Journal of Engineering Sciences, Assiut University, Vol. 38, No. 4, pp. 989-1000, July 2010.

SEMANTIC WEB BASED SEARCH AGENT SYSTEM

Majid A. Askar[1], Hesham A. Hassan[2], and Samhaa R. El-Beltagy[2].

[1]*Computer Science Department, Faculty of Computer and Information, Assiut University, Egypt.*

[2]*Computer Science Department, Faculty of Computer and Information, Cairo University, Egypt.*

(Received May 4, 2010 Accepted June 5, 2010)

The term "search engine" is traditionally used to refer to crawler based search engines, manually maintained directories, and hybrid search engines. However, current search engines do not fully satisfy the users' needs especially in terms of accuracy and specificity of the results. This paper proposes an approach to build an intelligent search agent system on top of the Semantic Web. The presented system consists of five main parts: the Annotator, the Ontology Parser, the Indexer, the Search Agent, and the Data Repository. Two kinds of search are implemented: keyword based and concept based search. The keyword based search matches a user's query terms to concepts while concept based search allows a user to choose the concept that s/he want to search for together with some attributes for this concept.

***KEYWORDS**: Information Retrieval, Semantic Search.*

1. INTRODUCTION AND MOTIVATION

The goal of the semantic web is to enable structural and semantic definitions of documents providing completely new and powerful possibilities: Intelligent search instead of keyword matching, query answering instead of information retrieval, document exchange between departments via ontology mapping. Using these technology internet agents can understand web content, access databases and co-operate with each other to perform specific tasks.

The Semantic web has thus become an important reality and an essential demand for many users on the internet. Also an important demand for many people is search. Many users need an intelligent search agent system that manages the search process. Because semantic search promises to revolutionize information retrieval (by complementing it rather than by replacing it), even search engines that currently dominate the web, the more notable of which are Google, Yahoo, and recently Bing, are making a move towards semantic search [1],[2]. This paper proposes an approach to build a search agent system that utilizes the Semantic Web. The proposed system uses ontology and annotations made within a specific domain. The system consists of five main components: data repository, annotator, ontology parser, Indexer, search agent. Related work is found in section 2. The system architecture is presented in section 3. Section 4 represents the case study. Conclusion and future work is in section 5

2. RELATED WORK

A proposed architecture for a semantic information retrieval system based on intelligent agents is presented in [3]. Using a graphical interface the user submits a query to the system and s/he can also specify a numeric value, which indicates the depth at which each site is to be inspected. The user can also specify the language of pages to be found and the context that indicates the search area.

The architecture described in [4] uses three main agents, where each agent is in charge of a different task. The user agent allows users to access the document ontology; it shows information about a document and makes annotations about the document's properties. The ontology agent is used to retrieve domain ontologies and their structure. The search agent searches for the metadata of a document as a response to a message from user agent querying about a document. The Java Agent Development Framework (JADE [5]) was used for implementation of the agents.

The architecture of another proposed search system is shown in [6] this uses the spread activation algorithm. The first two steps of the search process happen exactly in the same way as in traditional searches, some how like [2]. The user expresses his query in terms of keywords that are fed to a traditional search engine. The result given by the traditional search engine is a set of node instances ordered by their similarity with the query. This set of nodes is supplied to the spread activation algorithm as the initial set of nodes for the propagation.

Swoogle [7] is a crawler-based indexing and retrieval system for Semantic Web documents, documents represented in Resource Description Framework (RDF) or Web Ontology Language (OWL). It extracts metadata for each discovered document, and computes relations between documents. Discovered documents are indexed by an information retrieval system which can use either character N-Gram or URI refs as keywords to find relevant documents and to compute the similarity among a set of documents. One of the properties computed is the rank, a measure of the importance of a Semantic Web document.

