p2pDating: Real Life Inspired Semantic Overlay Networks for Web Search

Josiane Xavier Parreira
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
Saarbrücken, Germany
jparreir@mpi-inf.mpg.de

Sebastian Michel
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
Saarbrücken, Germany
smichel@mpi-inf.mpg.de

Gerhard Weikum
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
Saarbrücken, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

We consider a network of autonomous peers forming a logically global but physically distributed search engine, where every peer has its own local collection generated by independently crawling the web. A challenging task in such systems is to efficiently route user queries to peers that can deliver high quality results and be able to rank these returned results, thus satisfying the users' information need. However, the problem inherent with this scenario is selecting a few promising peers out of an a priori unlimited number of peers. In recent research a rather strict notion of semantic overlay networks has been established. In most approaches, peers are squeezed into a semantic profile by clustering them based on their contents. In the spirit of the natural notion of autonomous peers participating in a P2P system, our strategy creates semantic overlay networks based on the notion of "peer-to-peer dating": Peers are free to decide which connections they create and which they want to avoid based on various usefulness estimators. The proposed techniques can be easily integrated into existing systems as they require only small additional bandwidth consumption as most messages can be piggybacked onto established communication. We show how we can greatly benefit from these additional semantic relations during query routing in search engines, such as MINERVA, and in the JXP algorithm, which computes the PageRank authority measure in a completely decentralized manner.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: [Miscellaneous]; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Distributed systems*

General Terms

Algorithm, Design

Keywords

Semantic Overlay Networks, P2P Information Systems, Distributed PageRank Computation, Distributed Web Search

SIGIR Workshop on Heterogeneous and Distributed Information Retrieval 2005 Salvador, Bahia Brazil

1. INTRODUCTION

The peer-to-peer (P2P) approach, which has become popular in the context of file-sharing systems such as Gnutella or KaZaA, allows handling huge amounts of data in a distributed and self-organized way. In such a system, all peers are equal and all of the functionality is shared among all peers so that there is no single point of failure and the load is evenly balanced across a large number of peers. These characteristics offer enormous potential benefits for search capabilities powerful in terms of scalability, efficiency, and resilience to failures and dynamics. Additionally, such a search engine can potentially benefit from the intellectual input, e.g., bookmarks [6] and query logs [24], of a large user community. One of the key difficulties, however, is to efficiently select promising peers for a particular information need, given the total number of relevant peers in a network is not known a priori and peer relevance also varies from peer to peer.

Semantic Overlay Networks (SONs) [2, 4, 16, 37] appears as a network organization that improves query performance while maintaining a high degree of node autonomy. In a SON nodes with semantically similar content are "clustered" together. Queries are, then, routed only to the appropriated SONs, increasing the chances that matching information (e.g. files, documents) will be found quickly, and thus, the load on nodes having unrelated content is reduced. Such overlay networks are actually not only useful in search engines. It will be shown in this document that there are other applications that can also benefit from SONs.

We show how peers acting autonomously can form context-rich SONs, and how the proposed SONs can be utilized during query routing in P2P web search engines, and in distributed algorithm for PageRank authority computation.

There are many challenges when building SONs, regarding how nodes are assigned to SONs and to which SONs a query should be sent to. According to the initial idea proposed, peers should be evenly distributed among SONs, so we can answer queries fast, as we have to ask fewer peers, and each peer should belong to a small number of SONs, so that each peer has to handle only a few number of connections. However, in the real world, the distribution of peers over semantic classes is expected to be very skewed and dynamic as many peers will belong to some very popular topics that are constantly changing, whereas some uncommon classes will be less populated. Moreover, in most of the early approaches, nodes decide which SONs to join based on the classification of their documents. This leads to a fixed configuration of the Semantic Overlay Networks, so that the performance is highly dependable on a good choice of the classification algorithm, and requires a classifier that is the same for all peers. To overcome the restriction that the peers

are squeezed into these strict topic schemes, we propose a new method for building Semantic Overlay Networks that gives more autonomy for the peers, when deciding which SONs they should join. The method works by rearranging the connections between peers, according to the peers' criteria of a "good" neighbor (i.e. a "friend"), and using caching to remember the peers that were defined as friends. Possible measures for deciding if a peer should be considered a friend or not could be, for instance, level of overlap between documents from the peer and documents from the candidate for being a friend, the similarity between their documents, history of the peer, level of trust, etc. A peer has also the option to delete an already established link with a friend, if it has either changed its selection criteria or found more interesting peers.

The rest of the document is organized as follows: Section 2 briefly discusses related work. The p2pDating, our approach for creating dynamically evolving semantic overlay networks is then presented in more detail in Section 3, and a discussion about the different criteria for finding friends in the network is given in Section 4. Two applications that can directly benefit from p2pDating, namely the MINERVA system and the JXP algorithm, are presented in Sections 5 and 6, respectively. Moreover, we performed an experiment that shows how p2pDating can improve the performance of the JXP algorithm. The results are also presented in Section 5. Section 7 concludes this paper with a discussion about future work.

2. RELATED WORK

Recent research on P2P systems, such as Chord [33], CAN [29], Pastry [31], P2P-Net [13], or P-Grid [3] is based on various forms of distributed hash tables (DHTs) and supports mappings from keys, e.g., titles or authors, to locations in a decentralized manner such that routing scales well with the number of peers in the system. However, the approaches are limited to exact-match, single keyword queries on keys. This is insufficient when queries should return a ranked result list of the most relevant approximate matches [15].

