Supporting Information Retrieval in Peer-to-Peer Systems

Part 2

Wolf-Tilo Balke and Wolf Siberski

16.1.2008
Overview

1. Content Searching in Peer-to-Peer Applications
   1. Problems in Peer-to-Peer Information Retrieval
   2. Related Work in Distributed Information Retrieval

2. Index structures for Query Routing
   1. Distributed Hash Tables for Information Retrieval
   2. Routing Indexes for Information Retrieval
   3. Locality-Based Routing Indexes

3. Supporting Effective Information Retrieval
   1. Providing Collection-Wide Information
   2. Estimating the Document Overlap
   3. Prestructuring Collections with Taxonomies

4. Summary and Conclusion
1 Applications of P2P Technology

- **File sharing** was main application area of P2P technologies
  - Retrieval by simple metadata/keyword lookups
  - Exact match model, substring matching
  - Music sharing (Napster, Gnutella, KaZaa, …)
    - Mp3 files, small videos, etc.
    - Metadata like encoding quality or playing time
    - Keywords like song title, artist, and full text description

- **Old applications** used ‘more or less’ P2P infrastructures
  - Napster – central index lookups, content on peers
  - Gnutella – hierarchy of peers (supernodes)
  - From 2000 ‘real’ schema-based P2P-structure with load-balancing (Fast Track, e.g. KaZaa, Morpheus)
1.1 Information Retrieval in P2P Systems

- Information Retrieval deals with complex documents
  - Meta-data can only capture some aspects of a document, but not anticipate all semantic searches
    - E.g. sports-related newspaper article, but no names, locations, etc.
  - Support for full-text searches needed
  - Ranked retrieval model
    - Similarity between documents
    - Degree of match with respect to query or user’s information needs

- Find the best-matching document from the best-connected peer
  - Unlike in file sharing emphasis is on the document quality
  - If there are multiple sources offering similar quality documents, choose best peer according to connection, etc.
1.1 Challenges

- Challenges due to distributed nature of P2P environment
  - Efficient query evaluation scheme
    - How to disseminate a peer’s query?
    - Central inverted index of documents is expensive to maintain
    - Simple flooding of all queries is not scalable, if not just some matching, but ‘best’ documents have to be found
  - Dealing with network churn
    - A peer can always alter the set of documents offered, or significantly change individual documents
    - Peers may join and leave the network, i.e. whole document collections may disappear, or can be added
1.1 Challenges 2

Integration of collection-wide information

- Peers are not able to calculate IR-style scorings from local knowledge, but needs some knowledge from the (virtual) merged collection
- Constant dissemination of collection-wide information needs a lot of bandwidth

Example: Popular IR measure TFxIDF:

\[ s_q(D) := \frac{TF_q(D)}{\max_{t \in D} (TF_t(D))} \cdot \log\left( \frac{N}{DF_q} \right) \]

Term frequency (TF) component can be calculated locally, inverse document frequency (IDF) needs information about the total (i.e. collection-wide) number of documents, where the term occurs.
1.1 Example: Problem of Collection-wide Information

- Example: Different news collections, query on keyword ‘basketball’
  - General news collection, e.g.
    - Many articles, only few about basketball, therefore IDF small
    - Keyword discriminates well between articles
  - NBA news collection
    - Few articles, almost all about basketball, therefore IDF high
    - Keyword hardly discriminates between articles
  - Merged collection: IDF medium
    - But how do independent collections (peers) exchange their information?
1.1 Example: Problem of Collection-wide Information

Top object

Querying Peer

global scoring

all objects identical

TF = 1  IDF = 6/3

local scoring

TF = 1  IDF = 3/2

TF = 1  IDF = 3/1

TF = 1  IDF = 3/2

local scoring

TF = 1  IDF = 3/1

TF = 1  IDF = 3/1

TF = 1  IDF = 3/2

local scoring
1.2 Distributed Information Retrieval

- Distributed information retrieval techniques grew increasingly important for searching Web sources
  - Abstracts of information sources
    - To support distributed retrieval sources have to register abstracts or keyword sets
    - Abstracts can either be kept in a central repository or distributed by gossiping algorithms, e.g., PlanetP [Cuenca-Acuna et al., ‘03]
  - Collection selection
    - Having no central index needs a sophisticated way of choosing the most promising collections for querying
1.2 Distributed Information Retrieval