"Semantic Search" is the name of an application described in [8]. The Semantic search application runs as a client of the TAP infrastructure [9]. TAP is a semantic web platform. It is an implementation of a querying and negotiation interfaces/protocols [8]. When the search query is received, the search front end sends the query to the search backend, and invokes the Semantic Search application. The described system uses the W3C's Resource Description Framework with the schema vocabulary provided by RDFS [11] as a means for describing resources and their inter-relations

Noesis [12] is a semantic search engine and resource aggregator for atmospheric science. Noesis uses a three step algorithm to search resources. The first step is query analysis where the user query is broken down to identify the concepts that are defined in the domain ontology. The second one is the semantics presentation where the annotated concepts from the query string are used to search the Ontology Inference Service. The Ontology Inference Service (OIS) is a SOAP-based web service interface to an inference engine. The third one is the resource search where the selected terms are then used for searching the resources. Recently, the Semantic MediaWiki (SMW), which "helps to search, organize, tag, browse, evaluate, and share" the

contents of wikis built using MediaWiki (such as Wikipedia) [13] has been extended to work with Arabic [14]. The "SMW adds semantic annotations that let you easily publish Semantic Web content, and allow the wiki to function as a collaborative database" [13].

3. SYSTEM ARCHTECTURE

The goal of search engines is to return results that are both accurate and complete. Using web semantics enables us to get more accurate results. The proposed system uses ontology and annotations made within a specific domain. The system consists of five main components as in fig. 1 namely the annotator, the ontology parser, the indexer, the search agent, and the data repository. Each of these is described in the following sub-sections.

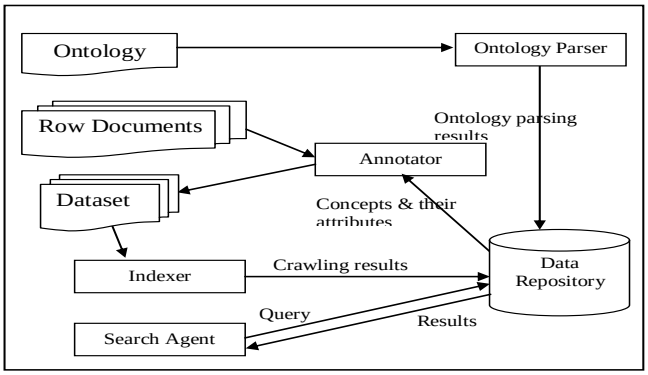

Figure 1. System Overview

Data Repository

The data repository represents the main data store where concepts and their attributes extracted from the ontology (see Ontology Parser section) are stored. In this repository concepts and their attribute values found in the crawled domain together with the page addresses where they are found in (see Indexer section), are also stored.

Ontology Parser

This module takes the ontology as input, applies parsing rules, and produces as output a standardized representation of the ontology which is stored in the data repository. The scenario is as follows:

- The parser parses the given ontology to extract the concepts and the attributes defined in it (see Figure 2). The parser does its job according to a predefined syntax in which the ontology is written (RDFS in this case).
- The extracted concepts and their attributes are then stored in the data repository. The related concepts are also stored together (the relationships between concepts are maintained in the data repository).

Other implementations of the ontology parser handling other ontology representation formats can be plugged into the system.

Ontology Parser Algorithm

- Get all the nodes in the ontology with prefix rdfs:Class and store them in an XML node list1.
- From list1 select the concept name & role (abstract, concrete) using the rdfs:Label & a:role prefix. 3- Get the rdfs:subClassOf property which represents the relative concept.
- Get all the nodes in the ontology with prefix rdf: Property and store them in an XML node list2.
- For each node in the list2 get the node cardinality through prefix a:maxCardinality & get parent for the concept instance attributes through prefix a:allowedClasses , rdfs:range.
- Store the concepts, attributes and relative concepts into the data repository.