In the following we briefly discuss some existing approaches towards P2P Web search. Galanx [38] is a peer-to-peer search engine implemented using the Apache HTTP server and BerkeleyDB. It directs user queries to relevant nodes by consulting local peer indexes.

PlanetP [17] is a publish-subscribe service for P2P communities and the first system supporting content ranking search. PlanetP distinguishes local indexes and a global index to describe all peers and their shared information. The global index is replicated using a gossiping algorithm.

Odissea [34] assumes a two-layered search engine architecture with a global index structure distributed over the nodes in the system. A single node holds the entire index for a particular text term (i.e., keyword or word stem). Query execution uses a distributed version of Fagin's threshold algorithm [18]. The system appears to cause high network traffic when posting document metadata into the network, and the query execution method presented currently seems limited to queries with one or two keywords only.

The system outlined in [30] uses a fully distributed inverted text index, in which every participant is responsible for a specific subset of terms and manages the respective index structures. Particular emphasis is put on three techniques to minimize the bandwidth used during multi-keyword searches.

[23] considers content-based retrieval in hybrid P2P networks where a peer can either be a simple node or a directory node. Directory nodes serve as super-peers, which may possibly limit the scalability and self-organization of the overall system. The peer selection for forwarding queries is based on the Kullback-Leibler divergence between peer-specific statistical models of term distributions.

Most relevant to our work is the idea proposed by Crespo and Garcia-Molina in [16]. However, our method goes beyond the fixed classification method for assigning peers into the SONs. Each peer defines its own criteria for joining a SON. We can think of the original idea as being a special case of our method, where all peers define their good neighbors as the ones whose documents belong to the same concept as their own documents.

Tang et al. [35] present another notion of semantic overlay, where documents are distributed among peers in a content-addressable network (CAN) according to their semantics, such that the distance (routing hops) between two documents in the network is proportional to their dissimilarity in semantics, and the document semantics is produced using latent semantic indexing(LSI). [22] chooses a similar approach but proposes a new dimension reduction technique that they incorporate directly in the construction of their overlay networks, so called Semantic Small Worlds. [4] proposes a technique that arranges peers into so called topic segments that consist of semantically related peers. Topic segments are constructed via clustering.

Triantafillou et al. [37] propose an architecture that is focused on harnessing all available resources in a P2P system where users contribute with their own content and own resources to the community. The presented architecture imposes a logical structure, based on document categories, node clustering, and their associations, and considers fair load distribution inside and across these node-clusters, so called *inter-cluster and intra-cluster load balancing*.

[36] considers query answering on RDF(S) data. It defines a method for query routing in which peers observe which queries are successfully answered by other peers and remember these peers in future query routing decisions. It is a lazy learning approach based on learning peers' query results.

[2] addresses the problem of building scalable semantic overlay networks and identifies strategies for their traversal using P-Grid [3].

In addition to this work on using semantics in distributed search, prior research on distributed IR and metasearch engines is relevant, too. [14] gives an overview of algorithms for distributed IR style result merging and database content discovery. [20] presents a formal decision model for database selection in networked IR. [27] investigates different quality measures for database selection. Notwithstanding the relevance of this prior work, collaborative P2P search is substantially more challenging than metasearch or distributed IR over a small federation of sources such as digital libraries, as these approaches mediate only a small and rather static set of underlying engines, as opposed to the high dynamics of a P2P system.

3. P2PDATING: AN OVERVIEW

The idea of p2pDating is to create semantic overlay networks in a P2P environment, where a peer has autonomy when deciding which SONs it wants to join. The approach works by having peers meeting other peers that they still do not know (like "blind dates"). If a peer "likes" another peer, i.e, if this other peer has information that is interesting for the peer, it might want to remember this peer, and insert it into the *friend list*. On the other hand, if the peer decided that the other peer is not interesting, it is most likely that it might not want to remember this peer, so no link is created or if there is already a link between them, it might be dropped. We believe that caching (i.e. remembering) of high quality peers is the natural way to create semantic overlay networks. It can either be seen as the criteria to create network links between peers, or also be used in existing peer-to-peer systems, as there is no need to change the

existing infrastructure. A real life example is that even if there exist a large number of pizza services, people usually have only one or two services that they call regularly. This is a kind of intuitive caching. If people have a few favorite services for each kind of food (Italian, Mexican, German) this can be seen as topic-specific "caching".

The process starts with a randomly connected network and runs infinitely since peers join and leave the network at a high rate. Semantic overlay networks will dynamically evolve from this process, as semantic links are more and more refined. In a dynamic P2P network, we expect that the SONs are continuously changing to adapt to the changes in the network, the peers' behavior, the peers' content, etc. The semantic links are represented by entries in the friend list. When a peer joins the network its friend list is empty and will be filled over time.

Each peer has two types of links: random links and semantic links. The random links are needed to keep the whole network together and are dictated by the underlying P2P network protocol.

