. (2005) 2–8
5. 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)
6. Lu, J., Callan, J.: Content-based retrieval in hybrid peer-to-peer networks. In: Proceedings of the 12th International Conference on Information and Knowledge Management. (2003)
7. Aberer, K., Alima, L.O., Ghodsi, A., Girdzijauskas, S., Haridi, S., Hauswirth, M.: The Essence of P2P: A Reference Architecture for Overlay Networks. In: Fifth IEEE International Conference on Peer-to-Peer Computing. (2005) 11–20
8. Salton, G., Yang, C.: On the specification of term values in automatic indexing. Journal of Documentation **4** (1973) 351–372
9. Salton, G., Allan, J., Buckley, C.: Approaches to Passage Retrieval in Full Text Information Systems. In: SIGIR'93. (1993) 49–58
10. Pôssas, B., Ziviani, N., Ribeiro-Neto, B., Wagner Meira, J.: Maximal termsets as a query structuring mechanism. In: CIKM '05. (2005) 287–288
11. Rajman, M., Bonnet, A.: Corpora-Base Linguistics: New Tools for Natural Language Processing. 1st Annual Conference of Association for Global Strategic Information (1992)
12. Aberer, K., Klemm, F., Luu, T., Podnar, I., Rajman, M.: Building a peer-to-peer full-text Web search engine with highly discriminative keys. (Technical Report LSIR-REPORT-2005-011)
13. Aberer, K., Klemm, F., Rajman, M., Wu, J.: An Architecture for Peer-to-Peer Information Retrieval. In: SIGIR'04, Workshop on Peer-to-Peer Information Retrieval. (2004)
14. 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)
15. 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)
16. Bender, M., Michel, S., Triantafillou, P., Weikum, G., Zimmer, C.: Improving collection selection with overlap awareness in P2P search engines. In: SIGIR '05. (2005) 67–74
17. 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)
18. Michel, S., Triantafillou, P., Weikum, G.: KLEE: a framework for distributed top-k query algorithms. In: VLDB '05. (2005) 637–648
19. Pôssas, B., Ziviani, N., Wagner Meira, J., Ribeiro-Neto, B.: Set-based vector model: An efficient approach for correlation-based ranking. ACM Trans. Inf. Syst. **23** (2005) 397–429