- Such abstracts can be compactly represented by Bloom Filters, i.e. bit vectors that allow membership queries
  - Each term is hashed with $n$ different functions and the position in the bit vector for each hash value is set to 1
  - Allows for false positives, but no false negatives
  - In Counting Bloom Filters objects can also be removed

$$\{x, y, z\}$$

```
0 1 0 1 1 1 0 0 0 0 0 1 0 1 0 0 1 0
```

$w?$
1.2 Distributed Information Retrieval

- **Benefit estimators** for collection selection use aggregated statistics about individual collections for selection, e.g. CORI measure [Callan et al., ‘95]

CORI calculates collection score $s_i$ for collection $i$ w.r.t. query $q$:

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

with

$$T_{i,t} := \beta + (1 - \beta) \times \frac{\log(cdf_{i,t} + 0.5)}{\log(cdf_{i,t}^{max} + 1.0)}$$

and

$$I_{i,t} := \frac{\log(n+0.5)}{\log(n+1.0)}$$

where $n$ is the number of collections, $cdf$ the collection document frequency, $cdf^{max}$ the maximum $cdf$ and $cf_t$ the collection frequency of term $t$
1.2 Distributed Information Retrieval

- **Metacrawlers**
  - Designed in connection with selecting most promising text collections from the Web
  - Do not crawl the actual documents in each collection, but just collect some metadata about each collection (number of documents, most prominent keywords, etc.)
  - Indexes are much smaller, but also less accurate than the complete inverted keyword indexes
  - E.g. the Glossary of Server Servers (GLOSS) [Gravano et al., ‘99] reduces index sizes by two orders of magnitude, selects most useful collection in over 80% of cases
Overview

1. Content Searching in Peer-to-Peer Applications
   1. Problems in Peer-to-Peer Information Retrieval
   2. Related Work in Distributed Information Retrieval

2. Index structures for Query Routing
   1. Distributed Hash Tables for Information Retrieval
   2. Routing Indexes for Information Retrieval
   3. Locality-Based Routing Indexes

3. Supporting Effective Information Retrieval
   1. Providing Collection-Wide Information
   2. Estimating the Document Overlap
   3. Prestructuring Collections with Taxonomies

4. Summary and Conclusion
2  Index Structures for Query Routing

- Traditional index structures cannot be readily employed in P2P systems
  - High degree of distribution
  - High degree of volatility (churn)
  
- Distributed paradigms needed to route queries to appropriate peers
  - Simple flooding method does not scale
  - Distributed hash table lookup
  - Using indexed routing information
  - Using shortcut overlays
2.1 Distributed Hash Tables for Information Retrieval

- Distributed hash tables
  - Route queries to appropriate peers with number of hops logarithmic in network size
  - No peer needs to maintain more than logarithmic amount of routing information
  - But…
    - Exact match queries only
    - All new content has to be published, if peers join/change
    - Old content has to be unpublished, if peers leave
    - Documents added/removed will contain a lot of different terms to be published/unpublished. Thus, usually many index peers have to be addressed
    - Conjunction of query terms needs to access many peers, but there is still no guarantee that a single document with the conjunction exists
2.1 Distributed Hash Tables for Information Retrieval

- **Improvement: Hybrid P2P infrastructures [Loo et al.,’04]**
  - Efficiency of DHT is worst, if highly replicated items are requested
    - Experiments show worse behavior than flooding, degrading with churn
  - Querying and content allocation follow Zipf-distribution
    - Only few highly replicated and often queried items
    - ‘People are looking for hay, not for needles’ (S. Shenker)
  - Hybrid P2P infrastructures use DHTs only for the less replicated and rarely queried items, all other queries are flooded
  - Still, DHTs have to be maintained for the majority of query terms
2.2 Routing Indexes for Information Retrieval