3.1 The Semantic Routing Table

The friends are annotated with statistics to form a *semantic routing table* (SRT) in which the peers are ordered according to their usefulness. SRTs are maintained in a soft-cache. Figure 1 shows an example of a semantic routing table.

	ip	port	overlap	similarity	credits	last used	usage freq
Peer A	XXXXXX	8040	4%	70%	434	two days	34
Peer B	XXXXXX	8090	1%	30%	344	yesterday	12
Peer C	XXXXXX	8040	7%	50%	121	today	4

Figure 1: Example of a semantic routing table containing statistics about known friends.

There are many ways to define a friend, as will be discussed in Section 4. The values displayed in the example are just an illustration.

When a new friend is found, an entry containing information about this peer is added to the table. Friend lists have a fixed length, which means that current friends might need to be dropped (according to some criteria) from the table, so that new friends can be added. Dropping a friend corresponds to remove an abstract link in the network. Each peer creates its friend list totally independent from other peers. In particular, "friendship" is not in general symmetric. If peer A adds peer B into its friend list, A is not automatically inserted into B's list. It is up to B to decide whether to add A or not. This means that the links created by p2pDating form a directed graph.

3.2 Finding New Friends

Friend lists, besides defining the links in the SONs, can also be used to find new friends in a very intuitive manner, by routing over semantic routing tables of different peers: if a peer A finds an interesting peer B, is very likely that the friends of B will be interesting to A as well. Therefore, besides adding B into A's friend list, we also add B's friends in a so called *candidate list*. Then, for the next meeting, a peer can choose to meet a friend of one of its friends, instead of picking a peer at random since being a friend of a friend is a strong recommendation. Candidate lists, like friend lists, also have fixed length, which means that if the maximum number is reached, candidates have to be dropped. Instead of using these candidate lists, a peer might consider searching for new friends only when needed, e.g., if the peers in its semantic neighborhood join and leave the system at a extremely high rate. This will reduce the cost for maintaining the candidate list, e.g., removing dead links.

3.3 Putting Everything Together

Algorithm 1 shows the procedure of picking a peer for the next meeting. According to some predefined probabilities, it can be a peer from the candidate list or a peer from the friend list, or a random peer in the network.

Algorithm 1 The *choosePeerToMeet*() procedure

1: $P \leftarrow$ a peer from the candidate list, with probability α
2: $P \leftarrow$ a peer from the friend list, with probability β
3: $P \leftarrow$ a random peer in the network, with probability $(1 - \alpha - \beta)$
4: return P

We can think of scores for peers in the candidate list, based on the scores of the peers where they were defined as friends. Thus, we can select the peer with the highest score when choosing a peer from the candidate list. Its important that peers have an updated view of the network, as peers can change their contents or eventually leave the network. Therefore, peers have to visit their friends from time to time. However, friends will be visited during query execution anyway, so that these update can be integrated into the standard querying process. Another possibility is to assign a *time to live* (TTL) to every friend so that peers can automatically be dropped, or visited to re-assess their usefulness. In addition, the probability of picking a peer at random should not be equal to zero, as many peers might not be reachable by only following the chain of friends.

Algorithm 2 shows the pseudo code for the p2pDating algorithm. A peer chooses another peer for the next meeting and contacts it. Then it decides whether the peer is a friend or not, based on the peer's content. If so, the peer is added to the friend list and the friends of the peer are added to the candidate list. The process of adding peers to the friends or candidate list checks if the maximum number of the peers on the list has been reached, and removes peers, if necessary.

Algorithm 2 p2pDating Algorithm

1: **repeat**
2: $P \leftarrow$ *choosePeerToMeet*()
3: contact P
4: **if** isFriend(P) **then**
5: add(P, friend list)
6: $C \leftarrow$ friends of P
7: add(C, candidate list)
8: **end if**

Replacement Strategies

To avoid that the friend/candidate lists grow forever we need a replacement strategy that keeps track about peers that are no longer interesting and thus replaced by other peers or just dropped from the cache, e.g., if it turns out that these peers have left the network. As we limited the size of a friend list, we use a ranking of friends so that if the size of the list reaches its maximum, the lowest-rank friend is dropped. The friend list's order is defined by a combination of measures that will be presented in the following section.

4. DEFINING GOOD FRIENDS

As explained in the previous section, when a peer A meets peer B in the network, it accesses B's content [1] and decides whether to establish a link to B or not, based on a measure of the quality/usefulness of peer B. This criteria can be a combination of different measures like good behavior in the past, collection similarity, overlap between the collection, and authority scores.

[1]By peer's content we mean the information that the peer had made visible for the others.

Figure 1 also shows some measures that can be used to find the most promising peer for a particular query. It shows that it is everything but trivial to decide which peer to choose since, for instance, peer A is the best choice if we consider the number of credit points, whereas peer B seems to be most promising if we take a look at the overlap. So, obviously, there is great need for an aggregation function that combines the single measures in a meaningful way, since selecting a peer based only a particular measure can be misleading. For instance, it might be the case where a high quality peer has many documents that we already know, and a peer that offers a lot of new information have a lower quality measure.

In order to improve the efficiency of the usefulness assessment, we might not want to access a peer's complete collection but access each topic-specific subset individually. Although this creates additional cost, it will increases the accuracy of the quality assessments, since comparing the semantic similarity of two collections might be misleading in the case where collections are related to more than one topic. It is up to the peers how they classify their content, this extension does not require a globally given classifier. Whenever two peers meet they access the usefulness for each of the particular topic-specific subsets. This requires that each peer maintains more than one semantic routing table, more precisely, one for each topic it is interested in.

