DL meets P2P – Distributed Document Retrieval based on Classification and Content

Wolf-Tilo Balke, Wolfgang Nejdl, Wolf Siberski and Uwe Thaden

L3S and University of Hannover
Expo Plaza 1, D-30539 Hannover, Telefax +49(0)511/762-9779
{balke,nejdl,siberski,thaden}@l3s.de

Abstract. Peer-to-peer architectures are a potentially powerful paradigm for retrieving documents over networks of digital libraries avoiding single points of failure by massive federation of (independent) information sources. Today sharing files over P2P infrastructures is already immensely successful, but restricted to simple metadata matching. But when it comes to the retrieval of complex documents, capabilities as provided by digital libraries are needed. Digital libraries have to cope with compound documents. Though some document parts (like embedded images) can efficiently be retrieved using metadata matching, the text-based information needs different methods like full text search. However, for effective querying of texts, also information like inverted document frequencies are essential. But due to the distributed characteristics of P2P networks such 'collection-wide' information poses severe problems, e.g. that central updates whenever changes in any document collection occur use up valuable bandwidth. We will present a novel indexing technique that allows to query using collection-wide information with respect to different classifications and show the effectiveness of our scheme for practical applications. We will in detail discuss our findings and present simulations for the scheme's efficiency and scalability.

1 Introduction

Digital libraries today offer a wide variety of content that is usually accessible through central portals. Though this generally is a useful way to access individual sources, it hampers the creation of federated networks of libraries or document collections. But such federations would be especially valuable for finding documents best matching the user's information needs and possibly providing a number of different views or opinions on a topic. One possibility of facilitating federated searches are meta-crawlers. But documents only accessible via a certain portal interface often cannot be crawled, also known as the 'hidden Web problem'. Moreover, crawlers only periodically crawl collections and update their indexes, and thus cannot react flexible to new, previously unknown collections and content changes in existing collections.

Peer to peer systems are a powerful paradigm to address some of these problems. Not relying on central coordination federations of information sources are formed dynamically by independent nodes. At any time new sources can join the network and disseminate their documents in a more timely way than the crawling of central servers can ascertain. File sharing applications, where media files are retrieved based on simple meta-data annotations like file formats or names, have become increasingly popular due to their ease of use. Also digital library collections can benefit from the advantages of P2P infrastructures, since much of their content (like embedded images) can be annotated by meta-data.

However, compound documents in digital libraries generally also contain textual parts, where the flexible, distributed P2P paradigm still is not readily usable. On one hand this is because queries have to be evaluated over the network at search time, which in basic file sharing applications is usually done by flooding queries through the complete network (or at least within a certain radius). On the other hand almost all effective textual measures for information retrieval not only rely on statistics about the single documents, but also integrate statistics on the entire collection of all documents, e.g. how well individual keywords discriminate between documents with respect to the entire collection (inverted document frequencies). This so-called collection wide information cannot be derived locally.

To improve query efficiency techniques one way are central indexes and distributed hash tables (DHTs)[11, 8, 1]. Besides the speed up above naive query flooding an additional advantage is that by such a structure also collection wide information can be provided for subsequent querying. A major drawback is that such indexes use up a lot of the available bandwidth by the necessary administrative message exchange for upkeep, because every change in the federated document collection (e.g. content changes within some peer) has to be registered in the index. A contrasting way of gaining query efficiency are local routing indexes that avoid the overhead of constant index upkeep, but due to their local nature face problems with acquiring the necessary collection-wide information. Besides having to be efficient, querying schemes will also have to take into account that the information in digital libraries often is pre-structured. Libraries usually categorize documents following some standardized taxonomies, such that documents on similar topics might be distinguished e.g. by rather taking an economical or sociological point of view, etc. This structure is also used to sometimes resolve ambiguities of keywords.

