20. Supporting Information Retrieval in Peer-to-Peer Systems

Wolf-Tilo Balke (L3S Research Center and University of Hannover)

This chapter focuses on information retrieval techniques in Peer-to-Peer infrastructures. Peer-to-peer systems are already being used for a vast number of applications in content exchange, but most searching is done by simple keyword lookups. In contrast information retrieval means that not only some more or less matching objects have to be retrieved, but a list of the best matching objects over the entire network given a user's information needs. Since the 1960ies the information retrieval community considers ways to efficiently and effectively query document collections and designs dedicated retrieval systems like e.g. SMART [96]. Usually a query is seen as a (possibly weighted) set of keywords a user specifies to express his/her information need. Documents that contain those (or sufficiently similar) keywords are considered to be relevant to the user's information need as expressed by the query. Thus, for information retrieval in Peer-to-Peer infrastructures the challenge is not only to retrieve documents efficiently, but also to effectively find a set of best matching objects. Usually the degree of effectiveness is measured by a precision-recall analysis, where the precision of a retrieval algorithm is given by the ratio of relevant delivered documents with respect to the size of the delivered result set (i.e. number of correctly retrieved documents divided by the number of all retrieved documents). The recall is given by the ratio of the relevant delivered documents with respect to the all relevant documents in the collections (i.e. number of correctly retrieved documents divided by the number of all relevant documents).

20.1 Content Searching in Peer-to-Peer Applications

Peer-to-peer (P2P) systems are highly distributed computing or service substrates built from thousands or even millions of typically non-dedicated nodes across the Internet that may flexibly join or leave the system at any time. In contrast to centralized system architectures Peer-to-Peer networks try to avoid central services and will usually share local resources like computing power [30] or storage space (e.g. Freenet [124]). They are characterized by a high resilience against failures of single nodes, good scalability by joining resources and a high degree of autonomy for each peer.

R. Steinmetz and K. Wehrle (Eds.): P2P Systems and Applications, LNCS 3485, pp. 337-352, 2005.

20.1.1 Exchanging Media Files by Meta-Data Searches

The first big application of Peer-to-Peer technology was the free exchange of music files (mostly mp3 files, but also a limited amount of videos) over the Internet. As early as 1999 Napster [436] offered a platform to exchange files in a distributed fashion. Peers could offer files for download and directly download files from other peers in the network. The Napster platform was not Peer-to-Peer technology in the strict sense, because Napster still relied on a central server administrating addresses of peers and lists of offered files. The files offered were, however, not moved to the server, but all downloads were directly initiated between two peers. The content searches in the Napster network were made on a restricted amount of meta-data like filename, artist, or song title. Matching this limited meta-data with a user's query keywords content searches thus only decided, if there was a peer offering an adequate file and ordered possible downloads by the expected quality of the download connection.

Since a central index approach could only handle the Napster network's enormous success in terms of scalability by hierarchies of peers and provided a single point of responsibility for the content, from 2000 on the Gnutella [249] network began to build a file exchange platform on a real Peer-to-Peer structure. Content searches were performed by flooding queries from the originating peer to all neighboring nodes within a certain radius (the time to live (TTL) for a query). Also this approach proved not to be scalable beyond a certain point and the Gnutella network spectacularly broke down in August 2000 because of heavy overloads in low bandwidth peers. This breakdown led to the introduction of load-balancing and the construction of schema-based networks (Fast Track, e.g. KaZaA [343] or Morpheus [432]), where a backbone of high bandwidth peers (so-called super-peers) takes a lot of the query routing responsibility. Search in schema-based Peer-to-Peer networks will be discussed in a different chapter of this book.

