Control Natural Language Query Editor Guided by OWL/XML Path

Mr. Kasar Sandeep Popat
PG student of MET Institute of engineering
Nasik , India
Kasarsandeep1@gmail.com

Prof. M.U.kharat
MET Institute of engineering
Nasik , India
mukharat@rediffmail.com

***Abstract*—** It is become difficult when common user try to access semantic web application or any one ontology database-like gene ontology, plant ontology etc. Because either he/she don't know about ontology technology or about no of concepts present in the ontology database. So only expert user can handle the applications or database. To avoid this problem, I have developed the Controlled natural language query editor guided by xml path. Here ontology data is get used in the xml file format and then that file is get used to generate ontology tree. Ontology tree is work as suggester for user in the generation of their query. Query then fire on xml/ontology file to retrieve desire result.

Keywords- *Natural language interfaces (NLIs) , ontology languages , query formulation, user interfaces.*

*****

I. INTRODUCTION

The semantic web [1] is the next generation of current web standard and technology. The knowledge representation technology used in semantic web is ontology. Ontology [2] is the key component for creating meaning to knowledge bases; make searching fast also it improves crawling of the web. Ontology get used in the design of knowledge bases like e-learning, medical models, knowledge diagnostic system and semantic web.

To use different knowledge bases, casual user come across some difficulties, like lack of concepts known in the particular domain. So to avoid this difficulties many researchers try to bridge this gap by using the natural language interface-NLI[3].But NLI have some problems like linguistic variability and ambiguity. To avoid those problems Controlled Natural language CNL[4] is get introduced .CNL have two main properties 1) their grammar is more restrictive than that of the general language.2) their vocabulary contains only fraction of the words that are permissible in the general language.

In this paper I have presented I have presented XML path a CNL bases NLI that assist user in designing there query. First xml file is get browse ,from that file ontology tree is generated. User use that tree to generate a query, after using navigation algorithm desire result is get achieved. The paper is organised in following section 2) related work. 3)CNL editor.4)conclusion. And references.

II. RELATED WORK

In recent years, the utilization of NLIs and CNLs toward an effective human–computer interaction has received much attention in the context of the semantic web. Several platforms have been developed to function as either natural language ontology editors or natural language query systems. Two good examples in the first category are CNL editor (formerly Onto Path) and Guided Input Natural Language Ontology Editor (GINO) [5].Onto Path is in fact situated in the frontier between these two categories because it manages and creates RDF ontologies and it is also capable of defining queries from natural language sentences. It is composed of three main components in a layered architecture "*Onto Path-Syntax*" in the syntax layer "*Onto Path-Object*" in the object layer and "*Onto Path-Semantic*" in the semantic layer. In the upper layer a knowledge engineer and a domain expert can work together to define the domain ontology by using "*Onto Path-Semantic*." With this tool it is possible to build a new ontology or edit a previously existing one. After defining a set of concepts and their corresponding relationships the system returns the ontology in an RDF file. In the next layer "*Onto Path-Object*" assists domain experts which have no knowledge of ontologies in graphically expressing natural language descriptions by using nodes and arcs that correspond to the elements in the ontology. This graphical description is then stored as RDF triples. Finally in the lower layer "*Onto Path-Syntax*" guides users in the query generation process through a simple visual interface. The query is formed from the knowledge available in ontology and is translated into RDF.

The ontology-based CNL editor extends Onto Path in providing a context-free grammar with lexical dependence for defining grammars. By using defined grammars, the CNL editor enables the system to get structured data from writer's narratives with sophisticated, patternized, and informal expressions.

With all the editor provides guidance on proper choice of words and translates the results into RDF triples. The architecture of the CNL editor consists of five components: an interface through which the system recommends proper next words to the writer; a parser which processes an incoming sentence and

Determines the dependences; a predictor which examines the relations in the domain ontology to make a recommendation; a lexicon pool which sends the candidate next words to the interface; and a triple generator which generates RDF triples when the sentence is completed.

GINO allows users to edit and query any OWL knowledge base using a guided input natural language akin to English. The user inputs a query or sentence into a free form text field and based on the grammar the system's incremental parser offers the possible completions of the user's entry by presenting the user with choice pop-up boxes. These pop-up menus offer suggestions on how to complete a current word or what the next word might be. The GINO architecture consists of four parts: a grammar compiler which generates the necessary dynamic grammar rules to extend the static part of the grammar; a partially dynamically generated multilevel grammar which is used to specify the complete set of parse able questions/sentences and to construct the SPARQL statements from entered sentences; an incremental parser, which maintains an in-memory structure representing all possible parse paths of the currently entered sequence of characters; and an ontology-access layer which is implemented with jena .