In this paper we investigate the querying of federated library collections over a peer-to-peer network. But in contrast to central indexing schemes, our aim is to create a local indexing scheme that allows effective indexing with a minimum of management message overhead and even efficiently use collection-wide information. Moreover, we will exploit taxonomies to structure the individual collections and investigate how this pre-structuring interacts with the effectiveness of our local indexing scheme and how to deal with the trade-off between the total number of documents in each category and our respective index size. We will investigate the necessary message traffic, the quality of result sets (as opposed to the perfect results using a central index), and a number of other characteristics of our novel approach. Throughout this paper we will assume a strong cooperation between peers, e.g. in order to ensure consistency of ranking results, score values have to be calculated uniformly by all peers.

We will motivate our approach with the example of federated news collections. News items can also be compound documents and are usually categorized within certain topics like politics or sports. Since we want to focus on the textual retrieval, we use a collection of LA Times news articles from the TREC-5 collection for our evaluations. By assuming periodic changes of user interests we can also experiment with the arrival or removal of complete corpora from our federation. Since queries in such practical applications usually form a Zipf distribution (i.e. considering the total set of queries very little queries are posed very often, whereas most queries are posed only once in a while), we will present extensive experiments for such a distribution. Given our innovative results, peer-to-peer networks are on the verge of forming efficient infrastructures for federations of digital libraries utilizing even collection-wide information without the communication overhead of central indexes.

2 Information Retrieval in a Distributed Environment

In large document collections information retrieval techniques are mandatory for efficient retrieval. Over centralized repositories these techniques have been investigated since the 70ies and work quite effective, e.g. using inverted file indexes for subsequent retrieval [4]. Maintaining these indexes, however, is a major problem in distributed systems, especially peer-to-peer networks that often share vast numbers of documents and have a high volatility with respect to peers joining and leaving the network. In contrast to static document collections every peer joining or leaving the network registers its document collection or removes it, thus indexes have to be updated very often.

For local query evaluation schemes a particular problem arises when collection wide information is an integral part of the query processing technique. For instance in the case of TFxIDF [16] the term frequency may be locally evaluated for each specific document, however, for the document frequency a snapshot of the entire current content of all active peers needs to be evaluated. Of course this would immediately annihilate any benefits gained by sophisticated local querying schemes.

Consider a simple example to show how local scorings fail, if collection-wide information has to be considered in the retrieval process. Assume the case that we have just two peers that should return their best matches with respect to the most popular information retrieval measure TFxIDF. This measure is a combination or two parts, the term frequency (TF, measures how often a query term is contained in a certain document), and the inverted document frequency (IDF, inverse of how often a query term occurs in the document collection). This measure needs to integrate collection-wide information and cannot be determined locally.

As an instance take a simple conjunctive query Q for the terms 'a' and 'b' posed to two peers P_1 and P_2 that has to be evaluated locally at each peer. Let's assume that P_1 contains three documents D_1, D_2 and D_3, and P_2 also contains three documents D_4, D_5 and D_6. For simplicity of our example let us further assume that in $D_1 \text{to} D_6$, our two keyword occur mutually exclusive in the documents: D_1, D_2 and D_6 contain the keyword 'a', whereas D_3, D_4 and D_5 contain the keyword 'b'. Moreover, assume that all documents are of the same length and the keywords occur in the same number in all documents, such that the respective term frequency is the same for all documents. Evaluating our query Q locally we have now to rank the documents in each peer. Since the keywords are mutually exclusive in our document base and the TFs are equal for each document, the ranking is only determined by the weighting factor of the occurring term in the IDF.

In peer P_1 we have two documents out of three containing term 'a', i.e. an IDF of $\frac{3}{2} = 1.5$, and only one document containing 'b', i.e. an IDF of $\frac{3}{1} = 3$. That means that with respect to Q P_1 ranks D_3 as better than D_1 and D_2. Symmetrically P_2 ranks D_6 higher than D_4 and D_5, because here 'b' occurs in two documents and 'a' only in one. Integrating the results from $P1$ and $P2$ we get a higher ranking of D_3 and D_6 than of the four other documents. In contrast, performing query Q over a central collection containing all six documents D_1 to D_6, we find that both query terms 'a' and 'b' occur in three of the six documents, i.e. have an IDF $\frac{6}{3} = 2$. Since the TF is still the same, all six documents will be correctly assigned the same score.