20.1.2 Problems in Peer-to-Peer Information Retrieval

Previous applications for media exchange dealt mostly with exact or substring matching of simple meta-data descriptions of media files. When it comes to the exchange of (predominantly) textual documents , meta-data is not enough, but fulltext searches have to be supported. Though meta-data can capture some basic traits of a document (e.g. that some text is a 'newspaper article' related to 'sports'), they cannot anticipate and capture all the aspects of a text a user might be interested in. Thus, in information retrieval all terms that can be in some way important for a document should be searchable (i.e. indexed). The second major difference to meta-data-based retrieval is that information retrieval cannot use an exact match retrieval model, but has to rely on ranked retrieval models. These models introduce the notion of a

certain degree of match of each document with respect to the query. The higher the degree of match the better the document is expected to satisfy a user's information need. Thus in contrast to simple media file exchanges, where the connection speed was the most interesting factor for choosing a peer for download, information retrieval really has to find the document with best degree of match from any peer within the entire network. A well balanced expected precision and recall of such a content search is thus a major indicator for the effectiveness of the search capabilities.

Very early information retrieval research encountered the necessity to not only take information in each document into account, but also use some background information regarding the entire collection, prominently e.g. the discriminatory power of each keyword within the specific collection. For example considering different news collections, the occurrence of the keyword 'basketball' in a document will have a good discriminatory power in a general news collection, a severely lesser power to discriminate between documents in a sports news collection and virtually no discriminative power within a collection of NBA news. One of the most popular information retrieval measure thus is the well-known TFxIDF type. This measure is a combination of two parts (typically with some normalizations), 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 documents of the specific collection). Intuitively a document gets more relevant the more often the query term(s) occur in the document and the less often the query terms occur in other documents of the collection (i.e. the more discriminating query terms are with respect to a collection). Though TF can be determined locally by each peer, the IDF measure needs to integrate collection-wide information and cannot be determined locally. A typical instance of the TFxIDF measure (with $s_q(D)$ as the score for query term q in document D and N as the total number of documents in the collection) is e.g. given by:

$$s_q(D) := \frac{TF_q(D)}{max_{t \in D}(TF_t(D))} * \log(\frac{N}{DF_q})$$

Collection-wide information is thus essential to provide proper document scores. In information retrieval research the problem of disseminating collection-wide information was first encountered when retrieval systems moved beyond stand-alone systems 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, even if new documents are included into the collection [607]. The required level of dissemination, however, was found to be dependent on the document allocation throughout the network [606]: random allocation calls for low dissemination, whereas higher dissemination is needed if documents are allocated based on

content. In Peer-to-Peer networks such a random-like distribution does usually not hold. In practical applications peers often will not carry a random portion of the entire document collection, but rather a set of documents that represent their individual interests.

Consider e.g. the news servers example from above. In the WWW there are some big centralized news serves like e.g. CNN or the New York Times that deal with all kinds of news, i.e, cover a wide range of topics. Even if news items change, the overall distribution of topics and keywords can be assumed to change only slowly. In contrast in Peer-to-Peer applications peers usually only provide a couple of sets of topically close documents they are interested in. That means that if a peer joins or leaves the network the collection-wide information may considerably change. Thus, a lazy dissemination in settings like the WWW 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 like in Peer-to-Peer applications, the collection-wide information has to be disseminated and regularly updated. On the other hand this information does not necessarily always 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.

Thus, in contrast to previous work in distributed information retrieval, not only the distributed aspect of the retrieval, but also the peers' autonomy and the relatively high network churn are major problems in Peer-to-Peer information retrieval. The problems for information retrieval can be roughly classified into four main categories:

- **Ranked Retrieval Model:** Exact match models will immediately lead to a valid result object once any document has been encountered fulfilling the query predicate. When answering queries in a ranked model a large number of documents have to be compared to find the 'best matching' document or the peers offering them. Moreover, queries often consist of a conjunction of different keywords to express a user's information need, hence the retrieval model has to allow for assessing a document's degree of match for complex queries.
- **Efficient Evaluation Scheme:** An efficient ranked retrieval model does not allow for simply flooding queries until a suitable peer for download is found, because a prohibitive number of peers would have to be addressed. Since the best matching document to a query could be encountered querying the most distant peer, guaranteeing correct retrieval results by flooding queries can only be facilitated by addressing every peer in the network. An efficient evaluation scheme thus needs ways to select most appropriate peers for querying.
- **Reliability Facing Network Churn:** Centralized index structures for the Peer-to-Peer network can provide all necessary information about what documents are available, but this solution does not scale, provides a single

