ARAGOG SEMANTIC SEARCH ENGINE
Beyond the Limits of Keyword Search

Madhur Garg, Jaspreet Singh, Jitesh Sachdeva, Harsh Mittal
Department of Computer Engineering, Netaji Subhas Institute of Technology, University of Delhi, Delhi, India

Sanjay K. Dhurandher
Department of Information Technology, Netaji Subhas Institute of Technology, University of Delhi, Delhi, India

Keywords: Semantic Web, Search, Semantic Search, Semantic Page Ranking, Ontology Ranking.

Abstract: The world is heading towards a phase of pure automation and artificial intelligence. In this context the science of exploring the possibility of computers interpreting the meanings of sentences is a topic of great interest. The search engines are in no way left behind from its impact. The prospects of having a semantic search engine that could explore the proper context of an input query and produce relevant results is being constantly looked for. In this backdrop we present our prototype – Aragog, which is even a step ahead than the conventional idea of a semantic search engine. This not only makes the user free from the hassle of browsing through hundreds of irrelevant results, but also generates results in an order that would match its intended context, with a high probability. The engine has been designed and tested in its nascent stage and the results have been found to be exemplary. Additionally, we have incorporated many other features such as synonym handling and explicit result display that make it all the more tempting to emerge as the next generation's search engine.

1 INTRODUCTION

Since its very inception the notion of a search engine has been to provide the web users with an interface that could look for appropriate content on the web. The AOL, Google and Yahoo! search engines have all restricted this idea to keyword searching which in the present scenario seems outdated and incomplete. The present generation search engines present a huge amount of search results to the user in response to the query, most of which are at times highly irrelevant, and the user has an ordeal in sifting through these result sets to arrive at some page of his interest.

The limitation of contemporary search engines has forced researchers to look for new alternatives. Semantic engines- which propose to derive meanings out of sentences seem to fit in place to alleviate out of this hitch. This can be better understood in the light of some examples. For instance, consider a query "*Winner of maiden T20 cricket world cup*". Intuitively, the user is interested in knowing the direct answer of the query which in this case turns out to be *India*. However, our dry run over some of the conventional search engines yielded us results which cater to the official T20 world cup site, site links for watching T20 world cup, web pages which are flooded with information not worth the user requirement. This clearly highlights the indispensability to look for better alternatives.

A semantic search engine uses ontologies which are a set of concepts mapped together and can be referenced as such to derive semantic associations among different words and concepts. The resources on the web, i.e., the web pages, are crawled and looked for annotations done on them, if any. These annotations are then used to set the words to that ontology with which they have been tagged. These tags are used later on for page searching. The matching and searching algorithms of Aragog have been explained in later sections.

The remaining paper is organized as follows. In Section 2, we discuss about the previous work done in this area. Next we present our motivation towards this work that has been taken up in Section 3.

Garg M., Singh J., Sachdeva J., Mittal H. and K. Dhurandher S. (2009).
ARAGOG SEMANTIC SEARCH ENGINE - Beyond the Limits of Keyword Search.
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development*, pages 21-27
DOI: 10.5220/0002290600210027

Section 4 talks about our proposed Semantic Search Engine Aragog's architecture. Section 5 explains the algorithms and the heuristics used in the various modules. Section 6 discusses the results of our implementation of Aragog. Finally we conclude our work and follow that up with the future work that can be done on this engine to enhance its capabilities.

2 PREVIOUS WORKS

The architecture for a semantic search engine was proposed in (Qazi Mudassar Ilyas, et al., 2004). However, this architecture proposes a two level interaction with the user whereby the user needs to separately mention both the search query and the domain in which the searching shall be performed.

Another important work in this field is (Li Ding, et al., 2004). It is a working implementation but it does not incorporate searching according to semantics. It restricts itself to keyword searching in the semantic web documents.

(Lee & Tsai, 2001) define the Lee's Model which uses a matching algorithm to reflect the semantic similarity of web page content but it is unable to do the same completely. Thus, it is not possible to satisfy user queries' to an appreciable extent.

This all reveals that semantic search engines are still far away from reality and a better solution needs to be found out.

