Ontology-based Interpretation of Keywords for Semantic Search

Thanh Tran, Philipp Cimiano, Sebastian Rudolph and Rudi Studer

Institute AIFB, Universität Karlsruhe, Germany
{dtr,pci,sru,rst}@aifb.uni-karlsruhe.de

Abstract. Current information retrieval (IR) approaches do not formally capture the explicit meaning of a keyword query but provide a comfortable way for the user to specify information needs on the basis of keywords. Ontology-based approaches allow for sophisticated semantic search but impose a query syntax more difficult to handle. In this paper, we present an approach for translating keyword queries to DL conjunctive queries using background knowledge available in ontologies. We present an implementation which shows that this interpretation of keywords can then be used for both exploration of asserted knowledge and for a semantics-based declarative query answering process. We also present an evaluation of our system and a discussion of the limitations of the approach with respect to our underlying assumptions which directly points to issues for future work.

1 Introduction

Part of the Semantic Web vision is to provide web-scale access to semantically described content. In particular, this implies understanding users' information needs accurately enough to allow for retrieving a precise answer using semantic technologies. Currently, most web search engines are however based on purely statistical techniques. While they are not able to figure out the meaning of a query, they can provide answers by returning the statistically most appropriate answer to a user's query—based on some measures for computing similarity in vector space (cf. [1]). Information Retrieval (IR) techniques applied to the Web have gained a reasonable degree of maturity which is clearly corroborated by the success of search engines such as Google, Yahoo and the like. These search engines are in fact providing a baseline quite difficult to outperform. Due to the nature and the maturity of the underlying statistical techniques, they are more robust and scale to the size of the Web, as opposed to semantic technologies.

For restricted domains which can be formalized using ontologies, there is nevertheless hope that semantic technologies can be put into work to allow for more semantics-based search. One of the crucial steps within such an endeavor is to precisely capture the user's information need (see also [2]). But how does the user express his information need? If we look at the wide-spread usage of web search engines, we can conclude that users are definitely used to express their information need via simple queries based on keywords. However, while there is substantial recent work on interpreting full natural language questions semantically w.r.t. an ontology (cf. [3], [4]) or database schema [5], not as much work has been carried out with respect to the formal interpretation of keyword queries. A notable exception is the approach described in [6], which we discuss further in the related work section.

In this paper, we present an approach for interpreting keyword queries using background knowledge available in ontologies. Based on a few assumptions about how people describe their information needs, we present an approach which translates a keyword query into a DL conjunctive query which can be evaluated with respect to an underlying knowledge base (KB). The evaluation of our approach has been carried out on the KB of the semantic portal at `http://www.aifb.uni-karlsruhe.de/` and shows first promising results which we discuss w.r.t to our underlying assumptions. In addition, we present a system which shows how the interpretation of keywords can be used for a combination of intuitive exploration and search in KBs.

The paper is structured as follows: we begin with a discussion of related work in Section 2. Then, a generic approach for the interpretation of queries with background knowledge is presented in Section 3, followed by a detailed description of the translation of keyword queries to DL conjunctive queries in Section 4. Then in Section 5, we present the implementation of the approach as well as its evaluation. A discussion of the results points us directly to open issues for future work. We conclude in Section 6.

2 Related Work

Recently, substantial work has been performed on the translation of natural language questions to formal queries using an ontology or a database (cf. [5], [3], [4], [7]). While these approaches have been shown to yield remarkable results, it is not clear if users always want to specify a full natural language question. In fact, the success of commercial search engines shows that users are quite comfortable with using keywords. Thus, it seems important to also develop approaches which are able to interpret keywords such that they can be answered through a query to a database or a KB.

In this regard, there exists work on the translation of keywords to XML-based queries, e.g. to interpret keywords as X-Queries on XML data [8]. This is related to our approach because also the structure of (XML) elements is considered to interpret the relations among keywords. However the structure exploited there is less complex than the many relations among entities given by ontology axioms that we explore for our translation. Also, there has already been work on translating keywords to semantic queries. For instance, Royo et al. propose to map keywords to corresponding WordNet synsets [9]. While they claim to also be able to discover relations between keywords, it is not clear how this is achieved, especially given the fact that WordNet does not include any non-taxonomic relations besides part-of relations.

