Two-layered architecture for peer-to-peer concept search

Janakiram Dharanipragada
Indian Institute of Technology, Madras
djram@iitm.ac.in

Fausto Giunchiglia
University of Trento, Italy
fausto@disi.unitn.it

Harisankar Haridas
Indian Institute of Technology, Madras
harisankarh@gmail.com

Uladzimir Kharkevich *
University of Trento, Italy
kharkevi@disi.unitn.it

ABSTRACT

The current search landscape consists of a small number of centralized search engines posing serious issues including centralized control, resource scalability, power consumption and inability to handle long tail of user interests. Since, the major search engines use syntactic search techniques, the quality of search results are also low, as the meanings of words are not considered effectively. A collaboratively managed peer-to-peer semantic search engine realized using the edge nodes of the internet could address most of the issues mentioned. We identify the issues related to knowledge management, word-to-concept mapping and efficiency in realizing a peer-to-peer concept search engine, which extends syntactic search with background knowledge of peers and searches based on concepts rather than words. We propose a two-layered architecture for peer-to-peer concept search to address the identified issues. In the two-layered approach, peers are organized into communities and background knowledge and document index are maintained at two levels. Universal knowledge is used to identify the appropriate communities for a query and search within the communities proceed based on the background knowledge developed independently by the communities. We developed proof-of-concept implementations of peer-to-peer syntactic search, straightforward single-layered and the proposed two-layered peer-to-peer concept search approaches. Our evaluation concludes that the proposed two-layered approach improves the quality and network efficiency substantially compared to a straightforward single-layered approach.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval

*The authors are listed in alphabetical order of last name.

Semsearch '11 co-located with WWW'11, Hyderabad, India

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

peer-to-peer, semantic, search

1. INTRODUCTION

Global scale search has become a vital infrastructure for the society in the current information age. But, the current search landscape is almost fully controlled by a small number of centralized search engines. Centralized search engines crawl the content available in the world wide web, index and maintain it in centralized data-centers. The user queries are directed towards the data-centers where the queries are processed internally and results are provided to the users. The functioning of the search engines is controlled by individual companies. This situation raises a lot of serious issues which need to be studied and addressed carefully, given the importance of search as a vital service.

The individual companies which control the centralized search engines decide upon what to(not to) index, rank etc. Moreover, the complete details of the ranking and preprocessing methods are not made available publicly by most of the search engines. This leads to political and security concerns. With the increase in the information needs of the global community, the demands on the search engine infrastructure have increased manifold over the years. Since, search involves a large amount of computational, storage and bandwidth resources, the cost of acquiring and maintaining resources are very high. Recently, the energy consumption of search engines is causing environmental concerns. Since, the search service is provided using a small number of datacenters, the chance of the service being completely down/inaccessible is also high.

The syntactic search techniques used in major search engines consider words and multi-word phrases in the documents and queries to determine relevance. But, this leads to low quality of search results due to the ambiguity of natural language. For example, "bank" could mean "depository financial institution" or "slope bordering a water body". Semantic search techniques like Concept search[8] address this problem by incorporating knowledge bases to identify concepts (eg: concept "bank-2" means "slope bordering a water body") and relationships between concepts(eg: poodle-1 ISA

dog-2). Thus, search can be performed based on the concepts instead of words thereby improving quality of search results. However, realizing such semantic search techniques over a centralized search engine is difficult because of multiple reasons. These techniques require availability of human generated knowledge bases which cater to the interests of the users. But, it is observed that the users of search engines exhibit a "long tail of interests", i.e., a diverse range of numerous niche interests. It will be extremely tough and expensive, if not impossible for a centrally managed search engine to develop and maintain rich knowledge bases addressing the long tail of user interests. Another difficulty is that the semantic search techniques are often more computationally expensive compared to syntactic search techniques making them difficult to be offered at global scale.

A peer-to-peer semantic search engine could address most of the issues raised above related to centralized search engines. A peer-to-peer semantic search engine uses the free shared resources available in the edge nodes(peers) of the internet to realize a distributed semantic search using the background knowledge of the peers. In this paper, we investigate the applicability of a peer-to-peer semantic search engine in addressing the issues related to global scale search. The new issues which need to be addressed while realizing a peer-to-peer semantic search engine are discussed. We show the importance of a multi-layered approach in realizing peer-to-peer semantic search. Experiments show the advantage of a two-layered approach to peer-to-peer concept search over a straightforward single-layered approach.

We explain the motivation and details of the architecture in Section 2. Sections 3 and 4 describe the design choices for the current implementation and the implementation details respectively. We present the details of the evaluation performed and the results in Section 5. We review the related work in Section 6 and state our conclusions in Section 7.

2. ARCHITECTURE

In a peer-to-peer semantic search engine, the edge nodes of the internet(peers) participate in providing the semantic search service in addition to being consumers of the service. The peers collaboratively maintain the background knowledge base required for realizing semantic search. The open and shared nature of the infrastructure and peer-to-peer community will address the centralized control and transparency issues inherent in centralized search engines. Since, the peers participate in providing the search service in a voluntary fashion, the overhead and cost involved in acquiring and maintaining dedicated resources in centralized data centers are avoided. As the idle computational power available in non-dedicated edge nodes is used in providing the search service, the power usage could be less compared to a centralized approach. Since, the background knowledge of the peers taking part in search is used to enhance search quality using semantic search, the difficulty of centralized search engines in addressing the long tail of niche user interests is absent in a peer-to-peer setting. But, the realization of a peer-to-peer semantic search scheme is not straightforward and involves addressing various associated issues as discussed below.

2.1 Issues with a single-layered approach

A straightforward way of realizing a peer-to-peer semantic search system will be to reuse the substantial existing work in the area of peer-to-peer syntactic search and extend it to semantic search. The index can be partitioned among the participating peers and search can be done in a distributed manner. The background knowledge of the peers can also be partitioned and managed among the peers supporting collaborative creation and maintenance of common knowledge. Though, this straightforward approach (henceforth called single-layered approach) addresses the problems mentioned earlier, it has the following issues which need to be addressed.

Knowledge management at large scale: The common knowledge which every one agrees on is minimal as it is difficult to have consensus at a large scale. This is because of the large amount of time and effort required to get a wide understanding and meet requirements in a large diverse group as explained in [3]. Hence, a commonly agreed monolithic knowledge base will be insufficient for realizing semantic search.

Difficulty in identifying concepts from words: A major issue in semantic information retrieval approaches like Concept search is that, given limited context, it is tough to identify the concept from the words in queries and documents(word sense disambiguation problem). For example, from a query "bank" without any context, it is difficult to infer the intended meaning. Statistical techniques for word-sense disambiguation(WSD) also suffer from low accuracy.