point of failure and needs a high communication overhead for keeping track of content changes in the network (e.g. peers changing their local content by adding or deleting documents or peers joining or leaving the network). Thus distributed index structures that are still reliable even in the face of network churn, have to be used.
– **Integration of Collection-Wide Information:** Queries cannot be answered by the individual peers having only local knowledge, but a peers needs up-to-date collection-wide information for correct scoring. Constantly disseminating this collection-wide information needs a high amount of bandwidth, if the network is rather volatile, with a high number of peers joining or leaving the network. Moreover, quick dissemination is necessary, if peers show a certain locality in their interests and provide document collections for specific topics, instead of a broad variety that resembles the topic distribution of the network.

20.1.3 Related Work in Distributed Information Retrieval

The problem of distributed information retrieval occurred already early in information retrieval literature and was mainly concerned with the merging of results and database content discovery. Together with the emergence of the World Wide Web as a highly distributed information source the research was intensified and the following paragraphs will revisit some important approaches that are also common in today's Peer-to-Peer information retrieval.

Abstracts of Information Sources. To support distributed information retrieval individual collections often have to send abstracts to a central (or distributed) repository. The abstract of a collection is usually simply the set of terms in the collection's inverted index. The most renown technique of efficiently representing these abstracts are Bloom filters [77]. A Bloom filter is data structure in the form of a bit vector compactly representing a set and allowing membership queries. In our case the set represented is the set of terms in a peer's inverted index. The filter computed is created by deriving n different indexes for each term using n different hash functions, each yielding a bit location in the vector. All bits at these positions in the Bloom filter are set to 1. The membership of a query term now can be efficiently determined by hashing the query term using the same n functions and comparing it bitwise with the filter. If there exists a position where a bit is set for the query term, but not in the filter, the query term is definitely not a member of the peer's abstract, which was used for creating the filter. Otherwise, with a certain probability it is member of the peer's abstract (Bloom filters allow for false positives. The probability of false positives is decreasing with growing n). Today Bloom filters are a popular technique for exchanging summaries of a peer's document collection. The PlanetP system for instance uses such

Bloom filters for retrieval and disseminates them throughout the community using gossiping algorithms [140].

Collection Selection. If no central index of all collections' contents is given, choosing 'just the right' collections for querying is a major problem. For the use in distributed environments like the WWW several benefit estimators for collection selection have been proposed. Basically these estimators use aggregated statistics about the individual collections to estimate the expected result quality of each individual collection. Expected qualities can then be used for deciding which collections to select for querying or for determining a querying sequence of the collections. The most popular benefit estimator is the CORI measure [101], which computes the collection score s_i for collection i with respect to a query q as:

$$s_i := \sum_{t \in q} \frac{\alpha + (1-\alpha) * T_{i,t} * I_{i,t}}{|q|}$$

$$\text{with } T_{i,t} := \beta + (1-\beta) * \frac{\log(cdf_{i,t}+0.5)}{\log(cdf_{i,t}^{max}+1.0)} \text{ and } I_{i,t} := \frac{\frac{\log(n+0.5)}{cf_t}}{log(n+1.0)}$$

where n is the number of collections, cdf the collection document frequency, cdf^{max} the maximum collection document frequency and finally cf_t denotes the collection frequency of query term t, i.e. the number of collections that contain the term. See [101] for appropriate choices of α and β.

Later [100] proposed to use a different formula for computing $T_{i,t}$ subsequently leading to better results:

$$T_{i,t} := \frac{cdf_{i,t}}{cdf_{i,t}+50+150*\frac{|V_i|}{|V^{avg}|})}$$

where V_i is the term space of the collection i, i.e. the distinct terms in the collection's inverted index. V^{avg} is the average term space of all collections whose inverted index contains term t. However, it is important to notice that statistics like the collection frequency cf_t or the average term space size V^{avg} have to be collected over all peers. That means they are collection-wide information that cannot be determined locally but has to be disseminated globally or estimated. Also the CORI estimators are widely used in Peer-to-Peer information retrieval, because they allow choosing collections with a sufficient quality, while having to exchange only a very limited amount of statistical data.