The approach closest to ours is the SemSearch approach presented by Lei et al. [6]. In fact, we agree with the analysis of Lei et al. that common approaches to semantic search are not particularly intuitive or user friendly as they either require posing formal (logical) queries or limit the expressive power of the user by using forms for example (compare the analysis of the semantic search state-of-the-art in [6]). Our approach is similar to SemSearch in the sense that we also aim at answering complex keyword queries by translating them into a logical query. However, our approach mainly differs in the way the query is computed. In SemSearch, the keywords are first interpreted as either instances, concepts or properties, respectively, which yields nine possible templates to be instantiated for the case of queries consisting of two keywords. Templates in fact fix the structure of the resulting query a priori, i.e. it is assumed that entities denoted by keywords can be connected through a direct relation in the ontology. As queries with more than two keywords lead to a combinatorial explosion of the different possible combinations of entities, and thus would require a large number of templates, some heuristics are suggested to handle these complex queries (see [6] for more details). In contrast to SemSearch, we build on a more generic graph-based approach to explore the connections between the entities in the query. Our approach does not fix the structure of the queries in the form of templates a priori and does not assume the availability of direct connections between entities. In fact, the vicinity of the entities that is to be explored is based on a variable d, which can be set by the user. Within this range, many possibly indirect connections might be discovered and used for the generation of the formal query.

3 Answering User Queries in Ontology-based Systems

In this section, we present an abstract framework describing the process of ontology-based IR, where the user poses a question to the system and the system answers the question using knowledge formalized in a logical language. In particular, we focus on scenarios where the language of the user question does not match the query language supported by the system. For this purpose, we define our ontology-based IR process as consisting of four models and describe the assumptions underlying our approach. We then present a generic approach for translating a user question into a formal system query.

3.1 Models in Ontology-based Information Retrieval

In line with models in classical IR, namely the query and resource model [1], we discuss four different models involved in ontology-based IR.

The Mental Model O_U: The mental model O_U corresponds to the *information need* that a user has in mind at the beginning of an IR task. Since the concrete mechanisms underlying human thought are far from completely understood, for the sake of the approach presented in this paper we postulate only very abstract properties of this model: O_U can be conceived as a set of (thought) entities that are relevant for the current information need and embedded in an association structure. These entities might be related to real world objects or to more abstract concepts. The entities in this association structure can be conceived as what the user knows. We assume that the user is looking for (some of the) entities missing in this structure, which we refer to as *gaps*.

The User Question Model Q_U: The user question model Q_U consists of elements, which in turn are constructed out of language primitives $\mathcal{P}_U$ of a language $\mathcal{L}_U$ (language of the user). This model is the result of the user translating elements in O_U to elements in $\mathcal{P}_U$. Moreover (depending on the expressive means of $\mathcal{L}_U$), there might be also elements in Q_U explicitly denoting gaps (like, e.g. question words in a natural language). Naturally, Q_U must not be empty.

The System Resource Model O_S: This model consists of elements constructed out of language primitives $\mathcal{P}_S$ of a formal KR language $\mathcal{L}_S$ (language of the system).

Independent from a concrete formal language used, these elements can be conceived as a set of entities of a given ontology. As opposed to the abstract mental model O_U, the entities and structure of O_S are explicitly given and directly accessible. These elements constitute the knowledge (the KB) the system uses to answer the user question.

The System Query Model Q_S: This model represents the final question processed by the formal query engine of the system. It consists of elements constructed out of language primitives $\mathcal{P}'_S$ of a query language $\mathcal{L}'_S$. When there is a formal semantics for $\mathcal{L}'_S$ (query language of the system) it must be compatible with the semantics of $\mathcal{L}_S$ for the query Q_S to be processable by the system. In particular, some elements in Q_S must correspond to elements in O_S. In fact, formal queries in many systems are specified using ontology elements of the underlying KR language. However, the query language $\mathcal{L}'_S$ may have primitives additional to the ones available in $\mathcal{L}_S$. In particular, there must be primitives to specify the gaps, e.g. variables.

Note the correspondence of these models and the consequences for ontology-based IR: The more the entities and structure in O_U match the entities and structure in O_S, the higher the chance that O_S can be used by the system to fill the gaps, i.e. to answer the query. Also, the more related the syntax and semantics of $\mathcal{L}_U$ and $\mathcal{L}_S$, the more straightforward is the mapping from Q_U to Q_S, i.e., the interpretation of the user query. Yet, in the following, we will restrict our attention to scenarios where the query language of the user Q_U and the language of the system Q_S differ considerably and propose to use an ontology-based system to interpret and answer the user question.

3.2 A Generic Approach for Ontology-based Query Interpretation

In this section, we are not concerned with the actual answering step where the query engine processes the system query. Instead, we present a generic approach to deal with the preceding step, namely translating the user question to the system query. Similar to query processing, we propose an approach which relies on the knowledge in the KB for question interpretation. We will start with the clarification of our assumptions before the presentation of our approach.

Assumption (A1) — Ontology-Mental Correspondence: This assumption requires both an entity-wise and a structural correspondence between the mental model O_U and the system resource model O_S. That is, elements and the associative structure in O_U correspond to ontology entities and the structure in O_S, respectively.