As shown, collection-wide information is essential to provide proper document scores. But the index holding this information does not necessarily need to be completely up-to-date; obviously there is a trade-off between index information that is 'still current enough' given the network volatility and the accuracy of the query results. Research on what dissemination level is required in Web IR applications to allow for efficient retrieval showed that a complete dissemination with immediate updates is usually unnecessary, even if new documents are included into the collection [14]. Moreover, the required level was found to be dependent on the document allocation throughout the network [13]: random allocation calls for low dissemination, whereas higher dissemination is needed if documents are allocated based on content. Thus a lazy dissemination usually has comparable effectiveness as a centralized approach for general queries, but if only parts of the networks containing most promising documents with similar content are queried, the collection-wide information has to be disseminated and regularly updated.

3 Approach

As shown in the previous section, we need collection-wide information at each peer to do a correct score computation. The challenges are

– how to compute this information
– where to store it, and
– how to distribute it in the network.

The key to success is the observation that we don't need a complete inverted index to process a query [7, 2]. For example, to calculate the correct scores, peers P_1 and P_2 need only IDFs for terms a and b, but not for all terms occurring in their documents.

Storage To store this information, we use a super-peer network approach: in a P2P-network peers often vary widely in bandwidth and computing power. As discussed in [17] exploiting these different capabilities can lead to an efficient network architecture, where a small subset of peers, called super-peers, takes over specific responsibilities. In our case, we assign the index management responsibility to the super-peers. The super-peers form the network backbone, and each document provider peer is directly connected to one of them.

Distribution A query consists of a category of the taxonomy which should be searched and a conjunction of keywords that are searched in the content of the documents. Before distributing such a query, a super-peer adds the necessary collection-wide information from its index to it. If it isn't yet in the index, an estimation is provided.

The category in the query is used as a filter for two purposes: First to reduce the number of peers (and thus documents) which must be searched and ranked. Second the user can use the category to avoid ambiguities. If a user is interested in the local sport-team called 'Jaguars', the appropriate category will avoid hits Jaguar-cars and the animal Jaguar. The keywords which are specified in the query will only be used for the documents which are in the named category.

Computation Responding peers do not only deliver matching documents, but also each add local data necessary to compute the collection-wide information (for TFxIDF this is the document frequency for each query term and the document count). On the way back to the originating super-peer, this data is aggregated. Thus, the originating super-peer gets everything it needs to compute the complete aggregate, and can store the computed result in its index.

The next subsections discuss in detail how this approach is applied to the digital library network context.

3.1 Query Processing

Query distribution at super-peer Each super-peer maintains an IDF index containing IDF values for the keywords. This is done separately for each category, not for all documents in all categories. Thus, the key for this index is built from a category and one keyword. As mentioned above a query contains the IDF values for the query-terms. The IDFs are taken from the IDF index, or estimated if a keyword is not yet in the index. In the latter case, the average IDF is used as estimation.

Query processing at peer At each peer, first only documents in the specified category are taken into account. The top-k documents are determined using the TFxIDF algorithm, but based on the IDF values from the query. If this gets enough (=k) results in the queried category, these are sent to the super-peer. If the number of documents matching the query is smaller than k, the query is relaxed, first to subcategories and then to the super-categories. This process is repeated until the peer has k results or the root of the taxonomy is reached. The entries found in all newly searched categories are sorted by their similarity to the originating category using the following measure (from Li et al. [5]):

$$sim(c_1, c_2) = e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}}$$

where l is the shortest path between the topics in the taxonomy tree and h is the depth level of the direct common subsumer. α and β are parameters to optimize the similarity measurement (best setting is usually $\alpha = 0.2$ and $\beta = 0.6$).