In addition, peers' classification schemes can be annotated with edge weights that correspond to the topic-subtopic similarity. Thus, peers can leverage the (now) topic-specific semantic routing table even in the case where the query does not correspond directly to a topic for which the peer has created a SRT since the edge weights can be interpreted as some kind of confidence measure that gives weight to the semantic query routing, or to the traditional routing respectively. In case where the query fits to a specific topic, SRTs from sub-topics or from more general topics can be incorporated into query routing using a weighted quality assessment. This decreases the risk of querying the wrong (or not perfectly matching) peers caused by a sub-optimal query to topic classification.

In addition to these peer-to-peer quality assessments, we can also think of having global measures about the top queries, top terms, top authoritative peers, etc. This is particularly of interest in search engines, where query results for very frequent queries can be cached.

The following subsections describe some of the other possible measurements that can be used to identify good friends in the network.

4.1 History

As recently proposed by [36], remembering excellent behavior in the past is a natural way to find friends. We can, for instance, give *credit-points* to peers for good cooperation. This also decreases the impact of malicious peers and can be seen as an incentive mechanism as it can be used to prioritize incoming queries from friends.

4.2 Overlap

Avoiding the retrieval of duplicate documents is a crucial issue in large scale distributed information system. We consider autonomous peers having their own local collection, generated by focused web crawls. The problem inherently associated with this scenario is that collections can have a high mutual overlap, thus, it is likely that the query initiating peer will retrieve documents that it already knows from its local collection. High quality documents are useless if there are already known: there is no need in querying a peer when it is known before-hand that this peer has an extremely high overlap with regard to the own collection. The mutual overlap between peers has to be taken into account while selecting promising peers for a particular query. Overlap aware techniques [5] avoid retrieving a lot of redundant information so that a certain level of recall can be reached by querying fewer peers, compared to the non-overlap-aware approach. For instance, if the most promising peers have exactly (or nearly) the same collections only the first peer can deliver valuable results whereas the following peers will not contribute with any new documents. Note that we consider parallel query execution, where the query is sent to multiple peers in parallel, since sequential query execution will cause additional latency.

The overlap between two collections can be estimated using min-wise independent permutations [11, 12] that have proven to offer an accurate overlap estimation and at the same time are pre-computable and require only small bandwidth consumption. Using Bloom filters [9], compressed Bloom filters [25] or some random sampling technique might be another option. Recent work [5] proposes a technique for predicting the query-specific mutual overlap between peers.

We can also consider the notions of *Containment* and *Resemblance* as appropriate measures of mutual set correlation [11] that are defined as $Containment(S_A, S_B) = \frac{|S_A \cap S_B|}{|S_B|}$ and $Resemblance(S_A, S_B) = \frac{|S_A \cap S_B|}{|S_A \cup S_B|}$.

4.3 Semantic Similarity

We can measure the thematic similarity between two peers by comparing their bookmarks or their complete document collections. More specifically, we can compare the peers in three aspects: 1) regarding their URL sets, 2) regarding the term frequency distributions in the documents referenced by the bookmark lists, or 3) the term frequency distributions in their complete collections.

As for term distributions, we could use the relative entropy, also called the Kullback-Leibler distance [21], as a measurement for information inequality. The peer A issues to find thematically related peers so that the benefit of a candidate peer B is inversely proportional to the Kullback-Leibler distance that is calculated using the term frequency distributions in all documents in a collection or the documents referenced by the bookmark lists (and optionally also all hyperlink successors of these pages).

Note that the bookmark-based measure [6] does not only reflect the similarity between the bookmark lists themselves, but also the similarity between the index contents of peer A with the index contents of peer B, if the index of a peer has been constructed by crawling the Web with the local bookmarks as crawl seeds. Clearly, comparing bookmarks is much more efficient than comparing entire indexes.

Link Distribution inside Semantic Communities

Moreover, [19] shows that the web is self-organizing in the sense that web communities are formed automatically. These communities can be easily identified by considering the link distribution within these pages. It is shown that web pages have more links to other pages inside their community than to pages that are outside their community.

To give an example, we have created 10 topic specific collections by conducting a web crawl and in parallel classifying the collected documents into classes using our focused Crawler BINGO [32]. The classifier has been trained manually and the resulting collections are strongly related to the topics. Overall, the 10 collections contain $253,875$ documents and $2,416,910$ links between these pages. Figure 2 shows the distribution of outgoing links from sport pages. As we can see, sport pages mainly point to other sport pages, an observation that one can support by personal experiences when surfing through the world wide web. Please note that this skewed link distribution in inherently associated with the WWW.

In the following two sections we describe two applications that can benefit from our semantic overlay networks.

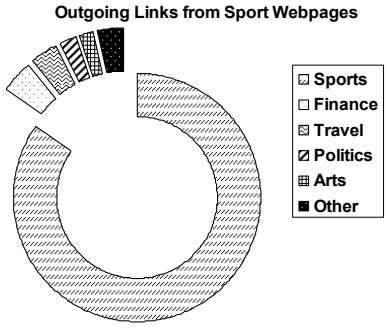


Figure 2: Outgoing Links from sport pages mainly point to other sport pages.

