Peer-to-Peer Similarity Search over Widely Distributed Document Collections

Christos Doulkeridis
Dept.of Informatics
AUEB
Athens, Greece
cdoulk@aueb.gr

Kjetil Nørvåg
Dept.of Computer Science
NTNU
Trondheim, Norway
Kjetil.Norvag@idi.ntnu.no

Michalis Vazirgiannis
Dept.of Informatics
AUEB
Athens, Greece
mvazirg@aueb.gr

ABSTRACT

This paper addresses the challenging problem of similarity search over widely distributed ultra-high dimensional data. Such an application is retrieval of the top-k most similar documents in a widely distributed document collection, as in the case of digital libraries. Peer-to-peer (P2P) systems emerge as a promising solution to delve with content management in cases of highly distributed data collections. We propose a self-organizing P2P approach in which an unstructured P2P network evolves into a super-peer architecture, with super-peers responsible for peers with similar content. Our approach is based on distributed clustering of peer contents, thus managing to create high quality clusters that span the entire network. More importantly, we show how to efficiently process similarity queries capitalizing on the newly constructed, clustered super-peer network. During query processing, the query is propagated only to few carefully selected super-peers that are able to return results of high quality. We evaluate the performance of our approach and demonstrate its advantages through simulation experiments on two document collections.

Categories and Subject Descriptors

H.3.3 [**Information storage and retrieval**]: Information search and retrieval; H.3.1 [**Information storage and retrieval**]: Content Analysis and Indexing

General Terms

Algorithms, Design, Performance

Keywords

Distributed and peer-to-peer search, semantic overlay networks

1. INTRODUCTION & MOTIVATION

This paper addresses the challenging problem of similarity search over widely distributed ultra-high dimensional data. Such an application is retrieval of the top-k most similar documents in a widely distributed document collection. The massive amounts of distributed high-dimensional data, such as digital libraries and web accessible text databases, motivate our work towards an infrastructure for efficient similarity search in peer-to-peer (P2P) environments. The overall goal is for a set of cooperative computers to support advanced search mechanisms that go beyond exact matching and involve ranking.

P2P systems emerge as a promising solution to delve with data management in cases of high degree of distribution. Contrary to centralized systems or traditional client-server architectures, nodes participating in a large P2P network, store and share data in an autonomous manner. Such nodes can be information providers, which do not wish to expose their full data to a server or be part of a centralized index. Therefore, a grand challenge is to provide efficient and scalable searching, in a context of highly distributed content, without necessarily moving the actual contents away from the information providers. Then the problem of lack of global knowledge – in the sense that each peer is aware of only a small portion of the network topology and content – needs to be dealt with. The solutions proposed in such an unstructured P2P environment usually lead to approaches that incur high costs, and thus deter the design of efficient searching over P2P content.

In this context, the first challenge is to organize content in an unsupervised, decentralized and distributed way. Unstructured P2P systems in their basic form suffer from high search costs in terms of both consumed bandwidth and latency, so in order to be useful for real applications, more sophisticated search mechanisms are required. An approach that has been proposed recently is the use of *Semantic Overlay Networks* (SONs) [3, 6], where peers containing relevant information are grouped together in overlay networks. Once SONs have been created, queries can be forwarded only to the most similar SONs to the given query, thus reducing the query cost and, at the same time, increasing the quality of results (mainly in terms of precision). More advanced search mechanisms, that go beyond exact matching, such as similarity search, can then be deployed on top of the newly generated SONs.

In this paper, we propose a novel approach for constructing SONs in an unstructured P2P network, aiming to use SONs as the underlying infrastructure for efficient similarity search. A main objective of our work is the distributed and decentralized generation of SONs. Peers with similar content eventually become part of a logical cluster, resulting in a super-peer architecture, where each super-peer becomes responsible for a thematically focused group of peers, namely a SON. In this way, a similarity query can be guided to the N most similar - to the query - overlays, thus achieving quality of results comparable to centralized search and at the same time avoiding the excessive cost of exhaustive search.

