Two-Level Smart Search Engine Using Ontology-Based Semantic Reasoning

Haiping Xu and Arturo W. Li
Computer and Information Science Department
University of Massachusetts Dartmouth, North Dartmouth, MA 02747, USA
{hxu, ali}@umassd.edu

***Abstract*—Traditional search engines are typically keyword-based, which cannot understand the semantics of a query or relationships among queried concepts, causing much of the related information to be absent from the search results. As opposed to traditional keyword-based search engines, in this paper, we introduce an approach to developing a smart search engine using ontology-based semantic reasoning. The search engine can understand the semantics of a concept or multiple concepts entered by a user, and classify and reason about the concepts within a knowledge domain specified using ontology. The search engine consists of two levels, namely the semantic reasoning level and the traditional keyword searching level. The semantic reasoning level supports reasoning about subsumption, supersumption, semantic equivalency and property relationships among concepts. Once new concepts related to the queried concepts have been inferred from the ontology, they can be further matched using a traditional keyword-based search mechanism. To demonstrate the feasibility of our approach, we adopt the domain of computer science courses for semantic search, and show that our approach may produce more relevant search results to user queries.**

Keywords-Search engine; smart search; semantic reasoning; ontology; domain knowledge.

I. INTRODUCTION

The growth of Internet has made search engines one of the most frequently used web applications over the past decades. Search engines can be used to search information on the web, including documents, images, audios, videos, and related web pages. Traditional search engines such as Google search engine, typically use the keyword-matching approach and ranking algorithms to retrieve the desired information for user queries. They have been quite successful by providing users simple search interface and useful search results. However, since traditional search engines cannot understand the semantics of a query or relations among queried concepts entered by a user, they often offer low recall and precision, with much of the related information being absent from the search results or too many irrelevant and ambiguous results being presented [1],[2]. For example, the word "Football" may refer to the sport played in North America with an oval ball or the Olympic sport played with a spherical shape ball known as soccer in America. When a user types the query "Football Leagues" into a traditional search engine, expecting some search results related to the Olympic sport, the highly ranked search results are unfortunately all about American football leagues with no relation to the Olympic sport. In order to return desirable results (e.g., the European Football League) for this user, it is required that the search engine can clearly understand the ambiguous meaning of the word "Football." In another example, suppose a computer science student wants to search for information related to a concept called "pushdown automaton." Unfortunately, the student cannot remember this terminology, though he knows that such a model contains some states and a stack. Using semantic search, when the student types a pair of concepts "State" and "Stack" within the domain of computer science courses, the reasoner should efficiently infer the concept of "pushdown automaton" and present it to the student for further querying. Different from traditional search engines, semantic search engines utilize semantic information of certain domain knowledge specified using ontology. With the support of domain knowledge, user queries are precisely and unambiguously analyzed in order to provide better precision and recall rates for search results. Thus, ontology-based semantic search can highly improve search accuracy of the query and deliver results that are more relevant to the user queries.

In this paper, we propose an ontology-based methodology combined with traditional keyword-matching approaches to obtain more accurate and relevant search results for user queries. The proposed smart search engine model consists of two levels, namely the semantic reasoning level and the traditional keyword searching level. As one of the advantages of our approach, users are not required to be familiar with the exact concepts to be searched when formulating queries. Instead, the smart search engine can automatically infer the desired concepts by analyzing and reasoning about user-provided concepts within the given knowledge domain. To demonstrate the advantages of our approach, we develop a prototype smart search engine that supports reasoning about subsumption, supersumption, semantic equivalency and object property relations among concepts, and adopt the domain knowledge in computer science courses to demonstrate how semantic search may benefit computer science students.

