Volume 4, Issue 8, August 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Semantic Tagging and Sigmoid Ranking Measure to Automatic Mapping of Natural Language Query to SQL Query

[1]B. Sujatha, [2]Dr. S. Vishwanatha Raju, [3]S. Nagaprasad
[1, 2, 3] Computer Science Engineering Department of CSE,
Hyderabad, India

Abstract: ***The automatic mapping of natural language questions into programming language is a significant problem for the retrieval of required data from any type of database. This reduces the risk of studying the query language completely so researchers have been developed various techniques for automatic mapping of NLP queries to database query. Accordingly, we have developed a SQL query parser using the semantic tagging and sigmoid function-based ranking measure. At first, NLP query is pre-processed and informative words are identified then to find semantic words from the ontology. After that, candidate queries are generated based on the lattice structure which have network of connection linked with family of group-like algebraic structures. Once the candidate queries are generated, optimal SQL query is selected based on the rule definition which is given by the domain experts. The selection criteria is mathematically formulated as ranking measure that is developed by including the number of rules which follows the rule definition and reverse sigmoid function. The experimentation is conducted with database, containing 10, 000 of data records individually in three tables with two set of queries. For both the query set, the proposed approach reaches the maximum accuracy of 80% with less computation task.***

Keywords: ***Database, SQL query, Natural language interface, Ontology, Lattice structure, Ranking measure, Accuracy***

I. INTRODUCTION

IN the early days of database systems, there was no idea of naive end-users accessing the data directly. This was prepared by a skilled programmer writing a special computer program. The reason for this was the 'navigational' nature of the data model applied by these early database systems. The user not only requires knowing about the configuration of the data in the application and also, he/she moreover required to know several programming tricks. At the time, the improvement of the relational model of data had a main impact on database systems. The only storage structure is the table in the relational model and this was something that still naive users could understand. Comparatively, simple declarative query languages, such as SQL were developed for this division of user [3]. After applying the SQL language to most of the businesses which require these kinds of applications, the user who doesn't know about the SQL query still faces challenges to employ those business data. At present, NLIDBS [3] is brought in to automatically mapping natural language into programming language semantics which is a main and interesting challenge in the field of computational linguistics as it may have a direct impact on industrial and social worlds.

For instance, accessing a database needs machine-readable instructions that not everybody is assumed to know. Users should be competent to create a question in natural language devoid of knowing either the underlying database schema or any difficult structured machine language. The progress of natural language interfaces over databases (NLIDBs) [17-21], that interpret the human intent into machine instructions employed to answer questions, is certainly a classic problem that is turning into of greater importance in today's world. To signify the domain, a dependable NLIDB system should offer sufficient coverage of patterns. Secondly, the system should permit for interactivity with the user. The user must be competent to understand the system reactions in the case of failure of generating SQL query. Thirdly, overcoming the problem of producing mapping rules from syntax based parse tree to SQL [2]. Hence, the goal is to produce an entire and correct natural language representation of a random SQL query, where wholeness means that should present all data which is essential for the user to understand the substance of the query.

Literature offers various algorithms to map natural language to database query in order to accomplish those objectives. Most of the algorithms used are the rules, interactive knowledge, template, statistical algorithm and ontology to plan the NLP query to database query. In the rule-based interface, SPARQL [10] was erected typically based on keyword search and question answering, applies reduction and alternative rules to develop the legibility of the verbalization, and a realization step which produces the last natural language representation of the query. Moreover, in [4], four interfaces were brought in each permitting a dissimilar query language and give usability from an end user's point of view. The approach given in [5] routinely gets domain knowledge from user interactions for the interactive knowledge-based approaches, and integrates the knowledge learned to develop the generic system. In [2], an approach was progressed for creating conversation-based natural language interfaces to relational databases by joining goal oriented conversational agents and knowledge trees. For the template-based interface, in [9], they have offered a work in building a template-based system for translating English sentences into SQL queries for a relational database system. In addition, in [11], they

explained a method to erect and semi-automatically annotate these kinds of data, containing pairs of NL questions and SQL queries.