The main contribution of this work is a scalable approach for distributed SON creation in unstructured P2P networks, and al-

LSDS-IR '08, October 30, 2008, Napa Valley California, USA

gorithms for recursively merging clusters of peer data and creating links between peer groups, in order to generate well-connected SONs with similar content that span the entire P2P network. The innovation of this paper relative to previous related work on organizing content in large P2P networks and P2P similarity search lies in: 1) our approach to P2P clustering is *completely unsupervised* (we make no assumptions of common ontology or common knowledge among peers) and contrary to other approaches, such as gossiping [25], random meetings [2, 18], assumptions on peer contents' probability distributions [11], it guarantees that the resulting clusters span the entire network, and 2) while most P2P similarity search approaches rely on *moving content or detailed indexes to remote peers* usually employing a DHT infrastructure [17, 20], our approach relies on an unstructured P2P network that evolves into a super-peer architecture, hence the peer's actual data do not need to be stored at remote peers arbitrarily defined by a hash function. The merits of our approach are evaluated on two large document datasets (GOV2 and Reuters), thus proving the applicability of our ideas and demonstrating both the result quality and the performance gains experimentally. In particular, by contacting the best 3 super-peers (SONs), we manage to achieve 85-96% of the recall[1] that would be achieved if all data were available in a central location.

The organization of the rest of this paper is as follows: In Section 2, we give an overview of related work. In Section 3, we describe in detail the process of distributed semantic overlay network generation and how the unstructured P2P network evolves into a clustered super-peer network. Section 4 describes how similarity search can be efficiently performed using the SONs. In Section 5, we present experimental results, acquired through simulations. Finally, in Section 6, we conclude the paper and outline future research directions.

2. RELATED WORK

Content organization in P2P systems has attracted a lot of attention recently. Semantic Overlay Networks (SONs) [3] have been proposed as an approach for organizing peers in thematic groups, so that queries can be forwarded to only those peers having content within specific topics. A different notion of SONs related to peers that are logically interconnected through schema mappings is presented in [1].

Liu *et al.* [12] propose HSPIR, a hierarchical SON based on CAN [19] and Range Addressable Network. Support for semantics is achieved by using Latent Semantic Indexing (LSI). Tang *et al.* [24] propose a system for SON creation on top of structured P2P networks using LSI. Improvements to this approach are presented in [12, 14]. However, some of the inherent problems of LSI, like processing cost, choosing number of dimensions etc., make it difficult to employ this technique in a large-scale dynamic document repository. More importantly, both [12] and [24] assume a central LSI computation, which presumes that all document representations or a sample of reasonable size must be assembled at a central location, which is not scalable for a large P2P system. In contrast to these approaches, we do not presume a central point that collects *all* documents, and we believe that a feasible P2P approach for search using LSI should focus on distributed LSI computation. Hence, this is a family of LSI-based approaches that cannot be compared to our work directly.

Although several papers describe how to use SON-like structures, little work exists on how to actually create SONs in an unsupervised, decentralized and distributed way in unstructured P2P networks. Cholvi *et al.* [2] propose the use of *acquaintances* as an extension to Gnutella-like networks to improve searching. Peers with similar contents are linked together, so that searches for a particular topic can be routed to more relevant peers in less time. A similar approach has been described in [18]. Other relevant approaches include gossiping algorithms [25], which have been proposed as an alternative to flooding for routing queries in unstructured P2P networks. A shortcoming of these approaches is that they are restricted to networks of limited size, as they provide no guarantees that remote peers will get acquainted in very large P2P networks. Recently, a self-organizing super-peer network architecture is presented, named SOSPNET [8], which deals with the issue of how clients connect to a super-peer.

In previous work [6], we have proposed the DESENT protocol for generating hierarchical semantic overlays. In this paper, we present novel algorithms for SON merging that do not rely on a fixed hierarchy, rather the focus is on building SONs that eventually form a super-peer network, with a super-peer being responsible for each SON. This results in a more robust and self-organizing architecture that has a less hierarchical structure and is better in terms of load-balancing and fault tolerance. Moreover we describe how similarity search can be performed over the newly formed SONs and perform a large-scale evaluation of the approach.