- Routing indexes are local collections of (key, peer) pairs
  - *Key* is either a keyword or a query
  - *Peer* is the address of a peer that either offers relevant results, or routes the query to other peers with relevant result

- In contrast to flooding only ‘interesting’ directions are queried
  - Often distinguished between links in the default network (directions of content providers) and overlay structure of direct links to content providers (‘shortcuts’)

- First introduced by [Crespo & Garcia-Molina, ‘02] to choose best neighbors in the default network for query forwarding
  - Index maintenance is of local nature and index coverage is usually high due to Zipf distribution of requests
  - Correctness of index is influenced by network volatility/churn
2.2 Routing Indexes for Information Retrieval

- Routing index policies in the face of network churn
  - With *restricted index sizes* new entries are collected and always stored. If the maximum size is reached, some stale information is replaced
    - A simple strategy always replaces the currently oldest index entries
    - ‘Least recently used’ (LRU) strategy assigns higher usefulness to entries that have been successfully used recently
    - Optimal index size is a problematic parameter
  - Indexes with *unrestricted size* have to combat network churn differently
    - ‘time to live’ assigns an expiry time for each new index entry
    - ‘forgetting factors’ can periodically weigh down reliability of link information
2.2 An Algorithm for Correct Query Routing

- Goal: progressive distributed top-\(k\) ranking of documents

- Putting techniques together to design an efficient top-\(k\) algorithm
  - Minimal number of object transfers
  - Optimal number of object accesses

- Features of the P2P based approach
  - Optimized Query-Routing
  - No global Index
  - Query-driven term-indexing
2.2 Bird’s View

1. Distribute query through the network (Routing)

2. Every peer scores documents locally (Ranking)

3. Hierarchical construction of the final result (Merging)

4. Optimized query routing (Index)
2.2 Building Blocks

- Structured network
- Local ranking
- Query-driven index
- Result
- Merging

P2P Information Retrieval
2.2 Network Structure

- Observation: peers strongly differ in availability, bandwidth, computing power, ...
- Hierarchical network structure with super-peers
  - Query routing
  - Result merging
  - Indexes
2.2 Network topology

- Super-peers as hypercube (HyperCuP protocol)
- Resilient against leaving peers
- Broadcast with \((n-1)\) messages, \(\log_2(n)\) hops

![Diagram showing network topology with super-peers and minimal spanning tree](image)
2.2 Local Ranking

- Super-peer asks for local rankings of peers' collections
- Top-\(k\) results (plus metric-dependent information) are returned to SP
- Arbitrary similarity measures can be used
  - TFxIDF
  - Similarities in taxonomies
2.2 Result Merging

- Results will be merged at the super-peers
  - Unique scoring function
  - Maximum of k messages per SP-SP edge
2.2 Indexing

- Super-peers keep indexes
  - IDF (collection wide information)
    - IDF-values for query terms
  - Top peers (routing)
    - List of peers that already have contributed to a previous top-k result
  - Others possible, e.g. for taxonomies

- Index entries are query-driven
2.2 Routing Indexes Example: Top k Query Routing

- Example for routing indexes in P2P networks with super-peer backbone holding routing indexes
- Progressive P2P top-k algorithm [Balke et al., ‘04]
  - If query q is indexed, distribute query and collect results
  - Otherwise flood query and
    - Compute ranks at local peers
    - Merge results at super-peers
    - Use statistics for new entry in routing index (routing information, collection-wide information, etc.)