In the category of natural language query systems, Portable natural language interface to Ontologies (PANTO)[6] and NLP-Reduce are two representative examples. PANTO is a system that takes ontologies and natural language queries as input and whose output is a series of SPARQL queries. When ontology is selected as the underlying knowledge base, PANTO uses the so-called "*Lexicon Builder*" to automatically extract entities out of the ontology in order to build a lexicon. This lexicon is used to make sense of the words that appear in a natural language query. Once the user has entered a natural language query, PANTO produces a parse tree which is then translated into SPARQL. NLP-Reduce, on the other hand is a domain-independent NLI for querying semantic web knowledge bases. Its architecture consists of five parts: an interface which allows user to enter full natural language queries, sentence fragments, or just keywords; a lexicon which is automatically built by extracting all explicit and inferred subject–property-object triples that exist in the knowledge base; an input query processor which reduces a query by removing stop words and punctuation marks; a SPARQL query generator which generates SPARQL queries from the input text; and an ontology access layer, which uses Jena and the Pellet reasoner .In other similar approaches are examined and the usefulness of NLIs is analyzed.

The authors came to the conclusion that casual end users strongly prefer querying using full sentences rather than keywords or any other means. In several related systems are analyzed and the exploitation of NLIs in a range of capabilities (e.g., the authoring of knowledge content the retrieval of information from semantic repositories and the generation of natural language texts from formal ontologies) is reviewed. In this report, the idea that CNLs could replace conventional semantic web ontologies was also explored but finally dismissed.

III. CNL EDITOR

The global architecture of the proposed system is shown in Fig. 1. CNL editor is composed of four main components:

A. *Ontology tree*
B. *Grammar checker*
C. *Ontology repository*
D. *Result generator*

In a nutshell the system works as follows. First user make login into system. He browse the ontology xml file, Then using that file ontology tree is get generated .Now this Ontology tree is useful for user for typing the keyword in search box.

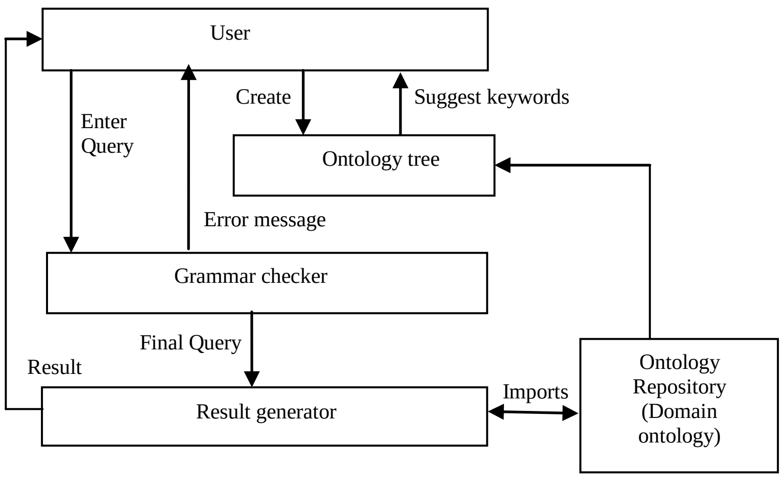


Fig.1.System Flow

User generates his query then clicks the search button. Now grammatical correctness of sentence is get check by grammar checker. Then query is get fire on the ontology files and we get the desire output result.

A. *Ontology tree*

Ontology is used to give meaning to the knowledge base. In ontology –different concepts are get represented in hierarchical manner, just like tree. In ontology tree generation xml document are get used .Here I am going to extract the different node present in the xml document, those all nodes are get shown in hierarchical manner. That is going to form an ontology tree. This tree is useful for suggesting keywords to the end user. So user can easily formulate his query in search box.

B. *Grammar Checker*

The main objective of the Grammar checker is to verify the grammatical correctness of the sentences generated. Grammar checker applies some rule to check the sentence. Actually the sentence is get divided into number of tokens. Then those tokens are arranged into group of tree according to the bacus normal form grammar (BNF-grammar).Then correctness of the tree is checked. In this project sentence is should be in the syntax as follows –view /any/resort/located_in/mountain[range=Himalaya].If this syntax is not get follows, the grammar showing the syntax error as-no token found in code or just no match found to the normal user.

C. *Ontology Repository*

The Ontology repository represents the knowledge base of the system. It includes the domain ontologies that are going to be queried by the end user through our tool. Domain ontologies can be created in the protégé ontology editor tool or we can access already present ontologies in the protégé. Ontologies from protégé are in the Owl/xml or RDF/xml format so those are converted into the complete XML file. After those xml file are get stored in the knowledge base. Now ontology repository is ready to use. For ontology tree generation, data searching, this repository is very important. Due to xml file use repository is becomes very handy in use, we can use it on any one platform.

D. Result generator

This module is generating final result and displays it on screen. Here Navigation algorithm is get used to search the final output. Output is display as the ontology tree. XPathdocument method is used for accessing the xml file, XPathNavigator method for navigation of xml file and XpathNodeIterator method is get used for searching the node in xml tree. Using those methods with xml tree navigation algorithm we can find result from ontology file. Then that result is shown back to the end user.

E. Algorithm

Xml tree navigation algorithm is get used here for searching the desire ontological node.