Network bandwidth usage of peer-to-peer search techniques: Although there has been substantial amount of work done in the area of peer-to-peer (syntactic) search, the network bandwidth consumption of these techniques remain high due to the distributed nature of the index. This affects the applicability of the approach at internet scale.

We propose a two-layered architecture for peer-to-peer concept search to address the above mentioned issues related to single-layered peer-to-peer concept search.

2.2 Two-layered architecture for peer-to-peer concept search

In the two-layered architecture for peer-to-peer concept search, the peers are organized as communities based on common interests. Each community maintains its own background knowledge base(DBK) pertaining to its specific interests distributed among the peers of the community. The DBK of a community is created and collaboratively managed by the peers in the community. Thus the long tail of peer interests can be supported in the community-based knowledge management scheme. Since, the DBKs in different communities evolve independently based on the specific interests of the smaller set of peers within the communities, the "consensus problem of monolithic knowledge base" present in the single-layered approach is absent.

The word-to-concept mapping of documents and queries within the community is performed based on the DBK of the community. This allows to perform better word-to-concept mapping within the community. For example, within a "Finance" community, "bank" could mean "depository financial institution" and related concepts almost surely. Performance of WSD can be significantly improved if it is performed within a specific branch of knowledge as explained in [4].

The index of the documents shared by the peers in a community is distributed among the peers within the community and the existing peer-to-peer syntactic search approaches are extended and reused for performing distributed search. Since, the indexes are distributed and searched only within

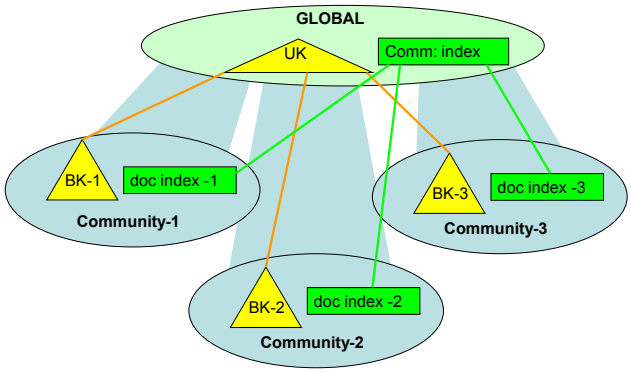

Figure 1: Two-layered architecture for peer-to-peer concept search

scope of various communities instead of the entire network, the bandwidth consumption overhead associated with the existing techniques is substantially reduced.

Maintaining background knowledge bases and document indexes independently in different communities leads to the issue of performing search across communities. If the DBKs are created and maintained independently by the different communities, performing semantic-enabled search for communities will not be possible as there is no commonly agreed knowledge base to represent and search for communities. Hence, we introduce a top layer consisting of a universal knowledge base(UK) and the global community index into the architecture forming a two-layered architecture. UK consists of minimal knowledge which is accepted globally in the network. The different DBKs are initially bootstrapped from the UK and developed independently by the different communities. Search for communities is enabled using the global community index which includes the representations of all the communities based on the UK. The community index consists of aggregate information from the document indexes from all the communities. The upper layer is maintained in a distributed manner across all communities.

The two-layered architecture is illustrated in figure 1. The lower(community) layer supports search within community, i.e., given a query and community, it searches the document index within the community using the DBK of the community and returns potentially relevant documents within the community. The upper(global) layer supports search for communities, i.e., given a query, it searches the global community index based on the UK and return the potential communities having the relevant documents to the query. Using the above two search methods, the users have the following options for search.

Option 1: The user can select a previously known community and scope the search within the community.

Option 2: The user can first query and obtain the list of communities relevant to the query. Then, the user can select the appropriate communities manually and search within them as in *Option 1*.

Option 3: The user wants to perform a search over the entire network without further intervention. The ranked list of communities relevant to the query are obtained and search is performed automatically on the top-ranked communities in parallel. The individual results are combined and ranked, and only the final results are presented to the user.

3. DESIGN CHOICES IN THE IMPLEMENTATION OF THE ARCHITECTURE

To implement the two-layered architecture, design choices need to made on the overlay network to use(interconnection among peers), index partitioning and search scheme, and knowledge management scheme. We developed proof of concept implementations of both single-layered and two-layered approaches using similar design choices(details described below). This allowed a fair comparison of the two approaches under similar settings. But, note that further improvements can be made in the design choices and implementation (mentioned at appropriate places) to improve the performance of both the approaches.

Index partitioning and search: Index partitioning is used to distribute the search over parts of the index present in different peers. The two major index partitioning schemes are *partition-by-document* and *partition-by-term*. In *partition-by-document*, the entire set of documents are divided into different smaller collections. All the smaller sub-collections are indexed separately and stored on separate peers. The query needs to be sent to all the sub-collection indexes in different peers to obtain complete search results. This leads to high network bandwidth consumption if the number of peers are large.

In *partition-by-term*, the portion of the index associated with a term, ie, the list of documents having the particular term(posting list) in the whole collection is stored on a particular peer. Thus, the index is partitioned term-wise and stored on different peers. The query is forwarded to the peers responsible for the query terms to obtain the results. But, for multi-term queries, posting list intersection over the network is required which result in high network load. Several optimizations(see Section 6) address this issue and is still an active area of research. We use the *partition-by-term* scheme which is popular among the research community in our current implementation of both single-layered and two-layered approaches.

Overlay network: To support a *partition-by-term* scheme, each term needs to be assigned to a unique peer in the network. This assignment has to be realized in a distributed manner and should handle node join/leave. Structured overlays like Distributed Hash Tables(DHTs) [19, 16] serve this purpose. DHTs realize a distributed hash table-like functionality by assigning the responsibility of different keys in the hash table to different peers and handle dynamic join and leave. Each peer maintains a routing table consisting of $O(logN)$ entries where N is the total number of peers. The key lookups(reaching the peer responsible for the key) are performed within $O(logN)$ hops with probabilistic guarantee. To maintain the document indexes of different communities separately and enable search across communities in top layer, we use a two-layered overlay inspired from our earlier works on computational grid(Vishwa[15]) and data grid (Virat[18]). In the two-layered overlay, different communities form separate overlays in the lower layer supporting indexing, search and knowledge management within the communities. To ensure management of global community index, search for community and UK management, the different community-level overlays are connected together forming a top-layer consisting of peers from all communities. The overlay supports key-lookup operations at both layers.

$putC(c, key, value)$ stores the $value$ associated with key

in the node responsible in community c.

$getC(c, key) \rightarrow value$ retrieves the value associated with key from community c.

$putG(key, value)$ stores the $value$ associated with key in the node responsible in the global layer.

$getG(key) \rightarrow value$ retrieves the value associated with key from the global layer.