The super-peer then gets the top-k of the peer or a number $n < k$ of documents matching. This query-relaxation is shown in the following code:

```
Initialize a ResultSet results;

Set searchRoot := Category from query

do
  Initialize a new set searchCategories
  Add searchRoot to searchCategories
  while (number of results < k and searchCategories is not empty
  begin
    Initialize new set allChildren
    for all cat in searchCategories do
    begin
      // retrieve hits matching category exact
      Initialize ordered list matchingDocuments
      for all doc in documents
      begin
        if (document-category = cat
           AND document contains terms from query
           AND number of matching documents < k
           OR doc.score(query.terms) > matchingDocuments.getLastDoc.score))
           then add document to matchingDocuments
      end
      results.addHits(retrieveExact(cat, query))
      allChildren.add(cat.children);
    end

    searchCategories = allChildren;  // go one level down in category tree
    removesearchRoot from searchCategories; //do not to traverse subtree twice
  end

  searchRoot := parent of searchRoot // go one level up in category tree
while(not k results AND searchRoot != nil);

trim results to k  // in case we collected more than k entries

return results;
```

Result merging at super-peer A super-peer retrieves max. k hits from each of its peers and combines them to the top-k. As described above it is possible that peers also send results which are coming from another category as requested. In this case, the super-peer first takes all hits which match the queried category. If this results in a set smaller than k it takes the next best-matching hits from each peer and combines them using the described sorting.

IDF index update The IDF index can be updated in two ways:

1. By summing up document frequencies and document counts delivered from connected peers and super-peers, the super-peer where the query originated computes IDFs for each query term and updates its IDF index. If the difference between computed IDF and estimated IDF value exceeds a threshold, the query is redistributed, this time using the computed IDF values.
2. if a super-peer receives a query it checks, if the IDFs contained are marked as estimated. If this is not the case, these values are used to update the IDF index.

3.2 IDF index entry expiration

Viles and French have shown that in a large document collection IDF values change slowly [14]. In our context, this is not strictly applicable, because there are two kinds of changes that may influence our collection-wide information significantly:

1. *New documents with similar content: new peers join the network.*
   Imagine a large federation of library servers which offer articles from different newspapers. Let's assume we already have a newspaper like the NY Times in the collection. What can happen if peers join the network offering a new newspaper, i.e. the LA Times? In this case we can be sure that the articles usually will be on nearly similar topics except a few local news. Thus, we do not really have to update our IDFs since the words in the articles are distributed the same way as before.
2. *New documents or new corpora: New library servers join the federation or new documents are included in existing collections, whose content is very different from existing articles and thus shifts IDFs and changes the discriminators.*
   Let's look at an example: Assume there is an election e.g. in France and people use our P2P-news-network to search for news regarding this election. This normally will be done using queries like 'election France' and results in a list of news that contain these words. In this case there would be a lot of news containing France, thus 'election' is the discriminator, and the IDFs will give us the correct results. Now think of another election taking place in the US in parallel. The term 'election' will no longer be the best discriminator, but the term 'France' then gets more important.

In these cases we have to solve the problem that entries in the IDF index become outdated over time. We can handle both cases in the same way: Each IDF value gets a timestamp when the term appears for the first time and the term/IDF-pair is stored. After a specific expiration period (depending on the network-volatility) the item becomes invalid and the entry gets deleted. In this way we force IDF recomputation if the term occurs again in a query. By adjusting the expiration period we can trade off accuracy against performance. We reduce the expiration period for terms occurring more frequently, thus ensuring higher accuracy for more popular queries.

3.3 Query Routing Indexes

So far, we still distribute all queries to all peers. We can avoid broadcasting by introducing additional routing indices which are used as destination filters:

– For each category in our taxonomy the *category index* contains a set of all peers which have documents for this category. It is not relevant if this peers did contribute to queries in the past.
– In the *query index* for each posed query the set of those peers which contributed to the top-k for the query are stored.

Query Distribution The super-peer first checks if all query terms are in the IDF index. If this is not the case the query has to be broadcast to permit IDF aggregation. We also broadcast the query if none of the routing indexes contain applicable entries.

If an entry for query exists in the query-index, it is sent to the peers in this entry only, since no other have contributed to the top-k result for the current query.

Otherwise, if the query category is in the category index, the query is sent to all peers to the corresponding category entry.

Index Update For each delivered result set, a query index entry is created, containing all peers and super-peers which contributed to the result.

For the category index, we need to know all peers holding documents of the specified category, even if they didn't contribute to the result set. Therefore, we collect this information as part of the result set, too, and use it to create category index entries.

As with the IDF index, the network volatility causes our routing indexes to become incorrect over time. We use the index entry expiration approach here as well to delete outdated entries.

