Volume 3, Issue 6, June 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Semantic Web Search Engine

D.Pratiba[1], Dr.G.Shobha[2]
Department of CSE,
RVCE, Bangalore, Karnataka, India

Abstract- ***Semantic search promises to produce precise answers to user queries by taking advantage of the availability of explicit semantics of information in the context of the semantic web. Existing tools have been primarily designed to enhance the performance of traditional search technologies but with little support for naive users, i.e., ordinary end users who are not necessarily familiar with domain specific semantic data, on- tologies, or SQL-like query languages. This paper presents SemSearch, a search engine, which pays special attention to this issue by hiding the complexity of semantic search from end users and making it easy to use and effective. In contrast with existing semantic-based keyword search engines which typically compromise their capability of handling complex user queries in order to overcome the problem of knowledge overhead, SemSearch not only overcomes the problem of knowledge overhead but also supports complex queries. Further, SemSearch provides comprehensive means to produce precise answers that on the one hand satisfy user queries and on the other hand are self-explanatory and understandable by end users. A prototype of the search engine has been implemented . An initial evaluation has shown promising results.***

Keywords-Semantic search, ontology, SPARQL-Query, RDF.

I. INTRODUCTION

One important goal of the semantic web is to make the meaning of information explicit through semantic mark-up, thus enabling more effective access to knowledge contained in heterogeneous information environments, such as the web. Semantic search plays an important role in realizing this goal, as it promises to produce precise answers to user's queries by taking advantage of the availability of explicit semantics of information. For example, when searching for news stories about phd students, with traditional searching technologies, often only news entries in which the term "phd students" appears. Those entries which mention the names of students but do not use the term "phd students" directly will be missed out. Such news entries however are often the ones that the user is really interested in. In the context of the semantic web, where the meaning of web content is made explicit, the semantic meaning of the keyword (which is a general concept in the example of phd students) can be figured out. Furthermore, the underlying semantic relations of metadata can be exploited to support the retrieving of information which is closely related to the keyword. Thereby, the search performance can be significantly improved by expanding the query with instances and relations.

A number of semantic search tools have been recently developed [1],[2],[3],[4],[5],[6]. Our review of the state-of-art semantic search tools reveals that while these tools do enhance the performance of traditional search technologies, they are however not suitable for naive users, i.e. ordinary end users who are not necessarily familiar with domain specific semantic data, ontologies, or SQL-like query languages. The semantic search engine which is presented here, SemSearch, provides several means to address this issue:

1) SemSearch tackles the problem of knowledge overhead by supporting a Google- like query interface. The proposed query interface provides a simple but powerful way of specifying queries.
2) SemSearch addresses the problem of existing semantic-based keyword search engines by supporting complex queries. It provides comprehensive means to make sense of user queries and to translate them into formal queries.

Thus, SemSearch makes it possible for ordinary end users to harvest the benefits of semantic search and other semantic web technologies without having to know the underlying semantic data or to learn a SQL-like query language.

II. BACKGROUND STUDIES

This section, investigates how current semantic search approaches address user support. Four categories of semantic search engines have been identified, according to the user interface they provide: i) form-based search engines, which provide sophisticated web forms that allow users to specify queries, in the format of choosing ontologies, classes, properties, and values; ii) RDF-based querying languages fronted search engines, which provide sophisticated querying languages to support semantic search; iii) semantic-based keyword search engines, which en-hance the performance of traditional keyword search techniques by making use of available semantic data; and iv) question answering tools, which exploit available semantic mark-up to answer questions asked in natural language format.

The SHOE search engine is one of the first form-based semantic search engines. It provides sophisticated web forms that allow users to specify queries. Such forms however are only suitable for those users who are fairly familiar with the back-end ontologies and knowledge bases. Naive users have difficulties in understanding these forms. Further, they have difficulties in formulating queries using their own view on the information they aim to find. The Corese search engine is an example of RDF-based querying language fronted search engines. Other examples include the engines built in CS AKTive Space [7] and the SemanticWeb.org portal. Such search engines usually provide a sophisticated querying language to support semantic data querying. However, in order to be able to ask queries with these search engines, end users will have to be fairly familiar with both the back-end ontologies and the provided querying language. The TAP search engine and the search engine presented in are examples of semantic-based keyword search engines. The search process of such search engines often comprises two major steps: i) finding an instance match for the user keyword and ii) retrieving instances which are closely related to the instance match of the user keyword. Such search engines often provide comprehensive means to support the clustering of search results. AquaLog and ORAKEL [8] are examples of ontology-based question answering engines. They make use of natural language processing technologies to reformulate natural language queries into ontological triples (e.g., in AquaLog) or into specific query languages (e.g., in ORAKEL). While these tools appear to be ideal for naive users, their performance on searching is heavily influenced by the performance of the used natural language processing techniques. All the tools described above are able to enhance the search performance by making use of available semantic data and their underlying ontologies. With the partial exception of ontology-based question answering tools, state-of-art tools are however not suitable for naive users. One problem is knowledge overhead, which is requiring users to be equipped with extensive knowledge of the back-end ontologies and knowledge bases (e.g. form-based search engines) or specific SQL- like querying languages (e.g. RDF-based query language fronted search engines) in order to be able to formulate queries or to understand the search result.

