Evaluating Profiling and Query Expansion Methods for P2P Information Retrieval

Hans Friedrich Witschel, Thomas Böhme*
witschel@informatik.uni-leipzig.de, thomas.boehme@theoinf.tu-ilmenau.de

ABSTRACT

This paper addresses the issue of peer profiles, i.e. compact representations of a peer's locally offered content, and their use in P2P information retrieval and routing. Experiments with different profile building and query expansion methods show that compression of profiles is possible without losing too much retrieval performance and that query expansion using global co-occurrence data can improve results by approx. 10%.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]: Selection Process, C.2.4 [Distributed Systems]: Distributed Applications.

General Terms: Algorithms, Experimentation.

Keywords: Peer-to-peer information retrieval, profiles, query expansion, intelligent query routing, semantic topologies.

1. INTRODUCTION

The emergence and growing popularity of peer-to-peer filesharing systems was chiefly motivated by the users' interest to share copies of music and other multimedia files. However, recent considerations have been concerned with the question whether the P2P paradigm can and should also be adopted for ordinary information retrieval, i.e. for sharing and searching text documents.

A number of strong arguments like scalability, availability and easy and uncensored publishing were found in favour of P2P information retrieval and led to the development of technologies like those described in [7, 24, 16] or [19].

In this paper, we address the issue of peer *profiles*, i.e. compact representations of a peer's locally offered content. Profiles have been used in a variety of settings in P2P information retrieval (cf. for instance [9, 2, 7] or [17]), primarily for facilitating the search process: using peer descriptions, blind search methods like those applied in Gnutella [12] can be avoided by selecting only those peers that seem to offer relevant content, i.e. whose profiles are similar to the query under inspection.

Some P2P information retrieval approaches like [8], [19] or [15] do not use profiles: instead of storing representations of a peer's content, they associate queries (search keys) with peers that have answered these queries in the past. This allows efficient retrieval of popular items. We argue, however, that an information retrieval system should also be able to satisfy requests for unpopular data, i.e. data that has rarely or never been requested before.

1.1 P2P infrastructure within our project

Before going into details about how profiles can be built, we would like to introduce an algorithm for restructuring and search in P2P networks that completely avoids flooding and thus ensures scalability. It has been developed as part of a project supported by DFG[1] and was described and evaluated in detail in [27].

The algorithm is based on the concept of *small worlds*: by searching for its own profile – i.e. sending out queries that consist of the profile – and receiving answers from other peers, each participant in the network fills up a part of its routing table with addresses and profiles of neighbours that offer content similar to its own. Peers thus organise into clusters of semantic similarity.

Another part of the routing table is reserved for some arbitrarily chosen neighbours (random shortcuts) that provide links between different clusters of peers. The combination of these two neighbour selection strategies is intended to result in a small world network structure (see [18] for a theoretical model) that can be exploited for an efficient search algorithm.

Each peer that receives a query, first scans its local index for matching documents and then forwards the message to *just one* of its neighbours: the one whose profile best matches the query, i.e. the one deemed most likely to have an answer to it. This continues until the time-to-live (TTL) of the query expires.

Because peers are organised into clusters of semantic similarity, queries will find many relevant results once they have reached the right cluster because peers that can contribute answers to a given query are likely to know others that also can.

Addresses of peers that have been visited are inserted into the query's *Log* so that circles can be avoided. However, backtracking is not performed: if there are no more valid choices for forwarding the query (e.g. because all of

*supported by Deutsche Forschungsgemeinschaft (DFG)

P2PIR'05, November 4, 2005, Bremen, Germany.

[1]DFG project grant nr. 255712

a peer's neighbours have already been visited) the query is immediately returned to the requesting node. To avoid dead ends, special attention has to be devoted to providing a well-connected network.

1.2 Goals and Requirements

In this approach, profiles are used both to build semantic clusters of peers (*structure building*) and to forward queries (*search*). From this scenario, we can deduce some requirements that peer profiles have to meet. These characteristics are general in that they are also useful in other approaches that use compact peer representations, be it for searching or for structure building.

- Completeness: when searching for an arbitrary item, there should be a match between query and profile whenever the corresponding peer has data that contains or matches the item. This requires a profile to contain complete knowledge of the contents offered by that peer.
- Compactness: In order to be attached to messages and stored in other peers' routing tables, profiles should be very compact.
- Generality: for structure building, profiles have to be general: when applying a similarity measure to a pair of profiles, this should yield non-zero values for two peers that offer similar content. This normally requires some sort of overlap between the two profiles (no matter what they consist of). Overlap will only occur frequently enough if the elements contained in profiles are somewhat general.

These requirements seem contradictory at first glance: when compressing the contents of a peer, we will obviously loose information, i.e. completeness. Completeness also calls for very specific data to be included into the profile which contradicts generality.

On the other hand, we argue that query expansion can offer solutions to this dilemma: by finding the right terms by which to expand a given query, we can compensate for the missing completeness of profiles.