Although there have been many efforts on semantic search, research on this topic is still in its premature stage. SHOE search engine [3], introduced by Heflin and Hendler, has a standalone architecture and is one of the first form-based semantic search engines. It provides sophisticated web forms that allow users to specify queries. Similar to SHOE, the OntoIR system [4] proposed by Garcia and Sicilia also belongs to the form-based semantic search engine category. It improves the SHOE approach by providing a selection-based interface

for the users. Although the above form-based approaches can be useful for those who are familiar with the domain ontology, they are not quite usable for typical Internet users. In contrast, our approach is not form based and does not require users to be familiar with the domain knowledge; thus, our approach allows users to interact with the system in a similar way to traditional search engines. There are also a few semantic search engines with standalone architecture and a RDF-based querying language. Swoogle is a crawler-based semantic web search engine that discovers and indexes documents containing RDF data [5]. When a new semantic web document is discovered, Swoogle analyzes it and extracts the needed data, and then it computes the metadata and derives the statistical properties. Similar to Google, Swoogle uses two ranking algorithms, namely OntoRank and Term Rank, to rank search results. Likewise, SWSE (Semantic Web Search Engine) consists of components that support web crawling, data enhancing and indexing, search interface, and browsing and retrieval of information [6]. It operates over RDF web data, and adapts large-scale web search engines to the case of structured data. Although the semantic search engines mentioned above can operate on RDF data, they do not support reasoning about relations among concepts in order to infer new knowledge. In contrast, our approach uses the Web Ontology Language (OWL) to specify domain ontologies, which is a knowledge representation language for authoring ontologies or knowledge bases. Thus, our approach supports reasoning about user queries, and can classify and infer related concepts to enhance search performance.

II. Ontology Development

Ontology can be used to formally and explicitly specify a conceptualization, and precisely describe the concepts and relationships that exist in a particular domain of knowledge. Ontology allows computers to understand and infer about meaning of information by providing an organized framework for classification and reasoning of given concepts and relationships. In the following sections, we describe important relations between concepts, and show how to use object property restrictions to define new concepts.

A. Concept Definition and Class Hierarchy

Ontology is typically organized as a hierarchical structure, which contains a set of concepts and relationships that can be inherited to their child elements. The combination of all concepts and their relationships constitute the knowledge base of the domain. Concepts can be represented in the form of classes organized within a class hierarchy, where classes are connected with each other by class-level relationships, namely, subsumption relation (i.e., *is-subsumed-by* or *is-subclass-of*, denoted as $\sqsubseteq$), supersumption relation (i.e., *is-superclass-of*, denoted as $\sqsupseteq$), and semantic equivalence (i.e., *is-equivalent-to*, denoted as $\equiv$) [7]. We follow a top-down approach to develop the class hierarchy for the domain knowledge. The top-down development approach starts with the definition of the most general concepts in the domain and subsequent specialization of the concepts. In an ontology-based class hierarchy, class-level relations between concepts are explicitly defined using the keywords `subClassOf` and `equivalentClass`. For example, the class `FiniteAutomaton` can be defined using ontology language OWL as a subclass of the `Automaton` class. Since it has the same semantic as a finite state machine, it is also defined as an equivalent class of the `FiniteState-Machine` class. Based on the above definitions, a reasoner can infer that all properties of the `Automaton` class and the `FiniteStateMachine` class are also properties of the `FiniteAutomaton` class due to subsumption and semantic equivalence relations among the classes, respectively.

B. Relation Between Concepts

Concepts defined as classes in a hierarchical order are not sufficient for describing certain domain knowledge. Properties between instances of classes may also be needed to describe further relations between different concepts. Relations between concepts can be specified using object properties in OWL, which are usually defined as actions. For example, the two concepts `Automaton` and `Language` can be defined as the domain and the range of the object property relation `recognizes`, respectively. Since all subclasses inherit the properties of their superclasses, when defining the domain and the range of an object property relation, we usually choose the most general classes that satisfy the relation from the class hierarchy. In addition, we also define the object property `recognizes` as an inverse relation of the object property `isRecognizedBy` since a language can be recognized by an automaton. Furthermore, the object property `recognizes` can be defined as a functional property because each automaton recognizes only one language, i.e., the language of an automaton is unique.