3 MOTIVATION

Aragog has been conceived and developed with the following motivations:

1) A Semantic Search Engine should minimize on the number of user interaction levels for better usability. For a particular query, the domain in which the search is to be performed should be deduced automatically. This will bring the Semantic Search Engine at par with the existing keyword based search engine.
2) Synonyms of the keywords should be considered while deriving the semantics of query. Synonyms should be handled in the sense that for a given query, the results of all synonymous queries should also be displayed.
3) Apart from web resources results, a semantic search engine should also provide the user with the exact answer of the query. This would make it a search engine cum answering agent.
4) The query result display should enhance the user experience. This should include ranking amongst the domains and also, ranking within an individual domain. Along with this, relevant text from the web documents should also be displayed to the user.

4 PROPOSED ARCHITECTURE FOR ARAGOG

As discussed in the previous section, the motivation for building Aragog has come from various shortcomings and limitations with other existing semantic search engines. The proposed architecture of Aragog has been shown in Figure 1.

This architecture supports various features such as *ontology ranking, synonym handling, semantic answer finder etc*. The various modules are described below:

Query Preprocessor: This is the module which interacts directly with the user. The query preprocessor is responsible for accepting a query from the user. An acceptance list for this is maintained in a relational database and contains entries for tokens for which corresponding concepts exist in the *ontology collections*. The user's query tokens are verified with the acceptance list and only the tokens which are found in the acceptance list are accepted. Remaining are rejected by the preprocessor. This helps in rejecting helping verbs such as 'is', 'am', 'are' etc. The acceptance list is created/maintained/updated by the Ontology Crawler Module discussed later.

This module also takes care of the scenario when a user query contains similar meaning tokens. For example if the query posed by a user is "*maximum highest score of Sachin Tendulkar in Test*". Here, the concepts '*maximum*' and '*highest*' both correspond to the same meaning. Hence, redundancy is there in the tokens. The query preprocessor also removes the similar meaning tokens to avoid complexity and inefficiency that may occur at a later stage. This module also provides features such as support for double quotes in queries.

Ontology Ranker: This module is a major improvement over the previously proposed versions. An ontology ranker understands the user's query's context and finds the ontologies which contain the desired concepts. Apart from finding the relevant ontologies, this module also ranks the ontologies for