In the rest of this paper, we will first examine some related work and then describe and evaluate different profiling and query expansion techniques.

2. RELATED WORK

Selecting information resources on the basis of so-called *resource descriptions* (i.e. profiles) has been studied extensively as a part of *distributed information retrieval* (cf. e.g. [1, 13]). In distributed IR the need for compactness is rarely a problem because descriptions are exchanged between servers with high bandwith and storage resources. However, work like [20] shows that pruning resource descriptions can also be of interest in distributed IR and that it does not greatly degrade precision.

Profiles in distributed IR often consist of a so-called *unigram language model*, i.e. a complete list of words contained in a resource's document collection, together with their document frequency. It has been shown in [28] that query expansion can greatly improve distributed IR.

There are also efforts in distributed IR that work with more compact resource descriptions: The system Q-Pilot presented in [23] offers access to specialised search engines. As many of these do not allow complete unigram language models to be obtained, Q-Pilot works with lists of terms acquired from backlink pages, i.e. web pages that contain links to the respective search engine. In this setting, the authors find that query expansion can improve results by up to 40%. These results indicate that query expansion can be particularly useful when knowledge about information resources is sparse. This is why we chose to use it in our experiments.

In P2P information retrieval, on the other hand, much more attention has to be devoted to compressing peer descriptions. In PlanetP [7], for example, these descriptions consist of a complete unigram language model of a peer, i.e. all the terms from its documents. There is no frequency information, however, and profiles are compressed using Bloom filters. Peers in PlanetP have complete knowledge about all other participants in the network and their term lists. This allows for a globally correct ranking algorithm but is not designed for scaling to a large number of participants.

Increasing the compactness of profiles often involves the use of ontologies or keyword categorisations that subsume multiple semantically related terms under one common category or concept. In [2], for instance, Crespo and Garcia-Molina use categories like "databases" for building peer summaries that are exchanged between peers. The authors do not elaborate on where categories come from and how documents are classified.

Bibster, a P2P system for exchanging bibliographic data, is described in [9]: Bibster uses ontologies built from bibliographic metadata (e.g. obtained through local BibTeX entries) and the ACM topic hierarchy to generate a so-called "expertise model" – i.e. a very compact profile – for each peer. Similar ways to represent peer contents can be found in Edutella [11].

Ontologies derived from manually crafted topic hierarchies often suffer from lack of flexibility and poor coverage (we will elaborate on this later). Advanced indexing techniques like Latent Semantic Indexing (LSI, [10]) allow for the automatic extraction of semantic subspaces (or concepts) from document collections. LSI was originally developed to solve the synonymy problem. It does so by mapping synonymous terms to the same dimension, i.e. it reduces the dimensionality of the original vector space. In [24], the use of pLSI is proposed: peers are responsible for covering certain dimensions of the reduced term space and query terms can also be projected into this space which allows for extremely compact peer descriptions but also poses other problems that we will discuss later.

In the next section, we will weigh some of these profiling methods against each other and later examine whether and how they can be combined with effective query expansion.

3. PROFILE BUILDING METHODS

As we have seen in the last section, there seem to be two major possibilities for building compact peer profiles:

1. Use terms that are actually contained in the peer's document collection. We may decide individually which proportion of them to use and how to select the most significant ones (see 3.1).
2. Classify documents according to some shared ontol-

ogy or (keyword) categorisation and represent peers by the most prominent categories of their documents (see 3.2).

3.1 Vector space models

The first of these possibilities is normally based on the *vector space model* (VSM) for information retrieval in which the terms that occur in the documents of a given collection (or a subset of them, or even a controlled indexing vocabulary) form the basis of a vector space. This vector space is used to represent documents and queries as a vector of weights. Similarity between vectors is often measured as the cosine of the angle between them.

As this vector space has a very high dimensionality, only the non-zero entries of document and query vectors are normally stored. The famous TF/IDF measure [22] is often used to compute weights which can be used to further reduce the size of vectors by cutting off entries with low weights.

In a distributed environment, however, the calculation of global IDF values is impossible since it involves counting the document frequency of terms, i.e. the number of documents in the whole collection that contain a given term. This is not feasible because peers do not have a global view on the entire collection.

Tang et al. [24] or Kronfol [19] have proposed to use representative samples of collections to estimate IDF: Tang suggests to periodically retrieve a sample of documents from a randomly chosen set of peers in the system and then use them to estimate IDF values. Kronfol calculates IDF values in advance from a large and representative document collection and then equips each peer with a lexicon that associates terms with their IDF.

In our experiments, we have opted for the latter of these possibilities: by analysing a very large and well-balanced corpus (which we will call *reference corpus* from now on), it is possible to estimate the frequency of words in everyday-language. Since language changes only very slowly, the statistical model we obtain can also be used in dynamic environments.