Mapping at syntactic level is carried out and then machine learning algorithms are applied in order to develop an automatic translator [13] of natural language questions into their associated SQL queries. An approach [16] was suggested by modelling queries in a higher-order version of Codd's tuple calculus and applied synchronous grammars extended with lambda functions to signify semantic grammars. In addition, an approach [1] was modelled for routinely explaining a factoid question in natural language to an SQL query that regains the correct answer from a target relational database (DB). In [14], they have applied structural kernels and their amalgamations for inducing the relational semantics among pairs of NL questions and SQL queries. They have given an approach in [15], for translating natural language questions into SQL queries by using the MySQL framework for both hypothesis construction and thesis authentication in the task of question answering. Newly, ontology was furthermore employed for automatic translations of NLP query. In [8], they have used the ontology, with a series of relative assessments for the customization process of a commercial interface. In [12], they have offered a framework to query for information about a software system with OWL ontology and employ knowledge processing technologies from the Semantic Web to query it. In [6], they have developed architecture to offer end-users with a means to query ontology based knowledge bases by means of natural language queries.

After analyzing the literature, we have developed a SQL query parser using the semantic tagging and sigmoid function-based ranking measure in this paper. The proposed query parser does the automatic mapping of natural language questions into programming language. At first, NLP query is pre-processed using stop word removal and stemming process to identify the informative words. Then, informative words are given to ontology to find semantic words which are used to obtain more meaningful words. After that, candidate queries are generated based on lattice structure and the template. From the large set of candidate queries, the optimal SQL query is selected based on rule definition defined by the domain experts. This is done by the ranking measure which have the parameters of number of rules followed the rule definition and reverse sigmoid function. The paper is organized as follows: Section 2 presents general architecture of NLP interface to SQL query engine and section 3 define the problem with mathematical background. Section 4 describes the proposed approach of NLP interface to SQL query engine using semantic tagging and ranking measure. The experimental results and the validation of the results are given in section 5. Finally, the conclusion is presented in section 6.

II. GENERAL ARCHITECTURE OF NLP INTERFACE TO SQL QUERY ENGINE

The automatic translation of natural language into programming language is an important problem for the retrieval of information from any type of database. The database is well structured information arranged in a well planned way. But, the retrieval of relevant information from the database needs a programming language which requires much knowledge in the field. This prohibits the usage of the database from the person who is not aware about any language. In order to avoid that situation, natural language interface is used to build up here to map the normal questions to SQL queries automatically. To translate, the general architecture used here is shown in figure 1 which contains four basic blocks such as, NLP query block, Query parser, SQL query engine and SQL database. The process of getting information is explained with five steps in figure 1. Initially, NLP query is given to query parser which is the core block of this architecture. The query parser parses the natural questions into programming language using different type of automated natural language processing techniques. Then, SQL query is given to the query engine to do the process after connecting with the SQL database. The final output which is required for the user is given as output for the user those who are zero knowledge about the database or query.

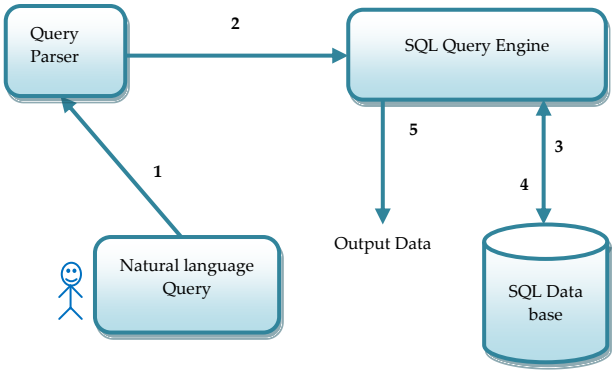

Fig 1. General architecture of NLP interface to query engine

III. PROBLEM DEFINITION

Let assume that the SQL database D having the required information of the user u in a structured way. The SQL database has a set of tables T and each table having table elements, $T.E$ which contains tuple of records relevant to the field. The information can be retrieved from the SQL database D by putting the SQL query Q_{SQL} which is in the form of $\{SELECT_s(OPERATION) \times FROM_f \times WHERE_e(CONDITION)_c\}$.