Metacrawlers. Closely related to the field of collection selection are so-called metacrawlers like e.g. GlOSS [259] (shorthand for Glossary of Server Servers). Metacrawlers have been designed in connection with the text database discovery problem, i.e. the problem of selecting most promising document collections from the WWW with respect to a query. The basic idea is that a metacrawler does not crawl the actual document collection and build a complete index over the documents, but rather collects only metadata about the individual collections like the number of documents in each

collection and how many documents for each keyword (above a certain critical threshold number) in a collection are present. Abstracting from the actual information which document contains the keyword, the indexes build by the metacrawler are much smaller than inverted keyword indexes, however, of course due to the aggregation of information also less reliable. For instance the information whether keywords appear conjunctively in any document of the collection is lost. But the resulting index can be handled centrally and the meta-data used for giving probabilities of finding suitable documents in each collection.

In GlOSS the usefulness of a collection for single keyword queries can be characterized by the number of documents that contain the keyword normalized by the total number of documents the collection offers. Building on the assumption that keywords appear independently in documents, the usefulness for multi-keyword queries is given as the product of the normalized numbers for each individual keyword [259]. This basic text database discovery using a central glossary of servers supports boolean retrieval and retrieval in the vector space model (vGlOSS). Experiments on the GlOSS system show that average index sizes can be reduced by about two orders of magnitude and produced a correct estimation (compared to a complete inverted document index) of the most useful collections in over 80% of cases. But still, since the glossary index is a central index, it needs to be updated every time a collection changes and thus does not lend itself easily to information retrieval in Peer-to-Peer infrastructures.

Although the work on distributed information retrieval and metasearch is definitely relevant related research, it addresses only the problem of integrating a small and typically rather static set of underlying retrieval engines and information sources. Such a small federation of systems is of course less challenging than a collaborative search process in highly dynamical Peer-to-Peer systems. We will take a closer look at specific techniques used in Peer-to-Peer infrastructures in the following sections.

20.2 Index Structures for Query Routing in Peer-to-Peer Infrastructures

Since traditional index structures cannot be readily employed in Peer-to-Peer systems, distributed paradigms must be used to find those peers in the network which offer suitable documents. Information retrieval queries then have to be routed directly to those peers. As stated before, given the network churn in typical Peer-to-Peer applications, the overhead of maintaining indexes in the presence of churn is a particularly important aspect.

20.2.1 Distributed Hash Tables for Information Retrieval

The simplest method of querying peer to peer systems is flooding queries iteratively from the querying peer to all adjacent peers until a certain number of hops (the 'time to live' for the query) is reached. While this solution is simple and robust even when peers join and leave the system, it does not scale and will only provide query answers within a limited radius around the querying peer. This can be fundamentally improved if content-based routing is allowed in the network. One of today's main technique for indexing such Peer-to-Peer systems are so-called distributed hash tables (DHTs) (see e.g. [505], [575]) which allow to route queries with certain keys to particular peers containing the desired data without requiring a central index. Typically, an exact match keyword search can be routed to the proper peers in a limited number of hops logarithmic of the network size, and likewise no peer needs to maintain more than a logarithmic amount of routing information. But to provide this functionality, all new content in the network has to be published at the node for the respective key, if new data on a peer arrives or a new peer joins the network. In case a peer leaves the network, the information about its content has to be unpublished. Moreover, if a new document is added to any peer's collection, it will usually contain a large set a of various terms that need to be indexed. Since in DHTs a hashing function decides on what peer the index for each term resides, chances are that a considerable number of peers holding some part of the DHT have to be addressed to fully publish all the information about the new document, see e.g. [237].