Content-based search in P2P networks [20] is usually related to full-text search [13, 23, 26], with most approaches relying on the use of structured P2P networks. Some research focuses on providing P2P web search functionalities, like in [15], where MINERVA∞ is presented, a P2P web search engine that aims at scalability and efficiency. In MINERVA∞, each peer decides what fraction of the Web it will crawl and subsequently index. Previous approaches regarding P2P web search have focused on building global inverted indices, as for example Odissea [22] and PlanetP [4]. A major shortcoming of all these approaches is that their efficiency degrades with increasing query length and thus they are inappropriate for similarity search. Recently, approaches have been proposed that reduce the global indexing load by indexing carefully selected term combinations [21].

Recently, an approach for P2P similarity search in metric spaces, called SIMPEER has been proposed [7]. Similarly to this work, SIMPEER employs a super-peer architecture and it is applicable for distributed document collections, since it is designed for metric spaces. However, SIMPEER focuses more on the performance of query processing for an existing super-peer network. Recently a number of papers on P2P similarity search has appeared, e.g., [9]. However, these approaches are not suitable for similarity search in document collections because of the very high dimensionality (which is equal to the vocabulary cardinality).

3. SON CONSTRUCTION

In this section we describe how the semantic overlays are created. Initially, peers are connected in an unstructured P2P network, hence each peer is only aware of a limited number of direct neighbors. Peers are assumed to store documents locally, as in the case of a distributed digital library. First we describe the local preprocessing on each peer joining our system (Section 3.1), followed by the SON construction process (Section 3.2). Peer dynamics and maintenance are described in Section 3.3.

Semantic Overlay Networks (SONs) refer to groups of peers with similar contents. At this point, it should be stressed that SONs do not necessarily imply use of semantics in the traditional sense (like ontologies), however this is the term first proposed in the literature [3] and we use it as such throughout the paper.

[1] *Recall* is the fraction of the documents that are relevant to the query that are successfully retrieved.

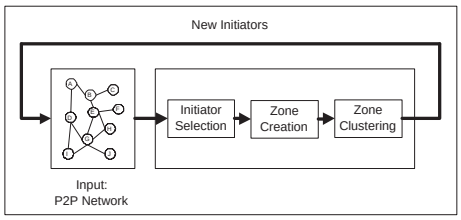

Figure 1: Illustration of overlay creation phases.

3.1 Pre-processing Phase

A pre-processing phase takes place on each peer independently of other peers. This phase includes typical techniques used in IR, such as document parsing, stop-word removal, tokenization, and calculation of local TF/IDF scores based on the peer's local contents only. Obviously other IR techniques or weight computation schemes can be employed, however in this paper we assume peers use TF/IDF. Then each peer runs a clustering algorithm on its local documents. Since local document clustering can be a costly process for a peer with many documents, it is left to the individual peers to decide how often this process should be performed, based on the rate of content change, so it does not necessarily have to be performed each time the global clustering process occurs.

In the end of this phase, each peer can provide to the network a set of clusters representing its local data. Each cluster c is represented by a *feature vector* F of tuples $F_i = (f_i, w_i, tf_i, df_i, D_i)$, where each tuple contains a feature/term f_i, the term's weight w_i, the term frequency tf_i and document frequency df_i, and the number of documents D_i represented by this feature vector. Although the set of terms and their respective weights are sufficient to represent the term's importance in the cluster, we keep some extra information in each tuple, in order to enable accurate weight computation in subsequent phases, as will be shown shortly.

3.2 Overlay Construction

SON creation is a multi-phase distributed process that runs recursively on specific sets of peers. Having as a starting point the initial unstructured P2P network, some *initiator* peers are selected in a pseudo-random way (*initiator selection phase*). Initiators create local topological zones over their neighboring peers (*zone creation phase*). Then each initiator collects the cluster descriptions of all peers in its zone, and executes a clustering algorithm in order to create new clusters that span the entire zone (*zone clustering phase*). Since the clusters of two (or more) peers may be merged into a new cluster, this implies that these peers become members of a SON, and the SON's contents are now represented by a new cluster description. In the subsequent steps, the initiators form the current (unstructured) P2P network, thus playing the role of peers in the initial setup. Therefore, the process described above runs on the initiators, thus new initiators are selected, that create zones and cluster zone contents in a completely similar way as shown above. Hence, zones and clusters are merged recursively until global clusters are obtained. Conceptually, the overlay creation steps are shown in Figure 1.