Instead of calculating TF/IDF, however, we used a likelihood ratio significance measure because of its proper mathematical foundation (cf. [5]). It compares the relative frequency of a term in a given document to its relative frequency in the reference corpus: terms whose relative frequency p_1 in the given document significantly surpasses the one in the reference corpus p_2, will be ranked highly:

$$sig(term) = 2(log_L(p_1, k_1, n_1) + log_L(p_2, k_2, n_2) - log_L(p, k_1, n_1) - log_L(p, k_2, n_2))$$

where k_i is the raw frequency of the term in the document and reference corpus, respectively, n_i is the size of these corpora and $log_L(p, k, n) = klog(p) + (n-k)log(1-p)$. Finally, we have $p = (k_1 + k_2)/(n_1 + n_2)$.

When weighted document descriptions are available, profiles can be built by adding up the respective vectors and then cutting off the least significant terms. The threshold for cutoff, i.e. the length of the resulting profiles can be chosen freely so that we can trade retrieval accuracy for storage costs.

It should be mentioned that other methods for building profiles with a VSM are thinkable: one could, for example treat the whole set of text files shared by a peer (or clusters of them) as one single document and then apply term extraction methods (i.e. likelihood ratio statistics) to this document. However, these procedures are computationally much more expensive, which is why we chose to use the simpler method of adding document vectors in our experiments.

3.2 Document classification

In many settings, documents can be classified according to some shared ontology or topic hierarchy. Peers will be equipped with an instance of this knowledge base which they use to classify their documents. Hence, profiles consist of the most prominent categories of a peer's documents.

LSI can also be used in a similar fashion: LSI representations of documents can be interpreted as *concept vectors* that indicate the importance of concepts for a given document. Adding up concept vectors of documents (as used by [4] in the TREC routing task) can be introduced as a possibility to build profiles for peers.

The calculation of LSI representations, however, also requires the knowledge of the complete document collection. As this is never feasible, we might try to use the same strategy as above: obtain an LSI basis (i.e. concept vectors for each term in a large and representative collection) and use it to fold in new documents by simply adding up the vectors of the terms they contain.

It would be more correct to use *SVD updating* (which is proposed by [24]) but as this requires to redistribute the complete basis of the reduced vector space after the insertion of new documents, it did not seem an interesting option to us.

4. QUERY EXPANSION

The original idea behind query expansion was to solve the so-called *vocabulary mismatch* problem: when searching for a certain keyword (e.g. "elevator"), one is normally also interested in all its synonyms (like "lift") or maybe other very closely related terms. By expanding the query, more relevant documents can be found.

When working with radically pruned profiles, however, the situation is slightly different: it is often desirable to expand queries not only by synonyms, but also by other related words – *topic words* as they are called in [28] – in order to increase the probability of matching the query with *any* profile that is similar enough to the query to start a gradient ascent search.

Depending on the kind of profile that is used, different query expansion methods are thinkable. We will now introduce various of them and discuss their advantages and drawbacks.

4.1 Categories

When working with ontologies or keyword classifications, queries should be expanded with their corresponding categories.

Using categories yields extremely compact peer profiles that are well human-readable. On the other hand, thesauri are difficult to obtain and often suffer from poor coverage (i.e. many queries cannot be expanded at all because keywords are not contained). They have to be designed manually and often fail to adapt to varying contexts: a general-purpose ontology will have very low coverage in an environment where specialists work on topics like e.g. "quantum mechanics". When designing ontologies, we are actually con-

fronted with the problem of deciding which categories the area of discourse consists of. This is often subjective and therefore difficult to realize.

4.2 LSI

When peer profiles consist of LSI concepts, query terms just have to be projected into the LSI semantic space. This representation can then directly be compared to profiles by applying the cosine similarity measure.

This too is very compact (since no strings are used) and no manual effort is needed. However, one still has to specify the dimensionality of the LSI space. As we have seen above, keeping a distributed LSI basis up to date is prohibitively expensive and approximations might not work very well. Additionally, LSI representations are not human-readable.

4.3 VSM

Profiles consisting of the statistically most significant keywords from a peer's document base are inherently incomplete. Multiple methods of query expansion have been proposed in the literature: they all aim at enriching the original query with words that are semantically closely related to the query words and thus increase the probability of obtaining a match with the (incomplete) profiles.

The use of manually crafted thesauri in this domain (as proposed by [25]) suffers from the same problems as mentioned above. Using co-occurrence data to automatically construct thesauri as suggested by [14, 21], however, seemed more appealing to us: using the same sampling method as for estimating word frequencies, one can analyse co-occurrences of words in a large corpus that is representative of the given domain [2] and store it in a co-occurrence matrix. This matrix can be interpreted as an "association thesaurus" (cf. [14]).