Assume that user u is not aware about the structure and the protocol of the SQL query Q_{SQL} but the user want to get information from the SQL database D . So, the user u is passing NLP query Q_{NLP} which is in the English language to the system. The NLP query Q_{NLP} is in the format of $\{S_i\}; \quad 0 < i < n$ where, S denotes a set of English words. In other

words, NLP query Q_{NLP} can be set of name of the tables, table elements, values of the tuple and other words, $S_i \in \{T_i, T.E_j, V_k, N_l\}$. The objective of the proposed system is to transform the query Q_{NLP} to SQL query Q_{SQL} as per the following formulation.

(1)

$$Q_{NLP} \Rightarrow Q_{SQL} = \begin{Bmatrix} SELECT_s\,(OPERATION)_o \times \\ FROM_f \times WHERE_e\,(CONDITION)_c \end{Bmatrix}$$

The above equation means that selecting of s from the table f after satisfying the condition c and then doing the operation o. This can be represented as,

$$\exists s \in I, \quad \exists o \in J, \quad \exists f \in T, \quad \exists \omega \in e, \quad \exists c \in g,$$
$$s.t. \quad \Pi_s\left(\xi_o\left(\sigma_f\left(g_c\right)\right)\right) \qquad \text{answers} \qquad Q_{SQL}$$

Where, Π represents the selection, σ represents projection, ξ represents operation and g represents the conditions of the SQL query.

IV. PROPOSED APPROACH OF NLP INTERFACE TO SQL QUERY ENGINE USING SEMANTIC TAGGING AND SIGMOID RANKING MEASURE

The proposed approach of automatic mapping of NLP queries to SQL queries using the semantic tagging and reverse sigmoid-based ranking measure is discussed in this section. As like the general architecture, the core step of the NLP interface is query parser which is the major contribution of this work. The query parser is designed with four important steps like, informative word segmentation, semantic tagging, candidate query generation and optimal query selector. These four steps are the heart of the proposed approach designed with more benchmark algorithms and the unique ranking measure. The overall architecture of the proposed approach is given in figure 2. Here, NLP query Q_{NLP} given by the user u is given to query parser. The first step of the query parser is to remove the stop words and converting the remaining words to the root form. Then, semantic tagging is performed for every words based on three steps. Subsequently, candidate queries are generated using rule matching strategy to do the final filter using the ranking measure which find the useful SQL query for the input NLP query Q_{NLP}.

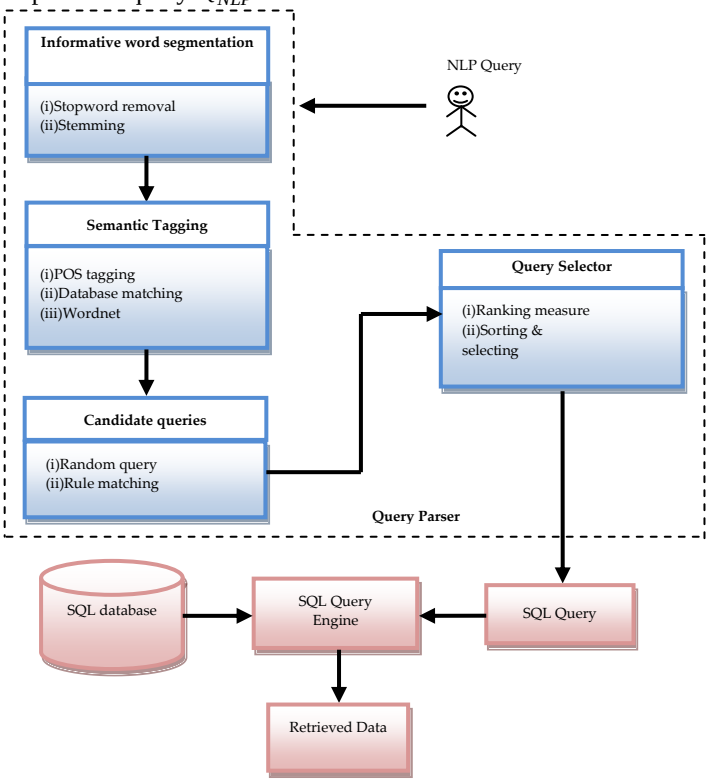

Fig 2. Proposed approach of NLP interface to SQL query engine

4.1 Informative word segmentation

Informative word segmentation is a process of identifying the important keywords from the input query given by the user Q_{NLP}. The input query contains a set of keywords as like the representation, $Q_{NLP} = \{S_i\}; o < i < n$. For every keyword, the matching process is done by matching the input word to the database of having the stop words, ST which have a set of stop words list that is prepared by the experts. In order to do the stop word removal [22], the following equation is applied. Using the equation, every word in the input query Q_{NLP} is compared with ST database. If the word is matched, then it is removed as stop word. Or else, it is still in the query word list.