Even though, the neighbor-maintenance and lookups are supported at two levels, the routing table size and lookup hops are still $O(logN)$. More details of the overlay used are present in the evaluation section(Section 5). Further details are available in [15, 18].

Knowledge management: The DBK of each community is maintained across the different peers which form part of the community. Each word or concept in the DBK is assigned to a peer in the community using the DHT get/put operations. The details are available in Section 4.2.4. The UK is also maintained in a similar manner in the top layer.

4. IMPLEMENTATION DETAILS

4.1 Community management

The communities are managed in a similar manner as public communities in flickr, facebook etc. A peer wanting to start a community provides a name and description and creates the community background knowledge by bootstrapping the relevant portions from the UK and extending it. The other peers can join the community initially by contacting the creator peer and later on by contacting any other peer within the community.

During the creation of the community, a new community overlay is formed and the community overlay gets connected to the global layer through an inter-community bootstrap process. A unique community identifier is generated based on the community name. The newly joining peers form part of the community overlay as well as the global layer. A peer joining the system participates in the overlay of one of the communities in which it belongs. When the peer wants to join more communities, it sends a subscription message to one peer each from the desired communities. The other peers act as proxies for the peer in the other communities and perform operations like indexing, search etc. on behalf of the peer. The proxy peer is dynamically chosen for each access and can be identified through routing in the upper layer. Henceforth, peer in a community means either a peer participating in the community overlay or a proxy peer in the community overlay acting on behalf of another peer.

4.2 Search within community

We extend our earlier work on (centralized)Concept search[8] to realize the underlying semantic search scheme within communities. Concept search extends syntactic search technique with available knowledge to realize semantic search. Since, Concept search falls back to syntactic search when no knowledge is available, it does not suffer from the difficulty faced by purely knowledge-based systems due to lack of initial knowledge during the starting phase(knowledge acquisition bottleneck problem). The essential details of the Concept search scheme used are described first, in Sections 4.2.1 to 4.2.3. The details related to realizing peer-to-peer concept search over the peers within the communities are described next, in Sections 4.2.4 to 4.2.6[1]. Further details are available in the technical report [6].

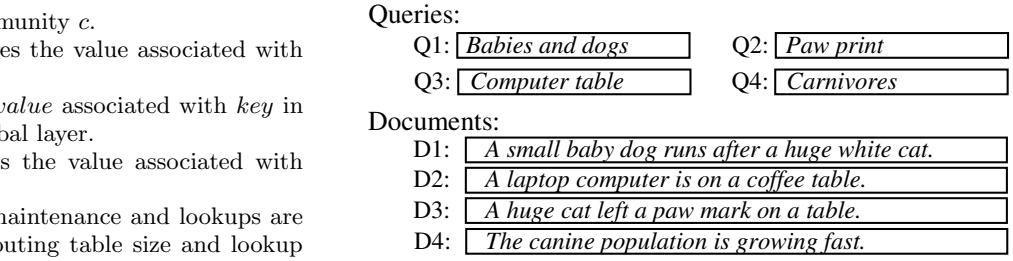

Figure 2: Queries and a document collection

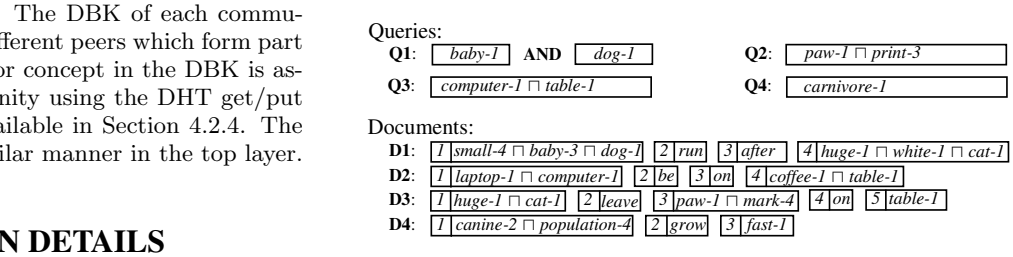

Figure 3: Document and Query Representations

4.2.1 Representation of knowledge

The knowledge base consists of description of concepts, relationships between concepts and word-to-concept mapping. For the sake of presentation, an atomic concept is represented as *lemma-sn* having the lemma($lemma$) of a word having the meaning of the atomic concept and its sense number(sn) within the knowledge base. For example, concept *bank-1* could refer to "depository financial institution" while *bank-2* refers to "sloping land near a water body". Description of an atomic concept also consists of its part of speech(POS) and natural language description(GLOSS). The relationships between concepts are represented through subsumption relations of the form $A_1 \sqsubseteq A_2$ where A_1 and A_2 are atomic concepts(For example, *dog-1* $\sqsubseteq$ *canine-2*). The concept relations form an acyclic graph with vertices representing concepts and edges representing subsumption relations. Word-to-concept mapping consists of the list of atomic concepts associated with each word.

4.2.2 Document and query representations

The concepts in documents and queries are extracted by analyzing the meanings of words and natural language phrases present. Single words are converted into atomic concepts using the knowledge base. If no relevant concepts are found for a word, then the word itself is used as a concept. If there are many possible candidate concepts for a word and the meaning of the word cannot be reliably disambiguated, then disjunction of at most three most probable concepts for the word are used. Noun phrases are identified and translated into complex concepts which are logical conjunctions of all the atomic concepts corresponding to the words. For example, the noun phrase "A little dog" is translated into the complex concept *little*-4⊓*dog*-1. After extracting the complex concepts, documents and queries are represented

[1]A preliminary version of the single-layered approach was presented in [7].

as enumerated sequences of these complex concepts as illustrated in figures 2 and 3. Figure 2 shows the original document and query collection. Figure 3 shows the resulting representations after complex concept extraction(examples from [8]). Rectangles in Figure 3 represent individual complex concepts and the number in the left square of each rectangle represents the position of the associated complex concept in the document.