The advantages of global co-occurrence data derive from the fact that no manual work is involved (i.e. coverage is expected to be high) and that it is not even necessary to specify the *number* of concepts that should be used for "categorising the world". It is also expected that this sort of statistical analysis generalises better than LSI, i.e. it should be possible to calculate associations based on co-occurrences in one document collection and use the data on a different one as long as the two collections are sufficiently similar. However, there is still a piece of global knowledge in the P2P network: the co-occurrence matrix has to be replicated on each peer. This infers high storage costs because the matrix will normally be very large. Dynamic environments where data includes items from very different and special domains also pose a problem because the collection from which co-occurrences are computed is static and only representative for a certain domain.

4.4 Local feedback

The only solution to query expansion that completely avoids the maintenance of a shared knowledge base on each peer is based on local feedback. The basic idea behind local feedback originates from *relevance feedback*, which adds terms to a query that appear in relevant documents. Instead of "waiting" for the actual relevance judgement by the user, Buckley et al. [6], for instance, propose to expand queries with concepts found in the top-ranked documents in the initial result set of that query (which are assumed to be relevant). This is sometimes also called *pseudo relevance feedback*.

In a P2P scenario, expansion can be done on the fly: at each peer, documents that are added to the result set can be scanned and terms that appear in many of them can be added to the query. Alternatively, peers that have contributed many documents could add terms from their profile to the query.

This approach is obviously very useful because no global knowledge has to be computed and maintained. However, it will only start to work when something has been found: due to the incompleteness of profiles, the query might initially be redirected to the wrong peers. Since these cannot contribute any relevant documents, no expansion will take place, the right peers can still not be found and so on. Another drawback results from the lack of *real* relevance judgements which might lead to expansion with terms from totally irrelevant documents.

As there are advantages and drawbacks to all methods described so far, we decided to perform some experiments and simulations in order to compare and evaluate them. These will be described in the next sections.

5. EXPERIMENTAL SETUP

5.1 Two scenarios

Our experiments cover two scenarios:

- The first one is the case of classical distributed IR where there is one central instance (often called *broker* or *librarian*) that has a global view on all peers' profiles and redirects incoming queries with their help: peers will be ranked according to their profile's similarity to the query and then visited in that order. This scenario was examined in order to uncouple structure building from search before analysing them jointly.

- The second scenario consists of real P2P information retrieval. Using a simulation tool that implements the algorithm sketched in section 1.1[3], profiles are first used for structure building. In a second step, each peer uses its acquaintances for routing: incoming queries are treated just like in the first scenario, with the difference that only local knowledge (i.e. the peer's direct neighbours) is available.

In both cases, we measured (cumulative) recall as a function of the number of peers visited (number of hops). For the second scenario, we were also interested in the characteristics of the network graph that evolved.

5.2 Initialisation

As the basis of our experiments, we used a German newspaper corpus consisting of 3429 texts, each of which was labelled with exactly one of 10 semantic categories: {*cars, finance, jobs, culture, politics, travel, sports, university, technology, science*}. The corpus was split into a training set (consisting of 2429 texts) and a test set (1000 texts). The former served as a basis for calculating "language models",

[2]here, we assume document topics to be restricted to a certain domain

[3]The tool was implemented using the network simulator OMNeT++ (http://www.omnetpp.org)

i.e. the LSI basis and co-occurrence data. The latter was distributed among the peers. The distribution of categories was balanced across the two sets. In the process of peer initialisation, we used the categories to simulate interest-based peer collections: each of the 1000 peers in our simulation first selected one or two of the categories as its "interests". Now each peer P was assigned 10 documents, each of which was chosen at random from all the documents that were labelled with one of P's interests.

For the second scenario peers were additionally equipped with three initial contacts, i.e. addresses of three other peers, picked at random. Each participant was allowed to have a maximum of 10 neighbours (three of which must be random shortcuts). For the *Baseline* strategy (see below), each peer was assigned 10 random neighbours.

Finally, we created three sorts of inverted lists for the documents in the test collection by applying an indexing technique based on likelihood ratio significances (see above). The best 16, 32 or 48 terms were chosen for each document and inserted into the inverted list. The inverted list consists of pairs (t_i, d_j) each of which indicates that term t_i occurs in document d_j.

In the following, the inverted list will be used as a substitute for a user's relevance judgements (which were not available): each document a query term occurs in is assumed to be relevant to that query. Furthermore, the inverted list was used to create a set of 200 test queries: as a very pessimistic assumption, we chose queries to consist of only one word [4]. These words were picked at random from the left column of the inverted list, which means that terms with high document frequency (i.e. t_i that occur in many pairs (t_i, d_j)) are more likely to be selected as query terms.

In the following, we assumed all peers to have a local full-text index, i.e. for a given query, each peer will correctly return all relevant documents that it possesses.

5.3 Strategies

We evaluated five combined profiling/query expansion strategies. For each strategy, the length of profiles (and hence the compression rate) was varied to cover the values 16, 32 and 48. These lengths were chosen because – for our practical work – we considered 50 items per profile an upper bound and were interested in knowing whether this size could be further reduced.