3.2.1 Initiator Selection

Given an unstructured P2P network of N_P peers, initiator selection is a pseudo-random distributed algorithm that runs on each peer and determines whether the peer will be assigned the role of initiator [6]. Initiators are peers with special roles, similar to super-peers, only they do not have fixed roles, but different initiators are selected each time the algorithm runs. This tackles the problem of being stuck with faulty initiators and it reduces the probability of permanent cheaters.

The global overlay network construction is assumed to start simultaneously at regular temporal intervals at all peers. The length of these intervals is a parameter of the protocol and can be as low as a few hours, thus ensuring that fresh contents are clustered and thus made retrievable. Because the aim of the clustering process is to achieve a global result, it is beneficial to perform the subsequent phases at the same time at the different peers in the network. Achieving (or even assuming) a common global time for temporal synchronization is not feasible in a large P2P network, and fortunately not necessary. Our technique to cope with this problem is to use a partial synchronization technique, making only the assumption that each peer has a clock that is accurate within a certain amount of time t_a (for example 1 minute deviation at most). In cases where peers have clock deviating more than t_a from actual time, this is detected by the fact that they are performing operations asynchronously to the other peers, and they are considered faulty.

3.2.2 Zone Creation

The goal of zone creation is to partition an unstructured P2P network into smaller subgroups. These subgroups are called *zones* and each zone is created by an initiator. Given a preferred zone size S_Z, initiator selection results in approximately one initiator – also called *zone initiators* – in every S_Z peers. During zone creation, the task of each initiator is to create a zone of peers by means of a local flooding-based algorithm. Using this algorithm, an initiator first captures its neighboring peers, then their neighbors, and so forth, as long as neighboring peers have not been captured by another initiator. These captured peers constitute the initiator's zone. Thus an initiator's zone is restricted by the boundaries of other initiators' zones.

Moreover, the algorithm ensures that any connected peer in the network finally belongs to some zone. We emphasize that a zone is a plain topological (not semantic) grouping of peers that are "around" a zone initiator, and obviously zones contain non-overlapping sets of peers. Both initiator selection and zone creation are completely distributed and they are performed using the algorithm presented in [6]. The basis of the zone creation algorithm is a probe message sent from each initiator and flooded until it is received by a node which is also initiator or has received a probe message from another initiator. Since a certain amount of peers are initiators and a zone will have limited size, the amount of messages is not very high.

Zone creation is a necessary step, in order to put an upper bound on the burden imposed on initiators. As will be shown next in detail, initiators assemble cluster descriptions of peers in their zone, and they perform clustering to create new clusters that span the contents of the entire zone. Thus, S_Z practically defines an upper limit on the number of peers that can belong to a zone, as well as the number of initiators. Hence, if peers in a network do not have high processing capabilities, S_Z can be set to a low value, in order to avoid excessive load on some initiators. Notice that in the case of a zone of excessive size, the initiator can decide to partition its zone, by splitting it into two or more zones.

3.2.3 Zone Clustering

After the zones have been formed, each zone initiator collects the feature vectors from the peers in its zone and creates new clusters based on similarity between the feature vectors. No restrictions are imposed on the choice of similarity (distance) function, so without loss of generality we use the cosine similarity (computed based on f_i, w_i) in the rest of this paper.

Algorithm 1 Cluster merging.

1: **Input:** Clusters C=$\{c_1 ... c_N\}$, Limit $L < N$, k
2: **Output:** Clusters C'=$\{c'_1 ... c'_L\}$
3:
4: **while** $sizeOf(C) < L$ **do**
5: getMostSimilarClusters(C, c_i, c_j)
6: Cluster $c_{new} \leftarrow c_j$
7: **for** ($t_i \in c_i$) **do**
8: **if** ($t_i \in c_j$) **then**
9: $w_{new} \leftarrow \frac{tf_i+tf_j}{\sum_{c_i,c_j} tf} \times log(\frac{1+D_i+D_j}{df_i+df_j})$
10: c_{new}.update(t_i, w_{new}, $tf_i + tf_j$, $df_i + df_j$, $D_i + D_j$)
11: **else**
12: $w_{new} \leftarrow \frac{tf_i}{\sum_{c_i} tf} \times log(\frac{1+D_i+D_j}{df_i})$
13: c_{new}.add(t_i, w_{new}, tf_i, df_i, D_i)
14: **end if**
15: **end for**
16: c_{new}.keepTopFeatures(k)
17: C.delete(c_i)
18: C.delete(c_j)
19: C.add(c_{new})
20: **end while**
21: $C' \leftarrow C$
22: **return** C'

