SEMANTIC FLOODING: SEMANTIC SEARCH ACROSS DISTRIBUTED LIGHTWEIGHT ONTOLOGIES

Fausto Giunchiglia, Uladzimir Kharkevich and Alethia Hume

July 2009

Technical Report # DISI-09-036

Semantic Flooding: Semantic Search across Distributed Lightweight Ontologies ⋆

Fausto Giunchiglia, Uladzimir Kharkevich, and Alethia Hume

Department of Information Engineering and Computer Science
University of Trento, Italy
{fausto,kharkevi,hume}@disi.unitn.it

Abstract. Lightweight ontologies are trees where links between nodes codify the fact that a node lower in the hierarchy describes a topic (and contains documents about this topic) which is more specific than the topic of the node one level above. In turn, multiple lightweight ontologies can be connected by semantic links which represent mappings among them and which can be computed, e.g., by ontology matching. In this paper we describe how these two types of links can be used to define a semantic overlay network which can cover any number of peers and which can be flooded to perform a semantic search on documents, i.e., to perform *semantic flooding*. We have evaluated our approach by simulating a network of 10,000 peers containing classifications which are fragments of the *DMoz* web directory. The results are promising and show that, in our approach, only a relatively small number of peers needs to be queried in order to achieve high accuracy.

1 Introduction

We can see the Web as a network of peers (a P2P network) where each peer stores various documents *about* a set of topics which are of interest to its users. Most often these documents are organized in classifications, i.e., tree-like topic hierarchies (e.g. email directories, file systems, web directories, and so on). An abstract example of user generated classifications of several peers can be seen in Figure 1. Such classification hierarchies, also called lightweight ontologies [19], have always been used by humans as the most effective and intuitive way to organize their knowledge according to their subjective view of a domain of interest [19, 16]. Nodes in the classification specify those (complex) concepts which the user is interested in. For example, the user of *Peer1* is interested in documents about *mice* and *hippopotamuses*. The whole classification specifies the user interest profile. For example, the user of *Peer1* is interested in various kinds of animals, and the user of *Peer3* is interested in cars. Notice, that a user can be interested in more than one topic. For instance, the user of *Peer2* stores documents about both animals and cars.

⋆ A short version of this article with the title *"Semantic Flooding: Search Over Semantic Links"* was published at the 1st International Workshop on Data Engineering meets the Semantic Web (DESWeb2010).

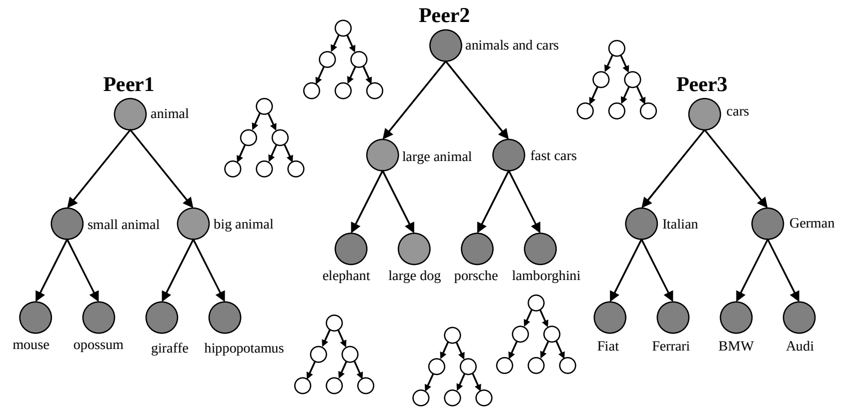

Fig. 1. P2P Network of User-Generated Classifications

The goal of this paper is to show how multiple classifications can be exploited to help the user in finding documents about the topics which are of interest to the user. For example, a user novice in some topic might benefit from finding a peer, the user of which is an expert in the given topic. Moreover, when searching for a particular document about the topic, even an expert might be interested in finding not only those documents which are stored in its local document collection, but also the documents which are stored on other peers. We propose an approach, that we call *Semantic Flooding*, which is based on the following three key ideas:

1. The first is that the links which connect nodes inside a classification together with the links which codify ontology mappings among classifications form a *semantic overlay network* which can be exploited to perform a semantic search on nodes.
2. The second is that semantic search on nodes is implemented by flooding the links of the semantic overlay network. Differently from "normal" flooding as it happens, for instance, in Gnutella [3], these links carry meaning and more precisely, codify the semantic relation (i.e., equivalence, more or less general) holding between any two nodes and allow, therefore, for "more informed" query propagation.
3. The third and last is that semantic search inside a node is performed by using *Concept Search* (*C-Search*) [15] (a semantics enabled information retrieval approach) thus exploiting as much as possible the advantages of a syntactic search and also a semantic search, as a function of the available background knowledge [17].

The paper is structured as follows. In Section 2, we define the semantic overlay network built out of the links in (and mappings across) the classifications. In Section 3, we show how this network can be exploited to perform semantic flooding. In Section 4, we show how links across classifications can be computed via semantic matching (as described in [18]) or via semantic search in distributed hash table (DHT) based on *P2P Concept Search* [14] approach. Section 5 provides the evaluation. In Section 6 we discuss the related work. Section 7 concludes the paper.

2 A Semantic Overlay Network

Formally classification can be defined as a rooted tree $C = \langle N; E; L \rangle$, where N is a set of nodes, E is a set of edges on N, and L is a set of labels expressed in a natural language, such that for any node $n \in N$, there is label $l \in L$ associated with n. Labels of nodes are used to describe an intended content of the node. An example of a user-generated classification is shown in Figure 2a. In order to allow automatic reasoning about classifications and their content,

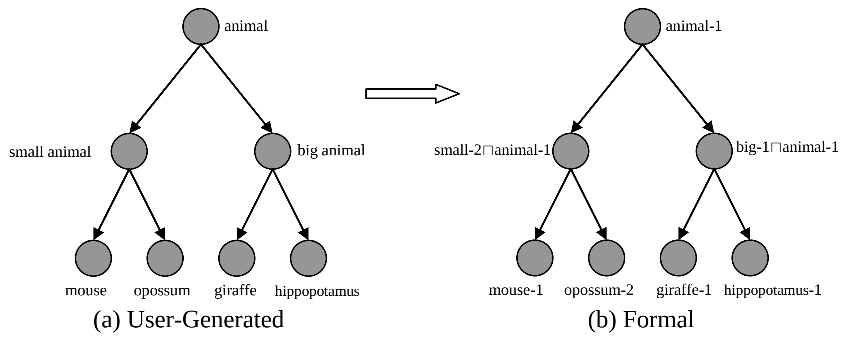

Fig. 2. Classification

each classification is converted into a Normal Formal Classification (NFC) [16] also called lightweight ontology [19]. A NFC is a rooted tree $NFC = \langle N, E, L^N \rangle$ where N is a set of nodes, E is a set of edges on N, and L^N is a set of labels expressed in Propositional Description Logic language L^C, such that for any node $n \in N$, there is label $C^n \in L^N$ associated with n.