- Data structures at super-peers
  - RequestResults: Peers which are queried for result (index information)
  - BestPeer: Peers which delivered recent best result
  - TopRes: Current top results
  - Delivered: Delivered results
2.2 Routing Indexes Example: Top k Query Routing

Find top 2 documents

Empty routing index at SP_4
2.2 Routing Indexes Example: Top k Query Routing

<table>
<thead>
<tr>
<th>RequestResults</th>
<th>()</th>
</tr>
</thead>
<tbody>
<tr>
<td>BestPeers</td>
<td>{P2}</td>
</tr>
<tr>
<td>TopRes</td>
<td>{((P3, d31, 0.5), (P4, d41, 0.4))}</td>
</tr>
<tr>
<td>Delivered</td>
<td>{((P2, d21, 0.7))}</td>
</tr>
</tbody>
</table>

P0, P1, P2, P3, P4

q1

<table>
<thead>
<tr>
<th>d11</th>
<th>0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>d12</td>
<td>0.3</td>
</tr>
<tr>
<td>d13</td>
<td>0.2</td>
</tr>
</tbody>
</table>

P2, P3, P4

<table>
<thead>
<tr>
<th>d21</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>d22</td>
<td>0.4</td>
</tr>
<tr>
<td>d23</td>
<td>0.3</td>
</tr>
</tbody>
</table>

P4

<table>
<thead>
<tr>
<th>d21</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>d32</td>
<td>0.6</td>
</tr>
<tr>
<td>d33</td>
<td>0.1</td>
</tr>
<tr>
<td>d42</td>
<td>0.5</td>
</tr>
<tr>
<td>d43</td>
<td>0.2</td>
</tr>
</tbody>
</table>
2.2 Routing Indexes Example: Top k Query Routing

**RequestResults**

<table>
<thead>
<tr>
<th>SP1</th>
<th>SP2</th>
<th>SP3</th>
<th>SP4</th>
<th>SP5</th>
<th>SP6</th>
<th>SP7</th>
<th>SP8</th>
</tr>
</thead>
<tbody>
<tr>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
</tr>
</tbody>
</table>

**BestPeers**

<table>
<thead>
<tr>
<th>SP1</th>
<th>SP2</th>
<th>SP3</th>
<th>SP4</th>
<th>SP5</th>
<th>SP6</th>
<th>SP7</th>
<th>SP8</th>
</tr>
</thead>
<tbody>
<tr>
<td>{P1}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
</tr>
</tbody>
</table>

**TopRes**

<table>
<thead>
<tr>
<th>SP1</th>
<th>SP2</th>
<th>SP3</th>
<th>SP4</th>
<th>SP5</th>
<th>SP6</th>
<th>SP7</th>
<th>SP8</th>
</tr>
</thead>
<tbody>
<tr>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
</tr>
</tbody>
</table>

**Delivered**

<table>
<thead>
<tr>
<th>SP1</th>
<th>SP2</th>
<th>SP3</th>
<th>SP4</th>
<th>SP5</th>
<th>SP6</th>
<th>SP7</th>
<th>SP8</th>
</tr>
</thead>
<tbody>
<tr>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
</tr>
</tbody>
</table>

**q1** (d11, 0.8)

<table>
<thead>
<tr>
<th>d11</th>
<th>0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>d12</td>
<td>0.3</td>
</tr>
<tr>
<td>d13</td>
<td>0.2</td>
</tr>
</tbody>
</table>

**d11 0.8**

<table>
<thead>
<tr>
<th>d21</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>d22</td>
<td>0.4</td>
</tr>
<tr>
<td>d23</td>
<td>0.3</td>
</tr>
</tbody>
</table>

**d12 0.3**

<table>
<thead>
<tr>
<th>d31</th>
<th>0.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>d32</td>
<td>0.6</td>
</tr>
<tr>
<td>d33</td>
<td>0.1</td>
</tr>
</tbody>
</table>

**d13 0.2**

<table>
<thead>
<tr>
<th>d41</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>d42</td>
<td>0.5</td>
</tr>
<tr>
<td>d43</td>
<td>0.2</td>
</tr>
</tbody>
</table>
2.2 Routing Indexes Example: Top k Query Routing