4.2.3 Relevance model

The relationship between complex concepts(denoted by C^x) is defined based on the relationship between the constituent atomic concepts(denoted by A^x) in the knowledge base(KB) as follows.

$$KB \models C^2 \sqsubseteq C^1 \iff \forall A^1 \in C^1, \exists A^2 \in C^2, \text{s.t. } KB \models A^2 \sqsubseteq A^1 \quad (1)$$

A document is relevant to a query if the document contains concepts which are specific than the query concepts according to knowledge base. The potentially relevant documents for a single complex query concept C^q in a document collection D, based on knowledge base KB is defined by

$$QA_{D,KB}(C^q) = \{d \in D \mid \exists C^d \in d, \text{s.t. } KB \models C^d \sqsubseteq C^q\} \quad (2)$$

When multiple complex query concepts are present in the query(e.g.,*baby-1* AND *dog-1*), the potentially relevant documents consist of the documents which are present in the query answer of all the individual complex query concepts as shown below.

$$QA_{D,KB}(Q) = \bigcap_{C^q \in Q} QA_{D,KB}(C^q) \quad (3)$$

The degree of relevance of documents(for ranking of results) is computed by adopting the cosine similarity from the vector space model for syntactic search. Atomic concepts in a query are weighted by the *tf-idf* weight measure. The only difference is that, frequency $tf'(A^q, d)$ of atomic concept A^q in document d is computed by taking into account the estimated *word-concept* and *concept-concept* similarities also.

$$tf'(A^q,d) = \sum_{A^d \sqsubseteq A^q} \frac{1}{10^{d(A^q,A^d)}} \cdot \frac{f(A^q,w^q)}{maxF(w^q)} \cdot \frac{f(A^d,w^d)}{maxF(w^d)} tf(A^d,d) \quad (4)$$

where w^q (w^d) is a word in the query (in the document) from which the atomic concept A^q (A^d) is extracted, $d(A^q, A^d)$ is the distance between concepts A^q and A^d in the concept hierarchy from KB, $f(A, w)$ is a value which shows how frequently the specified word w is used to represent the meaning A (incremented by one in order to avoid zero values), $maxF(w)$ is the maximum $f(A, w)$ for word w, and $tf(A^d, d)$ is the frequency of A^d in d. Informally, higher ranks are assigned for the following: (i) concepts A^d which are closer in the meaning to the concept A^q, (ii) concepts A^q which are frequent for word w^q, and (iii) concepts A^d which are frequent for word w^d.

4.2.4 Distributed maintenance of background knowledge

The background knowledge of each community is maintained in the DHT within the community. Each atomic concept is uniquely identified by a unique concept ID (A_{ID}) composed of peer ID of the peer which created the atomic concept and its local concept ID in the knowledge base of the peer. The concept ID(A_{ID}) is hashed to generate unique keys for each concept. The generated key is used to assign the responsibility of maintaining each concept to a different node in the community using the *putC* operation.

The peer responsible for an atomic concept maintains the associated POS and $GLOSS$. In addition, it stores IDs of the immediate specific concepts and IDs of the general concepts which are related to the concept. To handle relations, we can use a modified *putC* operation, $putC(c, A, B, Rel)$ which stores the ID of atomic concept B with relation Rel on the peer responsible for atomic concept A in community c, e.g., *putC(c,canine-2, dog-1, '$\sqsubseteq$')*, *putC(c,canine-2, carnivore-1, '$\sqsupseteq$')*. The list of all atomic concepts(IDs) related to each word(word-to-concept mapping) is maintained by hashing each word to obtain a unique key and maintaining the list in the node responsible for the key in the community DHT. This can be implemented using the *putC* operation, e.g., $putC(c, canine, \{canine\text{-}1,\ canine\text{-}2\})$. Similar modifications of the *getC* operation are used to access the DBK to obtain the list of all concepts related to a word and the description and more specific/general concepts of a particular concept.

4.2.5 Document indexing

After document representations are computed as in Figure 3, the indexing of documents within community is performed as follows. Every peer computes the set of atomic concepts which appear in the representations of its document collection. For every atomic concept A, the peer computes the set of documents which contain A. For every concept-document pair $\langle A, d\rangle$, the peer computes the set $S(d, A)$ of all the complex document concepts C^d in d, which contain A.

$$S(d, A) = \{C^d \in d \mid A \in C^d\} \quad (5)$$

For example, if d is document $D1$ in Figure 3 and A is *dog-1*, then $S(d, A) = \{small\text{-}4 \sqcap baby\text{-}3 \sqcap dog\text{-}1\}$. For every A, the peer sends document summaries corresponding to A, i.e., pairs $\langle d, S(d, A)\rangle$, to the peer p_A responsible for A in community DHT using the *putC* operation. The peer p_A indexes these summaries using a local positional inverted index[8]. The inverted index dictionary consists of atomic concepts from DBK (e.g., concepts *baby*-3 and *dog*-1). The posting list $P(A) = [\langle d, freq, [position]\rangle]$ for an atomic concept A stores the postings of complex concepts which contain A, where $\langle d,\ freq,\ [position]\rangle$ is a posting consisting of a document d associated with A, the frequency $freq$ of A in d, and a list $[position]$ of positions of A in d. For example, *P(dog-1)*=$\langle D1, 1, [1]\rangle$.

Overall, peers maintain the following information of the words and concepts they are responsible for:

1. For every word, the peer stores the set of atomic concepts (word senses) associated with the word.

2. For every atomic concept, the peer stores the description and a set of more specific and general concepts.

3. Document summaries $\langle d, S(d, A)\rangle$ for all the atomic concepts A (for which the peer is responsible) are stored on the peer and indexed in the local inverted index.

An example of the information, which can be stored on the peer responsible for a single word *bank* and for a single atomic concept *canine-2*, is shown in Figure 4.

Word senses	
bank	*bank-1, bank-2*
Concept descriptions	
canine-2	AID-192.., noun, {A mammal with..}
More specific concepts	
canine-2	*dog-1, wolf-1*
More general concepts	
canine-2	*carnivore-1*
Inverted index	
canine-2	$\langle D4, 1, [1]\rangle$
population-4	$\langle D4, 1, [1]\rangle$

Figure 4: Peer's information

4.2.6 Distributed search

For performing search within a community with document collection D and background knowledge DBK, the query answer for a complex query concept ($QA_{D,DBK}(C^q)$) is computed using the distributed document index as follows. Consider a subset $QA_{D,DBK}(C^q, A)$ of the query answer $QA_{D,DBK}(C^q)$. $QA_{D,DBK}(C^q, A)$ consists of documents which contain at least one concept C^d which is more specific than the complex query concept C^q and contains atomic concept A.

$$QA_{D,DBK}(C^q, A) = \{d \in D | \exists C^d \in d, \text{s.t. } DBK \models C^d \sqsubseteq C^q \text{ and } A \in C^d\} \quad (6)$$

If we denote the set of all atomic concepts A^d, which are equivalent to or more specific than concept A by $C(A)$, i.e.,

$$C(A) = \{A^d \mid DBK \models A^d \sqsubseteq A\} \quad (7)$$

then, it can be shown that, given Equation 6, the query answer $QA_{D,DBK}(C^q)$ can be computed as follows

$$QA_{D,DBK}(C^q) = \bigcup_{A \in C(A^*)} QA_{D,DBK}(C^q, A) \quad (8)$$

where A^* is an arbitrarily chosen atomic concept $A^q \in C^q$. For each A in $C(A^*)$, the query answer $QA_{D,DBK}(C^q, A)$ can be computed using the local inverted index(partition of the index) present at the node responsible for A. Thus, the final result can be obtained by sending the query to the nodes responsible for each A in $C(A^*)$ and combining the results obtained from all the nodes. But, each node stores the ID of only the immediate more specific concepts and hence multiple lookups will be required for finding $C(A^*)$. Hence, query is first sent to nodes responsible for immediate more specific concepts of A^*. The nodes responsible for the concepts return their own immediate more specific concepts also along with the partial query result. The query is then propagated to the newly obtained specific concepts and the process is continued in a recursive fashion. We choose the concept with the least number of more specific concepts from C^q as A^*, as it will minimize the number of iterations. The detailed algorithm where a peer p_I initiates a query C^q to obtain results QA is shown below.