Recent research in [393] shows that due to the publishing/unpublishing overhead, distributed hash tables lack efficiency when highly replicated items are requested. In practical settings, they have shown to perform even worse than flooding approaches degrading even further, if stronger network churn is introduced. Therefore, first hybrid Peer-to-Peer infrastructures have been proposed [394] that use DHTs only for less replicated and rare items, where DHTs are efficient, and rely on flooding in the rest of cases. But for the use in practical scenarios, recent investigation of file exchange behavior [113] show that rare items are also rarely queried ('People are looking for hay, not for needles'). Usually, the querying behavior in practical applications follows a Zipf distribution: there is a moderate number of popular items containing many replicas in the network, and a long tail of rarely queried items containing few replicas. Thus, though having a large potential in speed-up by using DHTs in queries for rare items, relying on flooding for the majority of queries does not seem a sensible approach and cannot support information retrieval queries.

Another problem with distributed hash tables is that the retrieval uses exact matches of single keywords, whereas information retrieval queries are usually conjunctions of several keywords. If such a query has to be answered using DHTs the peers offering content for each of the keywords have to be retrieved [347]. The intersection of the individual peer lists then may offer

documents also relevant to the conjunctive query. However, there is still no guarantee that a peer in this intersection offers relevant content because the publishing of the peer for each keyword may have been based on different documents. Thus even if a peer offers content for each single keyword, it is not clear whether it offers a single document containing the conjunction of the keyword. Of course, the documents of the respective keyword have to be evaluated with a suitable scoring function to assess their degree of match and thus the ranking of the final result. Obviously, this process is no efficient solution to the information retrieval challenge in Peer-to-Peer infrastructures. Moreover, typical search processes in document collections like browsing navigation or prefetching are complicated by the virtualization of the namespaces by DHTs (see e.g. [347] for a discussion).

20.2.2 Routing Indexes for Information Retrieval

A sophisticated strategy for accurately finding very commonly queried items can be provided using so-called routing indexes . A routing index is a local collection of (key, peer) pairs where the key is either a keyword or a query. The basic notion of a routing index is that in contrast to flooding all neighbors or selecting some randomly, the index points to an interesting peer or in the direction of interesting peers for a query. Peers thus can route a given query along connections that lead to collections of peers relevant for a query. Usually, it is distinguished between links in the default network pointing only in the direction of peers holding interesting collections and real links to some specific peers, forming an overlay and often referred to as shortcuts (since peers in the index do not have to be directly adjacent to the peer keeping the index). A topic-specific clustering of shortcuts represents a semantic overlay that may be completely independent from the underlying physical network.

Routing indexes were first introduced by [137] with the goal to choose best neighbors of a peer to forward a query to until the desired number of results is reached. While this approach only focused on routing, a lot of research soon focused on directly contacting relevant peers using semantic characteristics, like [274] or [444]. Subsequently, routing indexes were extended to different uses in Peer-to-Peer systems like top k retrieval [54]. The maintenance of such a routing index is of only local nature (that means that no publishing/unpublishing overhead like in DHTs is caused), and the recall for the indexed items is usually quite high. Since users in Peer-to-Peer environments are usually interested in popular queries and show a certain consistency in their interests, a routing index in most applications is a good solution.

But the question arises how to construct local indexes in a manner that is both effective in recall and efficient in performance. It is clear that in order to be effective in terms of recall, the local indexes should have a large amount of knowledge of the collections on different peers. On the other hand,

having collected a large amount of knowledge calls for constant updates to keep track of changes in the collections. Different routing index policies have been proposed to tackle this trade-off. Generally, it can be distinguished between restricted index sizes and unrestricted index sizes. For restricted index sizes index entries are collected and exchanged, if the maximum index size is reached but new information has been gathered. Getting rid of stale index entries is thus implemented by letting index entries compete with respect to their expected usefulness. One of the most often used strategy here is the LRU-strategy ('least recently used') that assigns higher usefulness to those index entries that have been successfully used in the recent past. The more recent, the more accurate and thus the more useful. However, the optimal size of such restricted indexes is a difficult problem and strongly dependent on the network's actual volatility that is hard to determine locally.