To convert a classification into a NFC, the *background knowledge (BK)* [17] of the user is used. BK represents the knowledge of the user about concepts and their relationships over a specific domain or a limited set of domains. Formally, BK can be defined as a 4-tuple: $BK = \langle W, C, R_{WC}, R_{CC} \rangle$, where W is a set of words (together with their part-of-speech, glosses, etc); C is a set of word senses (concepts), R_{WC} is a set of links from words to their senses (concepts); and R_{CC} is set of subsumption relations on the concepts in set C. An example of BK is shown in Figure 3. Atomic concepts are represented as *lemma-sn*, where *lemma* is the lemma of the word, and *sn* is the sense number. For instance, the atomic concept *dog-1* is used in the sense of a domestic dog, which is the first sense of word dog in the BK.

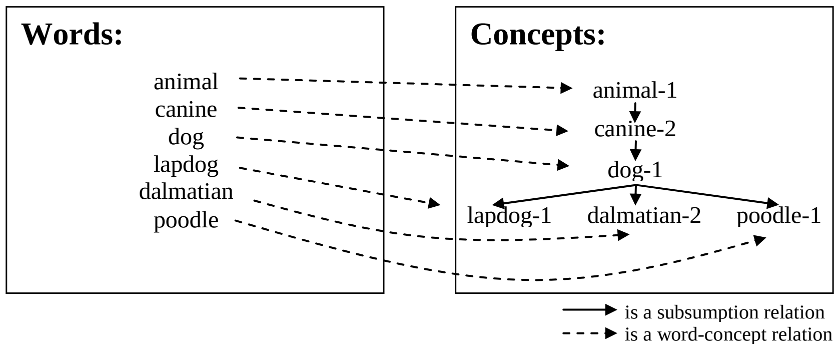

Fig. 3. Background Knowledge

The detailed discussion about how a classification can be converted into a NFC can be found in [16, 35]. In short, the conversion is performed as follows. Every label of a node in a classification is assigned a complex concept l^n expressed in L^C. Single words in the label are assigned atomic concepts from the peer's BK. Complex concepts are built by translating syntactic relations between words in the label into logical connectives of L^C. For example, the label "big animal" is translated into the complex concept $big\text{-}1 \sqcap animal\text{-}1$. A classification in which all the labels are converted into expressions in L^C is called Formal Classification (FC). An example of a FC created from the classification in Figure 2a is shown in Figure 2b. Notice that the meaning of a node in a classification is defined by a set of concepts on the path to the root and not only by a concept of the node's label. In order to encode meaning of nodes in the classification we use the notion of *concept at node* [18]. The concept at node C^n is defined as a conjunction of concepts at labels l^n for all the nodes on the path to the root from the given node:

$$C^n = l^{n_1} \sqcap l^{n_2} \sqcap \ldots \sqcap l^{n_i} \tag{1}$$

The resulting classification, in which concepts at node C^n are computed for all the nodes in the classification, is a NFC.

After a NFC has been created, documents can be automatically classified to nodes in the classification by using the get-specific principle [20]. Each document d is assigned an expression in L^C, which we call the document concept C^d. To assign a concept C^d to a document, first, a set of n key-phrases is retrieved from the document using text mining techniques (see, for example, [32]); the key-phrases are then converted to formulas in L^C, and, finally, C^d is computed as the conjunction of the formulas. According to the get-specific principle, documents are classified to nodes, such that complex concepts of these nodes are more general than complex concepts assigned to documents, and none of child nodes can describe the content of the document more specifically. Formally, a set of documents $S(n)$ classified in a sub-tree of node n is defined as follows:

$$S(n) = \{d \mid C^d \sqsubseteq C^n\} \tag{2}$$

If node n has a set of child nodes $C(n)$, then a set of documents $D(n)$ classified to node n, is defined as follows:

$$D(n) = S(n) - \bigcup_{n_i \in C(n)} S(n_i) \tag{3}$$

To make the peers in the P2P Network able to reason about the contents of each other, semantic links, expressed in the C-OWL language [9], can be created between related nodes in their classifications. C-OWL envisions a wide range of possible semantic relations that can hold between related nodes in different classifications. For the goals of this paper we concentrate on the following links: (i) equivalence links (represented as $A \xrightarrow{\equiv} B$), (ii) more general links ($A \xrightarrow{\sqsupseteq} B$), and (iii) more specific links ($A \xrightarrow{\sqsubseteq} B$). For example, in Figure 4, the link between nodes with labels "large dog" and "huge dalmatian" is used to specify that the concept of the former node ($large$-1 $\sqcap$ dog-1) is more general than the concept of the latter node ($huge$-1 $\sqcap$ $dalmatian$-2). Note that, according to Equation 2,

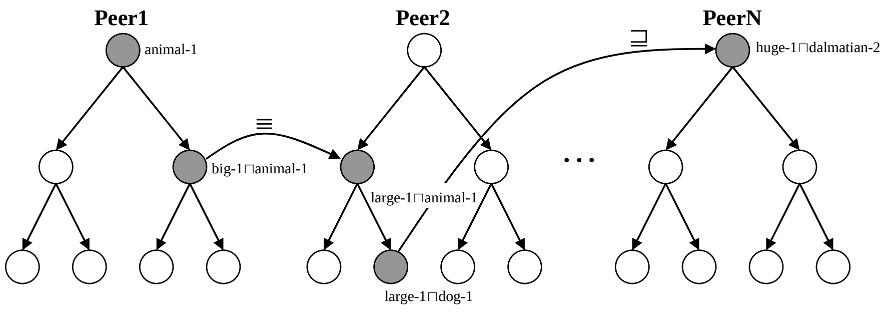

Fig. 4. A Semantic Overlay Network

all the documents which are classified in the subtree of the latter node can be classified also in the subtree of the former node.