Step 1 A peer p_I initiates the query process for complex query concept C^q.

Step 2 p_I selects A in C^q with the smallest number of more specific atomic concepts. C^q is propagated to the peer p_A responsible for A. On p_A, QA and auxiliary sets $\mathbf{C}_{ms}$ and $\mathbf{C}'_{ms}$ are initialized to $\emptyset$. A is added to $\mathbf{C}_{ms}$[2].

Step 3 p_A selects an atomic concept B from $\mathbf{C}_{ms}$ and repeats steps 4 and 5.

Step 4 p_A submits C^q to the peer p_B responsible for B. p_B receives the query concept C^q and locally (by using inverted index) computes the set $QA_{D,DBK}(C^q,B)$ as described in [8]. The results are sent back to p_A. Note that if B=A, then the query propagation is not needed. On receiving new results $QA_{D,DBK}(C^q,B)$, p_A merges them with QA.

Step 5 p_B also computes the set of atomic concepts which are more specific than B by querying locally stored (immediate) more specific concepts (e.g., see 'More specific concepts' in Figure 4). The results are also propagated to peer p_A where they are added to set $\mathbf{C}'_{ms}$. If B=A, then the set of more specific concepts are computed on p_A itself.

Step 6 If $\mathbf{C}'_{ms} \neq \emptyset$, then p_A assigns $\mathbf{C}_{ms} = \mathbf{C}'_{ms}$, $\mathbf{C}'_{ms} = \emptyset$ and repeats step 3. Otherwise the results are sent to the initiator peer p_I.

In order to optimize query answering, each peer p_A precomputes (and regularly updates) addresses of peers p_B which are responsible for immediate more specific concepts, and uses DHT to locate such peers only when pre-computed information is outdated. Steps 4 and 5 are done in parallel for all the concepts in $\mathbf{C}_{ms}$. We allow user to specify the maximum number of more specific concepts which can be used per atomic concept in C^q. In the current implementation, we use upto 10 more specific concepts per query concept.

An example showing how the query answer is computed is given in Figure 5. Peers, represented as small circles, are organized in the community DHT, represented as a circle. A query consisting of a single query concept C^q = *little*-4 $\sqcap$ *canine*-2 is submitted to peer P_I. Assume that the atomic concept, *canine*-2 has smaller number of more specific atomic concepts than concept *little*-4. In this case, C^q is propagated to a peer $P_{canine\text{-}2}$, i.e., the peer responsible for atomic concept *canine*-2. The query propagation is shown as a firm line in Figure 5. $P_{canine\text{-}2}$ searches in its local inverted index with C^q. No results are found in this case. $P_{canine\text{-}2}$ collects all the atomic concepts which are more specific than *canine*-2, i.e., atomic concepts *dog*-1 and $wolf$-1. Query concept C^q is propagated to peers $P_{dog\text{-}1}$ and $P_{wolf\text{-}1}$. $P_{wolf\text{-}1}$ finds no results while $P_{dog\text{-}1}$ finds document D_1. D_1 is an answer because it contains concept *small*-4 $\sqcap$ *baby*-3 $\sqcap$ *dog*-1 which is more specific than *little*-4 $\sqcap$ *canine*-2. D_1 is sent to P_A. The propagation of the results is shown as a dashed line in Figure 5. Both peers $P_{dog\text{-}1}$ and $P_{wolf\text{-}1}$ have no more specific concepts than *dog*-1 and $wolf$-1. Therefore C^q is not propagated to any other peers. P_A sends the final result, i.e. D_1, to peer P_I.

When the query contains multiple complex concepts, the query answer from all the complex concepts need to be intersected. In this case, first one concept is chosen from each

[2]If a word w in the query is not disambiguated to a single concept A, then the query is propagated to the peer p_w responsible for w and all non-disambiguated senses of w are added to $\mathbf{C}_{ms}$.

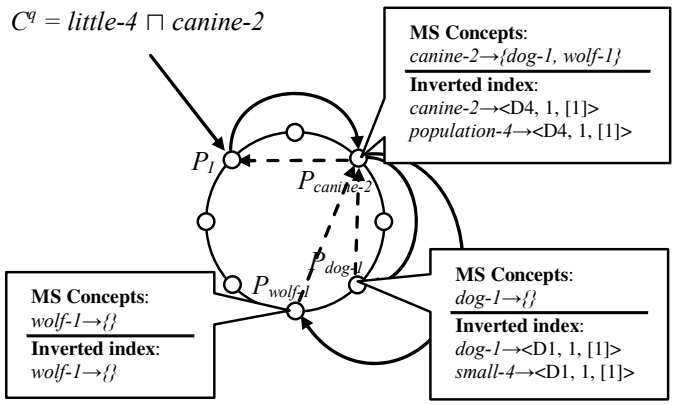

Figure 5: Query Answering

of the complex concepts in the query(as in step 2 in the algorithm). The responsible peers for each of these concepts and the corresponding lengths of posting lists are obtained from the DHT by issuing *getC* operations in parallel. The peers are traversed in the increasing order of length of posting lists in an attempt to reduce the network cost due to intersection. This *peer order* is encoded in the query message. The query is then sent to responsible peers in *peer order*. Within each peer, the search is performed and partial results are computed as mentioned in the previous algorithm. The resulting document list is intersected with the results obtained from the previous peer in *peer order*. In the Step 4 of the query-answer algorithm, if the results from the previous peer are smaller in size than the new results which need to be transferred from peer p_B to p_A, the previous results are sent to the peer p_B to filter the new results before they are transferred. The intermediate results obtained at each stage are sent to the next peer in *peer order*. The final results are ranked and transferred to the peer who issued the query, by the last node in the *peer order*. For an efficient implementation of the intersection, various optimization techniques available for peer-to-peer syntactic search can be reused.

