P2P Concept Search: Some Preliminary Results*

Fausto Giunchiglia
Department of Information Engineering and Computer Science
University of Trento, Italy
fausto@disi.unitn.it

Uladzimir Kharkevich
Department of Information Engineering and Computer Science
University of Trento, Italy
kharkevi@disi.unitn.it

S.R.H Noori
Department of Information Engineering and Computer Science
University of Trento, Italy
noori@disi.unitn.it

ABSTRACT

Concept Search extends syntactic search, i.e., search based on the computation of string similarity between words, with semantic search, i.e., search based on the computation of semantic relations between complex concepts. It allows us to deal with ambiguity of natural language. P2P Concept Search extends Concept Search by allowing distributed semantic search over structured P2P network. The key idea is to exploit distributed background knowledge and indices.

1. INTRODUCTION

Concept Search (*CSearch*) [1] extends syntactic search with semantics. The main idea is to keep the same machinery which has made syntactic search so successful, but to modify it so that, whenever possible, syntactic search is substituted with semantic search, thus improving the system performance. As a special case, when no semantic information is available, *CSearch* reduces to syntactic search, i.e., the results produced by *CSearch* and syntactic search are the same. In this paper, we propose an approach called *P2P Concept Search* which extends *CSearch* allowing semantic search on top of distributed hash table (DHT) [2].

2. P2P CONCEPT SEARCH

P2P Concept Search extends *CSearch* in several dimensions. First, we extend the reasoning with respect to a single background knowledge (BK) $\mathcal{T}$ to the reasoning with respect to the BK $\mathcal{T}_{P2P}$ which is distributed among all the peers in the network. Second, we extend the centralized inverted index (II) to distributed index build on top of DHT.

$$CSearch \xrightarrow{Knowledge(\mathcal{T}\rightarrow\mathcal{T}_{P2P}),Index(II\rightarrow DHT)} P2P\ CSearch$$

2.1 Distributed Background Knowledge

To access the background knowledge $\mathcal{T}$, stored on a single peer, *CSearch* uses the following three methods:

getConcepts(W) returns a set of all the possible meanings (atomic concepts A) for word W. For example, *getConcepts(canine)* $\rightarrow$ {*canine-1 ('conical tooth'), canine-2 ('mammal with long muzzles')*}. Note that in this example, atomic concepts are represented as *lemma-sn*, where *lemma* is the lemma of the word, and *sn* is the sense number in *BK*.

getChildren(A) returns a set of all the more specific atomic concepts which are directly connected to the given atomic concept A in $\mathcal{T}$. For example, *getChildren(carnivore-1)* $\rightarrow$ {*canine-2, feline-1*}.

getParents(A) returns a set of all the more general atomic concepts which are directly connected to the atomic concept A in $\mathcal{T}$. For example, *getParents(dog-1)* $\rightarrow$ {*canine-2*}.

In order to provide access to background knowledge $\mathcal{T}_{P2P}$ distributed over all the peers in the P2P network, we create distributed background knowledge *DBK*. In *DBK*, each atomic concept A is represented as a 3-tuple: $A = \langle A_{ID}, POS, GLOSS\rangle$, where A_{ID} is a unique concept ID; POS is a part of speech; and $GLOSS$ is a natural language description of A. In the rest of the paper, for the sake of presentation, instead of complete representation $\langle A_{ID}, POS, GLOSS\rangle$ we use just *lemma-sn*. *DBK* is created on top of a DHT. Atomic concepts are indexed by words using the DHT 'put' operation, e.g., *put(canine,* {*canine-1, canine-2*}*)*. Moreover, every atomic concept is also indexed by related atomic concepts together with the corresponding relations ('$\sqsubseteq$' or '$\sqsupseteq$'). We use a modification of the DHT 'put' operation *put(A, B, Rel)*, which stores atomic concept B with relation *Rel* on the peer responsible for (a hash of) atomic concept A, e.g., *put(canine-2, dog-1, '$\sqsubseteq$')*, *put(canine-2, carnivore-1, '$\sqsupseteq$')*.

After *DBK* has been created, *getConcepts(W)* can be implemented by using the DHT 'get' operation, i.e., getConcepts(W) = get(W). Both methods *getChildren(A)* and *getParents(A)* are implemented by using a modified DHT 'get' operation *get(A, Rel)*, i.e., *getChildren(A) = get(W, '$\sqsubseteq$')* and *getParents(A) = get(W, '$\sqsupseteq$')*. The operation *get(A, Rel)* finds a peer responsible for concept A and retrieve only those atomic concepts B which are in relation *Rel* with A.

2.2 Indexing and Retrieval

In *CSearch*, we can search for documents describing complex concepts which are semantically related to complex concepts in the user query. We assume that, when a user is searching for a concept, she is also interested in more specific concepts.

*A long version of this paper is available at http://eprints.biblio.unitn.it/archive/00001585/

Formally a query answer $QA(C^q, \mathcal{T})$ is defined as follows:

$$QA(C^q, \mathcal{T}) = \{d \mid \exists C^d \in d, \text{s.t. } \mathcal{T} \models C^d \sqsubseteq C^q\} \quad (1)$$

where C^q is a complex query concept extracted from the query q, C^d is a document complex concept extracted from the document d, and $\mathcal{T}$ is a terminological knowledge base (i.e., the *BK*) which is used in order to check if $C^d \sqsubseteq C^q$.