The set of the links which connect nodes inside a classification plus C-OWL links across classifications constitute a semantic overlay network which can be built on top of any underlying set of peers and their physical connections.

3 Semantic Flooding

We consider a semantic search in a P2P network as the process of finding documents, which are semantically related to the user information needs, and which are parts of a document collection distributed among all the peers in the network. When a user searches for documents, she, first, selects a node n in the classification. The root node of the classification serves as a default node for search if no other node is selected. Second, the user issues the query q. The query is converted into an expression in L^C using the same technique used for creation of concept at nodes, as described in the previous section. Let C^n be a complex concept at node n and C^q be a complex concept extracted from query q. The goal of the *Semantic Flooding* algorithm is to find documents d stored in the network, such that, concept of document C^d is more specific than the concept at node C^n and there exists a concept C described in d which is more specific than the query concept C^q. Formally a query answer $A(C^n, C^q)$ is defined as follows:

$$A(C^n, C^q) = \{d \mid C^d \sqsubseteq C^n \text{ and } \exists C \in d, \text{ s.t., } C \sqsubseteq C^q\} \tag{4}$$

The problem of a semantic search in the P2P network can be decomposed into three subproblems:

1. Identifying semantically relevant peers
2. Searching inside relevant peers
3. Aggregation of the search results

Let us consider these three subproblems in detail.

3.1 Identifying semantically relevant peers

We consider a peer to be semantically relevant to a query if there are nodes in the peer's classification which are relevant to the node selected by the user. Moreover, some of the documents classified in these nodes should be relevant to the user query. In order to store the information about potentially relevant peers, the initiator peer p_I creates a peer information list, defined as follows:

$$peerinfos(n) = [\langle p, nodeinfos(p, n), stat\rangle],$$

where p is a relevant peer, $stat$ is a status of p: NQ - peer is not queried, QU - peer is already queried, or RE - response is returned, and $nodeinfos(p, n)$ is a list which stores information about nodes n' from peer p which are semantically related to node n plus a set $\{l\}$ of incoming links l for node n':

$$nodeinfos(p, n) = [\langle n', \{l\}\rangle]$$

Initially, $peerinfos(n)$ contains information only about the peer p_I: $peerinfos(n) = [\langle p_I, [\langle n, \emptyset\rangle], NQ\rangle]$. After $peerinfos(n)$ is initialized, p_I starts an infinite loop, where a single iteration is performed as follows:

- Select the first (if any) peer info $\langle p, nodeinfos(p, n), stat\rangle$ from $peerinfos(n)$, such that, $stat = NQ$.
- If there are no such peer infos, wait until the $peerinfos(n)$ list is modified and perform the previous step again.

– Form a query request $\langle C^n, C^q \rangle$ and submit it to peer p.
– Change the status of peer p to $stat = QU$.

When peer p receives the query request, it locally computes a set of links L, such that, each of the target nodes has a complex concept which is more specific than the complex concept C^n. Note that, at the same time, the concept of the target node in a link can be equivalent, more specific, or more general than the concept of the source node. All the links in L are sent back to the initiator peer p_I. Peer p_I updates the $peerinfos(n)$ list by using information from links in L. $peerinfos(n)$ list is then sorted in a decreasing order of the number of incoming links. We assume that in this way, peers are queried in a decreasing order of their importance.

Every node n' in $nodeinfos(p, n)$ has only the documents with complex document concepts C^d which are more specific than the complex concept C^n. This is because, from $C^d \sqsubseteq C^{n'}$ and $C^{n'} \sqsubseteq C^n$, it follows that $C^d \sqsubseteq C^n$. In spite of this, links between nodes do not describe all complex concepts C, which can be found in the documents classified to these nodes. Therefore, it can be the case that node n' has no documents which are relevant to the query concept C^q. The portion of such nodes can increase when a concept C^n becomes more and more general. In the worst case, i.e., when $C^n \equiv \top$, all the nodes which can be reached by all the links can be added to $nodeinfos(p, n)$ and all the corresponding peers p can be queried. Semantic flooding in this case is reduced to normal flooding and, in general, can be very inefficient.

In order to implement a more efficient selection of semantically relevant peers, we propose to use a measure of semantic similarity $SS(C^{n'}, C^q)$ between complex concepts at node $C^{n'}$ and the complex query concept C^q (see, for example, [8]). As a simple example of a semantic similarity measure $SS(C^{n'}, C^q)$, let us consider the following measure:

$$SS(C^{n'}, C^q) = \begin{cases} 1 & \text{if } C^{n'} \sqsubseteq C^q \\ 0 & \text{otherwise} \end{cases}$$

Observe that for n' with $SS(C^{n'}, C^q) = 1$, concepts C^d, for all the documents classified to n', are more specific than query concept C^q. It is because, from $C^d \sqsubseteq C^{n'}$ and $C^{n'} \sqsubseteq C^q$, it follows that $C^d \sqsubseteq C^q$. Given that C^d is built from concepts C found in the document d, it is likely that d is relevant to query q. Note that the following measure of semantic similarity is actually used:

$$SS(C^{n'}, C^q) = \sum_{A^q \in C^q} \frac{1}{10^{\min\limits_{A^{n'} \in C^{n'}} (dist(A^{n'}, A^q))}} \tag{5}$$

Now, instead of just the number of incoming links, $peerinfos(n)$ list is sorted in a decreasing order of the peer scores computed as a sum of node scores $score(n', q)$. A node score $score(n', q)$ is computed as follows:

$$score(n', q) = (N_l + 1) * (SS(C^{n'}, C^n) + SS(C^{n'}, C^q)), \tag{6}$$

where, N_l is a number of incoming links for node n'. Note that only links for those nodes which are relevant for current search request are considered while sorting $peerinfos(n)$.