Assumption (A2) — Locality of Information Need: This assumption requires those ontology entities $O'_S \subseteq O_S$ that correspond to entities in the mental information need representation O_U to be connected over a maximum distance d. That is, for any two ontology entities $a, b \in O'_S$ there has to be a direct connection $\langle a, b \rangle$ or a sequence of x_i such that $a = x_0$ and $\langle x_0, x_1 \rangle, \langle x_1, x_2 \rangle, \ldots, \langle x_{n-1}, x_n \rangle$, and $\langle x_n, x_b \rangle$ and $n < d$. There might be several such sequences that connect gaps with the two entities the user knows (a, b). In such cases, we assume not only that there is a maximum distance but moreover that connections over smaller distances are more likely to contain the gaps that the user looks for.

The above assumptions are certainly too strict in the sense that users can not be assumed to fully think in term of ontological structures or in any KR language. However, we need to assume that they think in some structures which can be mapped to an ontology. Otherwise, a system would have no chance in interpreting and answering a user's query. In this sense, our assumptions seem justified from a practical point of view. Thus, if there is no such correspondence, the system cannot fill the user gaps, i.e. answer the query. In addition, A2 helps to restrict attention to only a particular part of the ontology, as discussed in our approach presented in the following.

Interpreting the User Question: We present a generic approach to translate Q_U to Q_S which consists of three high level steps. First, the elements in the user question Q_U are mapped to ontology elements from O_S (Step 1). Then, further ontology elements are explored to better cover the initial information need in the mental model O_U (Step 2). Finally, from this more refined ontological representation of the need, the query Q_S will be derived (Step 3).

In step one, we make use of the correspondence stated in A1 and map elements of the user queries Q_U to ontology elements $O'_S \subseteq O_S$. Note that the user question may only partially capture the mental model. Also, not all elements of the user question can be mapped to corresponding ontology elements. Therefore, the identified ontology elements O'_S yet do not account for the entire mental model. Since we want in some way to "reconstruct" the mental model and find out the gaps, further computation is required in these cases to find missing elements.

In step two, the assumption on the locality of information need (A2) is used to explore connections among O'_S identified in step one using further elements in O_S. Due to A2, only elements in O_S that are connected with the identified elements O'_S within a specified range (maximum distance d) have to be considered in the exploration. From this it also follows that after all the neighboring elements in this range have been explored for all O'_S, the discovered elements in O_S combined with O'_S can be assumed to approximate the user's mental model.

After reconstructing this mental model, the identified and discovered ontology elements need to be assembled into a formal query in the language $\mathcal{L}'_S$. The discussion on Q_S already pointed out that formal queries are specified using ontology elements (the *information part*). Additionally, they contain variables (the *question part*). As opposed to Q_S, O_S does not contain variables, and thus, the identified elements O'_S map to the information part. Note that in the exploration step, the discovered elements may correspond to thought entities the user knows but has not explicitly specified in the question. Also, they might correspond to gaps, i.e. entities the user does not know and out of which only some might be interesting to him/her. While all the others map to the information part, the elements corresponding to the answer the user looks for map to variables of the question part.

Illustrating Example: We illustrate our approach with a simple example as depicted in Figure 1, where a user wants to retrieve all publications authored by Philipp Cimiano which are associated to the project X-Media. Let's assume that the user, on the basis of his information need, issues the query Q_U ="Philipp Cimiano X-Media publications".

The elements in the query are then mapped to the ontology elements `Philipp Cimiano`, `X-Media` and `publication` respectively. These elements, however, yet do not fully correspond to the information need of the user. Also, they still cannot be assembled into a system query that yields answers the user looks for due to missing

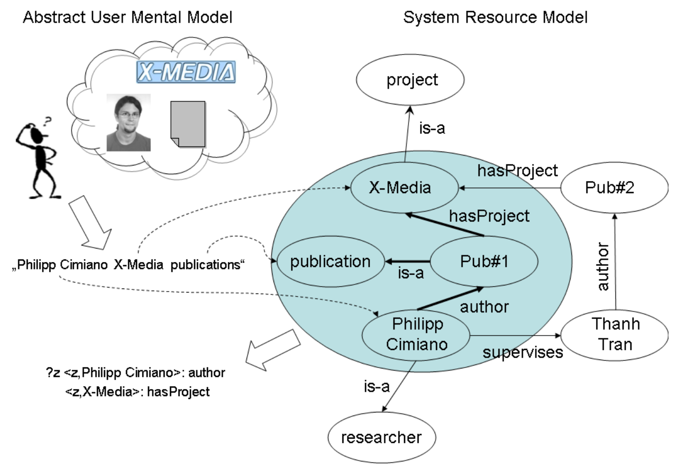

Fig. 1. Workflow for the query "Philipp Cimiano X-Media publications"

elements. These missing elements correspond to the entities in the mental model the user knows but does not specify such as the connection `is-a` between `X-Media` and `project`. In particular, the user does not make explicit the relation between `Philipp Cimiano` and `publication` connected in the ontology via the `author` relation. These missing elements correspond to what we call gaps, i.e. elements in the mental model that the user does not make explicit when specifying his/her information need. In our case, the user does not make explicit the connection `hasProject` between `X-Media` and `publication` while for sure s/he was thinking of it. Some of these gaps correspond to the information the user is looking for, i.e. `Pub#1` in our example. All these missing elements need to be made explicit in our translation into a formal query.