SP1
RequestResults {} ...
BestPeers {SP2}
TopRes {((P1, d12, 0.3))}
Delivered {((P1, d11, 0.8), (SP2, d21, 0.7))}

q1 {((d11, 0.8), (d21, 0.7))}

P0

SP1

SP2

SP3

SP4

SP5

SP6

SP7

SP8

P1

P2

P3

P4

<table>
<thead>
<tr>
<th></th>
<th>d21</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>d22</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>d23</td>
<td>0.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>d31</th>
<th>0.6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>d32</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>d33</td>
<td>0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>d41</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>d42</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>d43</td>
<td>0.2</td>
</tr>
</tbody>
</table>

L3S Research Center

P2P Information Retrieval
2.2 Routing Indexes Example: Top k Query Routing

**SP1 Routing Index**

<table>
<thead>
<tr>
<th>SP1</th>
<th>Routing Index</th>
<th>q1</th>
<th>{P1, P2}</th>
</tr>
</thead>
</table>

**SP2 Routing Index**

<table>
<thead>
<tr>
<th>SP2</th>
<th>Routing Index</th>
<th>q1</th>
<th>{SP4}</th>
</tr>
</thead>
</table>

**SP4 Routing Index**

<table>
<thead>
<tr>
<th>SP4</th>
<th>Routing Index</th>
<th>q1</th>
<th>{P2, P3}</th>
</tr>
</thead>
</table>

**BestPeers**

{SP2}

**TopRes**

{(P1, d12, 0.3)}

**Delivered**

{(P1, d11, 0.8), (SP2, d21, 0.7)}

**RequestResults**

{}
2.2 Query Routing

- At the first appearance of a queries peers only send out their input for IDF computation
- Super-peers aggregate IDFs and build index
- Whenever a query is repeated
  - SPs will send recent IDF-values together with query terms
  - Peers will uses IDFs for local score computation
- Disadvantage: at first occurrence of query it has to be sent twice
  - Zipf-Distribution minimizes number of queries concerned
- Advantages:
  - No effort for maintaining global IDF index
  - Values for often occurring queries are kept up-to-date
2.2 Query Routing and Network Churn

- Send queries only to peers that have already recently contributed to answering a query
  - Problem: the networks and each peer’s volatility
    - Solution 1: Send queries also to a randomly selected set of peers
    - Solution 2: “Best before”-timestamp
2.3 Locality-Based Routing Indexes

- Refinement of routing indexes by social metaphors
- Similar retrieval process like in real life
  - Every person has only limited knowledge of the environment
  - Who knows about a certain topic?
  - Who might know other people who know about the topic?
  - Try to build (short) ‘chains of acquaintances’ that will bring you close to the requested information
- Aims at building ‘social networks’ as overlays
- Peers semantically connected by certain topics form ‘small world networks’, e.g. [Milgram, ’67; Kleinberg, ‘00]
- Paradigm of interest-based locality
  - If a peer has relevant content for a user’s query, it very often also has some other content that this user might be interested in
2.3 Locality-Based Routing Indexes

- For information retrieval in P2P network this enables new routing in interest-based overlay structures
  - Route queries to peers with documents matching semantically close queries
  - Traces on practical data collections show that
    - peers get well-connected
    - the overlay graph shows highly-clustered characteristics with a small minimum distance between any two nodes
  - ‘Overhearing’ of communications routed through a peer can be used to enhance its local index
  - Randomly sending queries also to peers from the default network helps to extend knowledge and can remedy the effect of network churn
Overview

1. Content Searching in Peer-to-Peer Applications
   1. Problems in Peer-to-Peer Information Retrieval
   2. Related Work in Distributed Information Retrieval

2. Indexstructures for Query Routing
   1. Distributed Hash Tables for Information Retrieval
   2. Routing Indexes for Information Retrieval
   3. Locality-Based Routing Indexes

3. Supporting Effective Information Retrieval
   1. Providing Collection-Wide Information
   2. Estimating the Document Overlap
   3. Prestructuring Collections with Taxonomies