C. Class Definition Using Property Restriction

When we declare that class α satisfies condition φ, it is equivalent to say that α is a subclass of α' that satisfies a necessary and sufficient condition φ. This is because all instances of α must also be instances of α' since they all satisfy the necessary and sufficient condition of class α', namely, condition φ. Similarly, when concept c has property $p(c, c1)$, where c has an object property relation p with concept $c1$, concept c can be defined as a subclass of concept c' with property $p(c', c1)$ as its necessary and sufficient condition. In OWL, concept c' is called a *property restriction*, which is usually kept as an anonymous class. For example, when we define `PDA` that contains some `Stack`, we could specify `PDA` as a subclass of property restriction r_1 that satisfies property *contains*(r_1, `Stack`). Similarly, we can specify `PDA` as a subclass of a property restriction r_2 that satisfies property *isCoveredBy*(r_2, `CIS361`) since the concept `PDA` is covered by undergraduate course `CIS361`.

III. Ontology-Based Two-Level Smart Search Engine

A. Architectural Design

The framework for the two-level smart search engine is illustrated in Fig. 1. The system consists of a user interface module, a query handler, the domain ontology and two major levels, called the semantic reasoning module (Level 1) and the keyword-based query matching module (Level 2), respectively. In this paper, we focus on the first level, i.e., the semantic reasoning, and adopt an existing keyword-matching traditional

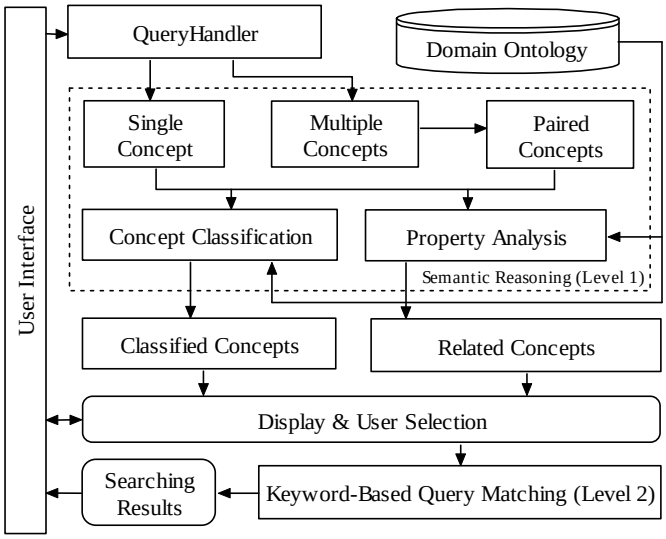

Figure 1. A framework for the two-level smart search engine

search engine, such as Google, Yahoo, Bing or DogPile, as the searching mechanism at the second level.

As shown in Fig. 1, the user interface allows a user to type in a single or multiple concepts in the form of string(s). The user inputs are filtered and parsed by the query handler, and matched with concepts defined in the domain ontology. When the user inputs are matched with a single concept, it is processed by the semantic reasoning module directly. On the other hand, when the user inputs are matched with two or more than two concepts, paired concepts are generated, and each pair of concepts is processed by the semantic reasoning module individually before the reasoning results are combined. For either a single concept or paired concepts, the semantic reasoning module first classifies the concept within the class hierarchy of the domain ontology, and infers the concepts that have the subsumption, supersumption, and semantic equivalence relations with the input concept(s). We called such inferred concepts *classified concepts*. Then it analyzes the properties of the input concept(s), and infers any concepts that are related to the input concept(s) by object properties. We call such inferred concepts *related concepts*. Once we derive all classified concepts and related concepts, they are displayed on screen to allow a user to select the most desired concept for further concept matching. In the second level of the smart search engine, a user-selected concept can be again sent to the semantic reasoning module (this option is not shown in Fig. 1) or it can be sent to a keyword-based query matching module, e.g., the Google search engine, to search for relevant information to the user query from the Internet.

