GuNQ – A Semantic Web Engine with a Keyword based Query Approach

Nipun Bhatia, Prashant Gaharwar, Pratyus Patnaik
Indian Institute of Information Technology, Allahabad, India
nipun@ug.iiita.ac.in, prashant.gaharwar@ug.iiita.ac.in, pratyus@ug.iiita.ac.in

Abstract

The Semantic Web is an extension of the current web in which information is given a well-defined meaning, better enabling computers and people to work in cooperation. In this paper we present, GuNQ - A GUI based Semantic Web Engine and a keyword based query approach to achieve ease of implementation and usage of Semantic Web applications. The GuNQ supports loading and validation of OWL/RDF ontologies, conversion to N-tuples, inferencing, indexing of classes and URI keywords of the ontology and querying using standard query languages like RDQL and SPARQL as well as keyword based querying. We incorporated the keyword based query approach for the user unfamiliar with the syntax of complex query languages viz. SPARQL. This enables users to Query using simple keywords and generate an equivalent query in RDQL/SPARQL format. We employed GuNQ as an effective framework for hassle-free development of various Semantic Web applications including the famous Wine Recommender using Food and Wine Ontology, a Semantic Web based Search Engine, Computer Intrusion Detection System, Tagged Image Search and Recognition. Finally, we evaluated the performance of our keyword based query approach in conjunction with GuNQ. We found a high consistency between keyword based and standard query result.

1. Introduction

The Semantic Web [1] is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. The aim of this paper is to enable the web reach its full potential by enabling the data in the web to be shared and processed by automated tools as well as by people. In this paper we present, GuNQ: A GUI based Semantic Web Engine and a Keyword based query approach. We employed GuNQ as an effective framework for hassle-free development of various Semantic Web applications. We then compare the results from GuNQ which is a Jena[2] based Semantic Web Engine and uses RDQL[3] and SPARQL[4] as query languages for query, with the results from our Keyword based query approach. GuNQ has been coded using the Jena 2.3 API for Java. The GuNQ is a Semantic Web engine capable of querying ontologies created using Web Ontology Language (OWL) [5], Resource Description Framework (RDF) [6] and Extensible Markup Language (XML) [7].

The GuNQ supports loading and validation of ontologies. During the validation, inconsistencies in the ontologies are pointed out. The ontology then is converted into RDF N-tuples form. On these generated N-tuples we apply inferencing to generate the inferred N-tuples. This inferencing enables us to derive additional RDF assertions which are entailed from some base RDF together with any optional ontology information and the axioms and rules associated with the reasoner. In addition, indexing of classes and URI keywords of the ontology is also done by using Hash-tables in GuNQ whose utility is evident while designing Semantic Web applications like Semantic Web Search Engine. N-Tuples essentially represent a directed graph. On these inferred N-tuples, we apply the keyword based query approach, which makes use of the fact that N-tuples represent a directed graph. The primary use of this mechanism is to support the use of languages such as RDFS and OWL, which allow additional facts to be inferred from instance data and class descriptions. The results of this query approach are compared with query results on the inferred N-tuples using a standard query language like the RDQL or its evolved counterpart SPARQL, also supported by GuNQ. In addition, many existing/new Semantic Web Applications were built using GuNQ to demonstrate the power of GuNQ in terms of time-saved and ease-of-use.

2. GuNQ

GuNQ [fig. 1] is a GUI based Semantic-Web engine which allows a user to load an ontology of their choice and query it. It allows the user to select an ontology of his choice and load it onto the GuNQ using a user-friendly GUI. The ontology then is converted into N-tuples

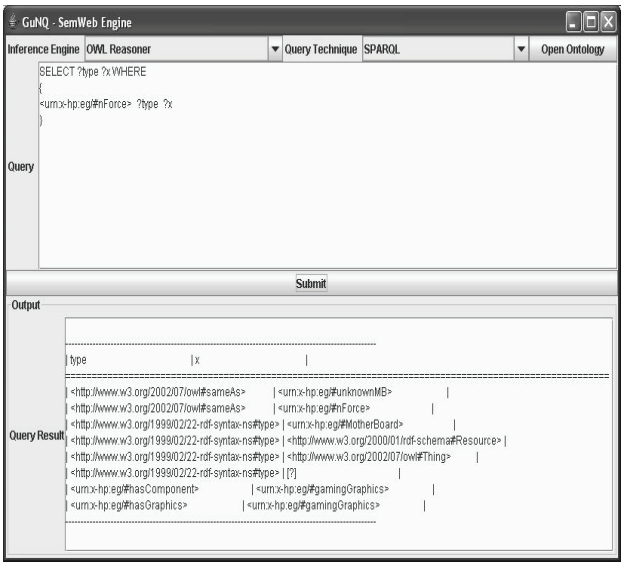

Figure 1

This conversion to N-tuples, makes the Ontology equivalent to a directed-graph. Every N-tuple consists of three parts: the subject, object and predicate. These are equivalent to: two nodes and the edge of a graph. The subject and object are the two nodes and the predicate defines the edge. The conversion to N-tuples is followed by inferencing and indexing. Indexing is incorporated using Hash-tables for quicker results in applications utilizing the feature.