Figure 2. Ontology Parser Algorithm

Annotator

The annotator is a manual tagging tool that is used to create an annotated dataset from an input set of documents. This dataset is then used by the indexer. The implemented annotation tool reads in the ontology from the data repository and creates a button for each concept in the ontology. It also takes in input documents to be annotated and displays this to the user along with the concepts. The user can then use the implemented graphical user interface to select portions of the text and annotate them. When the user selects a concept to annotate a portion of the text with, a template is presented to the user to allow him/her to fill in the values or related properties. Figure 3 shows an example of such a template. The output of this component is an XML file which is an annotated version of the input file.

The Indexer

The indexer takes in as input annotated XML files in some given domain and creates an index for entries in those files within. The indexing process as a whole takes place as shown in Figure 4 and involves the following entities:

- Home directory (Starting Folder): is the folder where domain specific annotated documents are kept.
- List of Files: A list containing those files residing within the home directory.
- Document processor: Is the actual indexing components. It extracts the concepts and their attributes form each file. The resulting concepts and their locations are then stored in the data repository. The annotation of a page is parsed using an XML parser to extract nodes which represent concepts. The attributes of each node are also extracted. These concepts and attributes with the page address are sorted in the data repository.

- List of concepts together with their locations: A list of pairs; each pair is a concept and the page where this concept is found. Also the attributes are added.
- Data repository: The main store of data.

Figure 3: An example of a template for filling in concept property values for use in the annotation process

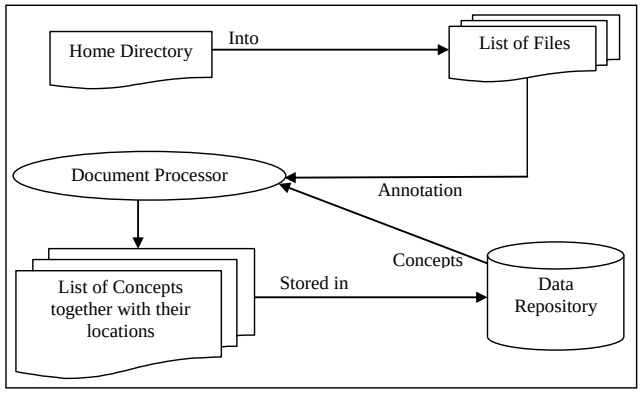

Figure 4: Indexer

Indexer Algorithm

* For every file in the home directory do the following.

1- Select the root node and place it in a node say Root.

2- For each child node within the node Root do the following

2.1 If the node is a concept or instance attribute then store the parent node id & name and the current node id & name and where they are found into the data repository.

2.2 If the node is an attribute then store the parent node id & name and the attribute name & value and where they are found into the data repository.

Figure 5. Indexer Algorithm

The Search Agent

Two types of search are implemented; advanced (concept based) and keyword based. Keyword based search resembles traditional search in that a user types his/her query as a set of keywords and then invokes the search process. However, in our work a user's query is first parsed to extract any concepts that it may contain (see figure 6). To do so, the search agent uses a concept parser. The outcome of the parser is a list of concepts is obtained. These concepts are then searched for in the data repository. The result is a list of links which are returned back to the user.

In advanced search the user can specify the concept that s/he is searching for and the attributes (if there are any) that are related to that concept. A query is formulated accordingly and sent to the data repository. This also results in the return of a list of links that are displayed to the user.

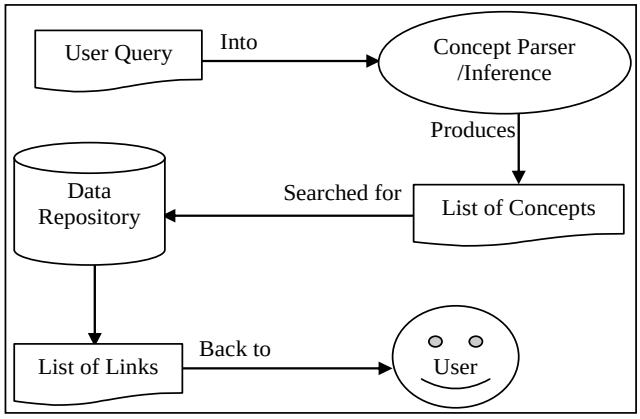

Figure 6. Search agent

Search Agent Algorithm

1- Connect to the data repository to get all the concepts.