B. Semantic Reasoning for Single-Concept Query

When a query contains a single concept, we call it a *simple query*. For example, if a user types in "Automaton," we consider it a simple query since "Automaton" represents a single concept. For a single concept *sc*, an inferred concept can be either a classified one or a related one. A classified concept subsumes, supersumes or is semantically equivalent to *sc*; while a related concept could be inferred due to a `someValuesFrom` or `allValuesFrom` restriction. In the following defined enumerated single concept relation, the aforementioned five different types of relations are denoted as `SUB`, `SUP`, `EQ`, `SOME` or `ALL`, respectively,

```
enum SingleConceptRelation { // for a single concept
   SUB, SUP, EQ, SOME, ALL}
```

The `SimpleRelation` and `InferredConcept` classes are defined in the following. Note that we also define an abstract class `Relation`, which serves as a superclass of the `SimpleRelation` class as well as more complicated relations described in Section III.C.

```
class SimpleRelation extends Relation {
   SingleConceptRelation sRel;
   OntClass queriedConcept;
   Relation getRelation() { … }
}

abstract class Relation {
   abstract Relation getRelation();
}

class InferredConcept {
   OntClass concept;  // the ontology class
   Relation relation; // relationship with the
                         concept(s) to be queried
}
```

The algorithm for inferring new concepts for a single concept is presented as Algorithm 1. When a user inputs a single concept, it is first processed (e.g., removing the space in the input string "Finite Automaton"), and matched with a predefined concept (i.e., `FiniteAutomaton`) within the domain ontology. Then the system uses a semantic reasoner to infer all classified concepts that subsume, supersume, and are semantically equivalent to the input concept. In the following steps, the algorithm checks all restrictions that are superclasses of the input concept, and infers all related concepts that have object property relations with the input concept by `someValuesFrom` or `allValuesFrom` restriction. Once all classified and related concepts are identified, they are returned as a list of inferred concepts for further processing.

Algorithm 1: Inference of New Concepts (Single Concept)

Input: User input representing a single concept *sc*
Output: A list of inferred concepts *lic* related to *sc*

1. match user inputs with single concept *sc* in the domain ontology
2. initialize *lic* to an empty list
3. infer all concepts such that each concept $c \sqsubset sc$, $c \sqsupset sc$ or $c \equiv sc$, and store them in sets *sSub*, *sSup*, and *sEq*, respectively.
4. **for each** *c* in *sSub*
5. set *c.relation.sRel* to `SUB`, and add *c* into *lic*
6. **for each** *c* in *sSup*
7. set *c.relation.sRel* to `SUP`, and add *c* into *lic*
8. **for each** *c* in *sEq*
9. set *c.relation.sRel* to `EQ`, and add *c* into *lic*
10. let *lres* be the list of restrictions from the set of superclasses of *sc*
11. let *inferc* be an instance of *InferredConcept*
12. **for each** restriction *res* in *lres*
13. **if** *res* is a `someValuesFrom` *inferc.concept* restriction
14. set *inferc.relation.sRel* to `SOME`, and add *inferc* to *lic*
15. **else if** *res* is an `allValuesFrom` *inferc.concept* restriction
16. set *inferc.relation.sRel* to `ALL`, and add *inferc* to *lic*
17. **return** list *lic*

C. Semantic Reasoning for Multi-Concept Query

When a user types in a query that involves two or more concepts, we call it a *complex query*. In this case, the semantic reasoning module first generates all possible paired concepts for efficient processing, and for each pair of concepts ($c1$, $c2$), it checks the following special cases that require only searching for a single concept.

Special Case 1 (SP1). When concept $c1$ is equivalent to concept $c2$ (i.e., $c1 \equiv c2$), we only need to search for single concept $c1$ or $c2$, and all classified concepts and related concepts for both $c1$ and $c2$ will be inferred.