Another problem is the lack of support for answering complex queries presented by current semantic-based keyword search engines. These search engines are often only able to accept one keyword as input and give back the semantic entities which are related to the keyword as results. Relation centric search that finds relations between multiple keywords is not supported. This greatly limits the scope of user queries. For example, current semantic-based keyword search engines typically could not even handle simple queries where two keywords are involved such as news about phd students.

III. AN OVERVIEW OF SEMSEARCH

One major goal of this work is to hide the complexity of semantic search from end users and to make it easy to use and effective for naive users. To achieve this goal, the following key requirements are identified:

1) Low barrier to access for ordinary end users. The semantic search engine should overcome the problem of knowledge overhead and ensure that ordinary end users are able to use it without having to know about the vocabulary or structure of the ontology or having to master a special query language.

2) Dealing with complex queries. In contrast with existing semantic-based keyword search engines which only answer simple queries, the semantic search engine should allow end users to ask complex queries and provide comprehensive means to handle them.

3) Precise and self-explanatory results. The semantic search engine should be able to produce precise results that on the one hand satisfy user queries, and on the other hand are self-explanatory. Thus, ordinary end users can understand the results (e.g. what they are and why they are there) without having to consult the back-end semantic data repositories or their underlying ontologies.

4) Quick response. Our semantic search engine should provide quick response to user queries, thus encouraging ordinary end users to harvest the benefit of the semantic web technology. This requires that the mechanism of semantic search is made as simple as possible.

To meet these requirements, the keyword-based searching route is chosen rather than the natural language question answering route, and deliberately avoided linguistic processing which is a relatively expensive process in terms of search. We overcome the limitation of current keyword-based semantic search engines by supporting a Google-like query interface which supports complex queries in terms of multiple keywords. Fig. 1 shows a layered architecture of our semantic search engine. It separates end users from the back-end heterogeneous semantic data repositories by several layers.

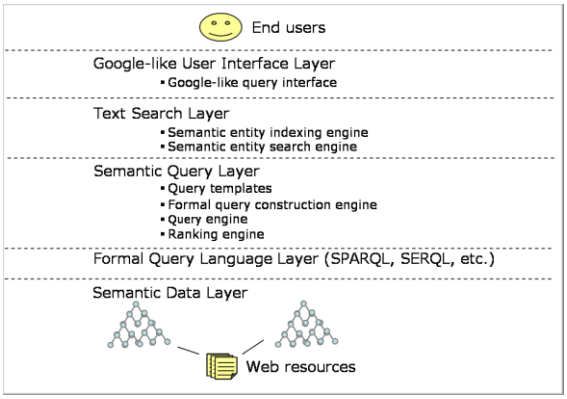

Fig. 1. An overview of the SemSearch architecture.

1) The Google-like User Interface Layer, which allows end users to specify queries in terms of keywords. The Google like query interface extends traditional keyword search languages by allowing the explicit specification of i) the queried subject and ii) the combination of multiple keywords.
2) The Text Search Layer, which makes sense of user queries by finding out the explicit semantic meanings of the user keywords. The central to this layer are two components: i) a semantic entity index engine, which indexes documents and their associated semantic entities including classes, properties, and individuals; and ii) a semantic entity search engine, which supports the searching of semantic entity matches for the user keywords.
3) The Semantic Query Layer, which produces search results for user queries by translating user queries into formal queries. This layer comprises three components, including i) a formal query construction engine, which translates user queries into formal queries, ii) a query engine, which queries the specified meta-data repository using the generated formal queries, and iii) a ranking engine, which ranks the search results according to the degree of their satisfactory on the user query.
4) The Formal Query Language Layer, which provides a specific formal query language that can be used to retrieve semantic relations from the underlying semantic data layer.
5) The Semantic Data Layer, which comprises semantic metadata that are gathered from heterogeneous data sources and are represented in different ontologies.