4 Evaluation

4.1 Simulation Environment

We use the TREC document collection volume 5 consisting of LA Times articles for our experiments. The articles are already categorized according to the section they appeared in, and we use this information as base for our document classification. To simulate a network of document providers, these articles are distributed among the peers in the network. The simulated network consists of 2000 peers, each providing articles from three categories on average (with a standard deviation of 2.0).

The simulation is based on the framework described in [10]. The super-peers are arranged in a HyperCuP topology [9]. The TFxIDF calculation based on inverse indexes was done using the (slightly modified) search engine Jakarta Lucene [1].

We assume a Zipf-distribution for query frequencies with skew of -0.0. News articles are popular only for a short time period, and the request frequency changes correspondingly. With respect to the Zipf-distribution this means that the query rank decreases over time. Query terms were selected randomly from the underlying documents. In our simulation, we generate 200 new most popular queries every 2000 queries which supersede the current ones and adjust query frequencies accordingly. This shift may be unrealistically high, but serves well to analyze how our algorithm reacts to such popularity changes.

4.2 Results

Index size Figure 1 shows how the IDF index at each super-peer grows over time. After 10000 queries it has grown to a size of 2015, only a small fraction of all terms occurring in the document collection. A global inverted index we would have had contained

[1] http://jakarta.apache.org/lucene/docs/index.html

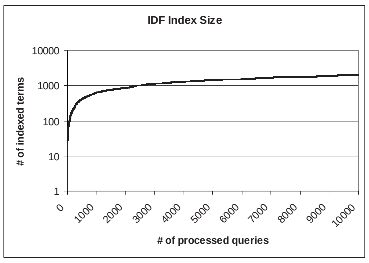

Fig. 1. Index size

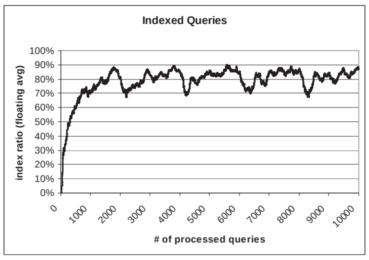

Fig. 2. Coverage of query index

148867 terms. This underlines that much effort can be saved when only indexing terms which are actually appearing in queries.

Index effectivity Both category and query index become quite effective. After nearly 2000 queries, the query index achieves a coverage of 80%. Figure 2 shows how each popularity shift causes a coverage reduction from which the query index recovers after about 1000 queries. This shows that a change in query popularity over time is coped with after a very short while.

As there are only about 120 different categories, after less then 1000 queries the index contains nearly all of them (Figure 3). We assume that news provider specialized on some topics change these topics only very infrequently. Therefore, peers do not shift their topics during the simulation. Thus, the category index serves to reduce the number of contacted peers continouosly, also after popularity shifts.

Figure 4 shows how many peers had to be contacted to compute the result. The influence of popularity shifts on the whole outcome can also be seen clearly. The category index takes care that the peaks caused by popularity shifts don't become too high. Summarized, the combination of both indexes yields a high decrease of contacted peers compared to broadcasting.

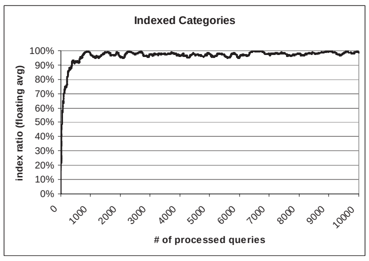

Fig. 3. Coverage of category index

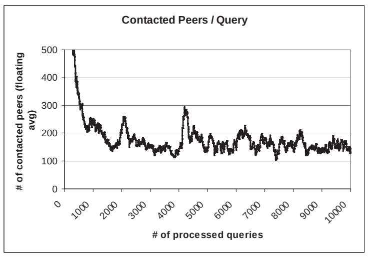

Fig. 4. Contacted peers per query

In the experiments described here we didn't introduce dynamics regarding the peers contents. Therefore, our algorithm yields exactly the same results as a complete index. In [2] (where we didn't take categories into account), we show that if 20% of the peers contents during a simulation run, the error ratio is about 3.5%.