For this purpose, in step 2, our approach starts the KB exploration from the individual `Philipp Cimiano` and leads to the relations `author` and `is-a`. Assuming the exploration width is 2, we also reach the elements `Pub#1` and `researcher` from `Philipp Cimiano`. From the other elements in the query, i.e. `X-Media` and `publication`, we reach the relations `hasProject`, `is-a` as well as the elements `project`, `Pub#1` and `Pub#2` from `X-Media`, and `is-a` and `Pub#1` from `publication`. This shows how step-by-step the exploration builds up a graph where all elements of the initial user query are connected.

In step 3, the (possibly many) subgraphs which connect these elements are computed. These subgraphs correspond to the different questions the user possibly has. As highlighted in the circle in Figure 1, in our specific example there is only one such subgraph. However, in other scenarios, and in particular if the exploration range *d* is set higher, we are likely to obtain several such subgraphs. In such cases, A2 allows to rank queries, since it postulates that connections over smaller distances are more likely to contain the answer the looks for. Finally, the graph is translated into a corresponding query, e.g. $Q_S = \langle \texttt{x,Philipp Cimiano}\rangle\texttt{:name} \wedge \langle \texttt{x, y}\rangle\texttt{:author} \wedge \langle \texttt{y,z}\rangle\texttt{:hasProject} \wedge \langle \texttt{z,X-Media}\rangle\texttt{:name} \wedge \langle \texttt{y:publication}\rangle$. While the previous steps are rather generic, the mapping from the graph elements to the information part and question part of the system queries depends on the query syntax of Q_S as well as the specific elements identified and explored in O_S.

This simple example demonstrates the high level steps as captured in the generic approach. It shall facilitate comprehension of more technical details of a procedure we propose for the specific translation of keyword queries to DL conjunctive queries presented in the following section.

4 Interpretation of Keywords Using DL Knowledge Bases

In this section, we present an instantiation of the generic approach described above to two specific languages Q_U and Q_S. Q_U is grounded to keyword queries, i.e. $Q_U = (k_1, k_2, ..., k_n)$ where the k_i's stand for keyword and represent the primitives $\mathcal{P}_U$. The language of the user $\mathcal{L}_U$ then simply consists in concatenations of the elements in $\mathcal{L}_U$. Thus, by keyword queries, we mean the standard type of queries supported by Google-style interfaces like the ones discussed in [10]. Further, Q_S is grounded to DL conjunctive queries. Such a query is defined as a conjunction of terms of the form $x : C$ or $\langle x, y \rangle : R$, where C is a concept, R is a role, and x, y are variables or individuals taken from $\mathcal{V}$ a set of variable names, or $\mathcal{I}$ a set of individual names. If we conceive the variables as individuals, these terms are assertional statements of a DL language, where the first kind is referred to as concept terms and the latter kind is called role terms.

For the translation of keyword queries to DL conjunctive queries, we make use of O_S, a KB containing knowledge formalized in the form of DL axioms. In particular, the description logic in our approach is $\mathcal{SHOIN}(\mathbf{D})$, the DL counterpart to OWL DL, such that, in addition to individuals and variables in query terms, we also have $j : D$, where j are data values taken from the set of values $\mathcal{J}$ and $D \in \mathcal{D}$ is the set of data ranges. Moreover, roles can be further divided into abstract roles (object properties) R and concrete roles (datatype properties) U such that possible terms occurring in a conjunctive query have the shape $x : C$, $j : D$, $\langle x, y \rangle : R$ and $\langle x, j \rangle : U$.

Before the detailed presentation of the approach, we discuss the specialization of A1 to the particular setting described above, i.e. the correspondence of the mental model and the DL knowledge base.

Assumption 1' (A1') We assume that users' mental models are organized in a way similar to DL knowledge bases. More precisely, this means that the thought entities of the mental model O_U correspond to $\mathcal{SHOIN}(\mathbf{D})$ ontology entities in the disjoint union of the sets $\mathcal{I}$ (individuals), $\mathcal{J}$ (data values), $\mathcal{C}$ (concepts), $\mathcal{D}$ (data ranges), $\mathcal{R}$ (object properties), and $\mathcal{U}$ (data properties) and the associations in O_U correspond to associative interconnections of the types $\langle i, C \rangle$, $\langle i_1, R, i_2 \rangle$ and $\langle i, U, j \rangle$ where $i, i_1, i_2 \in \mathcal{I}$, $j \in \mathcal{J}$, $C \in \mathcal{C}$, $R \in \mathcal{R}$, and $U \in \mathcal{U}$. As given by the $\mathcal{SHOIN}(\mathbf{D})$ syntax, such connections are specified using the DL-axioms $i \in C$ (concept membership), $\langle i_1, i_2 \rangle \in R$ (object property membership) and $\langle i, j \rangle \in U$ (data property membership).