Fig. 2 shows an overall diagram of the SemSearch search engine. It accepts keywords as input and produces results which are closely related to the user keywords in terms of semantic relations. The search process of SemSearch comprises four major steps:
1) Making sense of the user query, this is to find out the semantic meanings of the keywords specified in a user query.
2) Translating the user query into formal queries
3) Querying the back-end semantic data repositories using the generated formal queries
4) Ranking the querying results.

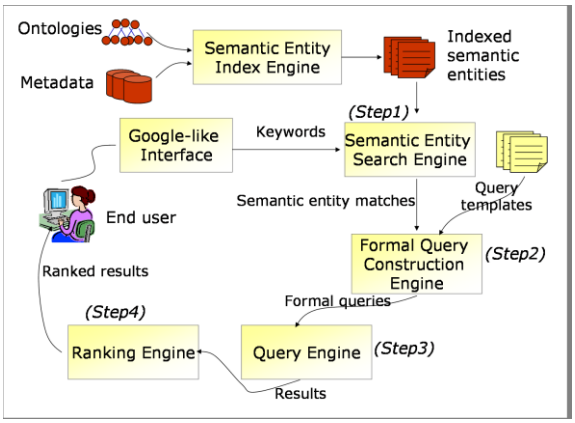

Fig. 2. An overall diagram of the SemSearch search engine.

Step1 is carried out within the Text Search Layer. The rest of the steps are associated with the Semantic Query Layer and are described in the following sections.

IV. THE GOOGLE-LIKE QUERY INTERFACE

The query interface of SemSearch extends traditional keyword search languages by allowing the explicit specification of i) the queried subject and ii) the combi- nation of keywords. By the term subject, the user can explicitly tell the search engine the expected type of the search results. For example, when the user specifies the keyword "news" as the subject keyword, he or she expects the search results to be the instances of the class entity that matches against the keyword "news". The search engine provides end users a simple way of expressing complex queries. In order to satisfy a query, each of the required keywords (please note that subject is a special required keyword) must be satisfied, which means that the search results must be semantically related to each of the required keywords. The SemSearch query interface provides a simple, flexible, and powerful approach for specifying user queries in semantic search. First, it overcomes the problem of knowledge overhead suffered in formal query fronted search engines and form-based semantic search engines, as it does not require end users to be familiar with any particular ontology, semantic data, or any special query language. Second, the query interface provides a more flexible way of specifying queries than the interface presented by form-based search engines. Indeed, it does not confine users to any pre-defined query subjects and values. Third, in contrast with current semantic-based keyword search engines which only accept one keyword as input, this query interface supports the specification of complex queries in the format of specifying multiple keywords and the expected type of results. Finally, this query interface is simpler than question answering tools as the search engine does not need to spend time calculating which of the keywords are in a user's query.

In this paper, we focus on the user queries in which there are at least two keywords involved, in order to better explain the distinctive features of our search engine. Regarding those queries that only comprise one keyword, the search engine develops an approach that is similar to the ones used in TAP and in to find instances that are closely related to the semantic entity matches of the keyword.

V. MAKING SENSE OF THE USER QUERY

As mentioned earlier in Section 3, making sense of the user query is the first step of the search process in SemSearch, whose task is to find out the semantic meanings of the keywords specified in a user query so that the search engine knows what the user is looking for and how to satisfy the user query. From the semantic point of view, one keyword may match i) general concepts (e.g., the keyword "phd students" which matches the concept phd-student), ii) semantic relations between concepts, (e.g. the keyword "author" matches the relation has-author), or iii) instances entities (e.g., the keyword "Enrico" which matches the instance Enrico-Motta, the keyword "chief scientist" which matches the values of the instance Marc-Eisenstadt of the property has-job-title). The ideal goal of this task is to find out the exact semantic meaning of each keyword. This is however not easy to achieve, as there may be more than one semantic entity which matches a keyword. Thus, we relaxed the goal as finding out all the semantic entity matches for each keyword. For the purpose of finding out semantic entity matches, we used the labels of semantic entities as the main search source. The rational for this choice is that from the user point of view labels often catch the meaning of semantic entities in an understandable way. In the case of instances, we also used their short literal values as the search source, so that when the user is searching for "chief scientist", the instance that has such a string as a value of its properties can be reached. In order to produce fast response, the search engine first indexes all the semantic entities contained in the back-end semantic data repositories, including classes, properties, and instances. It then searches the indexed repository to find out matches for keywords. Thus, two components are developed in the search engine, namely the semantic entity index engine and the semantic entity search engine. As it narrows the search sources to labels and short literals of semantic entities, the search engine is able to find out semantic entity matches for each keyword. These matches are the possible semantic meanings of keywords.

