Query-Driven Indexing for Peer-to-Peer Text Retrieval[*]

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

[†]Ecole Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
{gleb.skobeltsyn,vinhtoan.luu,martin.rajman,karl.aberer}@epfl.ch

[‡]University of Zagreb
Zagreb, Croatia
ivana.podnar@fer.hr

ABSTRACT

We describe a query-driven indexing framework for scalable text retrieval over structured P2P networks. To cope with the bandwidth consumption problem that has been identified as the major obstacle for full-text retrieval in P2P networks, we truncate posting lists associated with indexing features to a constant size storing only top-k ranked document references. To compensate for the loss of information caused by the truncation, we extend the set of indexing features with carefully chosen term sets. Indexing term sets are selected based on the query statistics extracted from query logs, thus we index only such combinations that are a) frequently present in user queries and b) non-redundant w.r.t the rest of the index. The distributed index is compact and efficient as it constantly evolves adapting to the current query popularity distribution. Moreover, it is possible to control the tradeoff between the storage/bandwidth requirements and the quality of query answering by tuning the indexing parameters. Our theoretical analysis and experimental results indicate that we can indeed achieve scalable P2P text retrieval for very large document collections and deliver good retrieval performance.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

P2P DHT IR Text Retrieval Query-Driven Indexing

1. INTRODUCTION

In contrast to centralized information retrieval engines, leveraging a P2P network to distribute the global index among a large number of participating peers offers a solution for decentralized search over web-scale document collections by sharing of peer resources. However, while P2P networks can provide very large storage capacity with the peer population potentially reaching millions of nodes, there is an ongoing debate about the overall scalability of P2P web search because of high bandwidth consumption. A recent study [4] has shown that, even when carefully optimized, P2P algorithms using traditional single-term indexes in structured P2P networks generate unscalable traffic during retrieval. Instead of indexing all single terms found in the document collection, which might lead to potentially very large posting lists, in [1] we propose the HDK approach[1] that generates low retrieval traffic by transmitting only truncated posting lists with top-ranked document references. HDKs are combinations of terms (or *keys* in our jargon) carefully extracted from the document collection and associated with posting lists of limited size. It is shown in [1] that the number of indexing keys grows linearly with the number of documents, which is acceptable under a reasonable assumption that the ratio between the total number of documents and peers in the network remains bounded, hence facilitating scalable indexing. However, according to our measurements, the number of indexing keys could still be impractical for large document collections.

Indeed, the extraction of term combinations solely based on document collection statistics can potentially create a large number of superfluous keys that will probably never be used in real queries. In the Distributed Cache Table (DCT) approach [2] we introduce an idea of indexing with term combinations tailored to the query distribution. However the DCT approach is rather suitable for middle size P2P text retrieval systems such as digital libraries as it relies on local post-processing of possibly large posting lists.

Following the ideas of the two approaches mentioned above, instead of exhaustive document-driven key generation, we suggest indexing only "non-superfluous" keys that can be detected and indexed on-the-fly by monitoring the recent query popularity statistics. The resulting indexing structure is considerably smaller compared to the HDK approach and causes only a marginal loss in recall for non-popular queries. Hence, the requirements in storage and bandwidth while indexing are substantially decreased by reducing the number of indexing keys. Moreover, due to the strictly bounded posting list size, query processing guarantees to consume limited bandwidth as it requires transmission of small portions of data only. These two factors make our approach a viable solution for web-scale P2P information retrieval.

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

[1]based on indexing with Highly Discriminative Keys

2. DISTRIBUTED INDEXING/RETRIEVAL

Each peer in the P2P network is responsible for 1) proper representation of its local document collection in the global distributed index and 2) maintenance of a selected set of indexing keys associated with the corresponding truncated posting lists. Initially, the peers collaboratively build a distributed single-term index associating all single term keys from the global collection with their top-k global document references. Each peer maintains a part of the global index containing a set of keys assigned by the hashing mechanism used in the underlying Distributed Hash Table (DHT).

As the result quality for some multi-term queries derived from the top-k single-term index might not be sufficient, the index is progressively populated with additional keys corresponding to popular term combinations, i.e., keys frequently occurring in recent user queries that will be useful to answer future queries. Thus, the subsequent indexing process is fully driven by the query statistics, and is performed in parallel with retrieval. More precisely, as soon as a peer receives a new query, it starts to explore the lattice of query term combinations (or the *query lattice*, see Figure 1) in decreasing subset size order starting with the query itself. For each node in the query lattice, the querying peer requests the posting list from the peer responsible for the corresponding term combination. If the term combination is indexed, the associated posting list is sent back to the originating peer and the part of the query lattice dominated by the combination is excluded from the further lattice exploration process. During lattice exploration each contacted peer updates the usage statistics for the requested combination.