3.2 Searching inside a relevant peer

On receiving a search request $\langle C^n, C^q \rangle$, peer p performs search for relevant documents in a local document collection by using the *C-Search* [15]. C-Search is an IR approach which is based on retrieval models and data structures of syntactic search, but which searches for complex concepts C rather than words W. The key idea is that syntactic matching of words is extended to semantic matching [18] of complex concepts, where semantic matching is implemented by using positional inverted index. The output of C-Search is a list of documents ordered by their relevance to the query. A list of top k ranked documents, nodes to which the documents are classified, and the information about frequencies of atomic concepts $A \in C^q$ in the retrieved documents and in the whole local document collection are sent back to the initiator peer p_I. Peer p_I updates the $peerinfos(n)$, i.e., the status of p is changed to $stat = RE$. In order to store the information about the relevant documents, the initiator peer p_I uses a document information list:

$$docinfos(q) = [\langle d, n', [\langle A, tf(A,d) \rangle] \rangle],$$

where d is a document which is classified to node n', and which is also relevant to query q, $tf(A,d)$ is a number which represents the importance of document d to an atomic concept $A \in C^q$. Moreover, in order to store the global information about the importance of atomic concepts $A \in C^q$, p_I uses term information lists for all A:

$$terminfos(A) = [\langle p, numDocs_p, docFreq_p(A) \rangle],$$

where $docFreq_p(A)$ is a number which represents the frequency of atomic concept A in the document collection of peer p which has $numDocs_p$ documents in total. When receiving new results, a peer p_I updates the $docinfos(q)$ and $terminfos(A)$ tables.

The search process terminates when: (i) the required number (e.g., 100) of documents is retrieved; or (ii) all the relevant documents are retrieved; or (iii) the search time exceeds some predefined limits; or (iv) the user terminates the process.

3.3 Aggregation of search results

After the search process is terminated, the peer p_I merges query answers from different peers into a single query answer. First, the cosine similarity $cos(d,q)$ from the vector space model is computed for every retrieved document d. Terms are weighted by the *tf-idf* weight measure used in *Lucene* [4], where an inverse document frequency $idf(A)$ is estimated as follows:

$$idf(A) = 1 + log(\frac{numDocs}{docFreq(A) + 1}),$$

where *numDocs* is computed as a sum of all the $numDocs_p$, and $docFreq(A)$ is computed as a sum of all the $docFreq_p(A)$. Second, the cosine similarity $cos(d, q)$ is combined with the score $score(n, q)$ of the node n to which the document is classified in order to compute the final score of the document $score(d, q)$:

$$score(d, q) = score(n, q) + cos(d, q)$$

Finally, documents are ordered according to the relevance score and presented to the user in the decreasing order of relevance.

4 Semantic Link Discovery

When a new peer joins the network, there are no semantic links connecting the nodes in the classifications of this peer with the nodes in classifications of other peers in the network. Another example where links can be missing is when a new node is created in a classification, e.g., because the user became interested in a new topic. In the following we discuss how new semantic links can be discovered in these and other similar situations.

If the two classifications which need to be connected are known in advance, then semantic links between these classifications can be computed by using semantic matching (*S-Match*) [18] approach. When the relevant classifications are not known, one way of computing semantic links is to run S-Match between the given classification and all the classifications of other peers. The problem with this approach is that the number of peers in the network can be huge and, therefore, running S-Match for all the possible combinations of classifications can become unfeasible.

In order to allow for an efficient discovery of semantic links, we propose to use *P2P Concept Search* [14]. P2P Concept Search extends C-Search [15] by allowing a distributed semantic search over structured P2P network. First, the reasoning with respect to a single background knowledge $\mathcal{T}$ is extended to the reasoning with respect to the background knowledge $\mathcal{T}_{P2P}$ which is distributed among all the peers in the network. Second, the centralized inverted index (II) is extended to a distributed inverted index built on top of a DHT [27, 30, 13, 36]. In this paper, P2P Concept Search is used in order to index and retrieve complex concepts at nodes $C^{n'}$. In this case, the query answer $A(C^n, \mathcal{T}_{p2p})$, produced by P2P Concept Search for complex concept C^n, contains a set of nodes n', such that, complex concepts at nodes $C^{n'}$ are more specific than C^n:

$$A(C^n, \mathcal{T}_{p2p}) = \{n' \mid \mathcal{T}_{p2p} \models C^{n'} \sqsubseteq C^n\}$$

If the user wants to discover semantic links for a node n, first, the query answer $A(C^n, \mathcal{T}_{p2p})$ is computed by using P2P Concept Search. The system then returns the ordered list of possibly relevant nodes to the user. Note that if no exact matches are found, partial matches are returned, i.e., when not all atomic concepts $A \in C^n$ are used. Finally, semantic links are created for those nodes which are selected by the user. Links created by users are indexed in the DHT by id's of target nodes. Note that these links can be used in the computation of the node score in Equation 6. After semantic links are discovered for a node (or a set of nodes), S-Match can be run between the user's classification and some of the classifications which are connected by the links.

5 Evaluation

In order to evaluate our approach, we conducted a set of simulation experiments, where the results of the proposed distributed *Semantic Flooding* algorithm and centralized *C-Search* algorithm were compared[1]. The key intuition here is to see how much we lost, in terms of the number of documents which are retrieved by a centralized search approach and which are missing from the results of a distributed search approach.

5.1 Evaluation parameters

In the evaluation, we measured the accuracy of search results returned by *Semantic Flooding* algorithm depending on the number of visited peers, where the accuracy is defined as follows:

$$Accuracy = \frac{|R_{CS} \cap R_{SF}|}{|R_{CS}|} * 100\%,$$

where R_{CS} are results returned by *C-Search* and R_{SF} are results returned by *Semantic Flooding* on the same data-set. Only the first 10 ranked results are used. The accuracy measure considers the results returned by *C-Search*, as the desired results and estimates the ability of semantic flooding algorithm to approximate these results by querying only a limited number of peers.