GuNQ allows the user to use an Inference Engine [8] of his choice. It provides support to use any existing DL (Description Logic) reasoners: OWL Reasoner/RDFS Reasoner. An external reasoner (e.g. Pellet Reasoner) can also be used for inferencing. The GuNQ is completely scalable and any external reasoner of choice can be attached. The inferencing enabled us to unearth the implicit relations inherent of any ontology. GuNQ supports standard query languages like: RDQL and SPARQL. The user can use any one and write the query with the correct syntax in the text area. For example, the user can input the following query in SPARQL:

SELECT
?type ?x
WHERE
{?type
<http://www.w3.org/1999/02/22-rdf-syntax-s#hasColor>
?x }

The GuNQ then generates all those N-tuples that have the relation <http://www.w3.org/1999/02/22-rdf-syntax-s#hasColor> as the predicate.

This Engine is fully integrable with other JAVA applications, which can sit on top of GuNQ to provide extended functionalities. Since it is highly extensible, there is vast room for improvement. There can be separate modules for providing Ontology Interoperability, Inferencing, Indexing and Querying.

3. Applications

We employed GuNQ as an effective framework for hassle-free development of various Semantic Web applications. The Keyword based querying approach enabled us to develop popular Semantic Web applications with unprecedented ease. The applications include,

I. Wine Recommender using Food and Wine Ontology [9].

With most of the task automated in GuNQ interface, the user has to provide certain keywords, related to the food and its properties and the system recommends the Wine that goes best with it.

Though, this has already been dealt with in full detail, nevertheless, GuNQ automates the whole task of Ontology loading, indexing, n-triples generation and inferencing, appropriate querying (based on Keyword approach) making the development of this application, a lot easier and time saving.

With hash-tables in place for Indexing, the application gives quick and accurate results.

II. Semantic Web based Search Engine.

GuNQ acts as an effective Semantic Web based Search Engine. Hash table based indexing (of classes, keywords, and properties) ensures that the results are obtained quickly for the given ontology.

GuNQ can be easily integrated with a Semantic Web Crawler to make the searches, truly global. One more area which needs examination is Ontology Interoperability, since, several Ontologies that seem different may be about the same domain. Integration of an intelligent Agent with GuNQ for Ontology Interoperability is very viable as and when the agent is completely developed.

III. Computer Intrusion Detection System.

GuNQ effectively works as a Computer Intrusion Detection System. Using Ontology described in [10], GuNQ indexes it. When a system log in RDF form is examined in context with the ontology, a quick and

accurate Intrusion-Detection report is obtained, using the indexed Ontology and inferencing.

IV. Tagged Image Search and Recognition.

In this application, a directory of Animal Images (as a working example) with an associated mark-up file with each image is stored on a machine. A corresponding Ontology for Animals was developed and loaded in GuNQ. This setup had two uses:

a. Image Search.

Any animal-based keyword/keywords query, covered in the Ontology, interprets the animal being searched for and then a simple Java tool, searches for the traits in mark-up files associated with the animal images. A match displays the animal image and also images of those animals that are somehow closely related to that animal.
e.g.
Query: trunk ivory
Result: Elephant Image
Analogical Results: A Walrus's Image, it also has tusks!

b. Image Recognition.

If the user describes an image using keywords and if GuNQ recognizes it as some animal from the image files then, it instantly recognizes the user's image as that animal.
Query: trunk ivory (as description of user's given image)
Result: It is an Elephant!

On similar lines, several Semantic Web Applications may be built in minimum time using GuNQ.

4. Our Keyword based Query Approach

For querying in a Semantic Web Portal, the user needs to have a broad intricate knowledge of the Querying languages like RDQL or SPARQL. It is indeed difficult for general novice user to query Semantic Web Systems because of its concepts of URIs [6] and using tagged information, which is very essential for making the machine understand what user wants to say: The Very Backbone of Semantic Web. The widely acceptable interface has to be the simplest: the plain text box where user can enter what he/she is looking for. In addition, the queries are also usually very short and only very few users want any more options.

In GuNQ, we have RDQL and SPARQL support for expert user. In addition, for novice user we have incorporated a Keyword based query system [fig. 2] on the topics, which even supports complex queries.