Note that when compared to A1, A1' imposes stricter structural properties on the mental model. Namely, its structure is frame-based in the sense that elements of the mental model correspond to the entities and relations of a DL A-Box. We think that as the frame-based nature of DL seems to be an intuitive formalism to describe knowledge, it might be also an intuitive way for users to think about (and to describe) the knowledge they are looking for. In what follows we describe the various steps of the concrete instantiation of the generic approach in more details.

4.1 Step One — Mapping Terms to KB Entities

Due to A1', we assume that keywords are mapped to ontology entities, namely individuals, data values, concepts, data ranges, as well as object and data properties. In particular, the mapping can be defined as a function f which maps elements of the user question model Q_U to entities of system resource model O_S, i.e. $f : Q_U \rightarrow O_S$. For practical purposes, it is crucial that this function is "robust" in the sense that it also considers syntactic and spelling variants.

Using the query engine, entities in the KB can be retrieved via their URIs. In particular, f can be implemented as a retrieval operation performed by the engine, e.g. simply by passing the URI as input to the repository API. In order to cope with syntactic and spelling variants, Lucene[1] is actually used as the index and search engine. That is, URIs and labels of entities are indexed, and using the fuzzy search feature of Lucene, a query is generated for each entered keyword. The engine returns ontology entities ranked according to syntactic similarity to the respective keyword. As there is only one minor syntactic difference in the example from the last section, the highest ranked entities for Q_U ="Philipp Cimiano X-Media publications" are indeed `Philipp Cimiano, X-Media` and `publication`. However, in other scenarios, this implementation of f based on syntactic similarity may not always find an appropriate mapping for each keyword. These mapped entities $O'_S := \{f(k_i)|Q_U = \langle k_0, ..., k_n\rangle\}$ will then be fed into the exploration step, which we will discuss in the following.

4.2 Step Two — Exploring Connections among KB Entities

Due to A1', we can restrict ourselves to the exploration of connections of the type $\langle i, C\rangle$, $\langle i_1, R, i_2\rangle$ and $\langle i, U, j\rangle$. Using these concept and property member axioms, we explore all ontology entities related to elements O'_S identified in step one according to the algorithm shown in Fig. 2.

Basically, the exploration encompasses the traversal to neighbors from each of the elements in O'_S. Then, depending on the type of the particular element $e \in O'_S$, different traversals are performed to build a graph connecting e with all the neighbors within the specified range d. For instance, given a concept, all individuals are retrieved via concept member axioms. Given a property, property member axioms are used to navigate to individuals and data values, respectively. Figure 3, for example, shows the pseudocode algorithm for the recursive traversal from a particular individual to its neighboring concepts, individuals and data values. Neighboring individuals and data values are retrieved using property member axioms. The value of d is reduced by one in each recursion step to ensure that this traversal is limited to a certain range. Note that, due to marking elements of O'_S globally as visited, any element of O'_S is traversed at most once. In the end, we obtain a graph g containing all entities out of O_S which have a graph-distance

[1] see http://lucene.apache.org/java/docs/

not greater than d to at least one of the elements of O'_S. We call this structure the d-neighborhood of O'_S. Possibly, if d is small, it might be the case that the computed graph is not connected.

```
KB Exploration($O'_S, d$)
1  INPUT a set of entities $O'_S$ matching the terms and the traversal width d
2  OUTPUT the graph containing all or some of $O'_S$
3  Intitialize new empty graph g
4  for $e \in O'_S$
5  do if e is a concept
6        then for all i being instances of e
7              do I-P-I Traversal($e, d, g$)
8        else if e is an object property
9                then for all $i, j$ with $\langle i, e, j\rangle \in O_S$
10                     do I-P-I Traversal($i, d, g$)
11                        I-P-I Traversal($j, d, g$)
12       else if e is a data property
13               then for all $i, j$ with $\langle i, e, j\rangle \in O_S$
14                    do J-P-I Traversal($j, d, g$)
15       else if e is an individual
16               then I-P-I Traversal($e, d, g$)
17       else if e is a data value
18               then J-P-I Traversal($e, d, g$)
19 return g
```

Fig. 2. KB Exploration algorithm

Note that the exploration simply incorporates all elements within a certain range. Thus, some discovered elements may not really be needed to connect elements in O'_S. Therefore, from this graph, only those paths are selected where the first and the last vertex correspond to an element in O'_S. In particular, a modified version of the depth first search (DFS) procedure over graphs is used for computing all paths $p \in P$ for each possible pair $(a, b) \in O'_S$ such that $p = (v_1, e_1, ..., e_n, v_n)$, where v_1 is constructed using a and v_n is constructed using b and none of the vertices is visited more than once. These paths are fed into the next step.