The number of queried peers can be used to estimate the number of messages M_{num} generated to answer a query in the best and the worse case scenarios. In the best case scenario, no link discovery is needed because all the relevant links are already computed. The number of generated messages in this case can be estimated as follows:

$$M_{num} = 2 * m,$$

where m is the number of queried peers.

In the worst case scenario, the relevant links need to be computed by the link discovery mechanism. If *P2P Concept Search* is used for link discovery, then the number of messages can be estimated by using the following formula:

$$M_{num} = \log p + k + (2 * m),$$

where p is number of peers in the network, m is the number of queried peers and k is the number of atomic concepts that are used by *P2P Concept Search* algorithm (see [14] for details). In the evaluation, k was limited to 10.

[1] Note that comparing performance of distributed and centralized information retrieval systems is a standard way of evaluating in P2P information retrieval (e.g., see [31]).

5.2 Dataset generation

In order to generate data-sets (which reproduce a realistic scenario) for the evaluation of the proposed semantic flooding algorithm, we used data from the Open Directory Project (ODP), also known as *DMoz* [2], and the public tags of about 950000 users from Delicious [1] obtained from [33]. Two sources were merged together by matching users from Delicious to nodes in DMoz classification. First, the intersection of urls from both sources was computed. Second, users which tagged a url in Delicious were matched to the node n in DMoz which classifies the document with the same url. As a result, we obtained 491414 users matched to 45545 nodes.

Four data-sets were generated by randomly selecting a sub-set of 10, 100, 1000, and 10000 users (one use is assigned to one peer). For each user u, the generation of its classification was performed as follows. Nodes matched to a user form the classification hierarchy. For each node n in the user's classification, we selected a sub-set of documents D_n classified to node n in DMoz. First, we selected all the documents that were tagged by the user and than we selected a random sub-set of spare documents (i.e., documents classified to n that were not tagged by user). Documents in the data-set were created by concatenating titles and descriptions of web-sites. On average, a classification of each peer had 21 nodes and 385 documents.

For each data-set, a *C-Search* index I_{CS} was created. All the documents in the data-set were indexed in I_{CS}. WordNet was used in *C-Search* as a background knowledge. Indexes of each single peer were created by filtering I_{CS}. Distributed Background Knowledge (DBK) [14], that provides access to the BK of each peer in the P2P network, was used to index concepts and relations from WordNet in the P2P network. By using DBK, a peer can exploit the knowledge of other peers in the network when an atomic concept is missing in the local BK of the peer. We can see the DBK as the sum of the BKs of all the peers in the network. By using the same BK in both centralized and distributed approaches we ensure the fairness of comparison of the results produced by these approaches.

A query set was generated by randomly selecting a set of N_q (100) queries from the AOL query log [26] for each data-set. One word queries; queries which contained punctuation, special symbols, or boolean operators (e.g., '+', and '?'); queries which contain the words shorter than 3 letters; and queries which had less than 10 results in I_{CS} were filtered out. For each query, we randomly selected a node n in *DMoz* classification, such that, a query request $\langle C^n, C^q \rangle$ have at least 10 relevant documents as computed by *C-Search*.

Given that our data-sets are generated from the data produced by the real users, nodes matched to users provide us with knowledge about real user interest profiles. Note that interests and accordingly classifications of different users can partially overlap, where the overlap has a higher probability for popular topics (e.g. *Top/Computers* and *Top/News*). In the following we analyze the distribution of peers' interests over the set of topics from DMoz. In Figure 5, we show the distribution of the popularity of topics in our data-set. Topics are ranked according to the popularity. The most popular topic occupies the first position

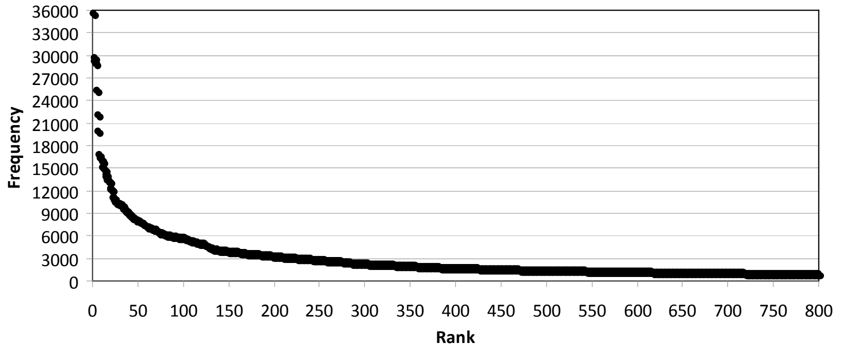

Fig. 5. Topics Popularity Distribution

in the ranking, followed by the second most popular, and so on. From Figure 5, we can observe the rapid decrease in the frequency from 35503 for the most popular topic to 29690 for the second most popular. Even more, only moving 50 places down in the ranking the frequency decreases to 8066. And there are only 121 topics that have more than 1% of peers interested on them. This behavior provides evidence of the existence of a "long tail" of unpopular topics.

In order to see how popularity of topics affect the performance of different approaches, we additionally generated two query sets for the data-set consisting of 10000 peers. The first query set consists of popular queries (i.e., queries related to topics that are in the first 200 positions in the popularity ranking, see Figure 5) and the second query set consists only of unpopular queries (i.e., queries related to topics in the position 400 or above in the popularity ranking, see Figure 5).