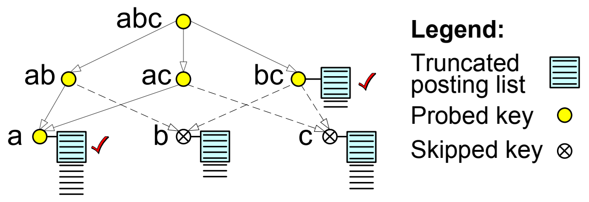

Figure 1: Processing of a query $\{a,b,c\}$

After the lattice exploration process is finished and all available posting lists relevant to the original query are received, the originating peer produces their union, re-ranks all records w.r.t the original query and presents the top-ranked results to the user. For example, assuming the term combination *bc* is indexed while *ab* and *ac* are not, the result for the query *abc* is produced from the union of truncated posting lists for the keys *bc* and *a* as shown in Figure 1.

A special *on-demand indexing* mechanism is executed when a certain key is detected as popular during query processing. The peer responsible for this key sends an indexing request to all the peers holding documents that contain the corresponding term combination and acquires a new posting list that stores a limited number of top-ranked document references. Thus, the key becomes *indexed* and can be used for future query processing. The list of peers to contact is maintained in a dedicated structure distributed in the network in order to avoid broadcasting of the indexing request.

To summarize, the processing of new queries triggers indexing of popular combinations, which, in turn, increases the overall answer quality. At the same time obsolete keys can be removed, resulting in an efficient indexing structure adaptive to the current query popularity distribution.

3. SCALABILITY AND EXPERIMENTS

The query-driven approach is scalable in terms of bandwidth consumption because: 1) the retrieval traffic generated while processing a query is low, since all transmitted posting lists are of bounded size, 2) the indexing traffic needed to generate and maintain the global distributed index is scalable due to the query-driven key selection.

The indexing traffic depends on the key activation rate, which is dictated by the distribution of term combinations in the query log. In [1] we showed that the exhaustive number of keys generated for a given document collection in the HDK approach grows linearly with the collection size. The query-driven key selection yields a substantial decrease of the number of keys compared to the HDK approach. In [3] we derived an upper bound on the number of activated keys for a given query log and showed that it scales linearly with the query log size under the assumption that the popularity of term combinations in the query load follows the Zipf law. Finally, the bandwidth required to execute the on-demand indexing mechanism grows linearly with the network size.

We also conducted several experiments to verify that the retrieval performance remains acceptable even with the index size reduced by 1-2 orders of magnitude compared to the HDK approach. Evaluations over a TREC collection showed that our retrieval performance is similar to a centralized single-term indexing system with the top-performing BM25 weighting schema. Moreover, we used the Google search engine to obtain top-ranked postings for real queries extracted from AOL query logs and showed that our query-driven approach delivers acceptable result quality with 70-80% overlap when compared to the original Google answers [3].

4. CONCLUSION

We presented a novel query-driven indexing strategy for P2P text retrieval with structured P2P networks. Our approach guarantees low bandwidth consumption during retrieval by limiting the size of transmitted posting lists. At the same time, we avoid maintenance of rarely used index entries by adapting the contents of the global distributed index to the query popularity distribution observed in the recent query log. Thus, the bandwidth consumption while indexing is significantly reduced as it is not wasted for the maintenance of superfluous index entries. Moreover, by adjusting the minimum popularity threshold of candidate term combinations found in the query log, we can balance the quality of query answering and the number of keys in the global index to ensure that the bandwidth consumption remains under practically acceptable bounds.

We are currently implementing query-driven indexing in the Alvis P2P search engine prototype (http://www.alvis.info).

5. REFERENCES

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

[2] G. Skobeltsyn and K. Aberer. Distributed Cache Table: Efficient Query-Driven processing of multi-term queries in P2P networks. In *P2PIR*, 2006.

[3] G. Skobeltsyn, T. Luu, I. Podnar, M. Rajman, and K. Aberer. Query-Driven indexing for Peer-to-Peer text retrieval. EPFL Technical Report, LSIR-REPORT-2006-014.

[4] J. Zhang and T. Suel. Efficient query evaluation on large textual collections in a Peer-to-Peer environment. In *P2P*, 2005.