4. Summary and Conclusion
3 Supporting Effective P2P Information Retrieval

- P2P information retrieval has to deal with the trade-off between
  - Efficient local maintenance of statistics / index information, where information can be stale (incorrect)
  - Expensive global maintenance of statistics / index information, where information always is accurate

- Needed is ‘just the right level’ of dissemination of statistics to guarantee a ‘sufficiently effective’ retrieval

- Some techniques help to support efficient retrieval
  - Providing adequate collection-wide information
  - Estimate document overlap between peers
  - Pre-structure collections by categories / taxonomies
3.1 Providing Collection-Wide Information

- Collection-wide information is important for retrieval quality, but cannot be calculated locally like e.g. IDFs
  - Some systems like e.g. PlanetP, do not use CWI directly, but circumnavigate the problem by using an inverted peer frequency
    \[ IPF_t := \log(1 + \frac{N}{N_t}) \]
    where \( N \) is the number of all peers and \( N_t \) is the number of peers offering documents on term \( t \)
  - If summarizations of peers (abstracts) are eagerly disseminated, each peer can locally decide values for \( N \) and \( N_t \)
  - The relevance of peers in multi-keyword queries is simply the sum of IPFs for the individual terms
  - Practical tests show an average overlap of about 70% between result sets retrieved with IDFs and those retrieved with IPFs
  - Using IPFs the scalability is however still limited
3.1 Providing Collection-Wide Information

- Tests in Web information retrieval, e.g. [Viles & French, ‘95], show that CWI stays relatively stable over the whole collection of Web Sites even with churn
  - Only joining/leaving corpora on completely new topics result in significant change

- Indexing CWI in a similar way as the routing information for queries is possible [Balke et al., ‘05]
  - In structured networks CWI can be aggregated along the backbone and indexed CWI can be distributed together with the query
  - New queries have to be flooded/routed twice
    - The first flooding collects and aggregates CWI
    - The second one provides the correct CWI for local scorings
  - Non-expired indexed CWI can always be used when available
3.2 Estimating the Document Overlap

- Assessing the novelty of collections also supports retrieval quality
  - Pre-computed statistics about expected result quality in each collection is often used to minimize the number of queried collections
  - Choosing collection with high overlap for querying will usually not improve result sets sufficiently to justify the access costs
  - Especially progressive searches, like top-k searches, profit from focusing on collections with small overlaps, since result merging procedures will ignore identical/similar results

- The novelty of a collection can only be calculated with respect to some reference collection(s)
  - e.g. those collection(s) already in a peers local routing index
3.2 Estimating the Document Overlap

- A definition of a peer \( p \)'s collection \( C_p \) with respect to a reference collection \( C_{ref} \) [Bender et al., '05]

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

- Since the information what exact documents a peer offers is usually not disseminated, the values have to be approximated from statistics.
  
  - E.g. if abstracts in the form of Bloom filters are given, a combined Bloom filter \( b_p \) can be calculated by bitwise logical AND between \( p \)'s Bloom filters for all keywords in a query.
  
  - Novelty then can be estimated by comparing it to as the union of those Bloom filters \( b_i \) of the set of \( s \) have already been retrieved.
  
  - The *degree of novelty* is given by counting locations where \( p \)'s Bloom filter has differing set bits
    \[
    | \{ k | b_p[k] = 1 \land b_{prev}[k] = 0 \} |
    \]
3.3 Prestructuring Collections with Taxonomies

- Retrieval in P2P systems generally considers two basic paradigms
  - Fulltext-based queries
  - Metadata-based queries

- Integrating these paradigms can support retrieval effectiveness
  - Structuring document collections
  - Disambiguation of query terms

- Peers often host collections of similar documents, e.g. similar kind of information (newspaper articles, etc.) on similar topics, etc.
  - Scalability and successful use of statistics are strongly improved, if a common system of categories to classify the documents can be used
  - Since categories are more or less similar to each other a taxonomy on categories allows for easily finding semantically similar documents
3.3 Prestructuring Collections with Taxonomies

- Topical similarity within the taxonomy is defined by
  \[ \text{sim}(c_1, c_2) = e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \]