Clustering feature vectors of different peers results in grouping the corresponding peers together. This is practically a grouping of peers based on their contents, therefore peers that are grouped together form a SON. Note that because each peer in general contributes more than one feature vector, a peer can be member of more than one SON.

Cluster merging (see Algorithm 1) consists of iteratively determining the two most similar clusters c_i,c_j from a set of clusters C (line 5) assembled at an initiator, computing the new feature vector, creating the new merged cluster, and then adding the new cluster to the list of clusters (line 19), until only L clusters are left. The new cluster c_{new} resulting from two merged clusters c_i and c_j will initially contain all terms in the feature vectors of c_i and c_j, with weights calculated in the way shown in Algorithm 1. Essentially, if a term t_i exists in both feature vectors of c_i and c_j, then its new weight and corresponding tuple in the merged cluster description is:

$$w_{new} = \frac{tf_i+tf_j}{\sum_{c_i,c_j} tf} \times log(\frac{1+D_i+D_j}{df_i+df_j})$$
$$F_{new} = (t_i, w_{new}, tf_i + tf_j, df_i + df_j, D_i + D_j)$$

whereas if t_i exists only in one cluster description, say c_i, then:

$$w_{new} = \frac{tf_i}{\sum_{c_i} tf} \times log(\frac{1+D_i+D_j}{df_i})$$
$$F_{new} = (t_i, w_{new}, tf_i, df_i, D_i)$$

This is practically a *progressive* TF/IDF calculation, and it is feasible since term frequencies (tf_i), document frequencies (df_i) and number of documents (D_i) are maintained in the cluster descriptions. As a final step, only the k top-weighted features of the new cluster description are kept.

As already mentioned, each cluster is associated with a SON, i.e. a set of peers. When two clusters c_i and c_j are merged, this necessarily implies that the SONs (S_i and S_j respectively) that they represent are linked together and form a new SON. In order to ensure peer connectivity within the new SON after merging, a number of d links are created between the two merged SONs. In this way, the probability that a SON will become disconnected due to peer failure is practically eliminated. The d links between the peers of the two SONs are formed by iteratively selecting from each SON, the two least connected peers, and then establishing a link that connects them. This is done in order to maintain a low peer connectivity degree. The algorithm ensures that there exists a path that connects each peer to any other peer in the new SON.

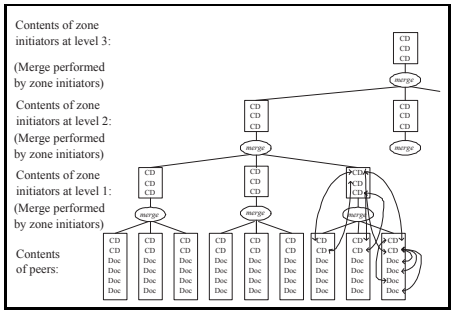

Figure 2: Recursive cluster merging.

Zone clustering at an initiator results into new *intra-zone clusters*, which are finally represented by cluster descriptors (CDs), one for each cluster c_i. A CD consists of the cluster identifier c_i, a feature vector F, and a random selection of N_r representative peers $\{R\}$ belonging to the SON, i.e., $\text{CD}_i = (c_i, F, \{R\})$. The set R of representative peers is created from a subset of (representative) peers from each of the merged SONs. The sole purpose of peers in R is to act as representative peers of the SON, hence they maintain connections to peers in the SON and they can forward any incoming message to all peers in the SON. In the special case when $N_r = 1$, the representative peer acts like a super-peer for the SON. It should be stressed that the value of N_r does not affect the quality of the generated SONs in any way. However, there exists a tradeoff when defining its value. High values minimize the probability of all representative peers disappearing (due to churn) during the lifetime of the CD. Low values avoid high communication cost when transmitting the CDs and maintenance cost when a representative peer fails.