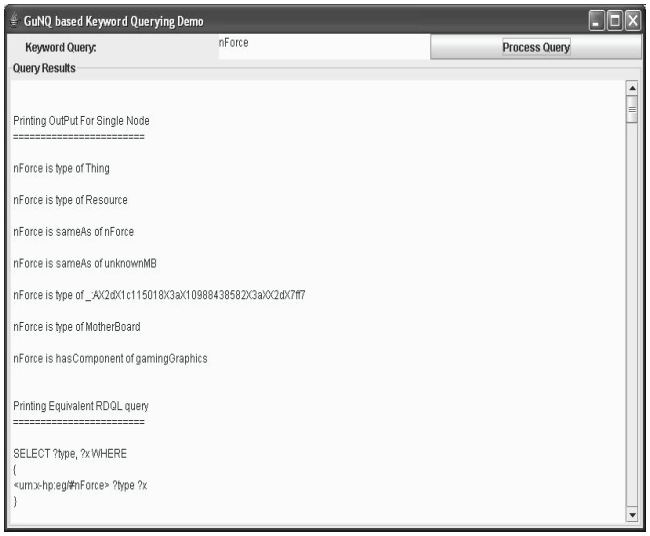

Figure 2

URI: <http://www.w3.org/1999/02/22-rdf-syntax-s#type>
KEYWORD: type

The Inferred N-Triples from the Engine, which essentially represent an RDF-directed Graph, are represented in the form of an adjacency matrix. It has been represented as a sparse matrix due to the inherent sparse nature. A user query contains Keywords. The Program detects the nodes and edges present in the keywords the user enters in his/her Query. It then finds the relationship between the nodes and edges, present in the keyword query, and gives the relationships between them (query result). It also generates Equivalent Query in RDQL/SPARQL format, which would generate the same set of results. For complex multi-leveled queries, general and ontology specific heuristics are used for filtering the query result. To prune out query results that might not interest the users. Then, similarly, a ranking of the Equivalent Query in RDQL/SPARQL format is done and, then, subsequently the query results are ranked.

5. Results and Conclusion

Experimentation was done by querying on Wine Recommender, Semantic Web based Search Engine, Computer intrusion detection system and Tagged Image Search and Recognition developed on GuNQ. For simple queries we had a 99.7% match between Keyword based query results and RDQL/SPARQL query results. In addition, a 100% match between Equivalent Query in RDQL/SPARQL format generated by the program and the intended RDQL/SPARQL query. For complex queries we had a 98.6% match between the ranked one Keyword based query results (without any heuristics based pruning) and RDQL/SPARQL query results. And a 99.3% match

between ranked one Equivalent Query in RDQL/SPARQL format generated by the program and the intended RDQL/SPARQL query.

Thus, this query approach not only enables Novice users to Query using simple keywords but also generates an equivalent query in RDQL/SPARQL format for their intended query, which the user can further exploit.

Moreover, GuNQ enables quick and easy development of Semantic Web applications, which is necessary for realizing Semantic Web.

6. Future scope

As we can see, there is not a 100% match between Keyword based query results and RDQL/SPARQL query results. This inconsistency results because the URI is being ignored while we try to process the query only based on the Keyword. This inconsistency may be amplified in a semantic web portal where there is a need for interoperation of two or more ontologies of the same domain. Thus, it also demands that Ontology Interoperability features be also used in GuNQ.

We are also trying to incorporate WORDNET to broaden the range of our keyword-based queries. In addition, future work can be directed towards an NLP (Natural Language Processing) based query processing.

7. References

[1] T. Berners -Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.

[2] Jena, http://jena.sourceforge.net/

[3] Libby Miller, Andy Seaborne, Alberto Reggiori, "Three Implementations of SquishQL, a Simple RDF Query Language", *ISWC2002*

[4] SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/

[5] OWL Web Ontology Language Reference, http://www.w3.org/TR/owl-ref/

[6] Resource Description Framework (RDF), http://www.w3.org/RDF/

[7] Extensible Markup Language (XML), http://www.w3.org/XML/

[8] Youyong Zou, Tim Finin, Harry Chen, "F-OWL: an Inference Engine for the Semantic Web", *FAABS III*, April 2004, Greenbelt Maryland.

[9] Wine Recommender Example, http://www.w3.org/TR/2004/REC-owl-guide-20040210/#WinePortal

[10] J. L. Undercoffer, A. Joshi, T. Finin, J. Pinkston. "A Target Centric Ontology for Intrusion Detection: Using DAML+OIL to Classify Intrusive Behaviors", *Knowledge Engineering Review*, January 2004