For the sake of getting quick response, we only use text search to find string matches for user keywords at the moment. We avoid using techniques like WordNet [9] based comparison to find matches. This might cost us some good matches, e.g., losing the match table if the user is searching for desk. But one to one comparison is time consuming and expensive in real-time scenarios. This is indeed a trade-off as well as a research challenge that we need to address in future.

VI. TRANSLATING THE USER QUERY INTO FORMAL QUERIES

In this step, the search engine takes as input the semantic matches of user search terms and outputs an appropriate formal query according to the semantic meanings of keywords. To achieve this task, the search engine needs to capture the focus of the user query (i.e., the type of the expected search results). As described earlier in Section 4, the subject keyword specifies the type of the expected search result. Thus, it is reasonable to expect that the queried subject is a general topic or concept (i.e. class). To better understand how to construct formal queries from user queries, we classify user queries into two types: i) simple queries which only comprise two keywords, and ii) complex queries where more than two keywords are involved. In the case of simple queries where the types of semantic entity match combinations are fixed, we developed a set of templates to support the formulation of formal queries. The situation is much trickier in the case of complex queries where there are many variables for keywords combinations. We have used the SPARRQL language as the formal query language in the prototype of the SemSearch search engine.

VII. IMPLEMENTATION AND EXPERIMENTAL EVALUATION

A prototype of SemSearch has been implemented, which uses Joseki and Jena . Jena provides a query language and a query engine for semantic data represented in RDF. Joseki provides a fast text search engine, which is used to build the semantic entity index engine and the semantic entity search engine contained in the Text Search Layer of SemSearch. The prototype has been applied to the semantic web portal of our lab. Fig. 3 shows a screenshot of the search results of the query example 'vietnamese on race course road'. As described earlier, the search engine not only gives back the information that the user is looking for but also gives back explanations, which makes the search results much more understandable than those in state-of-art tools. The search engine also summarizes all the data on the website. The search engine also provides support for search refinement. It provides a web form to allow the user to choose the meaning of the keywords and thus supports the user in reformulating a better search. To assess the performance of the semantic search engine, we carried out an initial study in the context of the semantic web portal. We used the questions that were gathered as the basis for experimental evaluation. Several re- formulations of each search were attempted if necessary. For each search, we assigned a score that describes the performance of the search engine: i) 0 - no result; ii) 1 - could get a result with heavy analysis; iii) 2 - could get a result with moderate analysis; and iv) 3 - good result. Taking into account only the questions for which an answer is possible, the average score was 2.1. The performance scores are only a qualitative assessment of how felt the system answered the questions so these results are biased. However, based on these rudimentary performance measures the semantic search is answering questions reasonably well where data is available. In particular, SemSearch was able to answer a high proportion of the questions despite its simplicity. It is intuitive and simple to learn. The user doesn't need to have a full grasp of the ontology to get started (though they need to know something). This is an affordance of the way that results are

presented in the interface in a way that informs the user about the terms in the ontology. That information can be used to help with search refinement. For example, the user might not remember relations like "generic-area-of-interest" but might remember that the word "area" is involved in a lot of relations about research topics and by browsing through the output of a search just for "area" can gather the information on the ontology vocabulary needed to formulate a better one.