2- Get the words that exist in the user query string and store them into a string array say WR1.

3- Get the stems of the words array WR1.

4- Compare the resulted stems with the concepts stems. The matched stems are used to indicate the concepts exist in the user query, these concepts are store into a string array say CR1.

5- Select the concept data and its location from the data repository using CR1 and WR1 as follow.

5.1- Select all the concepts as those in CR1 and its attributes as those in WR1.

5.2- if no result is found then select all the attributes as those in WR1.

5.3- if no result is found then select all the concepts as those in CR1.

5.4- if no result is found then prompt the user.

6- Show the results (if found) to the user.

Figure 7. Search Agent Algorithm

4. CASE STUDY

To demonstrate the usefulness of the developed tool, it was applied to a set of actual documents which represent department meetings of the computer science department in the faculty of computers and information Cairo University. First, a complete Ontology for department meetings was created and represented in XML format (see fig.8 for part of this Ontology). The department meetings documents were then annotated by concepts from that ontology. (See fig 9 for part of an annotated department meeting document). The annotated documents represent our dataset and it is what is indexed in our system.

The indexer stores its results in a database. Finally the user uses the search agent via either concept based search or key word search. If the user uses the concept based search s/he chooses the concept s/he is searching for and fills its attributes (if s/he so desires) then starts the search process. On the other hand if the user uses the keyword based search s/he enters his/her query represented in keywords and then starts searching.

Unlike traditional search engines that return to the user an entire document, our search system just returns to the user the annotated piece of information that s/he is probably interested in. Figures 11 and 12 show screen shots of the implemented search agent for concept-based search and key-word search respectively.

We conducted a very simple experiment to compare between our version of keyword search and the advanced concept based search. In this experiment 8 documents were annotated and then 7 queries were entered using the keyword interface and then again using the advanced concept based search interface used. Figure 10 shows the average mean precision for the results obtained from both systems. The outcome of this experiment showed that the concept based version returns more precise results. The reason for this can be attributed to the fact that this kind of search allows the user to enter exactly what s/he wants using a structured interface. We did not compare our results with a traditional search system as traditional search system targets documents, while we target specific pieces of information. However, we are planning on comparing our results with all semantic annotation systems we can get our hands on. This is part of our future work.

5. CONCLUSION AND FUTURE WORK

The use of web semantics to improve the web search can be considered as a step forward for enhancing web search especially with the existence of a rich ontology. The proposed system works on a specific domain with a known ontology and annotation. Applying this system, allows a user to reach the information of interest immediately, as unstructured data is transformed to a structured format during the annotation and indexing process. Having a richer ontology or a set of ontologies, the proposed system could work on different domains.