5. SONS FOR THE JXP DECENTRALIZED AUTHORITY RANKING

5.1 Reviewing the JXP Algorithm

JXP [28] is an algorithm for dynamically computing, in a decentralized P2P manner, global authority scores when the Web graph is spread across many autonomous peers. The algorithm runs at every peer and it combines local computations of the standard PageRank algorithm [10] and random meetings between peers in the network, to update peers' local knowledge. The main idea is as follows. It starts by adding to the local graph at each peer a special node, called *world node*, representing the part of the global graph that is not stored at and not known to the peer [2].

This world node has special features, regarding its own authority score and how it is connected to the local graph. As it represents all the pages that are not stored at the peer, all the links from local pages to external pages and make them point to the world node. In the same way, as the peer learns, during the random meetings, about external links that point to one of the local pages, these links are assigned to the world node. Every link from the world node is weighted based on how much of the authority score is received from the original page that owns the link. This gives a better approximation of the total authority score mass that is received from external pages. The world node also contains a self-loop link, representing links between pages that are not stored at the local graph. The authority score of the world node is defined as the sum of the scores of all external pages.

After the addition of the world node, the PageRank algorithm is performed on this extended graph and a first approximation to the authority scores of the pages is obtained.

The next step consists of meeting other peers in the network to exchange local knowledge and improve the local scores. Whenever two peers meet, they combine both local and external information. This is done at both peers independently of each other, so that the autonomy of peers and the asynchronous nature of communication and computation in a P2P network is preserved.

Local graphs are combined by simply forming the union between them. Combining the world nodes consists of merging their list of outgoing links, removing links that originally come from a page that is already represented in the combined graph, and adjusting the authority score of the combined world node to reflect the sum of scores of pages that do not belong to the combined graph. The new world node that results from this merging is then connected to the combined graph and the PageRank power iteration algorithm is again performed, yielding updated authority scores. The graphs are then disconnected and the local world node is recreated from the combined world node, by keeping only the links that point to a page in the local graph. The authority score of the world node is also recomputed. Everything else is then discarded. Figure 3 illustrates the process of combining and disconnecting local graphs and world nodes.

The algorithm is scalable, as the algorithm always runs on relative small graphs, independent of the number of peers in the network and the storage requirements are low, as peer do not keep the graphs of the peers they have met.

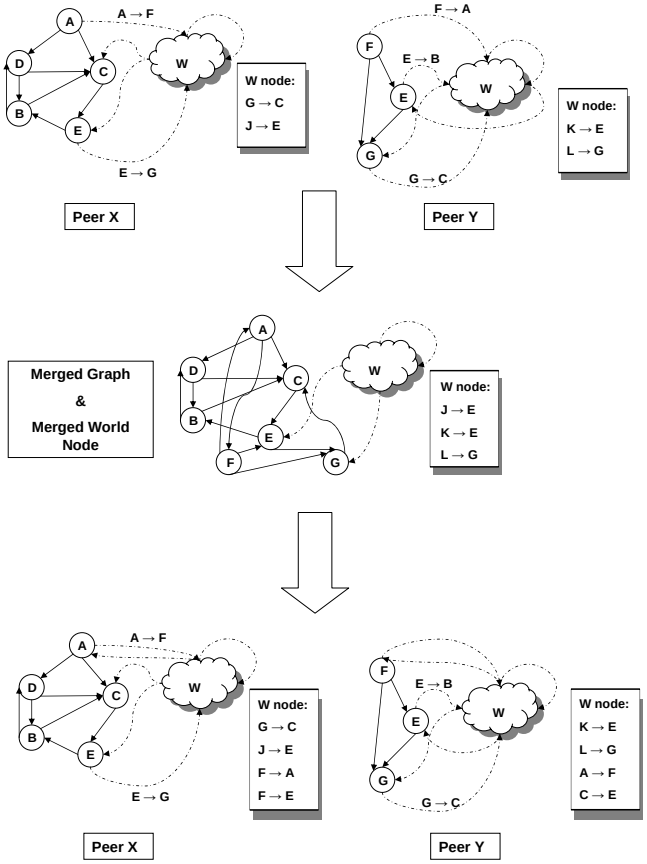


Figure 3: Illustration of the JXP meeting process.

Experiments have shown that, as a peer learns (through meetings) about other peers in the network, the locally computed authority scores converge to the true global PageRank scores and after a moderate number of meetings, a good approximation to the global PageRank is obtained.

5.2 Benefits from p2pDating

It is clear that, in the JXP algorithm, peers do not contribute with each other in the same way. How much a peer A will benefit from peer B for improving its local authority scores heavily depends on the degree of connectivity between their two local graphs and the level of overlap between them. Peers will not gain much from meetings when there are only a few links between the local graphs or if the local graphs are almost identical.

The performance of the algorithm can be improved if the peers are organized into semantic overlay networks, for ex-

[2]This is an application of the state-lumping techniques used in the analysis of large Markov models.

ample, when the semantic similarity is one of the criteria for defining good friends, as semantic similarity in combination with a certain overlap is a good indicator for the degree of web graph connectivity. In this way, peers are able to identify the best peers to meet, instead of choosing peers at random, which leads to fewer meetings to reach a good approximation of the global authority scores.