5.3 Evaluation results

The evaluation results for randomly selected queries are reported in Figure 6. We compared the performance achieved by Semantic Flooding when: (i) the query request $\langle C^n, C^q \rangle$ consists of a starting node n with concept C^n and of a query q with a concept C^q; (ii) the query request is $\langle \top, C^q \rangle$, namely the same as in (i) but with no starting node, i.e., $C^n \equiv \top$; and (iii) the same as (ii) but the semantic similarity $SS(C^{n'}, C^q)$ is not used. Note that in P2P networks of 10 and 100 peers, the total number of queried peers was set to 10, whereas in P2P networks of 1000 and 10000 peers, it was set to 50. From Figure 6, we can see that, when peers are selected without using the similarity function and also without a starting node specified (see "no semantic similarity" lines in Figure 6), accuracy decreases very quickly with the total number of peers in the network. The situation improves when semantic similarity is used and only starting node is missing (see "no starting node" lines in Figure 6). When the starting node n is selected, i.e., concept C^n is provided, the accuracy of *Semantic Flooding* becomes close to the accuracy of the centralized *C-Search* approach (see

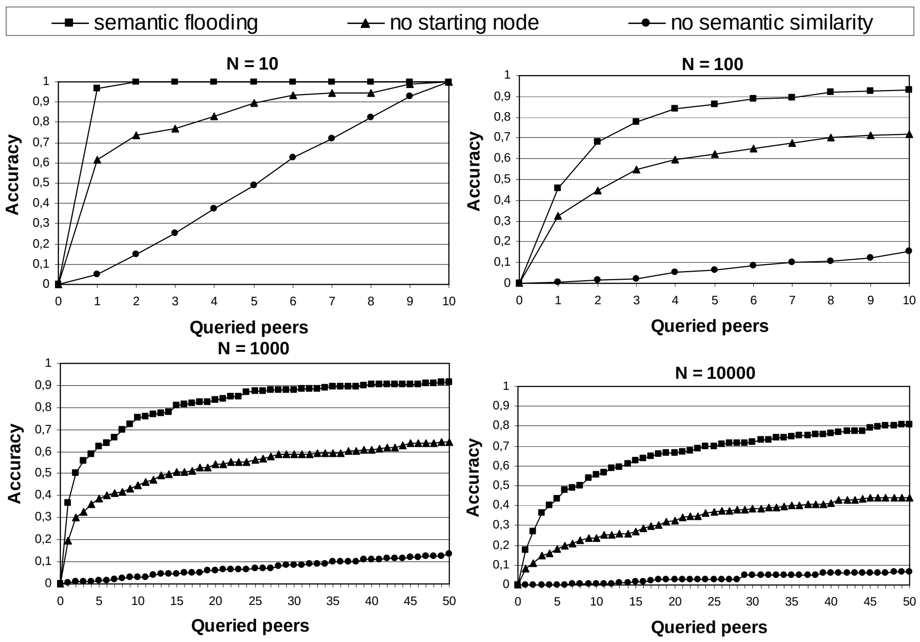

Fig. 6. Evaluation Results: Random queries

"semantic flooding" lines in Figure 6). In fact, in the network of 10000 peers, only 50 peers need to be queried in order to achieve 70% of accuracy. Note that if we need to retrieve one relevant result (i.e., 10% of accuracy), only one peer needs to be queried.

The evaluation results for popular/unpopular queries are reported in Figure 7. From Figure 7, we can see that even a normal flooding approach can achieve good results for popular queries. This is because there are many peers which can provide answers to such queries. On the other hand, for unpopular queries results provided by the normal flooding decrease substantially (i.e., the accuracy has decreased 4 times), whereas the Semantic Flooding approach still provide good accuracy (i.e., the accuracy has only decreased by 22%). Overall, we can see that the Semantic Flooding approach can provide good results for both popular and unpopular queries.

6 Related work

A number of P2P search approaches have been proposed in the literature (for an overview see [28]). The algorithm implemented by Gnutella is the classical example of a query flooding algorithm. In early versions of Gnutella, connections between peers were made mainly chaotically. A P2P network was completely *unstructured*, i.e., it did not have any predefined structure. The query sent by

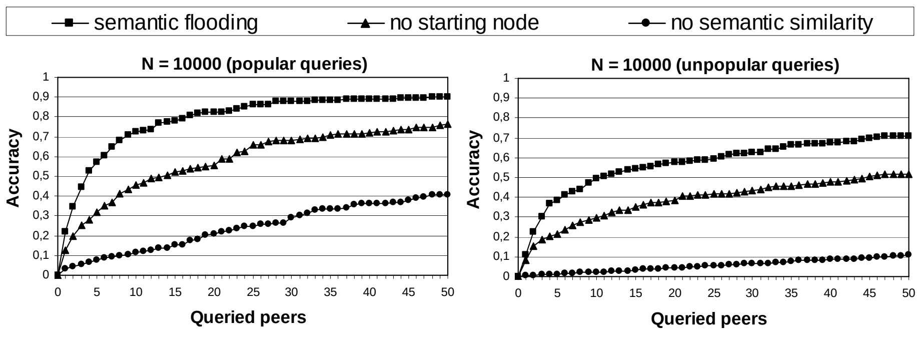

Fig. 7. Evaluation Results: Popular vs. Unpopular Queries

a peer was propagated to all the actively connected peers within a predefined number of *hops* from the query sender. The search process was *blind*, i.e., peers have no information related to the resource location. The lack of scalability was recognized as the main problem of the Gnutella. Various techniques were adopted in later versions of the Gnutella protocol in order to make the search process more scalable. *Super-peers* were introduced to utilize the heterogeneity between peers in computer power, bandwidth and availability. *Informed search*, i.e., when peers maintain additional information about resource locations which can be useful for the search, replaced *blind search*. In Gnutella, informed search is implemented by using Query Routing Protocol (QRP). Query Routing Tables (QRT) consisting of hashed keywords are exchanged between peers. During query routing, search request is propagated only to those peers which have all of the query words in its QRT. In [11], a peer uses Routing Indices to forward queries to neighbors that are more likely to have answers. Query topics are compared to neighbor's expertise to select relevant peers. In our approach, search for relevant peers is implemented by using semantic links created between nodes in classifications of different peers.

The basic idea of [5, 10, 29, 37, 12] is to organize peers into Similar Content Groups on top of unstructured P2P systems, i.e., a *peer clustering* approach is implemented. Peers from the same group tend to be relevant to the same queries. A query is guided to Similar Content Group that is more likely to have answers to the given query and then the query is flooded within this group. For instance, in Semantic Overlay Networks (SONs) [12] peers that have similar documents are clustered at the same group. A predefined classification hierarchy is used to classify the peers' documents. Thus two peers belong to the same SON if some of their documents classified under the same concept in this global classification. Peers can belong to more than one SON. In our approach, peers with similar content are connected to each other by creating semantic links between nodes in classifications of these peers. Differently from [12], a global classification is not required and users are free to create their own classification hierarchies.