4.3 Global search

Given a query, the search for communities involves returning a list of communities, such that, there is a high chance that the relevant documents will be found in these communities. The search for communities is performed in the upper layer based on the aggregate community index and UK. The aggregate community index is maintained as follows. In each community, when the posting list for an atomic concept A with id AID exceeds a specified threshold, the peer responsible for the concept, sends an "index community with concept A_{UK}" message to the overlay through $putG(A_{UK}, AID)$, where A_{UK} is the most specific subsumer in UK for concept A. The document frequency $df(A, c)$ for concept A in community c is also stored.

The current implementation uses a simple popularity-based method to search for communities. The potentially relevant communities are retrieved from the global layer in the same manner as the documents are indexed and retrieved inside each of the communities with some differences. One difference is that UK is used instead of BK of the community. The concepts from UK (and also words, if the corresponding concepts are not found in UK) are used for indexing the community in the global layer as explained before. Second difference is that, in the scoring function, the term frequency $tf(A, d)$ is replaced by $df(A, c)$ and the document frequency is replaced by the community frequency. Third difference is that, the *getC* operations are replaced by *getG* operations.

To support search in the entire network without user intervention in community selection (*Option 3* in Section 2.2), a subset of the relevant communities is selected. Only those communities whose score $score(c_i)$ is greater than a predefined threshold t, (i.e., $score(c_i) > t * max_i(score(c_i))$) are selected for search. Query is then propagated in parallel to all selected communities and search is performed within the communities as in Section 4.2. The top-k results from the communities are then propagated back to the requesting peer and are merged into a single ranked result list. Scores($score(d, c_i)$) for documents d from each community are normalized by the community score $score(c_i)$ as below

$$score(d) = score(c_i) * score(d, c_i) \quad (9)$$

The search for relevant communities is similar to the resource selection problem in case of federated peer-to-peer search on digital libraries[10] and metasearch. The solutions in these areas can be studied and enhanced by taking advantage of the knowledge base(UK) to enhance the accuracy of community selection.

5. EVALUATION

Evaluation is done mainly to compare two-layered approach with single-layered one, keeping various design choices same(for fair comparison). In our opinion, the closest existing work to our approach is Semantic Overlay Network (SON)[5]. However, there are significant differences between SON and two-layered architecture as explained in Section 6 and hence, no experimental comparison was made with SON. But, the evaluation clearly establishes the advantages of two-layered approach over a single layer of knowledge.

We developed three prototypes with similar design choices: single-layered P2P Syntactic Search, single-layered P2P Concept Search(*P2PCS*) and two-layered *P2PCS* . P2P Syntactic Search is based on Lucene[3] and implements *partition-by-terms* scheme for syntactic search(see Section 3). Single-layered *P2PCS* extends P2P Syntactic Search with a single knowledge base and performs concept search in the scope of the entire network [4]. Two-layered *P2PCS* extends the single-layered *P2PCS* implementation to realize the proposed community-based two-layered search. Experiments with real implementation could not be performed on thousands of nodes due to physical limitations. Hence, to obtain the results on quality, a custom simulator was developed by reusing parts of real implementation for both single-layered and two-layered approaches. To obtain the results on network performance, the simulations were performed by using a peer-to-peer simulator PeerSim [9]. The simulation setting and evaluation parameters are described below.

Simulation setting: The single-layered approaches (syntactic and semantic) were simulated by modifying the Pastry(DHT) [16] implementation available with PeerSim. The

[3] `http://lucene.apache.org/java/docs/index.html`

[4] To validate that there are no optimizations which can affect the fair comparison between syntactic and semantic approaches, we compared the results of syntactic approach with the results of semantic approach when the background knowledge was empty. No difference in quality and performance was found.

parameters used for Pastry were: (base = $2^b = 16$, id length = 128 bits, leafset size = 32) with the theoretical lookup cost of $O(\log_{16} N)$ hops.

To implement the two-layered architecture, using the hierarchical DHT based scheme, the Pastry-based implementation for single-layer was reused to form communities in the community layer, combined with a Chord(DHT) [19] implementation developed by us to realize the global layer. To ensure fairness while comparing two-layered approach with single-layered one, the theoretical lookup cost in hops for both the approaches need to be same for the same number of nodes. Hence, the fingers of each Chord node were placed at distances that are all integral powers of $(1 + 1/d)$ instead of 2, where $d = 15$, so that the theoretical lookup cost of Chord becomes $O(\log_{1+d} N) = O(\log_{16} N)$ as mentioned in [19]. The Chord parameters were: (base = 2, id length = 10 bits), making the total id length of two-layered overlay 138 bits. The node ids were randomly generated and experiments were performed after all the nodes joined the overlay with no subsequent join/leave. Each experiment was performed five times (with different seeds for random number generator) and the results were averaged.

Evaluation parameters:

quality: For evaluating quality of results, we used standard IR measures: the mean average precision (*MAP*) and precision at K (*P@K*), where K was set to 10 and 20.

network bandwidth consumption: The average number of postings *ANP* (document id and additional related information, e.g., score of document) transferred per query was taken as the measurement of network bandwidth consumption as they form the majority of network traffic for search (DHT lookup cost is comparatively negligible). ANP is independent of different optimizations and representations used for posting list transfer.

response time: The query processing involves both sequential as well as parallel operations (e.g., sequential posting list transfer in syntactic search, parallel search across communities). Also, each operation can include numerous sub-operations. The effective delay of a set of sequential operations (i.e., operations which need to be performed sequentially) is the sum of the individual delays of the operations, while the effective delay of a set of parallel operations (i.e., operations which can be performed in parallel) is the maximum of the individual delays. This can be combined recursively for each of the sub-operations to obtain the total response time.

The delay of a network operation depends both on the network latency (determined by RTT and queuing delay) and the data transfer time (depends on available bandwidth). While comparing the different approaches, the latency and data transfer time will depend on the various optimizations and representations used as well as the network conditions like bandwidth and latency. To indicate the effective response time independent of optimizations and network conditions, we define two response time indicators: sequential hops (*s-hops*) and sequential posting transfer (*s-postings*). *s-hops* approximates the contribution to response time due to the network latency, which in turn is determined by the number of hops. *s-postings* approximates the contribution to response time due to the bandwidth constraints, which in turn is determined by the posting lists to be transferred (major contributor to bandwidth consumption). The two response time indicators are calculated in the same manner as total effective response time is calculated (shown below).

TREC8 (401-450)		
	P2P Syn. Search	P2PCS
MAP	0.1672	0.2012(+20.3%)
P@10	0.3900	0.4360(+11.8%)
P@20	0.3140	0.3510(+10.4%)
ANP	3574.78	31905.28
s-posting	3574.78	20894.60
s-hops	5.25	17.75

Figure 6: Evaluation results: Syntactic vs. Semantic

The *s-hops* (*s-postings*) of a set of sequential operations is the sum of the individual hops (posting transfers) involved in the operations, while the *s-hops* (*s-postings*) of a set of parallel operations is the maximum of the individual hops (posting transfers) involved. This can be combined recursively for each of the sub-operations to obtain the total *s-hops* (*s-postings*). These measures though not precise, still help in comparing response times of different approaches.