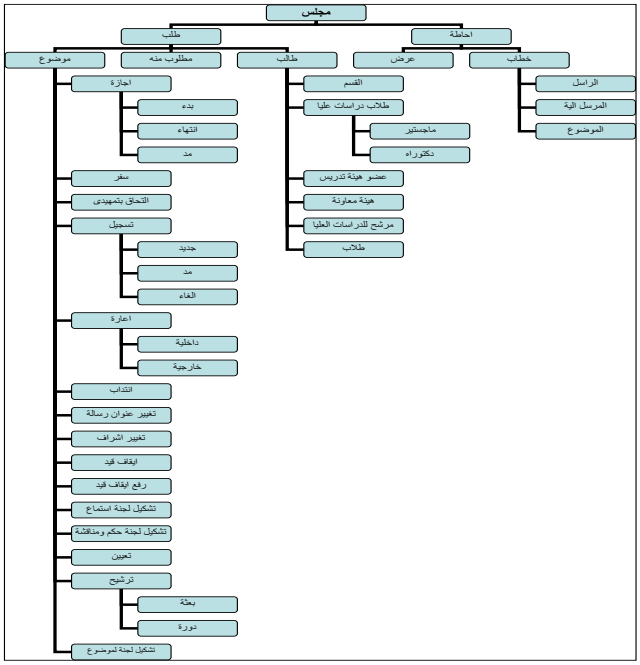

Figure 8. Used Ontology

```
<مجلس>
+ <الحضور>
+ <العام_الجامعي>
+ <المتغيبين>
- <برئاسة>
  <rdfs:isDefinedBy rdf:resource="\Ontology\StaffMembers.xml" />
- <عضو_هيئة_تدريس>
  <اسم>فاطمة عمارة</اسم>
    </عضو_هيئة_تدريس>
    </برئاسة>
- <تاريخ>
  <سنة>2009</سنة>
  <شهر>1</شهر>
  <يوم>27</يوم>
    </تاريخ>
- <يحتوى>
- <طلب>
- <بشان>
- <سفر>
- <تاريخ>
  <سنة>2009</سنة>
  <شهر>1</شهر>
  <يوم>27</يوم>
    </تاريخ>
- <تاريخ_السفر>
  <سنة>2009</سنة>
  <شهر>2</شهر>
  <يوم>15</يوم>
    </تاريخ_السفر>
```


Figure 9. Annotation Sample

Figure 10. Key-word Based Search

There are some open issues concerning semantic web search. Like the following:-

- Allow the use of multiple ontologies.
- Allowing the use of metadata with different semantic web languages and different specifications.

Automatic and precise annotation of documents through the use of a combination of natural language processing, information extraction and named entity recognition technologies