Special Case 2 (SP2). When concept $c1$ and $c2$ have a subsumption relation (i.e., $c1 \sqsubseteq c2$), we only need to search for single concept $c1$ since $c1$ inherits all properties of $c2$. In this case, all classified and related concepts of $c2$ will be automatically inferred for concept $c1$. Similarly, when $c2 \sqsubseteq c1$, we only need to search for single concept $c2$.

Special Case 3 (SP3). When there exists an object property relation $p(c1, c2)$ between paired concepts $c1$ and $c2$, we only need to search for single concept $c1$ since in this case, $c2$ will be automatically inferred as a related concept of $c1$.

On the other hand, if the relation between $c1$ and $c2$ does not belong to any of the above special cases, new concepts must be inferred using both of the two concepts. Four typical relations between an inferred concept c and the pair of concepts ($c1$, $c2$) are defined as follows.

Common Supersumption (CSUP). Concept c is an inferred classified concept such that $c \sqsupset c1$ and $c \sqsupset c2$, i.e., c is a proper superclass of both $c1$ and $c2$. Note that in this case, $c \not\equiv c1$ and $c \not\equiv c2$; otherwise, the relation must belong to either *SP*1 or *SP*2. For example, if $c \equiv c1$ and $c \sqsupset c2$, we have $c1 \sqsupset c2$, i.e., $c2 \sqsubset c1$, which is defined as *SP*2.

Common Subsumption (CSUB). Concept c is an inferred classified concept such that $c \sqsubset c1$ and $c \sqsubset c2$, i.e., c is a proper subclass of both $c1$ and $c2$. Note that in this case, $c \not\equiv c1$ and $c \not\equiv c2$; otherwise, the relation must belong to either *SP*1 or *SP*2. For example, if $c \equiv c1$ and $c \sqsubset c2$, we have $c1 \sqsubset c2$, which is defined as *SP*2.

Property Relation (PROP). Concept c is an inferred related concept, and $\exists\ p_1(c, c1)$ and $p_2(c, c2)$, where p_1 and p_2 are object property relations between c and $c1$, and c and $c2$, respectively. Note that p_1 and p_2 can be either the same or different object property relations.

Property-Subsumption Relation (PROP-SUB). Concept c is an inferred related concept such that $c \sqsubset c2$ and $\exists\ p_1(c, c1)$ or $p_1(c1, c)$, where p_1 is an object property relation between c and $c1$; or $c \sqsubset c1$ and $\exists\ p_2(c, c2)$ or $p_2(c2, c)$, where p_2 is an object property relation between c and $c2$.

Note that we do not need to consider the case of *Property-Supersumption Relation*, where an inferred related concept c is a superclass of $c1$ (or $c2$), and there exists an object property relation between c and $c2$ (or between c and $c1$). This is because in such cases, the relation must belong to the special case *SP*3. For example, in a paired concept ($c1$, $c2$), if concept c is a superclass of concept $c2$, and there exists an object property relation $p_1(c, c1)$, there must exist an object property relation $p_2(c2, c1)$ since $c2$ inherits all properties of c. This is exactly the case defined in special case *SP*3; therefore, we only need to search for single concept $c2$.

The algorithm for inferring new concepts from multiple concepts can be designed in a similar way by considering the special cases and the `CSUP`, `CSUB`, `PROP`, and `PROP-SUB` relations (due to page limitation, the algorithm is not presented in this paper). Briefly, the algorithm first generates a list of paired concepts, and then infers classified and related concepts for each pair of concepts ($c1$, $c2$). Note that in the special cases when $c1$ is equivalent to $c2$, $c1(c2)$ subsumes $c2(c1)$, or $c1$ and $c2$ has an object property relation, the algorithm automatically executes Algorithm 1 that infers classified and related classes for a single concept. Otherwise, the algorithm first infers classified concepts, namely, the common superclasses *csup* that subsume both $c1$and $c2$, and the common subclasses *csub* that supersume both $c1$and $c2$. Then the algorithm checks whether there exists any concept c that has object property relations with both $c1$ and $c2$. Finally, it infers any possible concept c that has a property-subsumption relation with the pair of concepts ($c1$, $c2$). All inferred concepts will be added into a list of inferred concepts, and returned as output of the algorithm.