5.1 P2P Syntactic Search vs. P2PCS

In this section, we compare the performance of P2P Syntactic Search and *P2PCS* (with WordNet stored in *DBK*). As a data-set for this experiment, we used the TREC ad-hoc document collection[5] (disks 4 and 5 minus the Congressional Record documents) and the query set from TREC8 (topics 401-450). The title for each topic was used as a query. The data-set consists of 523,822 documents and 50 queries. The peer-to-peer network was simulated with 10,000 nodes and queries were issued from randomly chosen peers. The results of the evaluation are shown in Figure 6. The experiments show that, on TREC ad-hoc data set, the results achieved by *P2PCS* are better in quality than those achieved by P2P Syntactic Search. The achieved improvement is statistically significant according to Wilcoxon signed rank test at the level of 0.1. The network overhead of concept-based search (measured by ANP) is 9 times that of syntactic search. The response time indicators are also higher. Thus, though concept-based search provides higher quality of results compared to syntactic search, the network performance is lower. Also, more techniques need to be explored for peer-to-peer concept-based search other than the term-partitioned DHT-based scheme used. Next, we show how the performance and quality can be improved using a two-layered architecture.

5.2 P2PCS vs. Two-Layered P2PCS

In this section, we compare the performance of single-layered *P2PCS* and two-layered *P2PCS* approaches. The data-set for this experiment was automatically generated by using data from *WordNet Domains* [2], *YAGO* ontology [20], and *Open Directory Project*[6] also known as *DMoz*. The DMoz was chosen for generating document collections because it is a collaborative effort of more than 80,000 editors who work independently in different domains and, therefore, the resulting document collections approximate the documents in the real communities well.

First, we created 18 background knowledge bases BK by using the following top-level WordNet Domains: *agriculture*, *architecture*, *biology*, *chemistry*, *computer science*, *earth*, *engineering*, *food*, *history*, *law*, *literature*, *mathematics*, *medicine*, *music*, *physics*, *religion*, *sport*, *transport*. Universal knowl-

[5] `http://trec.nist.gov/data/test_coll.html`
[6] `http://www.dmoz.org/`

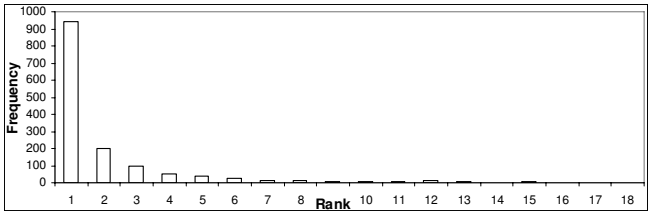

Figure 7: Ranking of communities

edge was created by using concepts and relations from WordNet which do not correspond to any domain (i.e., *factotum* in WordNet Domains). Knowledge from UK was added to every BK and then we enriched concepts in each BK with the more specific concepts (and corresponding relations) from YAGO ontology. Thus UK represents globally accepted knowledge while BK is specialized based on community interest.

Second, for each domain we selected a corresponding subtree in DMoz (e.g. *Top/Science/Agriculture* for *agriculture* domain, *Top/Society/History* for *history* domain). Documents classified to categories in the corresponding domain subtree were used as the document collection for the domain. The concatenation of DMoz description, title, and the first page of the corresponding web page was used as the document. The document collection size varies from 1387 documents in *architecture* domain to 71661 documents in *computer science* domain. The total number of documents is 318175.

Third, queries were generated by using labels of categories in the domain subtree. Queries were created by concatenation of each category and its parent's labels adding "," in between. Queries created from the labels which contain punctuation, special symbols, or boolean operators (e.g., '+', '*'), queries which contain the words shorter than 3 letters, and queries created from categories which contained less than 10 documents were eliminated. Two query sets were created. First query set contains 1432 queries (i.e., 100 or less randomly selected queries from each domain) and is used to evaluate the community selection process (see Figure 7). Second query set contains 18 queries (i.e., 1 randomly selected query from each domain) and is used to compare single-layered *P2PCS* and two-layered *P2PCS* (see Figure 8).

Finally, in order to generate a set of relevance judgments for the second query set, we used a mapping from categories for which queries are created to the documents classified to these categories by DMoz editors. For every category, we collect all the documents classified in the subtree of this category. All such documents are considered to be relevant to the query.

In Figure 7, we show the results of the community selection algorithm. We plot the number of times (y-axis) the correct (the one with the most relevant documents) community was ranked at position n (x-axis). As we can see from Figure 7, in 65% of cases, the correct community was ranked first. In 86% of cases, it was ranked within first three. Note that this was obtained using our simple community selection algorithm and could be improved as mentioned in Section 4.3.

For two-layered approach, each community had 1000 simulated nodes and hence, the total number of nodes was 18,000. The queries were issued from randomly chosen peers from one of the communities. For single-layered approach,

WordNet Domains + YAGO + DMoz				
	1-layer	2-layers (t=0.8)	2-layers (t=0.2)	2-layers (opt.)
MAP	0.1544	0.1573(+1.9%)	0.1736(+12.4%)	0.2039(+32.1%)*
P@10	0.2667	0.2500(-6.3%)	0.2833(+6.2%)	0.3222(+20.8%)
P@20	0.1750	0.1944(+11.1%)	0.2278(+30.2%)*	0.2556(+36.9%)*
ANP	30700.22	7545.94	22396.72	2151.83
s-posting	24582.70	3878.78	9246.89	1455.72
s-hops	17.31	24.99	24.39	23.05

Figure 8: Evaluation results: single vs. two layered(* Improvement is statistically significant according to Wilcoxon signed rank test at level of 0.1)

the peer-to-peer network was simulated with 18,000 nodes and the queries were issued from randomly chosen peers. In Figure 8, we show the results of the evaluation of single and two-layered approaches. For the two-layer approach we performed two experiments where a set of communities was automatically selected with thresholds $t = 0.8$ and $t = 0.2$ (see *Option 3* in Section 2.2). Also we performed an experiment where only the correct communities were selected, i.e., we evaluated the performance in the optimal case, when the user manually selects the set of communities to be queried (see *Options 1* and *2* in Section 2.2). From Figure 8, we can see that if the threshold is too high (e.g. 0.8), then we have a high risk of retrieving the results which are worse in quality than in the case of single layered approach. It is mainly because the correct communities can be missed by the community selection algorithm. If the threshold is relatively low (e.g. 0.2) it is likely that the correct community will be among the selected ones and the quality of the results is improved. The quality is further improved if the community is manually selected.