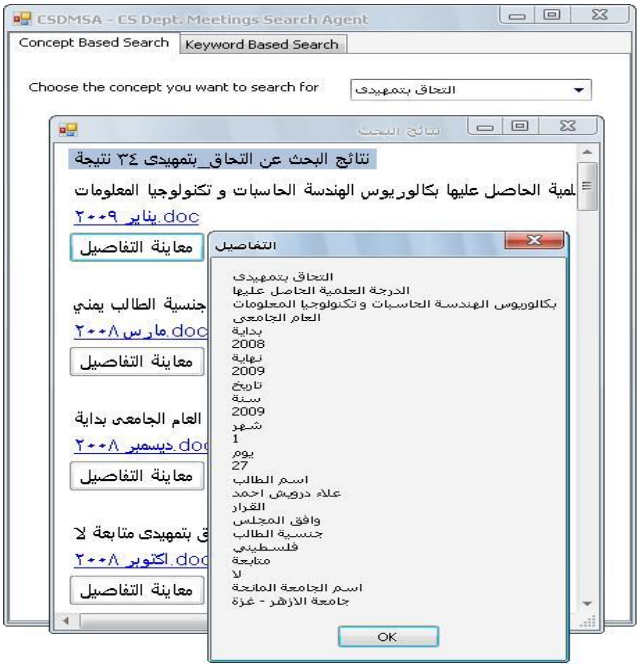

Figure 11. Concept-Based Search

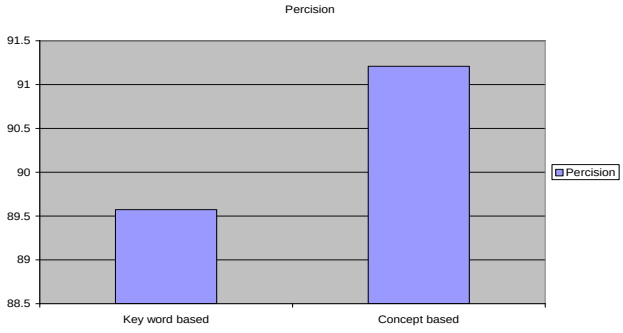

Figure 12. Average Mean Precision

References

1. Krill, P. "Microsoft to update Bing with semantic search", InfoWorld. http://news.techworld.com/applications/3211273/microsoft-to-update-bing-withsemantic-search/?olo=rss (2010)
2. Perez, J,. C. "Google Rolls out Semantic Search Capabilities", http://www.pcworld.com/businesscenter/article/161869/google_rolls_out_semantic_search_capabilities.html (2009)
3. Carmine Cesarano, Antonio d'Acierno, Antonio Picariello "An intelligent search agent system for semantic information retrieval on the internet " Fifth ACM CIKM International Workshop on Web Information and Data Management (WIDM 2003), New Orleans, Louisiana, USA, November, 2003. ACM 2003.
4. Juan L. Dinos, J. Fernando Vega-Riveros "A Document Ontology and Agent-Based RDF Metadata Retrieval" Thirteenth ACM Conf. on Information and Knowledge Management (CIKM'04), Washington DC, November 2004.
5. http://jade.tilab.com/
6. Cristiano Rocha, Daniel Schwabe and Marcus Poggi de Aragão "A Hybrid Approach for Searching in the Semantic Web" Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, USA, May, 2004. ACM 2004, ISBN 1-58113-844-X.
7. Li Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. C. Doshi, and J. Sachs. "Swoogle: A Semantic Web Search and Metadata Engine", Thirteenth ACM Conf. on Information and Knowledge Management (CIKM'04), Washington, November 2004.
8. Ramanathan V. Guha, Rob McCool, Eric Miller "Semantic search" In Proceedings of the Twelfth International World Wide Web Conference, WWW2003, Budapest, Hungary. ACM, 2003.
9. http://tap.stanford.edu/
10. Ramanathan V. Guha, Rob McCool "TAP: a Semantic Web platform" Computer Networks V. 42(5): P.557-577 2003.
11. http://www.w3.org/TR/rdf-schema/
12. Sunil Movva, Rahul Ramachandran, Xiang Li, Phani Cherukuri, Sara Graves: "Noesis: A Semantic Search Engine and Resource Aggregator for Atmospheric Science". NASA science technology conference 2007 NSTC2007.
13. Krȫtzsch, M. "Semantic MediaWiki", http://semanticmediawiki.org/wiki/Semantic_MediaWiki (2010).
14. Wiki news. (2008). "SMW now available in Arabic", http://semanticmediawiki.org/wiki/SMW_now_available_in_Arabic

نظام وكيل بحث مبنى على الشبكة الدلالية

فى ظل التطور السريع لشبكة المعلومات الدولية (الانترنت), ظهرت فكرة الانترنت الذكية والتى من خلالها يمكن لبرامج العملاء (Agents) ان تتعامل مع الانترنت مثل الانسان. وتعتمد فكرة الانترنت الذكية على توصيف البيانات الموجودة على الانترنت لجعلها مفهومة بالنسبة لبرامج الحاسب الآلى,

وبذلك يمكن لهذه البرامج مساعدة الانسان فى الأعمال التى يقوم بها على شبكة الانترنت. وعملية البحث من العمليات المهمة التى يجب أن تواكب هذا التطور لمساعدة المستخدم للوصول الى أفضل النتائج. وهذا البحث يقدم مقترح لعمل نظام وكيل بحث ذكى على الويب الدلالية. ويتم البحث بإحدى طريقتين وهما البحث بالكلمات المفتاحية والبحث المبنى على المصطلحات.

ويعمل هذا النظام من خلال أربعة مراحل أساسية:

- مرحلة تحليل معجم المصطلحات (انتولوجى) .
- مرحلة إنتاج المعانى التوصيفية للبيانات الموجودة بمحاضر مجالس الأقسام. ويتم تمثيل هذه المعانى التوصيفية باستخدام اللغات الموحدة المتعارف عليها والتى انشئت لهذا الغرض.
- مرحلة فهرسة المعانى التوصيفية الناتجة فى المرحلة السابقة.
- المرحلة النهائية وهى مرحلة البحث.