To confirm this idea, we performed an experiment in which the JXP and the p2pDating algorithms are combined such that whenever two peers meet in the network this meeting is used to improve both the JXP scores and the peer's friend and candidate lists. As our data collection, we chose a subset of the Amazon.com dataset. The Amazon.com dataset contains information about approximately 126,000 products (mostly books) offered by Amazon.com. The data was obtained in February 2005, through a Web Service provided by Amazon.com [1], and it is equivalent to a partial crawl of the corresponding web site. The graphs were created by considering the products as a node in the graph. For each product, pointers to similar recommended products are available in the dataset. These pointers define the edges in our graphs. Moreover, Amazon.com classifies every product into one or more predefined categories.

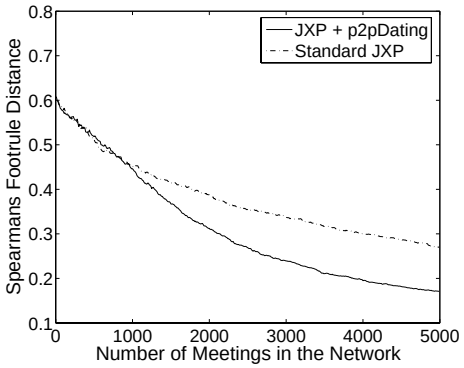
In our setup, each peer contains only pages from one out of five different categories, namely, *Cooking, Food & Wine*, *Mystery & Thrillers*, *Science Fiction & Fantasy*, *Sports* and *Teens*. Overlaps among the peers' local graphs were allowed and the criteria for defining a good friend was a combination of semantic similarity and overlap: peer A considers peer B as friends if B contain documents from the same category as A, and if the overlap between A and B is low [3].

We created a P2P network with 100 different peers. The peers were evenly distributed among the categories, which means that 20 peers contain pages from the same category. The total number of pages in the network was 5,223, and the maximum number of peers in the friend and candidate list was 10 and 20, respectively. We then, runned both the JXP and the p2pDating algorithms, using the procedure for choosing a peer for the next meeting described in Algorithm 1, with α equals to 0.5 and β equals to 0.3. This means that the peer chosen for the next meeting is one of the peers from the candidate list with 50% probability, one of the peers from the friend list with 30% probability, and a random peer in the network with 20% probability.

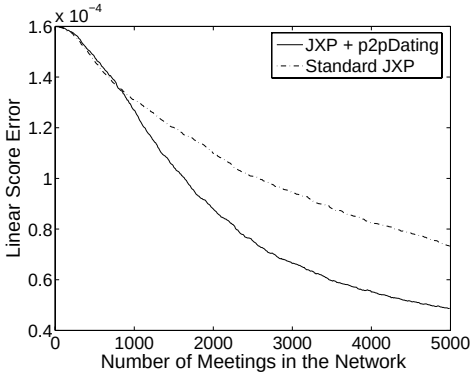
For evaluating the performance, after every interval of 10 meetings in the network, we combine the scores of all pages to construct a global ranking. If a page occurs in more than one peer we take the average of its score at the different peers as its global score. We then, compare this global ranking against the true global PageRank ranking, obtained by running the PageRank algorithm on the union of the peers' local graphs. For comparing these two rankings we measure the Spearman's footrule distance, defined as $F(\sigma_1, \sigma_2) = \sum_{i=1}^{N} |\sigma_1(i) - \sigma_2(i)|$ where $\sigma_1(i)$ and $\sigma_2(i)$ are the positions of the page i in the first and second ranking, and N is the total number of pages. We also measure the *linear score error*, i.e., the average of the absolute difference between the scores of the pages in the two rankings. $LS(\phi_1, \phi_2) = (\sum_{i=1}^{N} |\phi_1(i) - \phi_2(i)|)/N$, where $\phi_1(i)$ and $\phi_2(i)$ are the scores of the page i in the first and second ranking. We also evaluate the performance of the standard JXP, i.e., where the next peer to meet is always chosen at random. The results are shown in Figure 4.

We can see that the combination of the JXP and p2pDating algorithms outperforms the standard JXP, and a good approximation of the global PageRank scores can be achieved with a smaller number of meetigs in the network.

[3]The overlap is measured using the notion of *Containment* defined in Subsection 4.2.



(a)



(b)

Figure 4: Results showing the benefit of the p2pDating. Figure 4(a) shows the spearman's footrule distance and Figure 4(b) shows the linear score error.

6. SONS FOR MINERVA

6.1 Revisiting the MINERVA System Design

We briefly introduce MINERVA[4], a fully operational distributed search engine that we have implemented and that serves as a valuable testbed for our work[8, 7]. It assumes a P2P collaboration in which every peer is autonomous and has a local index that can be built from the peer's own crawls or imported from external sources and tailored to the user's thematic interest profile. The index contains inverted lists with URLs for Web pages that contain specific keywords.

A conceptually global but physically distributed directory, which is layered on top of a Chord-style Dynamic Hash Table (DHT), holds compact, aggregated information about the peers' local indexes and only to the extent that the individual peers are willing to disclose. MINERVA only uses the most basic DHT functionality, *lookup(key)*, that returns the peer currently responsible for *key*. Doing so, the term space is partitioned, such that every peer is responsible for a randomized subset of terms within the global directory. For failure resilience and availability, the entry for a term may be replicated across multiple peers.