4.3 Step Three — Deriving DL Conjunctive Queries from Connections

This step comprises three substeps. First, all different subsets of paths (called *connections*) are computed from P discovered previously. Then, for each subset, a query is derived. Finally, the resulting queries are ranked. The three substeps are described in the following:

Computing Possible Connections: A question can be derived when all elements O'_S identified in step one are connected. When merging all the paths P computed in step two, we however obtain a graph which may contain many different subgraphs connecting all the elements O'_S. Hence, it is a priori not clear which subgraph to choose as the correct interpretation of the keyword-based query. Therefore, we first compute all

```
I-P-I TRAVERSAL($i, d, g$)
 1  INPUT the individual i to be explored, the traversal range d, and the intermediate graph g
 2  OUTPUT updated graph g containing entities connected to i within the range d
 3  if  i not marked as visited and $d > 0$
 4    then
 5          mark i as visited within $O_S$
 6          $C_i := \{c \mid i$ instance of $c\}$
 7          add edge $(i, type, c)$ to g for all $c \in C_i$
 8          $P := \{(i, p, j) \mid \langle i, p, j\rangle \in O_S\}$
 9          for all $(i, p, j) \in P$
10          do if  j not marked as visited in $O_S$
11               then add a new edge $(i, p, j)$ to g
12                    if j is an individual
13                      then I-P-I TRAVERSAL($j, d - 1, g$)
14                      else  J-P-I TRAVERSAL($j, d - 1, g$)
```

Fig. 3. Individual-to-Individual traversal algorithm

these subgraphs and rank these at a second step. The subgraphs connecting the elements in O'_S are calculated by the recursive procedure shown in the pseudocode algorithm in Figure 4. The input to the algorithm is the set of paths P as computed previously as well as an initially empty set $R \subseteq O_S$ of vertices which have not yet been assembled into a graph connecting all the vertices in O_S as well as a subset C of already connected vertices. The recursion starts by selecting some edge connecting two arbitrary vertices and enters further recursions to add additional vertices. In this way, all the possible tree-shaped subgraphs connecting elements in O_S are determined.

Mapping Connections to Queries: Each of these connection graphs G_C are then translated to a corresponding DL conjunctive query Q_S as follows: an edge in G_C of the form $type(v_i, v_c)$ (representing the connection $\langle i, C\rangle$) is mapped to concept terms of the form $\langle \mathtt{x:C}\rangle$, where v_i is a vertex constructed using an individual, v_c is constructed using a concept, and x is an individual or a variable. The concept of v_c is used as concept of the term. When the individual of v_i matches some $e \in O'_S$, then it is used as constant, otherwise a variable is used for the term. As the same individual might be used in many edges, the same variable must be used for the same individual. Besides concept member axioms, also property member axioms are used to connect entities in the exploration. Edges constructed with these axioms are of the form $property_n(v_i, v_j)$, where v_i (v_j) is constructed either using an individual or a data value (the connections $\langle i_1, R, i_2\rangle$ and $\langle i, U, j\rangle$). These edges map to role terms of the form $\langle \mathtt{x:y}\rangle$: $\mathtt{R}$, where a vertex constructed using an individual is mapped to a variable or constant just as described above. When v_i (v_j) is constructed using a data value, it is simply mapped to constants of the role term. As the exploration incorporates only these two types of edges, this mapping is thus complete for the translation from G_C to Q_S. In our example, only one connection graph with the edges $name(uri1, Philipp\ Cimiano)$, $author(uri1, pub\#1)$, $hasProject(pub\#1, uri2)$, $name(uri2, X-Media)$, $type(pub\#1, publication)$ exists. Using the above specified mapping, this connection graph is translated to the final query: $Q_S = \langle \mathtt{x,Philipp\ Cimiano}\rangle\mathtt{:name} \wedge \langle \mathtt{x,\ y}\rangle\mathtt{:author} \wedge \langle \mathtt{y,z}\rangle\mathtt{:hasProject} \wedge \langle \mathtt{z,X\text{-}Media}\rangle\mathtt{:name} \wedge \langle \mathtt{y:publication}\rangle$.

```
CalculateSubGraphs($P, C, R, G, g$)
INPUT the paths $P$ calculated by DFS for all matching vertices $O'_S$
OUTPUT all different subgraphs connecting the vertices in $O'_S$
if $R = \emptyset$
  then $G = G \cup g$
if $g = \emptyset$
  then $G = newGraph$
       for $\{i, j\} \subseteq R$
       do for each path p between i and j (as calculated by DFS)
          do add (i,p,j) to G
             CalculateSubGraphs($P \setminus p, C \cup \{i, j\}, R \setminus \{i, j\}, G$)
  else for $i \in R$
       do for $j \in C$
          do for for each path p between i and j
             do
                add (i,p,j) to G
                CalculateSubGraphs($P \setminus p, C \cup \{i\}, R \setminus \{i\}, G$)
```

Fig. 4. Algorithm for Computing Connections

Rank Queries: Finally, the computed subgraphs have to be ranked. From A2, it follows that the smaller the length of the paths connecting the elements O'_S, the more likely they match the initial question in the mental model of the user. Thus, queries are ranked by the length of the longest path of the respective connection graph.