  - \( l \): shortest path between categories \( c_1 \) and \( c_2 \)
  - \( h \): level of common subsumer
  - Common values \( \alpha = 0.2, \beta = 0.6 \) (experimentally determined)
  - E.g. newspaper articles:
    - \( \text{sim}(\text{Politics, Sports}) \):
      - \( l = 2 \)
      - \( h = 1 \)
      - \( \text{sim} = 0.38 \)
3.3 Combination of Topics and Keywords

- Topics dominate keywords
- Cooperative Filter: Relax on topics until k results have been found

Example: [<Politics>, “London Olympics”]
3.3 Combination of Topics and Keywords

RequestResults

TopRes

Delivered

SP1

<table>
<thead>
<tr>
<th>RequestResults</th>
<th>TopRes</th>
<th>Delivered</th>
</tr>
</thead>
<tbody>
<tr>
<td>{P1}</td>
<td>{(P1, d11, [P, 0.8]), (SP2, d21, [P, 0.7]), (P1, d12, [P, 0.3])}</td>
<td>{(P1, d11, [P, 0.8]), (SP2, d21, [P, 0.7])}</td>
</tr>
</tbody>
</table>
Overview

1. Content Searching in Peer-to-Peer Applications
   1. Problems in Peer-to-Peer Information Retrieval
   2. Related Work in Distributed Information Retrieval

2. Index structures for Query Routing
   1. Distributed Hash Tables for Information Retrieval
   2. Routing Indexes for Information Retrieval
   3. Locality-Based Routing Indexes

3. Supporting Effective Information Retrieval
   1. Providing Collection-Wide Information
   2. Estimating the Document Overlap
   3. Prestructuring Collections with Taxonomies

4. Summary and Conclusion
4 Summary and Conclusion

- Evaluation of P2P top-k retrieval Scenario taken from [Balke et al., ’04]
  - Simulation of retrieval in a large P2P infrastructure
  - Structured P2P network
    - 100-2000 peers, 2-16 super-peers
    - ~100,000 documents (TREC collection of LA Times articles)
    - During each simulation run, 20% network churn
  - Queries
    - Top-1 and Top-10 queries
    - 2 terms per query on average, standard deviation 1.0
    - Terms randomly selected from document terms
    - Zipfian distribution, skew 1.0
4 Summary and Conclusion

- Frequent queries are indexed quickly, i.e. good index coverage
4 Summary and Conclusion

- Number of contacted peers decreases significantly (e.g. 50 of 2000 peers on average for the top-10 case).
4 Summary and Conclusion

• Experimental result accuracy
  ▶ Measured against globally indexed document collection
  ▶ Any change in ranking is considered incorrect (though typically the result are still useful)
  ▶ For a network with 20% network churn, on average 2-4% of results are incorrect.
4 Summary and Conclusion

- In today’s P2P systems only exact match keyword retrieval is prevalent (usually on meta-data)

- Information retrieval in P2P scenarios is needed
  - Individual, loosely coupled document collections need fulltext retrieval and ranking techniques
  - Applications range from shared working environments e.g. in project groups, to distributed digital libraries

- Almost all IR systems use at least some global statistics, in P2P infrastructures the dissemination of necessary statistics becomes a performance bottleneck
  - Trade-off between cached, but sometimes stale statistics and new, but expensively updated statistics needs to be managed
  - How much staleness does a ‘sufficient’ retrieval effectiveness allow?
4 Summary and Conclusion

- Choosing the ‘right’ collections for querying improves retrieval efficiency
  - Containing most promising documents with possibly little overlap
  - Small worlds offer quick connections to semantically close collections

- Query routing indexes can handle some network churn while providing results of sufficient quality
  - Local indexes can be efficiently maintained
  - Can exploit advantages by Zipf-distributed content allocations and querying behavior
  - Need to contact only small numbers of peers

- Supporting techniques like efficient CWI estimation/dissemination or taxonomies of document categories further improves retrieval