3.2.4 Recursive Cluster and SON Merging

At the end of the three phases outlined above, initiators have finished zone clustering and some *intra-zone clusters* and *intra-zone SONs* have been created. Next, this process is recursively applied to the initiators only (as shown in Figure 1), since they now constitute an unstructured P2P network, with links between neighboring initiators established due to the zone creation phase. The overall aim is to generate SONs that span the entire network, not just a zone. Therefore, this recursive process continues, thus resulting in a hierarchy of initiators of different levels. Zones and clusters are merged recursively, as shown in Figure 2. The CDs from each peer in a zone are combined into new clusters. Illustrated as an example on the right bottom part of the figure is the relationship between documents and CDs in peers, and the relationship between upper-level clusters and CDs from level below.

At each level of the hierarchy, a number of neighboring zones are merged to one higher-level zone, where a zone initiator from the previous level is selected to act as zone initiator for the merged zones. When a level-$(i+1)$ zone is created from a number of level-i zones, CDs created at the previous level are collected by the level-$(i+1)$ initiator and the most similar clusters from the level-i zones are merged. More specifically, this initiator assembles the feature vectors (i.e. cluster descriptions) of the initiators of the zones one level below, and the feature vectors of that level are clustered in order to create the next level clusters and SONs. These SONs now span all peers belonging to the level-i zones that merged into one level-$(i+1)$ zone.

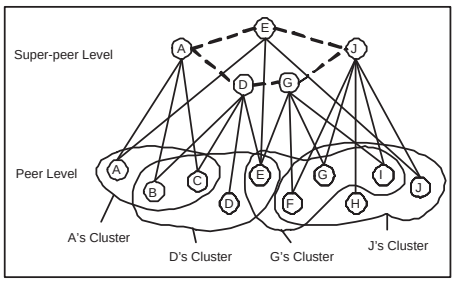

Figure 3: Final peer organization.

This recursive process continues until the final level with only one initiator/zone is reached (in practice the number of levels is small $\lfloor log_{S_Z} N_P \rfloor$). The result is a number of global clusters/SONs that span the entire network.

3.2.5 Final Organization

Due to the SON generation process, each peer P_i belongs to a set of SONs, that are sufficiently connected internally. However, in order to support queries directed to other SONs, the resulting SONs also need to be sufficiently interconnected. The top-level initiator picks from the set of representative peers of each SON, one peer that will act as super-peer for the specific SON. In practice more than one super-peers are chosen for each SON, in order to support load-balancing within each SON and ensure fault-tolerance. In addition, the initiator urges super-peers of the global SONs to create connections to other super-peers. These connections are used to ensure communication among SONs, in order to make any SON reachable from any other SON, and at the same time avoid its potential isolation from the rest of the network. In this way, the top-level initiator can essentially define the topological parameters of the super-peer network, such as number of super-peers per SON, super-peer connectivity degree, etc. For instance, if the super-peer topology is a hypercube [16], then each super-peer creates $log_2 N_{sp}$ links, where N_{sp} denotes the number of super-peers.

Then, each super-peer creates and maintains information about its peers, in order to be able to determine which peers are relevant to a query, without having to contact all peers. This is typically an index of peers' features. The final organization of the peers is depicted in Figure 3.

Furthermore, as an extra, optional step to the SON generation algorithm, we propose an *adaptive* strategy for document (and subsequently peer) membership in a cluster (SON). After the global SONs and their CDs have been determined, this information is broadcast to all peers in an efficient way using the zone hierarchy. Then each peer can recompute for each of its local documents, the most similar CD, and thus: 1) make this document become member of the associated cluster, and 2) the peer itself joins the respective SON. Thus the final organization of documents to global clusters (peers to SONs) is the result of this adaptive strategy, which can also be considered as a feedback mechanism to clustering. In the experimental part, we will show the advantages of the adaptive strategy, in terms of increased clustering quality.