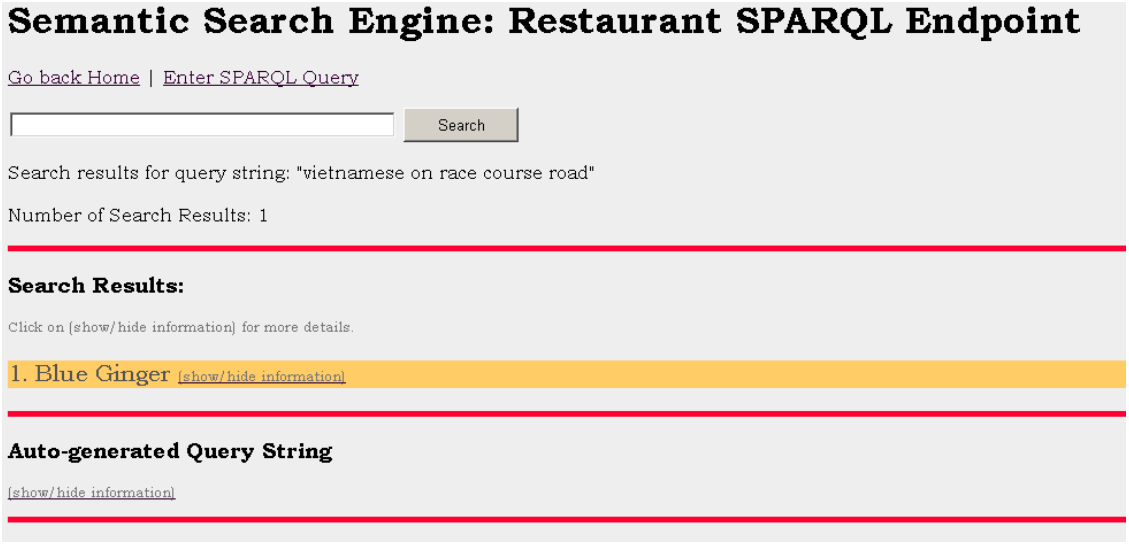

Fig. 3. A screenshot of the search results of the query example vietnamese on race course road.

Fig. 4. Expanded view of the same search result

VIII. CONCLUSIONS AND FUTURE WORK

The core observation that underlies this paper is that, in the case of semantic search that promises to produce precise answers to user queries, it is important to ensure that it is easy to use and effective for ordinary end users who are not necessarily familiar with domain specific semantic data, ontologies, or SQL-like query languages. Our survey of state-of-art semantic search tools however reveals that current tools provide little support for end users. In contrast with these efforts, our semantic search engine, SemSearch, provides several means to address this issue. First, SemSearch overcomes the problem of knowledge overhead by supporting a Google-like query interface. As described in Section 4, the proposed query interface provides a simple but powerful way of specifying queries. Second, SemSearch is able to produce precise answers for user queries by providing comprehensive means to make sense of user queries and to translate them into formal queries. In particular, as described in Section 6, the produced answers on the one hand satisfy user queries and on the other hand are self-explanatory and understandable by end users. Finally, SemSearch provides means (i.e., search refinement forms) to support end users in reformulating better queries. A prototype has been implemented. The experimental study indicates a encouraging results. Future work will focus on i) carrying out a more thorough evaluation which will help us to investigate the problems of each main component of the search engine and to improve their performance accordingly; ii) developing comprehensive means to perform semantic matching between keywords and semantic entities without compromising the performance of the search engine in terms of time; iii) developing more fine grained rules which on the one hand will help us to significantly reduce the number of keywords combinations and on the other hand will help us to identify and keep useful information in the

reduction process; and iv) developing a powerful ranking mechanism, which guarantees the best results always staying on the top.

REFERENCES

[1] O. Corby, R. Dieng-Kuntz, and C. Faron-Zucker. Querying the Semantic web with Corese Search Engine. In Proceedings of 15th ECAI/PAIS, Valencia (ES), 2004

[2] R. Guha, R. McCool, and E. Miller. Semantic Search. In Proceedings of the 12th international conference on World Wide Web, pages 700–709, 2003.

[3] J. Heflin and J. Hendler. Searching the Web with SHOE. In Proceedings of the AAAI Workshop on AI for Web Search, pages 35 – 40. AAAI Press, 2000.

[4] V. Lopez, M. Pasin, and E. Motta. AquaLog: An Ontology-portable Question Answering System for the Semantic Web. In Proceedings of European Semantic Web Conference (ESWC 2005), 2005.

[5] C. Rocha, D. Schwabe, and M. de Aragao. A Hybrid Approach for Searching in the Semantic Web. In Proceedings of the 13th International World Wide Web Conference, 2004

[6] L. Zhang, Y. Yu, J. Zhou, C. Lin, and Y. Yang. An Enhanced Model for Searching in Semantic Portals. In Proceedings of the 14th international conference on World Wide Web (WWW 2005), 2005

[7] M.C. Schraefel, N.R. Shadbolt, N. Gibbins, H. Glaser, and S. Harris. CS AKTive Space: Representing Computer Science in the Semantic Web. In Proceedings of the 13th International World Wide Web Conference, 2004.

[8] P. Cimiano. ORAKEL: A Natural Language Interface to an F-Logic Knowledge Base. In Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems, pages 401–406, 2004.

[9] C. Fellbaum. WORDNET: An Electronic Lexical Database. MIT Press, 1998.