In terms of unrestricted indexes, the peers keeping them locally have to combat network churn in a different way. For structured networks, the work on distributed retrieval in [54] proposes to use a backbone of superpeers for query routing where each superpeer keeps only a strictly locally maintained routing index. Queries are always routed along a minimum spanning tree of the backbone and the individual results are routed back the same way. Each routing index contains the recently asked query terms, all local peers that contributed to the result set of best documents, and the direction in terms of adjacent superpeers, where high quality documents came from. If a superpeer does not have a matching index entry for a query term, it forwards the query in all directions along the backbone and collects accurate information for its index, when the best documents have been determined and the results are routed back. Each index entry is assigned a certain time to live to combat network churn. Experiments show that this kind of index maintenance is especially suitable for Zipfian query frequency distributions like often encountered in Peer-to-Peer scenarios, where it essentially reduces the number of contacted peers and gives a recall comparable to central indexes.

Another approach using unrestricted index sizes are InfoBeacons [131] mainly maintaining a set of local indexes that are loosely coupled to the document sources on the peers. Like in the GlOSS approach [259] the indexes save space by containing only statistics about the documents in the underlying collections. This statistics can then be used to compute the expected usefulness of each known document source. To combat network churn or changes in underlying document collections [131] proposes to apply a 'forgetting factor' that periodically weighs down stale information about a source until it is finally 'forgotten'. In the same way the statistics about a source can be refreshed by evaluating answers to recent queries of that source (a so-called 'experience factor'). By closely investigating the result documents, an InfoBeacon index can not only learn about the query terms, but also about other terms that are contained in some result document.

20.2.3 Locality-Based Routing Indexes

A different approach for routing queries to most interesting document collections is based on the idea of social metaphors. Inspired by information retrieval processes between people in real life, social networks of peers can be used to get queries to the most relevant peers. Experiments about how information is gathered, showed that people in real world settings are quite capable of efficiently constructing chains of acquaintances that will bring them at least close to the desired information, though each person only had very local knowledge of the network (see e.g. [413] or [353]). The resulting so-called 'small worlds' have subsequently been employed to retrieve relevant information with respect to a peer's information need as expressed by the peer's queries.

Using small worlds for retrieval an often made assumption is the principle of interest-based locality. It posits that, if a peer has a particular document that is relevant to a query, it might very probably also have other interesting items that the querying peer is interested in. Building on this principle [571] studied interest-based overlays over Gnutella-style networks and proposes to generate interest-based shortcuts connecting querying peer and content-providing peer. Queries with semantically close information needs can then rely on already established shortcuts. Traces on practical data collections illustrate how peers in the overlay network get very well connected and that the overlay graph shows the highly-clustered characteristics of small world networks with a small minimum distance between any two nodes. The clustering coefficient of a peer is defined to be the fraction of edges that exist between its neighbors over the possible total number of edges. The clustering coefficient is a relevant measure for the degree of the small world characteristic as reflected by the network. Clusters in the shortcut graph can be said to correspond to clusters of interests and peers looking for content within their usual areas of interest, will be successful with high probability by using their shortcuts.

Other work, like e.g. [586], directly relied on social metaphors for routing queries to peers that can be assumed to offer interesting documents. A peer builds an index of shortcuts e.g. by 'remembering' content provider that have offered relevant documents in the past for a query or for queries on semantically similar topics, or by 'overhearing' communications between other peers that are just routed through the peer. Best peers that are likely to offer relevant documents, can then be queried by just following the shortcuts whose topic best matches the query semantics. Randomly sending queries also to some peers from the default network helps to extend the knowledge about relevant peers and is a limited help facing the problems of interest changes and network churn. Experiments show that such shortcut-based approaches can offer a decent recall and dramatically reduce the communication needed for answering queries.

20.3 Supporting Effective Information Retrieval in Peer-to-Peer Systems

20.3.1 Providing Collection-Wide Information

As has been stated, providing collection wide is essential for the retrieval effectiveness. 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.