The query answer defined in Equation 1, can be extended to the case of distributed search by taking into account that the document collection D_{P2P} is equivalent to the union of all the documents stored in the network and also that background knowledge $\mathcal{T}_{P2P}$ is distributed among peers.

$$QA(C^q, \mathcal{T}_{P2P}) = \{d \in D_{P2P} \mid \exists C^d \in d, \text{s.t. } \mathcal{T}_{P2P} \models C^d \sqsubseteq C^q\} \quad (2)$$

In *P2P CSearch*, complex concepts are computed in the same way as in *CSearch* (for more details see [1]). The only difference is that now if an atomic concept is not found in the local background knowledge $\mathcal{T}$, then $\mathcal{T}_{P2P}$ is queried instead. After complex concepts are computed, the indexing of documents is performed as follows. Every peer computes a set of atomic concepts A which appear in the representations of peer's documents. For every atomic concept A, the peer computes a set of documents d which contain A. For every pair $\langle A, d \rangle$, the peer computes a set $S(d, A)$ of all the complex document concepts C^d in d, which contain A.

$$S(d, A) = \{C^d \in d \mid A \in C^d\} \quad (3)$$

For every A, the peer sends document summaries corresponding to A, i.e., pairs $\langle d, S(d, A) \rangle$, to a peer p_A responsible for A in *DBK*. The peer p_A indexes these summaries using the local *CSearch*.

The query answer, defined in Equation 2, is computed by using a recursive algorithm described below. The algorithm takes as input complex query concept C^q and computes as output a query answer QA in five macro steps:

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

Step 2 For every conjunctive component $\sqcap A^q$ in C^q, p_I selects concept A in $\sqcap A^q$ with the smallest number of more specific atomic concepts. For every selected A, C^q is propagated to the peer p_A responsible for A.

Step 3 p_A receives the query concept C^q and locally (by using *CSearch*) computes a set of documents which belong to the query answer. The results are sent directly to p_I. On receiving new results, p_I merges them with QA. An (intermediate) result is shown to the user.

Step 4 p_A computes a set $\mathbf{C}_{ms}$ of all more specific atomic concepts B which are directly connected to the given atomic concept A in $\mathcal{T}_{P2P}$. $\mathbf{C}_{ms}$ is computed by querying locally stored (direct) more specific concepts.

Step 5 p_A propagates C^q to all the peers p_B responsible for concepts B in $\mathbf{C}_{ms}$, i.e., Step 2 is repeated on all p_B.

An example of how the query answer $QA(C^q, \mathcal{T}_{P2P}, A)$ is computed is given in Figure 1. Peers, represented as small circles, are organized in a DHT overlay, represented as a ring. A query consisting of a single query concept C^q = *little*-4 $\sqcap$ *canine*-2 is submitted to peer P_I. Let us assume that atomic concept *canine*-2 has smaller number of more specific atomic concepts then 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 1. $P_{canine\text{-}2}$ searches in a local *CSearch* index with C^q. No results are found. $P_{canine\text{-}2}$ collects all the atomic concepts which are more specific then *canine*-2, i.e., *dog*-1 and *wolf*-1. Query concept C^q is propagated to peers $P_{dog\text{-}1}$ and $P_{wolf\text{-}1}$. $P_{dog\text{-}1}$ 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_I, which presents it to the user. The results propagation is shown as a dash line in Figure 1. Both peers $P_{dog\text{-}1}$ and $P_{wolf\text{-}1}$ have no more specific concepts than *dog*-1 and *wolf*-1, therefore they do not propagate C^q to any other peers.

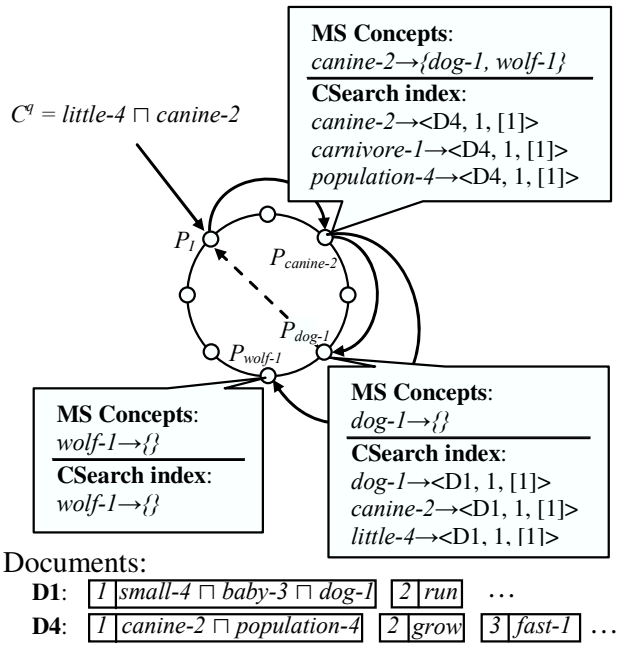

Figure 1: Query Answering

3. CONCLUSIONS

In this paper, we have presented an approach, called *P2P CSearch*, which allows for a semantic search on top of distributed hash table (DHT). *P2P CSerarch* addresses the scalability problem of *CSerarch* and the ambiguity problem of natural language in P2P syntactic search. Future work includes the development of document relevance metrics based on semantic similarity of query and document descriptions and evaluating the efficiency of the proposed solution.

4. REFERENCES

[1] Fausto Giunchiglia, Uladzimir Kharkevich, and Ilya Zaihrayeu. Concept search. In *Proc. of ESWC'09*, Lecture Notes in Computer Science. Springer, 2009.

[2] John Risson and Tim Moors. Survey of research towards robust peer-to-peer networks: Search methods. *Computer Networks*, 50:3485–3521, 2006.