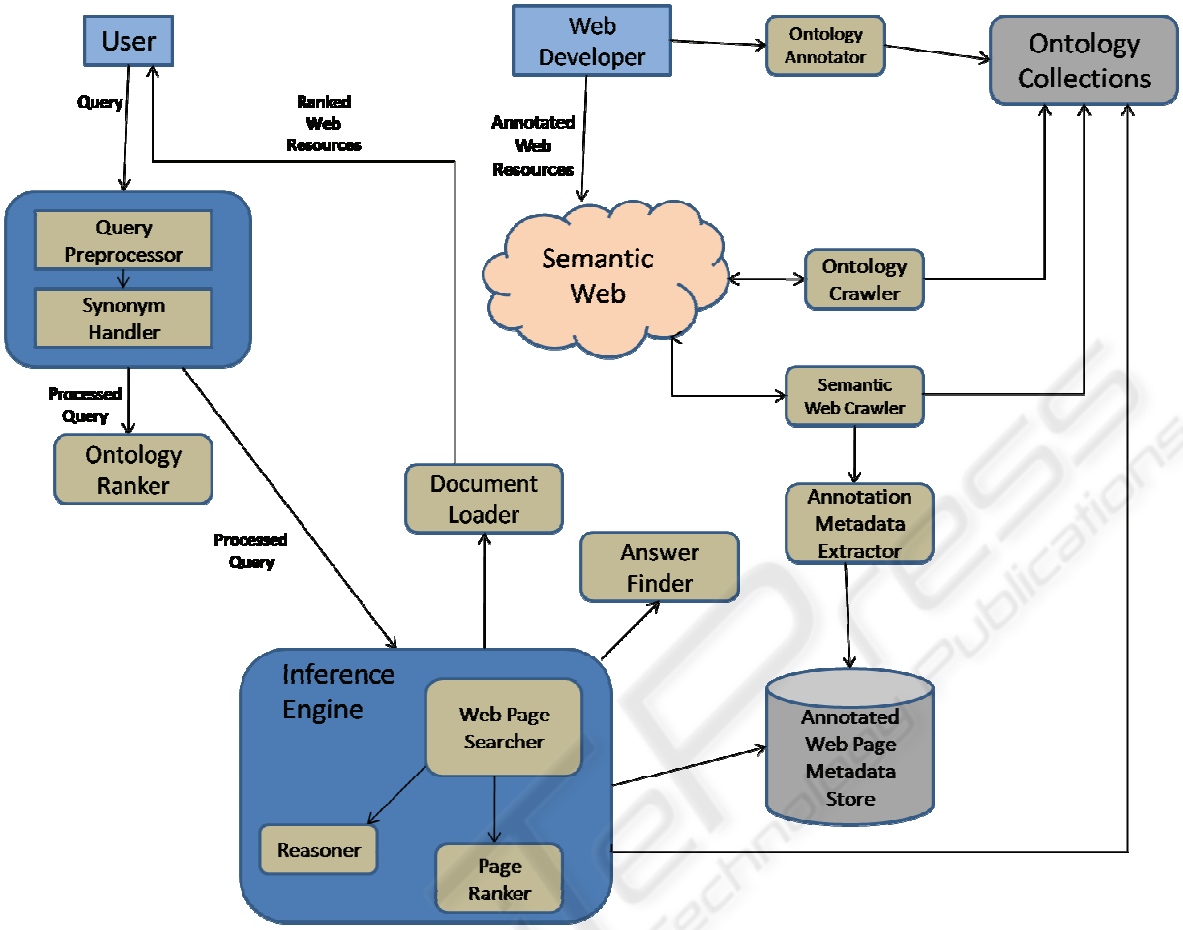

Figure 1: Architecture of Semantic Search Engine.

a given query's context.

As explained before, the previous versions expect the user to specify the ontology in which the concepts are to be searched. This module removes this second level interaction with user and hence, increases the responsiveness.

For example, if the user enters the query as "*Tiger Woods*". This query has two contexts. One refers to the golf ontology where *Tiger Woods* is a concept referring to a golf player. The second context refers to forest ontology where *Tiger* and *Woods* are two separate concepts.

The Ontology Ranker module uses various ontology ranking algorithms and heuristics to rank the ontologies. These have been discussed in later sections.

Inference Engine/Reasoner: The Reasoner's task is to infer new information from the given information to help understand the concepts in a better way. This is done by backward chaining to improve on the efficiency front.

For example, if in an ontology for the food Domain, we have a class hierarchy as Food --> Non-Vegetarian Food --> Sea Food. According to the transitive property of reasoning, it can be inferred that *Sea Food* is a type of *Food*.

Semantic Web Crawler: The Semantic Web Crawler crawls the web and searches for annotated pages on the web. Annotated pages are the web pages having concepts annotated on them. These annotations are done by the web developer while developing the pages using various tools available. The web crawler then retrieves such pages and passes these pages to the *Annotation Metadata Extractor Module*.

Ontology Crawler: This component is responsible for crawling web to find out Ontologies to build Ontology Collection. Apart from finding new Ontologies, this module performs the task of updating existing ontologies on finding new concepts.

Annotation Metadata Extractor Module: This module retrieves annotated web pages from the

semantic web crawler. It extracts the annotated metadata and concepts from the web page and updates the Metadata Store with the concepts extracted.

Annotated Web Page Metadata Store: This store is built by the Web Crawler Module and Annotation Metadata Extractor Module. This store contains the metadata in a relational database and consists of an index for all the annotated concepts in the webpage along with the URLs. The metadata also contains the information to be used by the *Page Ranker* Module.

Web Page Searcher: This module is responsible for finding the web resources suitable for the preprocessed query in a particular ontology context.

The searcher refers to the ontology and finds out the relevant concepts to be searched in the metadata store using the various Searcher Algorithms that have been discussed later. These relevant concepts are then searched in the *Annotated Web Page Metadata Store* and the corresponding URLs are retrieved.

Page Ranker: This module receives the list of URLs for a given query and ranks these URLs using various page rank algorithms discussed later. The point to be noted here is that the *Page Ranker* ranks only those URLs which correspond to a single ontology domain. Thus, the sorting done here is *Intra-Ontology Sorting* whereas the Ontology Ranker Module does *Inter-Ontology Sorting*.

Answer Finder: In certain cases, apart from the query result URLs, it is also helpful to get an answer of the query posed. For example if a user enters a query '*Director of Movie Black*', then along with the related web pages, it is really appreciated if the exact answer i.e. *Sanjay L. Bhansali* is given to the user. Answer Finder Module aims to provide this functionality.