5 Ontology-based Search and Exploration with Keywords

In this section, we discuss our implementation of the approach and show how it can be incorporated into a system for exploring and searching KBs. The system is evaluated and results are discussed in the last section in the light of the underlying assumptions.

5.1 Implementation

The presented approach for the interpretation of keywords with respect to a given ontology is integrated in our system called XXploreKnow!, which has been designed to support a combination of search and exploration in knowledge bases. A detailed description of this system will be published elsewhere. We will now describe a possible interaction of a user with XXploreKnow!.

At the beginning, the user enters keywords which are processed by the Lucene search engine. Ontology entities returned by this engine enter the exploration process, in which neighboring entities up to a width of d are considered[2]. As a result, the system visualizes a subgraph connecting the matched entities to the user and highlights the entities matching the keywords. Depending on the action performed by the user, e.g. clicking on the “search" or “xxplore" button, subsequent interactions consist of either further exploration of the graph or inspection of the search results. With “xxplore", the

[2] Currently, the parameter d must be configured in the implementation. It has been set to 3 in our experiments

user can expand nodes shown in the graph-based visualization to traverse to neighboring concepts and individuals connected via properties—as captured by concept restrictions and property member axioms. In addition, from an individual, the user can navigate to its types, and then along the concept hierarchy as specified by subclass axioms. By default, only assertional knowledge is retrieved (in order to keep the browsing performant) and shown in the visualization. During this exploration, the user can drag and drop elements from the visualization to the query view below the keywords to further refine the query. With “search", the user’s query is sent to the inference engine. In this case, the different possible queries are ranked as described in previous sections and presented to the user, who can choose among different queries. The results, which may contain also inferred facts, are then finally shown to the user in a separate view.

5.2 Evaluation

In order to carry out an evaluation of the system, we have asked colleagues at the institute AIFB to provide queries in the way they would interact with a system capable of processing keyword based queries, along with the natural language description of the query. The request was sent by e-mail and 12 people responded. Some queries which were obviously out of the scope of the knowledge base were removed, resulting in a total of 42 different queries. These queries were incorporated only as an evaluation set and not used for the development or tuning of the approach. Examples for queries with different number of keywords posed by our users are: *"projects Blohm"* (Retrieve all projects that Sebastian Blohm is working on), *"phone Rudi Studer"* (Retrieve the phone number of Rudi Studer) or *"publications SmartWeb Pascal Hitzler 2002"* (Retrieve all publications published by Pascal Hitzler within SmartWeb in 2002). For the evaluation, one of the authors manually assigned conjunctive queries according to the natural language description. A query generated by our approach is regarded as correct if it retrieved the same answers as the hand crafted query. In line with work on question answering ([5],[4]), we evaluate the approach in terms of precision, recall and F-Measure. Precision P is defined as the number of correctly translated keyword queries (based on equivalence of results) divided by the number of cases for which the system was able to construct a query. Recall R is defined as the number of correctly translated keyword queries divided by all the keyword queries of the evaluation set, i.e. 42 in our case. The $F_1 = \frac{2*P*R}{P+R}$ measure is then the harmonic mean between precision and recall.

In case the query is selected by hand from the different queries generated, our system obtains a precision P = 85%, a recall R = 52% and a F-Measure F_1 = 64%. In case we automatically choose the highest ranked question instead, the results are slightly lower with a precision of P = 69%, a recall of R = 43% and an F-Measure of F_1 = 53%.

5.3 Discussion

Our evaluation has been performed with the knowledge base from which our institute portal is automatically generated[3]. The underlying ontology is the SWRC ontology [11], which allows for the representation of researchers, their publications, active projects etc.

[3] see `http://www.aifb.uni-karlsruhe.de/`

The evaluation has been carried out involving our colleagues, who visit and update the portal pages frequently, but do not know the underlying ontology in detail. Thus, some of the queries the users in our experiments asked the system contain keywords which do not correspond to entities in the knowledge base. Obviously, this is a problem for our approach as it violates assumptions A1' as well as the generic A1. According to our recall measure (43-52%), at least about half of the keyword queries fulfill assumption A1' and can thus be mapped to appropriate ontology elements. The higher precision of 69-85% on the other hand shows that, given that A1' is fulfilled, the generated query is correct in most cases. In fact, we found that most of the errors in our approach are produced in step 1. This means that the Lucene engine does not return the appropriate ontology elements in some cases. This problem could be for example addressed by integrating additional lexical knowledge about words as found in resources such as WordNet [12].