In the above experiments, with $t = 0.2$, the network cost (ANP) is 72.95% as that of single-layered approach while it reduces to 24.58% when $t = 0.8$. Thus, with our simple community search algorithm itself, the network cost was drastically reduced. With *Option 2*, i.e., when the relevant community was manually selected, the network cost is only 7.01% as that of single-layered approach. This shows that with user intervention in community selection, the search quality as well as network cost reduction can be dramatically improved. Also, there is a lot of scope for better community search algorithms to improve the performance of *Option 3* further. Although, there is in an increase (increased by 33.2% for optimal) in *s-hops* for two-layered approach due to search for communities, the delay due to posting list transfer is substantially reduced (decreased by 94% for optimal) due to parallelization across communities. Thus, the experiments show the improvements obtained (both in quality and network performance) by using a two-layered architecture.

6. RELATED WORK

A number of variants of DHT inverted index based P2P search approaches have been proposed in the literature (e.g., [11, 22]). All of these approaches are based on syntactic matching of words and, therefore, the quality of results produced by these approaches can be negatively affected by the problems related to the ambiguity of natural language. *P2PCS* is based on semantic matching of concepts which allows it to deal with ambiguity of natural language. Since our approach extends syntactic search and does not replace it, the optimization techniques which are used in P2P syntactic search can be easily adapted to *P2PCS*.

Some P2P search approaches use matching techniques based on the knowledge about term relatedness (and not only syn-

tactic similarity of terms). For instance, statistical knowledge about term co-occurrence is used in [21]. Knowledge about synonyms and related terms is used in [12]. Unlike these approaches, *P2PCS* is based on semantic matching of complex concepts. Knowledge about concepts and concept relatedness is distributed among all the peers in the network. Note that, in principle, the knowledge about concept relatedness can be stored in DHT based RDF triple store (see e.g. [1]). But in our approach we use only one type of relation (i.e., subsumption) and therefore the ad-hoc approach for storing distributed background knowledge (as discussed in Section 4.2.4) will require less resources, e.g., we don't need to index triples by predicates as in [1].

Semantic overlay networks[5] follow the scheme of classifying nodes and queries based on semantics. The nodes are clustered together by forming links among each other based on the classification. The queries are classified and are directed to the corresponding cluster. This approach increases the scalability of the search compared to a pure unstructured approach. But, a globally accepted shared classification scheme is required which may not be feasible in a general case. This problem is reduced in our two-layered approach as the universal knowledge, which all the peers need to agree on, is minimal. The rest of the knowledge (community background knowledge) evolves independently in various communities.

In general, formation of peer groups to improve the efficiency of peer-to-peer search has been exploited in various approaches. The node grouping could be based on term distributions[13], group hierarchy[17] or clustering[14]. Unlike these approaches, in our architecture, search and indexing in each node group(community) is done based on background knowledge of the community, made interoperable through a universal knowledge base.

Federated peer-to-peer search[10] addresses the problems of resource representation, ranking and selection, result merging, heterogeneity, dynamicity and uncooperativeness for searching federated text-based digital libraries. But, the knowledge and overlay related issues differ in our case.

7. CONCLUSIONS

We have proposed and implemented a two-layered architecture for peer-to-peer concept search. Our experiments show that peer-to-peer concept search achieves better quality compared to traditional peer-to-peer keyword-based search. The proposed two-layered architecture improves the search quality and network performance compared to a straightforward single-layered approach. However, peer-to-peer search still has scalability issues and is an active area of research. We expect the two-layered approach to be a stepping stone in achieving pragmatic global-scale semantic search. Future works include implementing and comparing alternate design choices, introducing novel search mechanisms, elaborate experiments with peer and community dynamics and experimenting in a large scale deployment across universities.

8. REFERENCES

[1] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, and T. V. Pelt. Gridvine: Building internet-scale semantic overlay networks. In *ISWC*, 2004.

[2] L. Bentivogli, P. Forner, B. Magnini, and E. Pianta. Revising wordnet domains hierarchy: Semantics, coverage and balancing. In *COLING*, 2004.

[3] T. Berners-Lee and L. Kagal. The fractal nature of the semantic web. *AI Magazine*, 29(3):29, 2008.

[4] P. Buitelaar, B. Magnini, C. Strapparava, and P. Vossen. Domain-Specific WSD. *Word Sense Disambiguation: Algorithms and Applications*, 2006.

[5] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems. *LNCS*, 3601:1, 2005.

[6] J. Dharanipragada, F. Giunchiglia, , H. Haridas, and U. Kharkevich. Two-layered architecture for peer-to-peer concept search. Technical report, 2010.

[7] F. Giunchiglia, U. Kharkevich, and S. Noori. P2P Concept Search: Some Preliminary Results. In *Semsearch 2009 workshop*, 2009.

[8] F. Giunchiglia, U. Kharkevich, and I. Zaihrayeu. Concept search. In *ESWC*, 2009.

[9] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris. The Peersim simulator. `http://peersim.sf.net`.

[10] J. Lu and J. Callan. Federated search of text-based digital libraries in hierarchical peer-to-peer networks. In *ECIR*, 2005.

[11] T. Luu, G. Skobeltsyn, F. Klemm, M. Puh, I. P. Žarko, M. Rajman, and K. Aberer. AlvisP2P: scalable peer-to-peer text retrieval in a structured p2p network. In *Proc. of the VLDB Endow.*, 2008.

[12] W. Ma, W. Fang, G. Wang, and J. Liu. Concept index for document retrieval with peer-to-peer network. In *SNPD*, 2007.

[13] L. T. Nguyen, W. G. Yee, and O. Frieder. Adaptive distributed indexing for structured peer-to-peer networks. In *CIKM*, 2008.

[14] O. Papapetrou, W. Siberski, W. T. Balke, and W. Nejdl. DHTs over peer clusters for distributed information retrieval. In *AINA*, 2007.

[15] M. V. Reddy, A. V. Srinivas, T. Gopinath, and D. Janakiram. Vishwa: A reconfigurable P2P middleware for grid computations. In *ICPP*, 2006.

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

[17] S. Shi, G. Yang, D. Wang, J. Yu, S. Qu, and M. Chen. Making peer-to-peer keyword searching feasible using multi-level partitioning. In *IPTPS*, 2004.

[18] V. Srinivas and D. Janakiram. Node capability aware replica management for peer-to-peer grids. *IEEE Transactions on SMC Part A: Systems and Humans*, 39:807–818, July 2009.

[19] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, February 2003.

[20] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *WWW*, 2007.

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

[22] Y. Yang, R. Dunlap, M. Rexroad, and B. F. Cooper. Performance of full text search in structured and unstructured peer-to-peer systems. In *IEEE INFOCOM*, 2006.