$$\forall i \in S \quad \text{do} \quad \begin{cases} S_i \approx ST\{j\} \Rightarrow REMOVE \\ S_i \not\approx ST\{j\} \Rightarrow STAY \end{cases} \tag{2}$$

After this process, we obtain a set of informative words $IW \subseteq S$ which would be important and useful words for further process. Then, we do the stemming process [23] which is used to transform the words to its root form. The final output of informative word is stored in the set, SI .

$$\forall i \quad \text{do} \quad SI = ROOT(IW) \tag{3}$$

4.2 Semantic tagging of informative words using data matching strategy

The next step of the proposed approach is semantic tagging of the informative words SI identified from the NLP query Q_{NLP}. Here, three set of training tables are formed by putting table name in $DSI^{(TN)}$, name of table element in $DSI^{(TN.E)}$ and operation relevant SQL words in $DSI^{(O)}$.These three training tables are easily identifiable from the original SQL database itself. For the semantic tagging of every informative words SI , the matching between SI and $DSI^{(TN)}$ is done. If it is matched, then it is indicated as $SI^{(TN)}$. Similarly, the matching between SI and $DSI^{(TN.E)}$ is done. If it is matched, then it is indicated as $SI^{(TN.E)}$ and then, the matching between SI and $DSI^{(O)}$ is done. If it is matched, then it is indicated as $SI^{(O)}$.

$$\forall i \quad \text{if} \quad SI_i \approx DSI^{(TN)}{}_j \quad \Rightarrow \quad \left\{SI^{(TN)}\right\} \tag{4}$$

$$\forall i \quad \text{if} \quad SI_i \approx DSI^{(TN.E)}{}_j \quad \Rightarrow \quad \left\{SI^{(TN.E)}\right\} \tag{5}$$

$$\forall i \quad \text{if} \quad SI_i \approx DSI^{(O)}{}_j \quad \Rightarrow \quad \left\{SI^{(O)}\right\} \tag{6}$$

After this tagging process, the remaining keywords in the identified informative word list SI from the NLP query Q_{NLP} are given to wordnet ontology [7] which is the standard dictionary used in various research for obtaining the hypernyms and hyponyms. These two sets (hypernyms and hyponyms) for the input word is used to provide the general meaning and its core meaning of the input word so that wordnet has been proved effective in identifying the semantic information for any query word. According to the perspective, the untagged words are given to the wordnet ontology which would produce the semantic words of it as per the diagram given in figure 3.

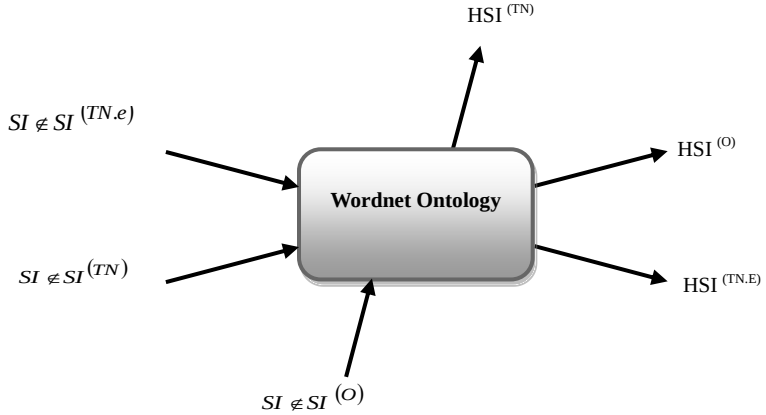

Figure 3. Semantic matching of query words

Then, the semantic words obtained from the ontology like, $HSI = \{HSI^{(TN)}, HSI^{(TN.E)}, HSI^{(O)}\}$ are again matched with the training tables.