5 Related Work

Since the concepts of the highly distributed P2P networks and the rather centralized IR engines are hard to integrate, previous work in the area is focussing on efficient dissemination of this information. There is a challenging trade-off between reduced network traffic by lazy dissemination however leading to less effective retrieval, and a large network traffic overhead by eager dissemination facilitating very effective retrieval. What is needed is ”just the right” level of dissemination to maintain a ”suitable” retrieval effectiveness. Thus previous approaches to disseminate collection-wide information rely on different techniques. We will briefly review the techniques from peer-to-peer systems, from distributed IR and Web Search engines and compare them to our approach.

For peer-to-peer systems there are different approaches. The PlanetP system [3] does not use collection-wide information like e.g. the inverted document frequency of query terms directly, but circumnavigates the problem by using a so-called inverted peer frequency estimating for all query terms, which peers are interesting contributors to a certain query. Summarizations of the content in the form of Bloom filters are used to decide what content a peer can offer, which are eagerly disseminated throughout the network by gossiping algorithms. Thus in terms of retrieval effectiveness this scheme describes documents on the summarization level, which is a suboptimal discriminator and by gossiping the system’s scalability is limited. The idea of PeerSearch [12] is comparable to our approach, but instead of a broadcast-topology CAN [8] is used in combination with the vector space model (VSM) and latent semantic indexing (LSI) to create an index which is stored in CAN using the vector representations as coordinates. Thus all collection-wide information has to be disseminated again leading to a limited scalability. Also summarizing indexes have been used to maintain global information about a set of documents like e.g. in [15]. Here so-called cell abstract indexes are used for approximate queries. The abstract of a set of documents is some statistics of all documents in the set and the abstract of a peer is an abstract of the shared document set residing in the peer. An abstract index of a P2P system then is an organization of all abstracts of peers in the system. All peers of a system can thus be formed into an overlay network. Every joining peer will be added to a cell that contains its abstract and subsequently queries are routed to those cells that contain their abstract. However, also in this case indexes for all cells have to be updated regularly leading to a high overhead of network traffic. Moreover, peers in the end cells will just deliver all documents to the querying peer not removing suboptimal objects and again causing unnecessary network traffic. As in our approach, [6] use super-peers (called ”‘hub”’ nodes) to manage indices and merge results. Depending on the cooperation capability/willingness of document providers (”‘leaf”’ nodes), hub nodes collect either complete or sampled term frequencies for each leaf peer. This information is used to select relevant peers during query distribution. By using query sampling hubs are able to give an estimate of relevant peers, even in case of uncooperative peers. As with the other systems, indices are built in advance, thus causing possibly unnecessary management messages.

From the perspective of information retrieval the problem of disseminating collection-wide information first occurred when IR moved beyond centralized indexing schemes over collections like e.g. given by TREC, and had to deal with vast distributed document collections like the WWW. Here due to the random-like distribution of content over the WWW, research on effective retrieval in Web IR applications showed that a complete dissemination with immediate updates is usually unnecessary, thus allowing for a little volatility [14], The required level of dissemination, however, was found to be dependent on the document allocation throughout the network [13]: random allocation calls for low dissemination, whereas higher dissemination is needed if documents are allocated based on content. In peer-to-peer networks this random-like distribution does usually not hold. We have argued in our news scenario, that in practical applications peers often will not carry a random portion of the entire document collection. Though some newspapers like the New York Times will cover a wide area of topics, specialized newspapers like the Financial Times will limit the range and some publications can even provide corpora that essentially differ in the topics and keywords contained. Moreover, though a lazy dissemination in terms of effectiveness usually is comparable to the centralized approach for general queries, our indexing scheme focusses only on parts of the networks containing most promising documents, thus the collection-wide information has to be disseminated and (at least) regularly updated. Hence, classical Web search engines like Google crawl the Web and individually index the sites, but then all indexing information is transferred over the network and managed in a vast centralized repository for subsequent retrieval. Novel approaches to distribute Web search engines like Google desktop will have to deal with the same problem of dissemination this information efficiently. Therefore, though designed for peer-to-peer infrastructures, our work here can be assumed to have an interesting impact on future developments in distributed Web search engines.