IV. Case Study

To demonstrate the feasibility of our proposed approach to developing the two-level smart search engine using ontology-based semantic reasoning, we present a case study of semantic search in the domain of computer science courses. In particular, we have defined the domain ontology based on two computer science courses taught by the first author for over 10 years at the University of Massachusetts Dartmouth (UMassD). The two courses are the undergraduate course CIS 361: Models of Computation and the graduate course CIS 560: Theoretical Computer Science. We expect a feasible smart search engine in such a knowledge domain can greatly benefit our computer science students in understanding fundamental concepts and their relations in computer science.

The smart search engine was developed using the Pellet reasoner as a Java-based OWL-DL reasoner, which provides reasoning services for OWL ontologies. The Pellet OWL2 Reasoner for Java works with the Jena framework that provides a programmatic environment for RDF, OWL, SPARQL, and includes a rule-based inference engine. The smart search engine processes a user input as concept(s) rather than keywords, and infers classified and related concepts. A user may select an inferred concept for further semantic reasoning or send it to a traditional search engine for retrieving related information from the Internet.

Once we have defined the domain ontologies for the course contents in CIS 361 and CIS 560, we can use our prototype two-level smart search engine for semantic search. Suppose a computer science student wants to search for some concepts that are related to a pair of concepts (`Stack`, `State`), some partial semantic search results for the paired concepts will be displayed as in Fig. 2.

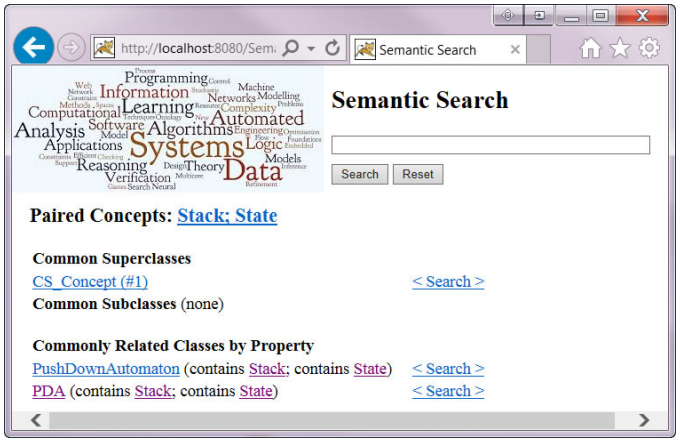

Figure 2. Seach results for a paired concepts "Stack; State"

From the search results, we can see that both `Stack` and `State` are subclasses of `CS_Concept`. The commonly related concepts inferred according to the *Property Relation* defined in Section III.C include `PushDownAutomaton` and `PDA` because both contain some `State` and some `Stack`. By clicking on the links of `PushDownAutomaton` and `PDA`, they can be further processed as a single concept. Alternatively, when an associated "Search" link is clicked, the corresponding single concept (e.g., `PushDownAutomaton`) can be searched as a keyword using a traditional search engine.

In another scenario, when a computer science student wants to search for the type of language that can be recognized by a PDA, he/she can simply type in a pair of concepts (`Language`, `PDA`) into the smart search engine, and some partial semantic search results will be displayed as in Fig. 3.

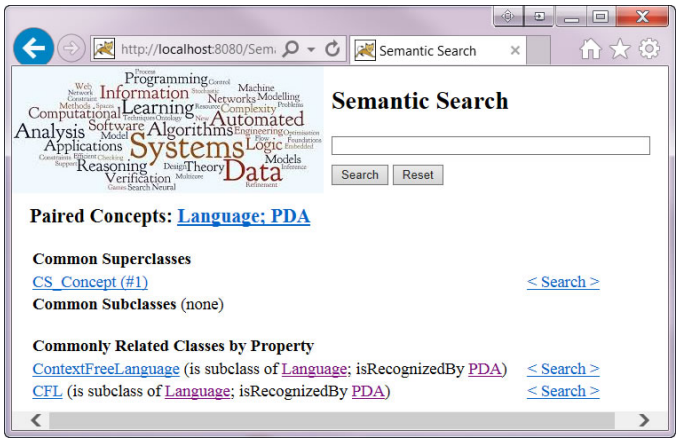

Figure 3. Seach results for a paired concepts "Language; PDA"