Directory maintenance, query routing, and query processing work as follows (cf. Figure 5). In a preliminary step (step 0), every peer publishes a summary (*Post*) about every term in its local index to the directory. A hash func-

[4]Project homepage available at http://www.minerva-project.org

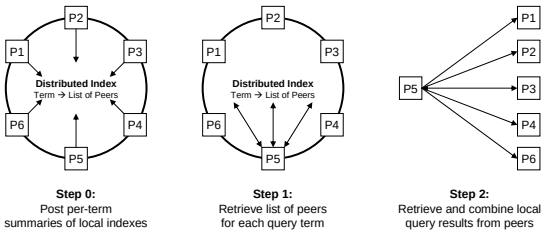


Figure 5: MINERVA System Architecture.

tion is applied to the term in order to determine the peer currently responsible for this term. This peer maintains a *PeerList* of all postings for this term from peers across the network. Posts contain contact information about the peer who posted this summary together with statistics to calculate IR-style measures for a term (e.g., the size of the inverted list for the term, the maximum average score among the term's inverted list entries, or some other statistical measure). These statistics are used to support the query routing process, i.e., determining the most promising peers for a particular query using database selection methods [8] from distributed IR [14]. This step, called *query routing*, yields a number of promising peers for the complete query. Subsequently, the query is forwarded to these peers and executed based on their local indexes (*query execution*; step 2). Note that this communication is done in a pairwise point-to-point manner between the peers, allowing efficient communication and limiting the load on the global directory. Finally, the results from the various peers are combined at the querying peer into a single result list. Due to efficiency reasons, the query initiating peer does not have to retrieve the complete PeerLists. Instead, it can run a distributed top-k algorithm to efficiently figure out the k most promising peers.

6.2 Benefits from p2pDating

The proposed techniques to created semantic overlay networks can be easily integrated into MINERVA as most of the information can be piggybacked onto the existing communication.

Random peer dates can be implemented by conducting a random lookup[5] in the underlying DHT or by executing a random query. This can be implemented as a background process or be combined with the traditional query processing.

Firework Query Routing

As described above, the query routing in Minerva uses standard IR techniques to figure out the most promising peers for a given user query. The selected peers are then queried by sending to them the complete query that will then be locally executed at the peers' collections. In our approach of semantic overlay networks, each peer is considered to have a logic neighborhood consisting of high quality and semantically related peers that can deliver reasonably good answers. The proposed notion of a *Firework Query Model* [26], seems to be excellently suitable here. We can use MINERVA's query routing process to find a few suitable peers for a given query and, subsequently, forward the query to their neighborhood. The query is routed straight into a small, but high quality community established by the primarily selected peer(s). In contrast to the work presented in [26], we do not have to rely on random links to find the appropriate peer cluster, as we can use MINERVA for that issue. This decreases the network resource consumption, as only a small number of lookups/hops are needed to determine appropriate peers inside such a semantic cluster.

[5]draw a random key and perform a lookup.

Pseudo-relevance Feedback

Recently, we have worked on pseudo-relevance feedback in the context P2P Web Search, where the query is first executed at the most promising peer that returns query results, and an expanded query created by pseudo-relevance feedback using its collection. Subsequently, the query initiator can forward the expanded query to the other promising peers, or can conduct an additional query routing step, thus using statistics for the new query terms that have not been retrieved earlier. Using SONs and the above discussed "firework approach" we can easily employ pseudo-relevance feedback: The most promising peer receives the query, expands it, and forwards it to the peers in its neighborhood that return the query results directly to the query initiator.

Pro-Active Dissemination

To further enhance query efficiency, statistical summaries and also the query results may be cached within a semantic cluster in a way that allows other peers not only to instantly benefit from the existing query results but also to benefit from click streams that were recorded on the occasion of similar queries. Statistical summaries may also be disseminated pro-actively among thematically related peers.

Enhanced Query Routing

Inside a semantic community, peers could use the cached information together with the disseminated statistics to find appropriate peers, without stressing the global MINERVA directory. Moreover, using a hybrid routing strategy that uses the MINERVA directory, and the semantic routing table (cf. Figure 1) seems to be a promising approach that combines the robustness of the MINERVA directory with the, for thematically related queries, high accuracy of peers within a semantic community. The lazily, or pro-actively learned information about friends evolves over time and is self adapting to the peers' peculiarities whereas the MINERVA directory is rather static as it relies on the published statistics, and thus, might be influenced by malicious peers posting invalid statistics about their local contents.

Decrease Storage Load

In addition, MINERVA can benefit from semantic classification of the peers' content if we restrict the post process to the terms that are related to the peers' topics. This reduces the number of overall posts and thus increases system performance and decreases the overall storage consumption and computational load.

7. DISCUSSION AND OUTLOOK

In this work we have presented an approach to create and maintain semantic overlay networks based on the notion of p2pDating, where peers maintain information about their friends to form a semantic network. Friends are chosen based on a whole variety of usefulness estimators, like overlap and semantic similarity. We have shown how we can leverage these friend networks in JXP and MINERVA. In particular, the experiment results that we presented have already shown the feasibility and the benefit of integrating the JXP with the p2pDating algorithm.