Inputs-

Q-query-expression
D-doc-filename
OT-ontology tree

Outputs-

RO-resulted ontology

Procedure-

```
GetElement(Q)
Navigator nav = set to root node,
Compile expression,
Fix iterator = root,

While
     iterator ++,
     Navigator nav2 = move to first node,
     Extract node  Є  OT,
     Add node To resulted Ontology  Є  RT,
     For
       each ontology node,
       Navigate node
       Add to RO,
End while,
If
   iterator.count= =0
   no match found.
end
```

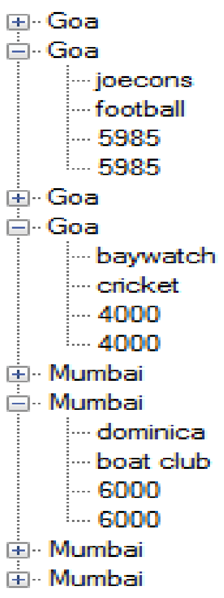
IV. RESULT

The basic input query type by the user should be as follow.

View/any/resort/located_in/beach

Fig.2.input

And the output/result for above search is as follows-



```
+ Goa
- Goa
    joecons
    football
    5985
    5985
+ Goa
- Goa
    baywatch
    cricket
    4000
    4000
+ Mumbai
- Mumbai
    dominica
    boat club
    6000
    6000
+ Mumbai
+ Mumbai
```

Fig.3.output

When common user try to access the ontological database, he come across the problem of limited knowledge of particular domain. Sometime user get confuse how to start to search the desire data. Here this project provides solution as shown in above figures. Controlled natural language provide the limited words with specific syntax for user as in fig.2.Furter user get assisted by the ontology tree. This ontology tree contains the full knowledge base, this knowledge base is presented in the tree format. So common user can navigate the tree for understand the knowledge base. After navigation he peek up the words which he want to search. Then he type those words to formulate the query. This query then get used to search the result. We get the result as shown in fig.3.The result is also in the form of ontology tree.

Project basically help to the casual user much more as compare to the expert user. When two or more user both expert and non expert are using the system, then time elapsed in typing a query is given in the following table-

TABLE I.

ANALYSIS OF RESULT

Input	**With guide ontology**		**Without guide ontology**	
Words	**Time elapsed in sec**		**Time elapsed in sec**	
In No	***Common user***	***Expert user***	***common user***	***Expert user***
1	27	10	600	300
2	40	22	720	420
3	53	30	780	540
4	70	40	900	660
5	100	60	1020	780
6	126	80	1200	900

V. CONCLUSION

Thus developed Control Natural Language Query Editor Guided by xml path that accepts natural language queries and outputs the results retrieved from ontological knowledge bases. In use of natural language as interface there are some problems as 1) It is hard to adapt them to new domains 2) there is typically a mismatch between the user's expectations and the capabilities of the natural language system namely, the habitability problem and 3) NLIs suffer from linguistic variability and ambiguities. This limits the expressiveness of the user.

CNLs aim to overcome some of the shortcomings of NLIs. CNLs are subsets of natural languages whose grammars and dictionaries have been restricted to reduce or eliminate both ambiguity and complexity. So it believes that the combined effects of NLIs and CNLs can help to bridge the gap between end users and the semantic web. So that, developed Control Natural Language Query Editor Guided by owl/xml path that accepts natural language queries generated by user and outputs the results retrieved from ontological knowledge bases. The use of ontologies for defining both grammar and knowledge about the domain application makes the platform portable and domain independent. Furthermore the logical basis of ontology languages allows for the elaboration of more powerful grammars.

REFERENCES

[1] N. Shadbolt, T. Berners-Lee, and W. Hall, "The semantic web revisited," *IEEE Intell. Syst.*, vol. 21, no. 3, pp. 96–101, Jan./Feb. 2006.

[2] X. Jiang and A.-H. Tan, "Learning and inferencing in user ontology for personalized semantic web search," *Inf. Sci.*, vol. 179, no. 16, pp. 2794–2808, Jul. 2009.

[3] A. Bernstein and E. Kaufmann, "Gino—A guided input natural language ontology editor," in *Proc. Int. Semantic Web Conf.*, vol. 4273, *Lecture Notes in Computer Science*, I. F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, Eds., 2006, pp. 144–157.

[4] U. Muegge, "Controlled language: The next big thing in translation?" *ClientSide News Mag.*, vol. 7, no. 7, pp. 21–24, 2007.

[5] A. Bernstein and E. Kaufmann, "Gino—A guided input natural language ontology editor," in *Proc. Int. Semantic Web Conf.*, vol. 4273, *Lecture Notes in Computer Science*, I. F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, Eds., 2006, pp. 144–157.

[6] C. Wang, M. Xiong, Q. Zhou, and Y. Yu, "Panto: A portable natural language interface to ontologies," in *Proc. ESWC*, vol. 4519, *Lecture Notes in Computer Science*, E. Franconi, M. Kifer, and W. May, Eds., 2007, pp. 473–487.

[7] http://msdn.microsoft.com/en-us/library/0ea193 ac%28v=vs.110%29.aspx

[8] Rafael Valencia-García, Francisco García-Sánchez, Dagoberto Castellanos-Nieves, and Jesualdo Tomás Fernández-Breis .IEEE transaction on system, man, and cybernetics—part A: system and humans, vol. 41, no. 1, jan 2011