- *Standard*: in that case, the vector of document d_j consisted of all the terms t_i for which an entry (t_i, d_j) existed in the inverted list(s). Profiles were calculated by adding up the vectors of a peer's documents and then cutting off terms with low significance values in order to arrive at the respective profile length. Similarities between vectors were computed by simply counting the number of common terms. Queries were not expanded.

- *Dornseiff*: As a second strategy, we combined these term vectors with categories from a keyword classification: we used a German thesaurus called "Dornseiff" [3], which consists of words grouped together in subject areas non-hierarchically. We reserved 1/8 of each document and profile vector for categories (or subject areas) that were derived by simply counting the categories of terms that occurred in a document. These vectors were then added and pruned as indicated above, but separately for the keyword and subject area part. Similarity was calculated according to:

$$sim(q,p) = \sum_{t \in q \cap p} 1 + \sum_{c \in q \cap p} \frac{1}{4} \quad (1)$$

  where t stands for a term and c for a category or subject area. Our intention behind this was to weigh direct matches between keywords (4 times) higher than matches between categories, i.e. to prefer specific matches to general ones. Query terms were expanded with their subject area(s) from "Dornseiff".

- *Cooccs*: This strategy works very similarly to the *Standard* strategy, with the only difference that queries will be expanded with words that often co-occur with query words in the training corpus. Again, a likelihood ratio measure was used for computing the significance of co-occurrences. Similarities were computed using the same formula as given in eq. 1, with c now representing a word that co-occurred frequently with one of the query words (i.e. direct keyword matches are again considered four times as valuable as matches arrived at by query expansion).

- *LSI*: In this case, the first thing that we did was calculate an LSI basis from the training set, i.e. obtain concept vectors for each term in the training collection. Documents from the test set were represented by projecting them onto that basis, i.e. by adding up the vectors of the terms they contained. They were then added to arrive at peer profiles and queries were represented by the corresponding term vector from the LSI basis. The cosine measure was used for calculating similarities. As LSI provides very compact representations and as it has been reported to work best with several hundred dimensions, we allowed vector lengths of 100 and 200 in addition to those specified before (16, 32 and 48).

- *Local Feedback*: As a last strategy, we used the *Standard* strategy, this time expanding queries "on the fly" by local feedback: terms occurring in relevant documents were added to the query as these documents were found on the peers. Those terms t, for which the number n_t of relevant documents they were contained in surpassed a threshold of 2, were added to the query, together with n_t as a weight. Similarity between queries and profiles was now obtained by calculating

$$sim(q,p) = \sum_{t \in q \cap p} n_t \quad (2)$$

  Note that again the original query word – being contained in all relevant documents – receives the highest importance when calculating similarity.

In addition to these strategies, we implemented two "boundaries": a *Baseline* where the next peer the query will be forwarded to is picked at random and a fully *Informed* search where profiles are not pruned and contain full frequency information. This means that for a given query word, a peer's profile indicates *exactly* how many relevant documents are

[4] This is pessimistic because coverage of thesauri will be higher and disambiguation easier for longer queries

available on that peer. Profiles in the *Informed* strategy had an average length of 200 entries: this tells us how much information is lost when cutting to our predefined lengths of 16, 32 and 48.

6. RESULTS

6.1 First scenario

For the first scenario, we forwarded all 200 queries to the central broker. Peers were then visited one after the other in the order induced by the ranking that was returned by the different algorithms, and the cumulative recall was recorded for each step and averaged over all test queries. Note that whenever two peers receive the same rank (i.e. the same similarity value), they will be visited in a random order.

Figure 1 shows the recall curves for the two baselines and the strategies *Standard, Dornseiff, Cooccs* and *LSI* for profile lengths 16 and 48. We can see that using co-occurrence data for query expansion outperforms all other strategies whereas the use of keyword categories from Dornseiff is quite useless: there is no visible difference between *Standard* and *Dornseiff* strategies. This can be explained when looking at Dornseiff coverage: only 45% of the query words were contained in the thesaurus. For the rest of the queries, expansion with Dornseiff categories does not seem to increase recall: many were assigned multiple (i.e. ambiguous) categories and there was no way to disambiguate.

LSI performs significantly worse, even when using 100 or 200 dimensions for the reduced space. This is probably due to the suspicion we had before: LSI cannot be used for sampling, i.e. for calculating a basis on a collection completely different from the one that will be used for retrieval.

Another observation that can be made is the fact that differences between strategies become less visible as the length of profiles increases, which is obvious because longer profiles are closer to completeness, i.e. data sparseness is not such a problem here.

6.2 Second scenario

6.2.1 Graph analysis

The first thing we analysed for the second scenario were the effects of structure building. As mentioned in section 1, peers build the network structure by actively searching for other peers that have profiles similar to their own. Profiles are treated as queries in this process, which means that the same strategies for query expansion could be applied.