$$\forall i \quad \text{if} \quad HSI_i \approx DSI^{(TN)}{}_j \quad \Rightarrow \quad HSI_i >> \left\{SI^{(TN)}\right\} \tag{7}$$

$$\forall i \quad \text{if} \quad HSI_i \approx DSI^{(TN.E)}{}_j \quad \Rightarrow \quad HSI_i >> \left\{SI^{(TN.E)}\right\} \tag{8}$$

$$\forall i \quad \text{if} \quad HSI_i \approx DSI^{(O)}{}_j \quad \Rightarrow \quad HSI_i >> \left\{SI^{(O)}\right\} \tag{9}$$

After this tagging, we check the informative word list that any words are missed out in this tagging process. If any words are missed out here, then tagging of that words is done as per the POS tagging process [24] which read the word and assign part of speech to that words such as, verb, noun, adjective and so on. This process provides the output of all the words tagged as, table name, table element name, POS tag or operation name. Now, every keywords of informative set is divided into four categories. These tagged words are then utilized to generate the candidate queries.

$$\left\{SI^{(TN \cdot e)}, SI^{(TN)}, SI^{(POS)}, SI^{O}\right\} = SI \tag{10}$$

4.3 Formation of lattice structure and candidate query generation

This section explains about the formation of candidate SQL queries to select the best one which is suitable for input NLP query. Here, candidate queries are generated based on lattice structure [25] which have network of connection linked with family of group-like algebraic structures. It is kind of meet and join-based connection of nodes visualized like, pairs, arrays, trees, maps, and graphs. In the lattice structure given in figure 4, left side nodes contains only the keywords of SQL query such as, SELECT, OPERATION, FROM, WHERE and CONDITION. The right side of the lattice contains the tagged words obtained from the previous step. Let assume that the right hand side of the lattice L have l nodes which are the cumulative sum of the words presented in $SI^{(TN \cdot e)}, SI^{(TN)}, SI^{(POS)}, SI^{O}$. Here, candidate query is derived based on the following format.

$$SELECT\ SI_i(SI)_j FROM\ SI_k\ WHERE\{SI_l = SI_m\} AND\{SI_n = SI_o\}$$

The entire keywords presented in the list SI are randomly filled up with the required place so that n number of candidate SQL queries can be formed. The resultant candidate queries are stored in variable $Q_c = \{Q_c{}^k;\ \ 0 < k < n\}$ where, n number of candidate SQL queries generated.

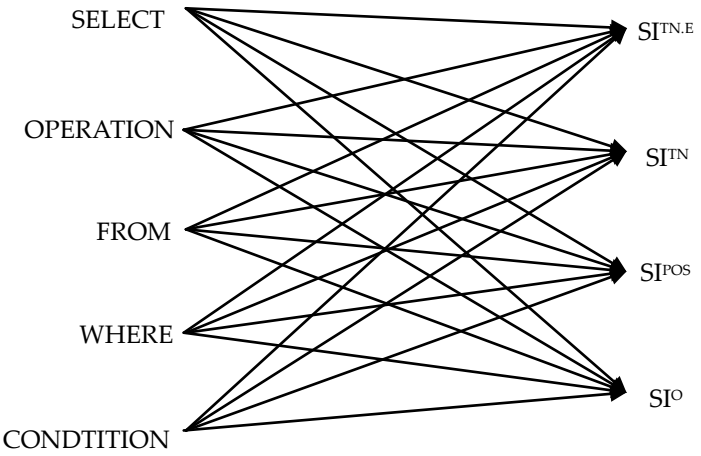

Figure 4. Lattice structure

4.4 Optimal ranking of queries using reverse sigmoid function

This is the important step of filtering or selecting a best SQL query from candidate queries generated from the previous step. The candidate queries are large in size so the right selection would be very challenging work. In order to select an optimal query, the rule-based system is used here. Rule-based system is the system of having the rules which are utilized to give scores for candidate queries. The performance of the system is completely depending on the number of useful rules given by the experts. The idea of selecting an optimal SQL query from candidate queries is based on the rule definition. The ranking measure is developed by including the number of rules which follows the rule definition. Here, $R(Q_{ci})$ is defined as the sum of the individual rules are matched with the rule base, $R(B)$. The rules are defined alike, SELECT should have the table element, OPERATION should have the operation name defined in the database, FROM should be table name and Conditions should held at least table element and numerical or categorical attributes. These rules defined by the database experts are matched with candidate queries directly. If the matching is done, then the value is one or else it returns the value of zero. These explanations are given in the following equations.