Summarizing, we emphasize the following features of SON creation: 1) the algorithm ensures that peers within a SON remain well-connected and that sufficient connections exist among SONs, 2) peers having the roles of zone initiators are chosen at random and perform their tasks completely independently of each other, 3) the whole process is performed in a distributed and decentralized manner, 4) although SON creation resembles a hierarchical approach, the hierarchy is actually made obsolete and is not used anymore, right after SON creation, avoiding excessive load on certain peers or single points of failure, and 5) as a final result, the peers in the initial unstructured P2P network have formed a super-peer network in a self-organizing manner, with super-peers responsible for peers with relevant content.

3.3 Peer Dynamics & Maintenance

While the SON creation process can be performed at regular intervals (as shown in [6] the cost of constructing the hierarchical structure is acceptable, the burden on each peer is in general less than the typical load from various web crawlers accessing the site), new peers joining should also become members of SONs and thus contribute to the results of similarity search. This is achieved by the peer joining the SON with the highest similarity to its contents.

Note that no other changes will be performed to existing SONs (and their CDs) when new peers join, so after a while a cluster description might become less accurate. However, SON creation is performed at regular intervals and it creates a new peer and data organization that reflects also the data of new nodes. In this way, new or modified documents, which have changed the feature vectors of existing peers, are also taken into account. This strategy considerably reduces the maintenance cost in terms of communication bandwidth compared to incremental reclustering, and also avoids the significant computational cost that could be the result of continuous reclustering.

Churn during overlay creation is handled by k-replication, while multiple connections between peers in a SON assure a low probability of SON partitioning.

4. SON-BASED SIMILARITY SEARCH

In this section, the process of similarity search is described in detail, first at peer level (Section 4.1), then at super-peer level (Section 4.2), followed by a query routing technique (Section 4.3) that reduces the total cost.

A similarity query originates from a querying peer Q_P. We assume that we have a super-peer network, with each super-peer responsible for peers that belong to a SON. Our focus is on processing similarity queries q_k for the top-k most similar documents to the query. Such a query retrieves the k documents d_i ($1 \leq i \leq k$) from the network with highest cosine values $cos(q_k, d_i)$. In our context, q_k represents a non-trivial multi-keyword query, such as another document.

4.1 Peer Query Processing

During the pre-processing phase, each peer has performed feature extraction on its local documents and has performed local clustering. Thus each document is represented by a feature vector, and the same holds for local clusters of documents. A similarity query q_k is forwarded to a peer by its super-peer. Then the peer needs to find the k most similar documents to the query. In order to avoid computing the query similarity with each local document ($cos(q_k, d_i)$), a similarity threshold T_s is employed, so that only those documents that belong to local clusters c_j with $cos(q_k, c_j) \geq T_s$ may belong to the result list. These documents are ranked according to their similarity to the query and only the top-k documents are returned to its super-peer.

4.2 Super-peer Query Processing

A query initiated by a peer P_Q eventually reaches a super-peer (i.e., SON). The super-peer first determines the subset of its peers that can provide results relevant to the query. Then the query is forwarded to these peers. Each peer will process the query locally and if it has matching results these will be returned to the super-

peer. These results must be merged into a top-k list of documents that will be the result of this super-peer.

In order to support this set of operations, a super-peer must perform two main tasks: *peer selection* and *result merging*. As already mentioned, a super-peer maintains an index over its peers' features. Essentially the super-peer has enough information to decide for each peer whether it contains relevant documents to the query. Therefore, when the query is processed by the super-peer, it is forwarded only to a selected set of peers that may contribute results to the final result set. Then each peer returns a list (of maximum length k) of documents and their scores respectively. The super-peer merges this information into a final top-k list of documents with highest scores with respect to the query. This top-k list is the final result of the particular SON for the given query.

4.3 SON-based Query Routing

A similarity query originates from a querying peer Q_P. In an unstructured P2P system, querying is performed by routing the query to appropriate peers and performing the query on each of these peers, and then returning matching results. In our context, processing the query is performed by first determining which SONs (or super-peers) may contain relevant data, followed by searching one or more of these SONs, as described above.

SON-based query routing refers to query routing at SON level, which aims to identify similar SONs to the query. In order to limit the number of SONs that will be searched, SON-based routing is performed in two steps.