Semantic Document Loader: This module is a normal document loader but with extra capability of loading relevant text of a URL depending on the query posed by the user. A local cache copy of each web resource is maintained in the Annotated Web page metadata store that is used by the semantic document loader. For example if a user places the query "*Movies of Shahrukh Khan*", the conventional search engine's document loader will display those section of the retrieved pages where either the keyword '*Shahrukh Khan*' or '*movies*' appears, but the Semantic Document Loader will display those sections in the result which contain the name of the movies of Shahrukh Khan.

In the next section, the various algorithms and heuristics adopted for Aragog shall be discussed.

5 PROPOSED ALGORITHMS AND TECHNIQUES FOR VARIOUS MODULES

5.1 Ontology Ranker

As stated in previous section, this module ranks the ontologies according to the query based on:

1) **The Presence of Keywords in an Ontology**:

Each word present in any of the ontologies has a set of numbers associated with it, where each number corresponds to an ontology. When the user enters a query we compute the intersection of the sets corresponding to each keyword of the query so as to determine the ontology containing all the words of the pre-processed query, which would result in narrowing down the list of ontologies which are required to be searched.

2) **The Position of Keywords in an Ontology:**

i) If the query consists of a single word, we calculate the depth of the keyword in various ontologies and the one having the keyword at the minimum depth is given the highest priority.

ii) If the query consists of multiple words, we calculate the Lowest Common Ancestor of the keywords (nodes represented by the keywords). The lowest common ancestor thus found must also be one of the keywords entered. Then the ontology having those keywords separated by minimum distance is ranked first.

5.2 Synonym Handler

The synonyms for a word (class name, instance name and property name) are inserted in the same node itself with the help of aliasing. This greatly reduces the time that would have been required in referring to a thesaurus for each word entered in the query.

5.3 Semantic Answer Finder

Here the ontology graph is traversed from the least common ancestor of the various nodes to the required instance and the property value of the required property is returned.

5.4 Page Ranker

1) The ontology (graph) is traversed from the property value of the required property (of the

required instance) followed by the property name to the least common ancestor of the keywords. A separate set S_i is constructed for the i^{th} node of the path, where $i=0$ corresponds to the property value node.

2) For each set S_i, a set of links L_i is searched, such that any of the words of S_i, appear in the annotation of those web pages.
3) If there were n nodes in the path, then two sets of sets are computed. Set V contains n sets, where the set V_i is computed from the intersection of the all the sets L_j, where $i \in [0,n]$ and $j \in [0,i]$, and set R contains $n-1$ sets, where R_k is computed from the intersection of sets L_m, where $k \in [1,n]$ and $m \in [1,k]$.
4) Since, the set V also incorporates the property values, so the links in the sets of set V will be preferred over the links in the sets of set R.
5) Each set V_i contains the set of links which contains the answer of the query i.e. the property value along with words from the 1^{st} node to the i^{th} node. Thus, V_n will give the set of links which are best suited for the query as it would contain the property value and all the keywords of the query (or their synonyms) which were present in the ontology.
6) Therefore, priority will be given to the links of the set V_i over the links of the set V_j, where i>j.
7) Similarly, the links in the set R_i will be preferred over the links in the set R_j, where i>j.

6 IMPLEMENTATION OF ARAGOG

A working implementation of Aragog has been developed. In this work we have covered three domains: Bollywood, Cricket and Food. The Ontologies were created for these domains. As OWL-DL is rapidly emerging as a standard to build Ontologies, we have used it to follow the worldwide standards. Protégé tool was used to design Ontologies.

For handling Ontologies and performing operations on them, we used Jena Ontology API . Jena provides us with the Reasoner based on the DL transitive rules. This Reasoner was used in our Inference Engine module.

We chose C#.NET as our development language and ASP.NET was chosen to build the web interface of the Aragog. As Jena API was available in Java, IKVM was used to convert the java source code into a .NET DLL. This development work has been carried on a Intel Dual Core 1.83GHz system having 3GB of DDR2 RAM and 320GB of SATA Hard Disk.

The Aragog search results were compared with Google for a set of queries. The results as presented below clearly highlight how Aragog outperforms the conventional keyword search engine.

Case 1: Imprecise Queries

Query 1: "Maiden T20 World Cup winner"
Ideal results: All web resources about India preferably in cricket domain.