Table 1 shows some figures that describe the different network graphs obtained by structure building. *Dornseiff* was omitted from this table for space reasons and because figures were very similar to the *Standard* strategy.

The *clustering coefficient* of a graph $G = (V, E)$ is defined according to [26]: Given the neighbourhood N_v of a node v, $N_v = \{u \in V | (v, u) \in E\}$ we get the local clustering coefficient C_v for node v:

$$C_v = \frac{\{(a, b) \in E | a, b \in N_v\}}{|N_v|(|N_v| - 1)} \quad (3)$$

The global clustering coefficient is the mean of C_v-values, averaged over all nodes $v \in V$.

Neighbour precision is a new measure which we introduce on the basis of peer interests: for a given peer P, it denotes the percentage of P's neighbours (except random shortcuts) that have at least one interest in common with P. Remember that interests were chosen in advance for each peer using the 10 semantic categories of our newspaper corpus. Thus, neighbour precision measures the purity of semantic clusters formed by the structure building technique. Finally, the number of strongly connected components and average distances between nodes were calculated.

Pr. length		Baseline	Standard	Cooccs	LSI
16	Cluster Coeff.	0.015	0.15	0.2	0.46
	# Components	1	51	108	896
	Neighbour Prec.	0.21	0.997	0.978	0.29
	Avg. distance	3.2	3.6	3.7	3.5
32	Cluster Coeff.	0.015	0.15	0.21	0.34
	# Components	1	63	219	842
	Neighbour Prec.	0.21	1	0.99	0.47
	Avg. distance	3.2	3.6	3.8	3.4
48	Cluster Coeff.	0.015	0.16	0.21	0.34
	# Components	1	88	230	697
	Neighbour Prec.	0.21	1	0.994	0.5
	Avg. distance	3.2	3.5	3.8	6.2

Table 1: Figures describing network structure

There are a number of interesting observations to be made:

- By looking at the number of components, we can see that the tendency of the graph to break up increases as more information is used, be it longer profiles or more elaborate query expansion. We should mention that in all cases there was one big strongly connected component and a number of isolated nodes: for instance, a number of 57 components indicates that there were 56 unreachable nodes and a big strongly connected component consisting of 944 nodes. Distances are (reasonably) short in all graphs but note that they were only computed using node pairs (A,B) where there is a path connecting A and B.
- The "semantic clustering" (neighbour precision) induced by the structure building works well in all cases except for *LSI*. That means that peers with common interests are really grouped together.
- *LSI* performs significantly worse than all other strategies as far as connectivity and neighbour precision are concerned. For LSI, however, connectivity improves as the number of dimensions grows. This continues for 100 and 200 dimensions (which is not shown here) and indicates that LSI should really be used with at least 100 dimensions. Distances between nodes, however, increase in that case, the average being 4.8 in the case of 100 and 5.0 in the case of 200 dimensions.

The conclusion we can draw from these figures is quite plain: for structure building, very simple methods should be used. Profiles can be short and direct matches between profiles are sufficient for building good semantic clusters while preserving a good connectivity and short paths between nodes.

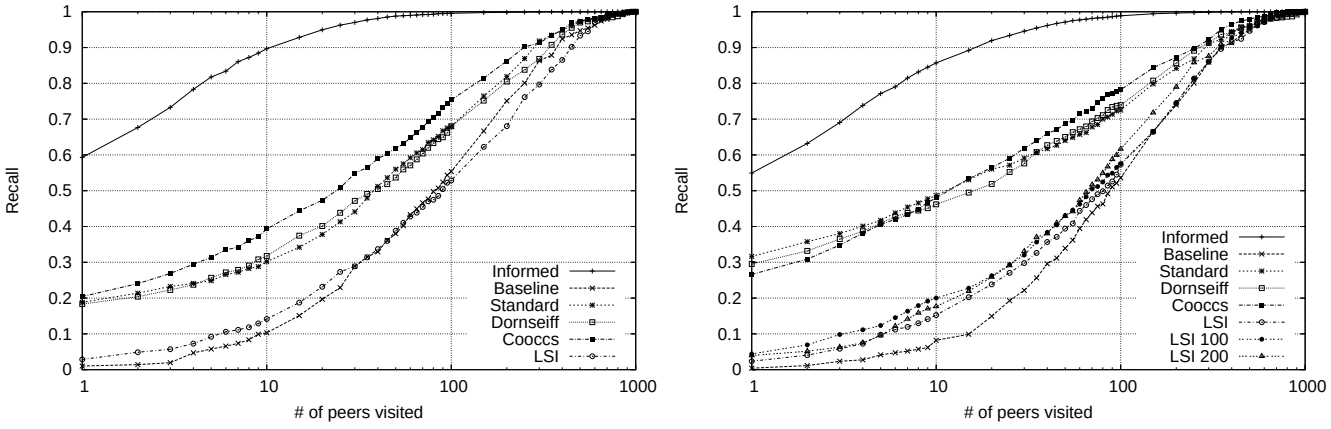


Figure 1: Recall as a function of the number of peers visited, averaged over all queries for profile length of (a) 16 and (b) 48

6.2.2 Recall

Finally, we measured recall as a function of the number of hops a query made in the network graph that was previously formed by our structure building algorithm. Each strategy was applied in the graph that was built by using that same strategy (we did not try any other combinations).