In the *first step*, a search for appropriate SONs is performed. A number of techniques can be used to find these SONs. Potential solutions include *routing indices* or some *gossiping approach* [25] that allow peers to become familiar with a small set of peers outside their cluster. However, such techniques and their variants impose a query horizon, i.e., they cannot guarantee that remote peers will be reached in very large networks. In order to overcome such limitations, we use the super-peer network to route queries. In the absence of more sophisticated mechanisms (such as hypercubes [16]), this routing can take the form of flooding. In this way, we can guarantee that the query will contact all SONs (or at least one of their super-peers), thus enabling access even to the most distant peers. Those super-peers that are sufficiently similar to the query, return their CD to the querying peer Q_P.

In the *second step*, Q_P determines the top-N most similar SONs (based on the results of step 1), and forwards the query to these super-peers for intra-SON query processing. The number of SONs to search is determined by the number of results returned (as for example for k-nearest neighbor queries), so that the number of SONs searched can be limited. Essentially, N is used to restrict the searching cost that would be induced by having to contact all SONs that were returned from the first step. In the experimental part, we demonstrate that even a small number of the most relevant SONs to the query is enough to return results of high quality.

5. EXPERIMENTAL RESULTS

In order to demonstrate the feasibility and efficiency of our approach, we have developed a simulation environment covering all the intermediate phases of SON generation (Section 3) and searching (Section 4). The simulator is developed in Java and all experiments were performed on Pentium IV commodity computers. The cost of constructing the semantic overlay hierarchy is comparable to DESENT construction, which was analyzed and shown to be acceptable in [6], and will due to space constraints not be discussed further in this paper.

The initial P2P network topology used in the experiments consists of N_P interconnected peers. We used the GT-ITM topology generator[2] to create well-connected random graphs of peers with a user-specified average connectivity. We used two network setups, 1000 (*small*) and 5000 (*large*) peers, to study the scalability of our approach with network size. Different topology types, such as power-law topologies, give similar results, due to the fact that the underlying topology only affects the zone creation phase.

As for the peers' content, we used two document collections in our experiments: 1) a subset of GOV2[3] consisting of 1,000,000 randomly selected documents, and 2) Reuters Corpus[4] (810,000 articles of news stories). The conclusions that can be drawn from experimental results from using the sets are mostly similar, so due to space constraints we will in this paper only present results from GOV2 except when there is a siginificant difference. Unless stated explicitly, results discussed are from using GOV2.

The experimental methodology consists of the following steps: 1) distribution of documents to peers, 2) local feature extraction and clustering on each peer, 3) SON creation, as outlined in Section 3.2, 4) adaptive clustering, and 5) query processing for P2P similarity search.

Regarding document distribution to peers we performed the distribution according to document's URL, so each peer is assigned documents from a small number of web sites. We performed local feature extraction using the MC toolkit, while for local clustering Gmeans is used [5].

We generated a query workload by randomly choosing 100 documents from each collection under concern. These documents are equivalent to queries of the form: *"given document X, find the top-k similar documents from the collection"*. We capitalize on the cosine similarity measure and we assume a similarity threshold T_s to determine which documents are considered similar to the query.

In the absence of relevance judgments for such queries, we need to determine the results returned by a centralized similarity search mechanism that employs the same distance function as in our approach. Thus we first evaluate the queries in a centralized setting. These results are considered the *correct* results for each query and they are used as a baseline. Ideally, the aim for the SON-based approach is to achieve exactly the results retrieved by a centralized mechanism.

We assess the clustering quality (in terms of clusters' separation) by computing the average pairwise cluster similarity, i.e. for each pair of clusters compute the similarity and take the average of all values. Low values indicate that clusters are well-separated. As regards retrieval quality, we measure:

- Recall, i.e., the fraction of the documents that are relevant to the query that are successfully retrieved.
- Recall@K. In our context, *Recall@K* is the fraction of the total K most relevant documents that a user found after scanning the best K retrieved documents.
- Precision@K. Precision is the fraction of the documents retrieved that are relevant to the user's information need. In our context, *Precision@K* is computed considering only of the best K retrieved documents.

Obviously, searching in all SONs would give perfect recall. However, this would be very costly, so we only consider the N most

[2] http://www.cc.gatech.edu/projects/gtitm/
[3] http://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm
[4] http://trec.nist.gov/data/reuters/reuters.html