CAN [27], Chord [30], Pastry [13], and Tapestry [36] use another approach to the routing and topology organization of P2P networks. This approach employs the DHT functionality (e.g. mapping keys onto values) on Internet-like scale. Such systems are highly *structured*. The topology is tightly controlled and documents (or information about documents) are placed at the precisely specified locations defined by their keys. A *data clustering* approach is implemented, i.e., similar data (meta-data) is placed in the same place. Search in these systems is limited to an exact key search. Mercury [7] supports multi-attribute range queries, e.g. each query is a conjunction of ranges in one or more attributes. Examples of how a full text retrieval can be implemented on top of structured P2P networks are described in [22, 23, 6, 31]. A straightforward way to implement syntactic search is to use the DHT to distribute peers' inverted indices in the P2P network [28]. In order to find a set of documents which contain a term we just need to contact the peer responsible for this term and retrieve the corresponding posting list. In order to search for more than one term, we, first, need to retrieve posting lists for every single term, and then to intersect all these posting lists. The above approach can potentially be very expensive in terms of required storage and generated traffic (see e.g. [22]). For instance, posting lists need to be transferred when peers join or leave the network. Searching with multiple terms requires intersection of posting lists, which also need to be transferred. In the case of huge posting lists, a bandwidth consumption can exceed the maximum allowed limits. In [22], it is shown that the efficiency of DHT can be even worse than the efficiency of a simple flooding algorithm. Some of optimization techniques (e.g., Bloom Filters), which can improve the performance of posting lists intersecting, are summarized in [22]. In [23], indexing is performed by terms and term sets appearing in a limited number of documents. Different filtering techniques are used in [23] in order to make vocabulary to grow linearly with respect to the document collection size. In Minerva [6] DHT holds only compact, aggregated meta-information about the peers' local indexes which is used to efficiently select promising peers from across the peer population that can best locally execute a query. In our approach, a DHT based P2P Concept Search is used only for indexing and retrieval of nodes from classifications and not documents. The problem with storage is reduced since inverted indices for nodes are usually smaller than those for documents which are classified to these nodes. Moreover, the generated traffic is reduced because, in P2P Concept Search, a query consisting of a single complex concept do not require the intersection of inverted indices.

All of the described so far 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. Some P2P search approaches use matching techniques which are based on the knowledge about term relatedness (and not only syntactic similarity of terms). For instance, statistical knowledge about term co-occurrence is used in [31]. Knowledge about synonyms and related terms is used in [24]. In our approach, the problem of ambiguity of natural language is dealt with by using semantic matching of complex concepts. Different semantic search approaches are also used in [38, 34, 25, 21]. A semantic link peer-to-peer network (P2PSLN) [38] specifies and manages semantic relationships between peers' data schemas. A semantic-based peer similarity measurement is used for efficient query routing. A schema mapping algorithm is used for query reformulation and heterogeneous data integration. Ontology-based P2P data management system (OPDMS) [34] is based on ontology mapping and query processing. Edutella [25] and Bibster [21] are built on JXTA framework and aim to combine meta-data with P2P networks. Each peer is described and published using an advertisement, which is an XML document describing a network resource. For example in the Bibster [21] system, these expertise descriptions contain a set of topics that the peer is an expert in. Peers use a shared ontology to advertise their expertise in the Peer-to-Peer network. In our approach, semantic links are created between semantically related nodes in classifications of different peers and not between data schemas of the peers, as in [38]. Moreover, differently from [21], we don't assume shared ontology.

In Table 1, we provide a summary of the search methods discussed in this section.

Table 1. Search Methods in P2P networks.

	Network structure	Clustering	Identifying semantically relevant peers	Search method
Gnutella [28]	Unstructured	-	Blind	Keyword
Routing Indices [11]	Unstructured	-	Informed	Keyword
SETS [5]	Unstructured	Peers	Informed	Keyword
Associative overlay [10]	Unstructured	Peers	Informed	Keyword
Interest-based overlay [29]	Unstructured	Peers	Informed	Keyword
ESS [37]	Unstructured	Peers	Informed	Keyword
SONs [12]	Unstructured	Peers	Informed	Keyword
P2PSLN [38]	Unstructured	Peers	Informed	Semantic
OPDMS [34]	Unstructured	Peers	Informed	Semantic
EDUTELA [25]	Hybrid	Peers	Informed	Semantic
Bibster [21]	Hybrid	Peers	Informed	Semantic
pSearch [31]	Structured	Data	Informed	Semantic
Concept Index in P2P [24]	Structured	Data	Informed	Semantic
CAN [27]	Structured	Data	Informed	Key
Chord [30]	Structured	Data	Informed	Key
Pastry [13]	Structured	Data	Informed	Key
Tapestry [36]	Structured	Data	Informed	Key
Mercury [7]	Structured	Data	Informed	Keyword
MINERVA [6]	Structured	Data	Informed	Keyword
AlvisP2P [23]	Structured	Data	Informed	Keyword

7 Conclusions

Document classifications (lightweight ontologies) together with semantic links which codify the mappings existing among nodes define a semantic overlay network which can cover any number of peers (e.g., in the Web). In this paper, we have shown how these links can be flooded and used to perform a semantic search on links. The resulting approach, that we call semantic flooding, seems very promising. In fact the experimental results, evaluated on networks of 10, 100, 1000, and 10000 peers containing classifications which are fragments of the *DMoz* web directory, show that *Semantic Flooding* is robust and gracefully degrades with the number of peers. The future work will include evaluating the sensitivity of the algorithm on the number of links and on their quality. Semantic Flooding will also be compared with P2P Concept Search, namely DHT based *C-Search*, with the goal to understand the conditions under which Semantic Flooding performs better (or worse) than P2P Concept Search.