We selected 20 peers at random and then let each of them search for all of our 200 test queries. Cumulative recall was recorded for each hop the query made. Macro averaging was then used, i.e. recall values were first computed for each query instance separately and then averaged over all instances.

Figure 2 shows the same curves as figure 1, this time for the second scenario. It now also includes the fifth strategy (*Local Feedback*).

We can see that the tendencies are very similar to the ones observed in the first scenario.

Local feedback performs no better than the *Standard* or *Dornseiff* strategies which can be explained as follows: approx. 50% of the queries have two or less matching documents, even though terms with high document frequency were more likely to be chosen as query terms. This means that in 50% of the cases, nothing happens: in order for *Local Feedback* to start working, at least two relevant documents have to be found. Table 2 shows some statistics on how many relevant documents exist for each query.

# relevant documents	# queries
1	73
2	25
3	16
4	12
>5	74

Table 2: Statistics on number of relevant documents per query

We can also see that the effect of weak connectivity – the graph breaking up into many components – affects the final retrieval performance: recall stays below 90% for the *Cooccs* strategy when using a profile length of 48 because some documents reside on unreachable peers. The effect is much stronger for LSI where the graph breaks up almost completely.

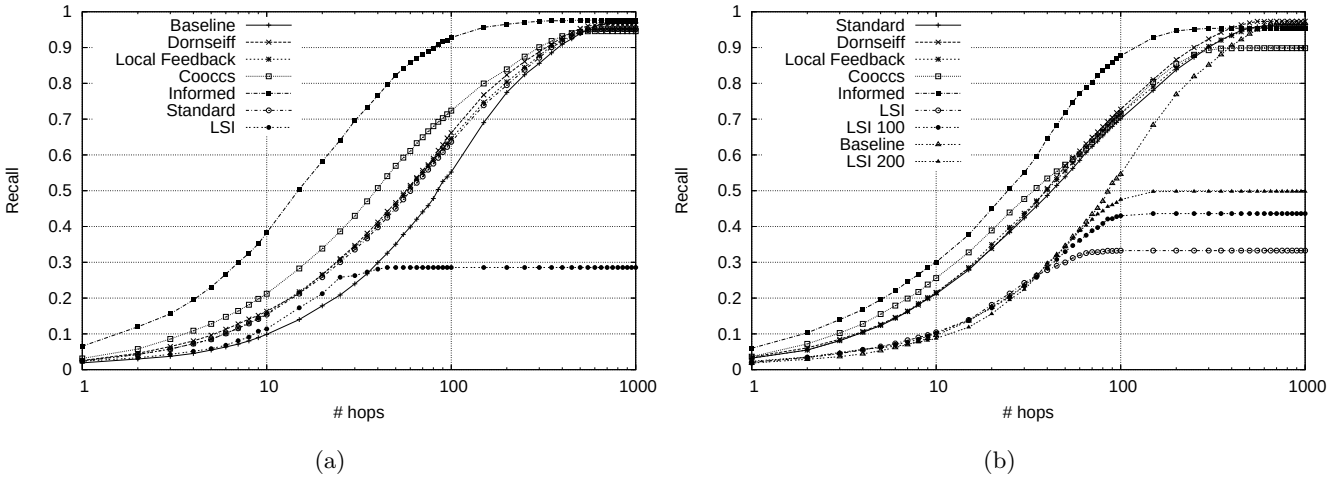
But note that a recall of 100% is also never reached because circles are avoided by the algorithm and backtracking is not performed when a query reaches a peer whose neighbours have all been visited.

Finally, we see that the *Informed* strategy performs worse in the second scenario, which indicates that this second task is more difficult. We introduced searching in a semantically clustered graph as a hill-climbing exercise, i.e. a task of first finding the right cluster of peers and then visiting them one after the other. For some queries, however, this seems to be difficult because they are not related to the topics that are used to form semantic clusters. Consider, for example, the query "Schlapphut" (floppy hat). Even for a human, it would be difficult to decide which of the categories introduced above it should be associated with (should it be *politics* or *travel*?) and it is not likely for a peer that has documents containing "Schlapphut" to know others that also have. This means that for a lot of queries, search will be quite blind, even when fully expanded peer profiles are available.

7. CONCLUSIONS

By analysing various techniques for profiling and query expansion in P2P information retrieval and applying them to structure building and searching, we found support for our claim that query expansion is likely to enhance recall in these distributed settings: when working with compact and incomplete peer descriptions, query expansion using co-occurrence data will improve recall by approx. 10%.