From the search results, we know that both the concepts `ContextFreeLanguage` and `CFL` are commonly related to the paired concepts (`Language`, `PDA`). The inference of these two new concepts is based on the *Property-Subsumption Relation* defined in Section III.C since both of them are subclasses of `Language` and can be recognized by `PDA`. By clicking on the link of `ContextFreeLanguage` or `CFL`, the inferred concept can be further searched as a single concept. Alternatively, when an associated "Search" link is clicked, the corresponding concept (e.g., `ContextFreeLanguage`) can be searched as a keyword using a traditional search engine.

V. Conclusions and Future Work

In this paper, we present an ontology-based methodology for semantic search that can produce more accurate and relevant search results for a concept-based query. The smart search engine consists of two levels, namely the semantic reasoning level and the traditional keyword searching level. The semantic reasoning level can understand the meaning of a concept or multiple concepts entered by a user, and classify and reason about the concepts in the corresponding knowledge domain specified using ontology. Once the semantic reasoning module infers the concepts related to a user query, in the second level, a traditional search engine can be used to search for addition information from the Internet. One major benefit of our approach is that a user does not have to be familiar with the domain knowledge when making queries. Instead, the smart search engine can reason about user queries, and produce those concepts that are closely related to the user inputs. To demonstrate the feasibility of our smart search engine approach, we develop a prototype smart search engine as well as ontologies in the domain of computer science courses. The case studies show that the smart search engine can effectively infer concepts that are semantically related to user queries.

As future work, we plan to develop a set of more complete domain knowledge for the computer science courses taught at UMassD, and show that such smart search engine can greatly help students to understand difficult computer science concepts. In addition to dealing with paired concepts, we will also design algorithms that can directly process a complex query with more than two concepts. Such algorithms should efficiently infer new related concepts from the given multiple concepts by reasoning about their relations with each other.

References

[1] Y. Lei, V. Uren, and E. Motta, "SemSearch: A Search Engine for the Semantic Web," *Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2006)*, Lecture Notes in Computer Science, Vol. 4248, Springer, Heidelberg, Oct. 2-6, 2006, Podebrady, Czech Republic, pp. 238-245.

[2] C. Mangold, "A Survey and Classification of Semantic Search Approaches," *International Journal of Metadata, Semantics and Ontologies (IJMSO)*, Vol. 2, No. 1 , 2007, pp. 23-34.

[3] J. Heflin and J. Hendler, "Searching the Web with SHOE," *Proceedings of the AAAI Workshop on Artificial Intelligence for Web Search*, AAAI Press, Menlo Park, CA, 2000, pp. 35-40.

[4] E. García and M. Sicilia, "Designing Ontology-Based Interactive Information Retrieval Interfaces," *Proceedings of the Workshop on Human Computer Interface for Semantic Web and Web applications (HCI-SWWA),* Lecture Notes in Computer Science 2003, Springer, New York, 2003, pp. 152-165.

[5] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P Reddivari, V. Doshi, and J. Sachs. "Swoogle: a Search and Metadata Engine for the Semantic Web," *Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management (CIKM'04),* Washington DC, USA, 2004, pp.652-659.

[6] A. Hogan, A. Harth, J. Umbrich, et. al., "Searching and Browsing Linked Data with SWSE: the Semantic Web Search Engine," *Journal of Web Semantics,* Vol. 9, No. 4, 2011, pp. 365-401.

[7] G. Antoniou and F. van Harmelen, *A Semantic Web Primer*, 2nd Edition, MIT Press, Massachuestts, 2008.