Ongoing work includes more experiments with different real web-data collections. The experiments are threefold, first we want to measure the performance gain for the JXP algorithm, second, we want to study how query processing in MINERVA can be improved using the presented techniques, and finally we want to combine JXP and MINERVA. The combination of JXP and MINERVA yields another nice side-effect as the JXP authority score can be used within the peers' local collections to have globally comparable document scores. In all experiments we want to keep track of

the evolving semantic structure that can be compared to an ideal one given by a clustering technique with global knowledge. Ultimately, we want to refine the presented heuristics to choose friends/candidates, including a sophisticated $benefit/cost$ ratio. Defining a meaningful cost measure, however, is a challenging issue. While there are techniques for observing and inferring network bandwidth or other infrastructural information, expected response times, depending on the current system load, change over time. One approach is to create a distributed Quality-of-Service directory that, for example, holds moving averages of recent peer response times.

8. REFERENCES

[1] http://www.amazon.com/gp/aws/landing.html.

[2] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, and T. V. Pelt. Gridvine: Building internet-scale semantic overlay networks. Technical report, EPFL, 2004.

[3] K. Aberer, M. Punceva, M. Hauswirth, and R. Schmidt. Improving data access in p2p systems. *IEEE Internet Computing*, 2002.

[4] M. Bawa, G. Manku, and P. Raghavan. SETS: Search enhanced by topic segmentation. In *SIGIR 2003*.

[5] M. Bender, S. Michel, P. Triantafillou, G. Weikum, Z. Christian. Improving collection selection with overlap awareness in p2p search engines. In *SIGIR*, 2005.

[6] M. Bender, S. Michel, G. Weikum, and C. Zimmer. Bookmark-driven query routing in peer-to-peer web search. In *SIGIR Workshop on P2P-IR 2004*.

[7] M. Bender, S. Michel, G. Weikum, and C. Zimmer. Minerva: Collaborative p2p search. In *Proceedings of the VLDB Conference (Demonstration)*, 2005.

[8] M. Bender, S. Michel, G. Weikum, and C. Zimmer. The MINERVA project: Database selection in the context of P2P search. In Datenbanksysteme in Business, Technologie und Web *BTW*, 2005.

[9] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 1970.

[10] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems* 1998.

[11] Broder. On the resemblance and containment of documents. In SEQUENCES '97: Proceedings of the Compression and Complexity of Sequences 1997.

[12] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences* 2000.

[13] E. Buchmann and K. Böhm. How to Run Experiments with Large Peer-to-Peer Data Structures. In *IPDPS 2004*, Apr. 2004.

[14] J. Callan. Distributed information retrieval. In *Advances in information retrieval, Kluwer Academic Publishers.* 2000.

[15] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, San Francisco, 2002.

[16] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems. Technical report, computer science department, Stanford university, October 2002.

[17] F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen. PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. Technical Report DCS-TR-487, Rutgers University, Sept. 2002.

[18] R. Fagin. Combining fuzzy information from multiple systems. *J. Comput. Syst. Sci. 58(1), 1999.*

[19] G. W. Flake, S. Lawrence, C. L. Giles, and F. Coetzee. Self-organization of the web and identification of communities. *IEEE Computer 35(3), 2002.*

[20] N. Fuhr. A decision-theoretic approach to database selection in networked IR. *ACM Transactions on Information Systems, 1999.*

[21] S. Kullback. *Information Theory and Statistics*. Wiley, New York, 1959.

[22] M. Li, W.-C. Lee, and A. Sivasubramaniam. Semantic small world: An overlay network for peer-to-peer search. In *ICNP 2004*.

[23] J. Lu and J. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *CIKM, 2003*

[24] J. Luxenburger and G. Weikum. Query-log based authority analysis for web information search. In *WISE, 2004*.

[25] M. Mitzenmacher. Compressed bloom filters. *IEEE/ACM Trans. Netw., 2002.*

[26] C. Ng, K. Sia, and I. King. Peer clustering and firework query model in the peer-to -peer network. Technical report, Chinese University of Hongkong, department of computer science and engineering, 2003.

[27] H. Nottelmann and N. Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. In *SIGIR, 2003.*

[28] J. X. Parreira and G. Weikum. JXP: Global authority scores in a p2p network. In *WebDB, 2005.*

[29] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *SIGCOMM, 2001.*

[30] P. Reynolds and A. Vahdat. Efficient peer-to-peer keyword searching. In *Middleware, 2003.*

[31] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware, 2001.*

[32] S. Sizov, M. Biwer, J. Graupmann, S. Siersdorfer, M. Theobald, G. Weikum, and P. Zimmer. The bingo! system for information portal generation and expert web search. In *CIDR, 2003.*

[33] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM, 2001.*

[34] T. Suel, C. Mathur, J. Wu, J. Zhang, A. Delis, M. Kharrazi, X. Long, and K. Shanmugasunderam. Odissea: A peer-to-peer architecture for scalable web search and information retrieval. Technical report, Polytechnic Univ., 2003.

[35] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *SIGCOMM, 2003.*

[36] C. Tempich, S. Staab, and A. Wranik. REMINDIN': Semantic query routing in peer-to-peer networks based on social metaphors. In *WWW, 2004.*

[37] P. Triantafillou, C. Xiruhaki, M. Koubarakis, and N. Ntarmos. Towards high performance peer-to-peer content and resource sharing systems. In *CIDR*, 2003.

[38] Y. Wang, L. Galanis, and D. J. deWitt. Galanx: An efficient peer-to-peer search engine system. *Available at http://www.cs.wisc.edu/ yuanwang.*