The use of thesauri did not prove fruitful in our setting because of poor coverage. Local feedback techniques failed because they only start to work when something has already been found (which is often too late). LSI, when applied to calculating a semantic basis and using it for calculating profiles, performed very poorly because LSI seems to be unable



(a) (b)

Figure 2: Recall as a function of the number of hops, averaged over all queries for profile lengths of (a) 16 and (b) 48

to generalise from one collection to the other. That means that the use of co-occurrence data for query expansion seems most promising to us.

As far as structure building is concerned, we found that comparing very compact profiles by very simple methods is sufficient for creating good semantic clusters of peers.

All in all, the task of routing queries using compact peer profiles remains a difficult one with many problems unsolved. However, the fact that a compression ratio of 1:4 (48 vs. 200) for profiles only led to a loss in recall of about 10% (using the best of our strategies), makes us assume that when building profiles, a large amount of terms can be safely ignored without losing too much information.

8. REFERENCES

[1] J. Callan. Distributed information retrieval. In W.B. Croft, editor, *Advances in Information Retrieval*, pages 127–150. Kluwer Academic Publishers, 2000.

[2] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *Proc. of the 28 th Conference on Distributed Computing Systems*, 2002.

[3] F. Dornseiff. *Dornseiff – Der deutsche Wortschatz nach Sachgruppen*. de Gruyter, Berlin/New York, 2004.

[4] S. Dumais. Latent semantic indexing (LSI): TREC-3 report. In *Proc. of the Third Text REtrieval Conference*, pages 219–230, 1994.

[5] T. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1994.

[6] C. Buckley et al. Automatic query expansion using smart: Trec 3. In *Proc. of TREC-3*, pages 69–80, 1994.

[7] F. M. Cuenca-Acuna et al. PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. In *12th International Symposium on High Performance Distributed Computing (HPDC)*, 2003.

[8] I. Clarke et al. Freenet: A Distributed Anonymous Information Storage and Retrieval System. *Lecture Notes in Computer Science*, 2009:46+, 2001.

[9] J. Broekstra et al. Bibster - A Semantics-Based Bibliographic Peer-to-Peer System. In *Proceedings of SemPGRID '04, 2nd Workshop on Semantics in Peer-to-Peer and Grid Computing*, pages 3–22, New York, USA, May 2004.

[10] S. C. Deerwester et al. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[11] W. Nejdl et al. Edutella: a p2p networking infrastructure based on rdf. In *WWW '02: Proceedings of the eleventh international conference on World Wide Web*, pages 604–615, 2002.

[12] Gnutella. www.gnutella.com.

[13] L. Gravano, H. García-Molina, and A. Tomasic. GlOSS: text-source discovery over the Internet. *ACM Transactions on Database Systems*, 24(2):229–264, 1999.

[14] Y. Jing and W.B. Croft. An association thesaurus for information retrieval. In *Proc. of RIAO*, pages 146–160, 1994.

[15] S. Joseph. NeuroGrid: Semantically Routing Queries in Peer-to-Peer Networks. In *Proceedings of the International Workshop on Peer-to-Peer Computing*, 2002.

[16] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti. A Local Search Mechanism for Peer-to-Peer Networks. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pages 300–307, 2002.

[17] I. King, C. H. Ng, and K. C. Sia. Distributed content-based visual information retrieval system on peer-to-peer networks. *ACM Transactions on Information Systems*, 22(3):477–501, 2004.

[18] J. Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000.

[19] A. Z. Kronfol. FASD: A Fault-tolerant, Adaptive, Scalable, Distributed Search Engine, 2002.

[20] J. Lu and J. Callan. Pruning long documents for distributed information retrieval. In *Proc. of CIKM'02*, pages 332–339, 2002.

[21] Y. Qiu and H.-P. Frei. Concept-based query expansion. In *Proc. of SIGIR-93*, pages 160–169, 1993.

[22] G. Salton, A. Wong, and C.S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[23] A. Sugiura and O. Etzioni. Query routing for web search engines: architectures and experiments. In *Proc. of the 9th international World Wide Web conference on Computer networks*, pages 417–429, 2000.

[24] C. Tang, Z. Xu, and M. Mahalingam. pSearch: Information retrieval in structured overlays. *ACM SIGCOMM Computer Communication Review*, pages 89–94, 2003.

[25] E. Vorhees. Query expansion using lexical-semantic relations. In *Proc. of ACM SIGIR*, pages 61–69, 1994.

[26] D. Watts and S. Strogatz. Collective Dynamics of 'Small-World' Networks. *Nature*, 393(6):440–442, 1998.

[27] H.F. Witschel. Content-oriented Topology Restructuring for Search in P2P Networks. Technical report, University of Leipzig, http://wortschatz.uni-leipzig.de/%7Efwitschel/papers/simulation.pdf, 2005.

[28] J. Xu and J. Callan. Effective retrieval with distributed collections. In *Proc. of SIGIR '98*, pages 112–120, 1998.