$$R(Q_{ci}) = \sum_{j=1}^{m} M_R{}^j \qquad (11)$$

$$M_R{}^j = \begin{cases} 1 & if \quad R_j = Q_{ci} \\ o & \text{if} \quad R_j \neq Q_{ci} \end{cases} \qquad (12)$$

In order to bring the ranking measure more effectively, the weighage formulae, called reverse sigmoid function is used. Here, weighate values are varied in between zero to two to give the importance. Using reverse sigmoid function as weight function, the values are spread through exponential way so that the selection would be easy. According to, the weightage value is then multiplied with the ranking measure $R(Q_{ci})$ to obtain an optimal rank measure $OR(Q_{ci})$.

$$K(j) = 2 * \left(\frac{1}{1 + \exp(R(Q_{ci}))} \right) \qquad (13) \qquad (13)$$

$$OR(Q_{ci}) = \frac{R(Q_{ci})}{K(j)} \qquad (14)$$

The finding of optimal ranking measure is repeated for all the candidate queries and then, the rules are sorted based on the optimal ranking measure. The candidate query which got the high value is selected as best SQL query for inputting to the SQL query engine, Q_{SQL}. The pseudo code of the proposed approach is explained in figure 5.

Input : NLP query, Q_{NLP}
Output : SQL query Q_{SQL}
Parameters :
SI → informative words in the NLP query
$SI^{(TN \cdot e)}$ →Table elements in the NLP query
$SI^{(TN)}$ → Table name in the NLP query
$SI^{(POS)}$ → POS tagged in the NLP query
SI^{O} → Operation tagged in the NLP query
Start
for all $S_i \in Q_{NLP}$
do
call *stop word removal*
call *stemming*
$SI \Leftarrow S_i$
end
end
for all SI_i
if $SI \approx DSI^{(TN.e)}$
$SI^{(TN.e)} \Leftarrow SI_i$
else
$SI = DSI^{(TN)}$
$SI^{(TN)} \Leftarrow SI_i$
else
else
$SI = DSI^{(O)}$
$SI^{(CS)} \Leftarrow SI_i$
else
call *POS tagging*
$SI^{(POS)} \Leftarrow SI_i$
else
Call *wordnet*
Update SI
Generate queries Q_c
For all Q_c
Find $OR(Q_c)$
End
Sort $OR(Q_c)$
Select the best Q_{ci}
$Q_{SQL} = Q_{ci}$
end

Fig 5. Pseudo code of the approach

IV. RESULTS AND DISCUSSION

This section presents the experimental results of the proposed automatic query translator. Here, experimental set up is explained clearly with algorithmic parameters and performance of the system is proved with required analytical parameters.

5.1 Experimental set up

The implementation of the proposed approach is done using JAVA in Eclipse editor. The SQL database and query engine is connected with JAVA programming for loading and querying it. The layout of the SQL database taken for the experimentation is shown in table 1. The input database contains three tables and every table contains10, 000 of data records. Customer table contains three elements such as, Name, Age and City and the Order table contains four elements such as product, name, price and time. Again, Sale table contains product, Quantity and rate as table elements. Then, sample of stop words stored in the database is given as,

$$ST = \{a, an, the, was, is, am, were, shall, on, of, having, with, ...\}$$

The database $DSI^{(TN)}$ contains customer, order and sale, $DSI^{(TN.E)}$ contains C.ID, name, age, city, product, price, time. Qty and rate. Further, $SI^{(O)}$ contains count, maximum, minimum and so on. Again, sample rules like, R(1): SELECT should choose the table element, R(2): OPERATION should use the operation name defined in the database, R3: FROM should use table name, R4: CONDITIONS should use the table element, numerical or categorical attributes are stored in the rule base $R(B)$. These values and information are created based on the input data before executing the proposed approach.