The PlanetP system [140] 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 (IPF). The inverted peer frequency estimates for all query terms, which peers are interesting contributors to a certain query. For each query term t the inverted peer frequency is given by $IPF_t := \log(1 + \frac{N}{N_t})$ where N is the number of peers in the community and N_t is the number of peers that offer documents containing term t. In PlanetP summarizations of the content in the form of Bloom filters are used to decide what content a peer can offer. Since these are eagerly disseminated throughout the network by gossiping algorithms, each peer can locally decide values for N and N_t. The relevance of a peer for answering multi-keyword queries is then simply the sum of the inverted peer frequencies for all query terms. Peers are then queried in the sequence of their IPFs and the best documents are collected until queried peers do no longer improve the quality of the result set. In terms of retrieval effectiveness [140] show that the approach is quite comparable to the use of inverted document frequencies in precision and recall and also the documents retrieved using IPF show an average overlap of about 70% to result sets retrieved using IDF. However, by using gossiping to disseminate Bloom filters the system's scalability is severely limited.

Structured Peer-to-Peer infrastructures allow for a more scalable way of providing collection-wide information than simple gossiping. Based on the notion that in answering a query current collection-wide information is only needed for the query terms, each superpeer can disseminate such information together with a query. [53] shows for a setting of distributed servers hosting collections of newspaper articles that employing an index collecting information like IDFs for certain query terms in parallel to the query routing index can provide sufficiently up-to-date collection-wide information. The basic idea of both indexes is the same: the routing index of a super-peer states what peers are interesting to address for a given query and the CWI index provides collection-wide data for each keyword. The data in the CWI index can change in two ways: like in routing indexes existing entries have only a certain time to live, such that stale entries are periodically removed. On the

other hand it can be updated evaluating the answers of the peers that the query was forwarded to. These peers can easily provide the result documents together with local statistics about their individual collections. This statistical information can then be aggregated along the super-peer backbone to give an adequate snapshot of the currently most important document collections for a keyword (e.g. document frequencies and collection sizes can be added up). As stated in [607] the collection-wide informations does usually only change significantly, if new peers join the network with corpora of documents on completely new topics. Since index entries only have a certain time to live, occasionally flooding queries about query terms not in the index (and disseminating only an estimation of the statistics needed), usually refreshes the CWI index sufficiently, while not producing too many incorrect results. Experiments in [53] show that by using an CWI index and disseminating the collection-wide information together with the query, even in the presence of massive popularity shifts the CWI index recovers quickly.

20.3.2 Estimating the Document Overlap

As another important factor for supporting the overall retrieval quality is assessing the novelty of collections as e.g. motivated in [66]. In collection selection approaches usually precomputed statistics about the expected quality of results from a collection is used to minimize the number of collections that have to be accessed. Minimizing the number of collection accesses (and thus the necessary communication) is even more important in Peer-to-Peer settings. Given typical popularity distributions with a high amount of replication of popular items in today's file sharing applications [113], it seems probable that also in document exchange such overlap between the individual peers' repositories will exist. However, accessing promising peers in an information retrieval process that show a high overlap in their collection is not going to improve the result sets. When deriving result sets from distributed sources, like e.g. in [54], the result merging will ignore documents that have occurred before and simply put out requests (and thus probably contact more peers) for more answers until enough distinct documents have been found.

The novelty of a collection a new peer offers always has to be computed with respect to a reference collection, i.e. collections that are already part of the querying peer's local routing index or more general the collection of already returned result documents. [66] defines the novelty of a peer p's collection C_p with respect to a reference collection C_{ref} as:

$$Novelty(C_p) := |C_p| - |C_p \cap C_{ref}|$$

However, since there is usually no information disseminated exactly what documents are given by a certain peer, this information has to be approximated by the information disseminated. Thus, for estimating what is actually

in a peer's collection with respect to multi keyword queries the index lists or summaries of the peer have to be investigated. Using Bloom filters as summaries [66] proposes to build a peer p's combined Bloom filter b_p with respect to the query as the bitwise logical AND of its filters for the individual keywords and then estimate the novelty by comparing it to $b_{prev} := \bigcup_{i \in S} b_i$ as the union of those Bloom filters b_i of the set of collections S that have already been investigated previously during the retrieval process. The degree of novelty can then be approximated by counting the locations where peer p's Bloom filter gives set bits that are not already set in the combined filter of previous collections:

$$|\{k|b_p[k] = 1 \wedge b_{prev}[k] = 0\}|$$

Analogously, the overlap between the collections can be estimated by counting the number of bits that are set in both filters. Of course this is only a heuristic measure as the actual documents have been abstracted into summaries. Having the same summary, however, does not imply being the same document, but only being characterized by the same keywords. That means those documents are probably not adding new aspects for the user's information need as expressed in the query. Generally speaking estimating the overlap and preferably querying peers that add new aspects to an answer set is a promising technique for supporting information retrieval in Peer-to-Peer environments and will need further attention.