6 Conclusion and Further Work

In this paper we have discussed the important problem of efficiently querying federated library collections using peer-to-peer infrastructures especially if collection-wide information is needed. Though federations of libraries servers would benefit from more dynamic and easy to use the P2P paradigm, previous approaches were only able to support information searches based on exact meta-data matchings. This is because centralized indexing techniques have a high communication overhead, whereas local routing indexes cannot deal with collection-wide information. We have described a practical use-case scenario for the problem and have presented an innovative local indexing scheme which flexibly includes collection-wide information. Our novel indexes are not created in advance, but are maintained query-driven, i.e. we do not index any information which is never asked for. This allows our algorithm to scale, even in more volatile networks. Another improvement is our introduction of a seperate category index that allows to prune large portions of the network and thus also enhances scalability.

To ensure result quality in networks of library collections, our future work will also need to consider issues as trust and reputation of individual servers. Super-peers can do consistency checks by continuously taking samples and subsequently exclude outliers from delivering results. Moreover, while in this paper we restricted our analysis on a unique classification scheme for the network, it also essential to support mapping between heterogeneous classifications.

References

1. K. Aberer. P-grid: A self-organizing access structure for p2p information systems. In *In Proceedings of the Sixth International Conference on Cooperative Information Systems (CoopIS)*, Trento, Italy, 2001.
2. W.-T. Balke, W. Nejld, W. Siberski, and U. Thaden. Progressive distributed top-k retrieval in peer-to-peer networks. In *Proceedings of the 21st International Conference on Data Engineering (ICDE 2005)*, 2005.
3. F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen. PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. In *Twelfth IEEE International Symposium on High Performance Distributed Computing (HPDC-12)*. IEEE Press, June 2003.
4. R. Korfhage. *Information Storage and Retrieval*. John Wiley, New York, 1997.
5. Y. Li, Z. A. Bandar, and D. McLean. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knwoledge and Data Engineering*, 15(4), 2003.
6. J. Lu and J. Callan. Federated search of text-based digital libraries in hierarchical peer-to-peer networks. In *European Colloquium on IR Research (ECIR 2005)*, 2005.
7. W. Nejld, W. Siberski, U. Thaden, and W.-T. Balke. Top-k query evaluation for schema-based peer-to-peer networks. In *Proceedings of 3rd International Semantic Web Conference (ISWC 2004)*, 2004.
8. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *Proceedings of the 2001 Conference on applications, technologies, architectures, and protocols for computer communications*. ACM Press, 2001.
9. M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. HyperCuP—Hypercubes, Ontologies and Efficient Search on P2P Networks. In *International Workshop on Agents and Peer-to-Peer Computing*, Bologna, Italy, July 2002.
10. W. Siberski and U. Thaden. A simulation framework for schema-based query routing in p2p-networks. In *1st International Workshop on Peer-to-Peer Computing & DataBases(P2P& DB 2004*, 2004.
11. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on applications, technologies, architectures, and protocols for computer communications*. ACM Press, 2001.
12. C. Tang, Z. Xu, and M. Mahalingam. Peersearch: Efficient information retrieval in peer-peer networks. Technical Report HPL-2002-198, Hewlett-Packard Labs, 2002.
13. C. L. Viles and J. C. French. Dissemination of collection wide information in a distributed information retrieval system. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*, pages 12–20. ACM Press, 1995.
14. C. L. Viles and J. C. French. On the update of term weights in dynamic information retrieval systems. In *Proceedings of the 1995 International Conference on Information and Knowledge Management (CIKM)*, pages 167–174. ACM, 1995.
15. C. Wang, J. Li, and S. Shi. Cell abstract indices for content-based approximate query processing in structured peer-to-peer data systems. In *APWeb*, volume 3007 of *Lecture Notes in Computer Science*. Springer, 2004.
16. I. Witten, A. Moffat, and T. Bell. *Managing Gigabytes*. Morgan Kaufman, Heidelberg, 1999.
17. B. Yang and H. Garcia-Molina. Designing a super-peer network. In *Proccedings of the 19th International Conference on Data Engineering (ICDE)*, 2003.