A further issue is related to our assumption A2, i.e. the assumption that the ontology entities the keywords map to are connected via paths of up to a given length *d*. We have experimented with a length *d* of 3 in our approach. Possibly, a higher recall could be achieved by using a higher value for *d*, but it is also probable that much more "background noise" would be introduced, thus making the selection of the relevant query more difficult. Overall, our assumptions have proved to be very valuable. Our first assumption (A1 / A1') states that users conceptualize their information need in terms compatible with the underlying ontology. While such an assumption is quite simplistic on the one hand and rather strict on the other, it turned out to be necessary as questions which do not fulfill this assumption are anyway out of the conceptual range of the system. From a practical point of view, this assumption is thus necessary. Assumption A2, which assumes that the ontology elements are connected with paths of a maximal length turned out to be crucial in order to restrict the search space to a specific part of the KB.

6 Conclusions

We have presented a generic approach for mapping queries in a user language into an expressive logical language. In particular, we have presented a particular instantiation of our generic approach which translates keyword queries into DL conjunctive queries using knowledge available in the KB. We have clarified in particular the assumptions on which our approach is based on. We have presented the current implementation of the system as well as first results of an evaluation of the translation process. The evaluation shows promising results w.r.t. precision, but still a lower recall which can be definitely increased by integrating lexical knowledge into the process of matching keywords to ontology elements. In the light of these evaluation results, we have argued that our assumptions are indeed reasonable and necessary for the interpretation and the answering of queries using ontologies.

Besides the integration of lexical knowledge to improve recall, we intend also to improve the runtime performance of our approach. will focus future work on boosting the performance. So far, the process of interpretation so relies mainly on assertional knowledge, resulting in a large number of A-Box queries that need to be processed during the exploration. We plan to exploit the available T-Box knowledge for a “guided exploration" of the connections between ontology entities to reduce the number of A-Box queries.

One major problem our approach suffers from is the fact that it does not consider that keywords can be ambiguous with respect to labels in the ontology and simply considers the first matching ontology element to start the exploration. Currently, in case of ambiguities, the exploration would have to be performed for each of the possible interpretations of a query term. However, the alternatives to explored might be exponential in the number of possible interpretations of the keywords. Future work will thus aim at a more appropriate treatment of ambiguities.

Finally, we will further develop the presented system to support an integrated approach for combined search and exploration in knowledge bases.

Acknowledgements The research presented in this paper was partially supported by the European Commission in the context of the X-Media (IST-2006- 026978) and the NeOn (IST-2006-027595) projects.

References

1. Baeza-Yates, R., Ribeiro-Neto, B.: Modern information retrieval. Addison-Wesley (1999)
2. Stojanovic, N., Studer, R., Stojanovic, L.: An approach for step-by-step query refinement in the ontology-based information retrieval. In: Proceedings of the IEEE/ACM Web Intelligence Conference (WI). (2004) 120–137
3. Lopez, V., Pasin, M., Motta, E.: Aqualog: An ontology-portable question answering system for the semantic web. In: Proceedings of the European Semantic Web Conference (ESWC). (2005) 546–562
4. Cimiano, P., Haase, P., Heizmann, J.: Porting natural language interfaces between domains – a case study with the ORAKEL system –. In: Proceedings of the International Conference on Intelligent User Interfaces (IUI). (2007) 180–189
5. Popescu, A., Etzioni, O., Kautz, H.: Towards a theory of natural language interfaces to databases. In: Proceedings of the International Conference on Intelligent User Interfaces (IUI'03). (2003) 149–157
6. Lei, Y., Uren, V., Motta, E.: Semsearch: A search engine for the semantic web. In: Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW). (2006) 238–45
7. Bernstein, A., Kaufmann, E.: GINO – a guided input natural language ontology editor. In: Proceedings of the International Semantic Web Conference (ISWC). (2006) 144–157
8. Li, Y., Yang, H., Jagadish, H.: NaLIX: An interactive natural language interface for querying xml. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, ACM Press (2005) 900–902
9. Royo, J., Mena, E., Bernard, J., Illarramendi, A.: Searching the web: From keywords to semantic queries. In: Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05), IEEE Computer Society (2005) 244–249
10. Clarke, C., Cormack, G., Tudhope, E.: Relevance ranking for one to three term queries. Information Processing and Management **36** (2000) 291–311
11. Sure, Y., Bloehdorn, S., Haase, P., Hartmann, J., Oberle, D.: The swrc ontology - semantic web for research communities. (In: Proceedings of the 12th Portuguese Conference on Artificial Intelligence (EPIA 2005)) 218–231
12. Fellbaum, C.: WordNet, an electronic lexical database. MIT Press (1998)