Aragog Results:

Domain: Cricket
Answer: India
Top result:
http://www.cricinfo.com/database/NATIONAL/IND/ (Figure 2)

Google Results:

Top Result:
http://cricket.yahoo.com/cricket/videos/fvideo/210609_SL_PAK_2inn_hl/3222 (Figure 3)

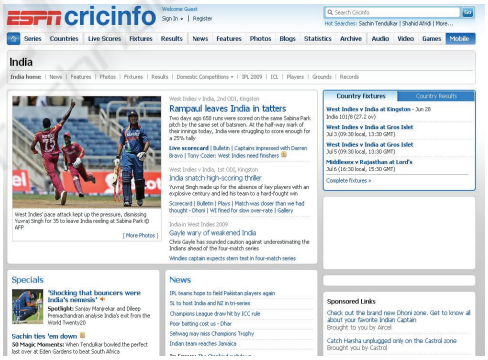

Figure 2: Screenshot of Aragog's top result for query 1.

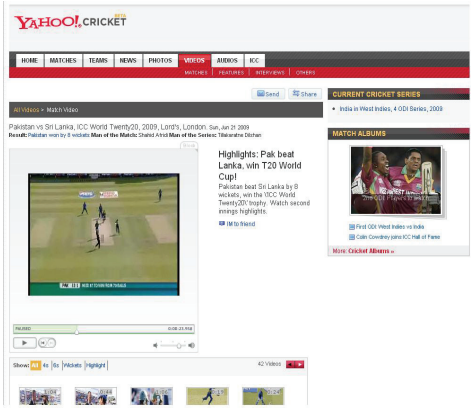

Figure 3: Screenshot of Google's top result for query 1.

Comparison:

It is clearly discernible from the results that Google results returned irrelevant pages which were not desired by the user as shown in Figure 3. On the other hand, Aragog returns a page on a cricket related website about India, as shown in Figure 2, which is first (inferred by 'maiden') T-20 world cup winner. Thus, Aragog succeeded in interpreting the correct meaning of the query and displaying the most relevant domain results.

Case 2: Queries containing some keywords spanning over multiple domains

Query 2: "Ingredients of 20-20"

Ideal Result: All pages containing any information regarding the contents of 20-20 biscuits (Brand: Parle-G)

Aragog Results:

Domain: Food
Answer: Wheat, Flour, Sugar, Butter, Milk
Top Result:
http://parleproducts.com/brands/biscuits_20-20Cookies.asp (Figure 4)

Google Results:

Top Result:
http://www.mindbodyhealth.com/MbhVision20.htm (Figure 5)

Figure 4: Screenshot of Aragog's top result for query 2.

Figure 5: Screenshot of Google's top result for query 2.

Comparison:

The difference in the quality of results is clearly perceptible. Aragog returns the correct answer page, as shown in Figure 4, whereas Google returns a web page as shown in Figure 5, which is not even remotely related to the food item in question.

Similarly, several other kinds of queries such as those concerning synonyms, intra domain ranking, inter domain ranking were also tested and the results demonstrate how Aragog outperforms the traditional keyword based search engine.

7 CONCLUSIONS

The idea of a novel semantic search engine has been proposed as well as implemented in this paper. The results have been found out to be refined, smarter and accurate than the conventional keyword based search engine. Aragog also lays the foundation of semantic search with minimum user intervention. Even the presentation of the results is such that the most apt results are available at a mouse's click. Further, on implementation the proposed Aragog was found to perform much better than the Google search engine.

8 FUTURE WORK

Aragog leaves us with a few possible future additions that can be made to broaden its searching horizons. Currently, Aragog searches in 3 domains – Bollywood, Cricket and Food. It can easily be extended to incorporate many more domains by adding respective ontologies to Ontology collection.

Aragog can also be easily extended to cover searching amongst videos. This can be done in two ways:

1) Aragog can search through the descriptions of videos available on the internet by simple text search.
2) A speech to text module can also be added to Aragog to allow it to get the contents of the video in a text format which can then easily be searched through to bring semantic search to video contents, which has never been done before.

REFERENCES

Mudassar Ilyas, Q., Yang Zong Kai, Adeel Talib, M., 2004. A Conceptual Architecture for Semantic Search Engine. In *Multitopic Conference, 2004. Proceedings of INMIC 2004. 8th International.* IEEE Press.

Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, Joel Sachs, 2004. Swoogle: A Search and Metadata Engine. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management.*ACM Press.

W.-P. Lee, T.-C. Tsai, 2003. An interactive agent-based system for concept-based web search. In *Expert Systems with Applications*. Elsevier.