References

1. Delicious: `http://www.delicious.com/`.
2. Dmoz: `http://www.dmoz.org/`.
3. Gnutella: `http://www.gnutella.com/`.
4. Lucene: `http://lucene.apache.org/java/2_4_0/api/org/apache/lucene/search/Similarity.html`.
5. M. Bawa, G. Manku, and P. Raghavan. Sets: Search enhanced by topic segmentation. In *Proceedings of The 26th Annual International ACM SIGIR Conference*, pages 306–313, 2003.
6. Matthias Bender, Sebastian Michel, Peter Triantafillou, Gerhard Weikum, and Christian Zimmer. Minerva: Collaborative p2p search. In *Proceedings of VLDB*, pages 1263–1266, 2005.
7. A. Bharambe, M. Agrawal, and S. Seshan. Mercury: Supporting multi-attribute range queries. In *Proceedings of ACM SIGCOMM*, 2004.
8. Alexander Borgida, Thomas Walsh, and Haym Hirsh. Towards measuring similarity in description logics. In *Proceedings of the 2005 International Workshop on Description Logics (DL2005)*, 2005.
9. P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. Contextualizing ontologies. In *Journal of Web Semantics*, volume 1, pages 325–343, 2004.
10. E. Cohen, H. Kaplan, and A. Fiat. Associative search in peer to peer networks: Harnessing latent semantics. In *Proceedings of IEEE INFOCOM*, 2003.
11. A. Crespo and H. Garcia Molina. Routing indices for peer-to-peer systems. In *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2002.
12. Arturo Crespo and Hector Garcia-Molina. Semantic overlay networks for p2p systems. Technical report, Stanford University, 2002.
13. P. Druschel and A. Rowstron. Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc.of ACM SIGCOM*, 2001.
14. Fausto Giunchiglia, Uladzimir Kharkevich, and Sheak Rashed Haider Noori. P2P Concept Search: Some preliminary results. In *SemSearch2009 workshop at WWW*, 2009.

15. Fausto Giunchiglia, Uladzimir Kharkevich, and Ilya Zaihrayeu. Concept search. In *Proc. of ESWC'09*, Lecture Notes in Computer Science. Springer, 2009.
16. Fausto Giunchiglia, Maurizio Marchese, and Ilya Zaihrayeu. Encoding classifications into lightweight ontologies. In *Journal on Data Semantics (JoDS) VIII*, Winter 2006.
17. Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. Discovering missing background knowledge in ontology matching. In *Proc. of ECAI*, 2006.
18. Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko. Semantic matching: Algorithms and implementation. *Journal on Data Semantics (JoDS)*, 9:1–38, 2007.
19. Fausto Giunchiglia and Ilya Zaihrayeu. Lightweight ontologies. In *The Encyclopedia of Database Systems*, 2008.
20. Fausto Giunchiglia, Ilya Zaihrayeu, and Uladzimir Kharkevich. Formalizing the get-specific document classification algorithm. In László Kovács, Norbert Fuhr, and Carlo Meghini, editors, *ECDL*, volume 4675 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 2007.
21. Peter Haase, Jeen Broekstra, Marc Ehrig, Maarten Menken, Peter Mika, Michal Plechawski, Pawel Pyszlak, Bjrn Schnizler, Ronny Siebes, Steffen Staab, and Christoph Tempich. Bibster - a semantics-based bibliographic peer-to-peer system. In *In Proceedings of the Third International Semantic Web Conference*, 2004.
22. Jinyang Li, Boon Thau, Loo Joseph, M. Hellerstein, and M. Frans Kaashoek. On the feasibility of peer-to-peer web indexing and search. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS 2003)*, 2003.
23. Toan Luu, Gleb Skobeltsyn, Fabius Klemm, Maroje Puh, Ivana Podnar Žarko, Martin Rajman, and Karl Aberer. AlvisP2P: scalable peer-to-peer text retrieval in a structured p2p network. In *Proc. VLDB Endow.*, 2008.
24. Wenhui Ma, Wenbin Fang, Gang Wang, and Jing Liu. Concept index for document retrieval with peer-to-peer network. In *Proc. SNPD '07*, 2007.
25. W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch. Edutella: A p2p networking infrastructure based on rdf. In *Proceedings to the Eleventh International World Wide Web Conference*, 2002.
26. Greg Pass, Abdur Chowdhury, and Cayley Torgeson. A picture of search. In *InfoScale'06: Proceedings of the 1st international conference on Scalable information systems*, New York, NY, USA, 2006. ACM.
27. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of SIGCOMM*, 2001.
28. John Risson and Tim Moors. Survey of research towards robust peer-to-peer networks: Search methods. *Computer Networks*, 50:3485–3521, 2006.
29. Kunwadee Spripanidkulchai, Bruce Maggs, and Hui Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *Proceedings of IEEE INFOCOM*, volume 3, pages 2166–2176, 2003.
30. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of SIGCOMM*, 2001.
31. Chunqiang Tang, Zhichen Xu, and Sandhya Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *Proceedings of ACM SIGCOMM*, pages 175–186, 2003.
32. P. Turney. Learning algorithms for keyphrase extraction. In *Information Retrieval*, volume 2, pages 303–336, 2000.
33. R. Wetzker, C. Zimmermann, and C. Bauckhage. Analyzing social bookmarking systems: A del. icio. us cookbook. In *Mining Social Data (MSoDa) Workshop Proceedings, ECAI*, pages 26–30, 2008.

34. H. Xiao and I. F. Cruz. Ontology-based query rewriting in peer-to-peer networks. In *Proceedings of the 2nd International Conference on Knowledge Engineering and Decision Support*, pages 11–18, 2006.
35. I. Zaihrayeu, L. Sun, F. Giunchiglia, W. Pan, Q. Ju, M. Chi, and X. Huang. From web directories to ontologies: Natural language processing challenges. In *6th International Semantic Web Conference (ISWC 2007)*. Springer, 2007.
36. B. Y. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, Computer Science Department, University of California, 2001.
37. Yingwu Zhu and Yiming Hu. Ess: Efficient semantic search on gnutella-like p2p system. Technical report, Department of ECECS, University of Cincinnati, 2004.
38. Hai Zhuge, Jie Liu, Liang Feng, Xiaoping Sun, and Chao He. Query routing in a peer-to-peer semantic link network. *Computational Intelligence*, 21:197–216, 2005.