20.3.3 Prestructuring Collections with Taxonomies of Categories

Retrieval in Peer-to-Peer systems considered two different kinds of paradigms: the meta-data-based queries and the fulltext-based queries. Often it is useful to consider them not as two orthogonally used paradigms, but to integrate them into a single query. A major problem in information retrieval where such an integration is helpful, is for instance the disambiguation of query terms. In Peer-to-Peer systems offering documents that show a certain similarity in terms of their types (like collections of newspaper articles, etc.), the retrieval process can essentially be supported by introducing a common system of categories that classify the documents. Given that categories are usually not entirely independent of each other a taxonomy of the categories can find related categories that are semantically closer than others. A query then can be given using keywords and the category the result documents should be in. The approach given in [53] shows that queries have to be answered in each category separately starting with the category specified in the query. Thus, the query routing index has to contain also category information. If no sufficient number of documents can be retrieved from that category the search has to be extended first to the children of the category and then to its parents. For each category own collection-wide information has to be

collected and disseminated, e.g. by building CWI indexes as described above per category.

Following [382] the semantic similarity for different categories c_1 and c_2 can be considered to be determined by the shortest path length as well as the depth of the common subsumer:

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

where l is the shortest path between the topics in the taxonomy tree, h is the depth level of the direct common subsumer, and $\alpha \geq 0$ and $\beta > 0$ are parameters scaling the contribution of shortest path length and depth, respectively. Using optimal parameter ($\alpha = 0.2$ and $\beta = 0.6$) this measure shows a correlation coefficient with human similarity judgements performing nearly at the level of human replication. Experiments in a scenario of federated news collections in [53] show that the retrieval process can be effectively supported, if documents can be classified sufficiently well by a taxonomy of common categories.

20.4 Summary and Conclusion

This chapter has given a brief survey of techniques for information retrieval in Peer-to-Peer infrastructures. In most of today's applications in Peer-to-Peer scenarios simple retrieval models based on exact matching of meta-data are prevalent. Whereas meta-data annotation has to anticipate the use of descriptors in later applications, information retrieval capabilities work on more complex and unbiased information about the documents in each collection offered by a peer. Thus, such capabilities offer much more flexibility in querying and open up a large number of semantically advanced applications.

Generally speaking, information retrieval differs from simple meta-data-based retrieval in that a ranked retrieval model is employed where not only some suitable peer for download needs to be found, but the 'best' documents within the entire network must be located. Moreover and in contrast to Gnutella-style infrastructures, querying has to be performed in a more efficient manner than simple flooding. Generally, only a small number of peers should be selected for querying. In addition, the querying method has to be relatively stable in the face of network churn and since rankings usually rely on collection-wide information, it has to estimated or efficiently disseminated throughout the network.

The basic retrieval problem is heavily related to previous research in distributed information retrieval as is used for querying document collections in the WWW. But the Peer-to-Peer environment still poses different challenges, especially because network churn causes a much more dynamic retrieval environment and centralized index structures cannot be efficiently used. Also, related work in Peer-to-Peer systems, e.g., distributed hash tables can not be

readily used either due to the limitations in scalability caused by publishing and unpublishing information in more volatile networks.

Thus, the main problem for Peer-to-Peer information retrieval today is managing the trade-off between the efficient maintenance of local indexes with only limited knowledge about the Peer-to-Peer network's global parameters and the expensive dissemination of dynamically changing global information about the network needed to guarantee a satisfying recall in result sets. Heuristic techniques like estimating the document overlap of collections or integrating taxonomies of document classifications into the retrieval process, have been proved to be helpful and should be further investigated.