Table 1. Layout of the input database taken

Customer		
Name	Age	City

ORDER			
product	Name	Price	Time

SALE		
product	Qty	Rate

5.2 Sample results

The intermediate results of all the processes involved in the approach are tabulated in this section to understand the concept very clearly. Here, table 2 shows NLP queries given as input and table 3 is the output of NLP queries after removing stop words. Then, it is given to stemming process which provides the output for the queries, tabulated in table 4. Then, the semantic tagging is done for all the words in the informative list. Table 5 presents NLP queries after performing the semantic tagging and candidate SQL queries are generated using lattice structure. The example of candidate queries generated is given in 6. The optimal ranking measure is computed for all the candidate queries using the developed ranking measure. The corresponding ranking values of the candidate queries are given in table 7. From the table 7, we select a candidate query which has the highest value as optimal SQL query for the input. Table 8 is the final SQL query automatically translated from the NLP query.

Table 2. NLP queries given as input

$Q_{NLP}^{(1)}$	Count the products which are bought on the time of august 2012
$Q_{NLP}^{(2)}$	What are the cities having the customer name "rao"
$Q_{NLP}^{(3)}$	Find the maximum rate of the product with the quantity of 10

Table 3. NLP queries after removing stop words

$Q_{NLP}^{(1)}$	Count products bought time august 2012
$Q_{NLP}^{(2)}$	cities customer name "rao"
$Q_{NLP}^{(3)}$	Find maximum rate product quantity 10

Table 4. NLP queries after stemming process

$Q_{NLP}^{(1)}$	Count product buy time august 2012
$Q_{NLP}^{(2)}$	city customer name "rao"
$Q_{NLP}^{(3)}$	Find maximum rate product quantity 10

Table 5. NLP queries after semantic tagging

$Q_{NLP}^{(1)}$	Count→verb (by POS tagging) product →Table element buy → verb (by POS tagging) time→Table element august →attribute 2012→ attribute

Q_{NLP} (2)	city →Table element customer → Table name → Table element "rao"→ attribute
Q_{NLP} (3)	Find → verb (by POS tagging) maximum →operation rate →Table element product →Table element quantity → Table element 10→ attribute

Table 6. Candidate SQL queries

Q_{NLP} (1)	SELECT count (product) FROM order WHERE "time=august" && "time=2012" SELECT buy (product) FROM order WHERE "time=august" SELECT count (product) FROM order WHERE "time=august" SELECT buy (product) FROM order WHERE "time= 2012"
Q_{NLP} (2)	SELECT city FROM customer WHERE "name=rao" SELECT name FROM customer WHERE "name=rao" SELECT name FROM customer WHERE "city=rao"
Q_{NLP} (3)	SELECT maximum(rate) FROM order WHERE "quantity=10" SELECT maximum(product) FROM order WHERE "rate=10" SELECT maximum(quantity) FROM order WHERE "product=10"

Table 7. Optimal ranking measure

	Candidate queries	Optimal Ranking measure
Q_{NLP} (1)	**SELECT count (product) FROM order WHERE "time=august" && "time=2012"**	**373.5329**
	SELECT buy (product) FROM order WHERE "time=august"	31.6283
	SELECT count (product) FROM order WHERE "time=august"	111.1963
	SELECT buy (product) FROM order WHERE "time= 2012"	31.6283
Q_{NLP} (2)	**SELECT city FROM customer WHERE "name=rao"**	**31.6283**
	SELECT name FROM customer WHERE "name=rao"	31.6283
	SELECT name FROM customer WHERE "city=rao"	31.6283
Q_{NLP} (3)	**SELECT maximum(rate) FROM order WHERE "quantity=10"**	**111.1963**
	SELECT produc) FROM order WHERE "rate=10"	31.6283
	SELECT maximum(quantity) FROM order WHERE	111.1963

	"product=10"	

Table 8. Automatically generated SQL queries

$Q_{NLP}^{(1)}$	Count the products which are bought on the time of august 2012	SELECT count (product) FROM order WHERE "time=august" && "time=2012"
$Q_{NLP}^{(2)}$	What are the cities having the customer name "rao"	SELECT city FROM customer WHERE "name=rao"
$Q_{NLP}^{(3)}$	Find the maximum rate of the product with the quantity of 10	SELECT maximum(rate) FROM order WHERE "quantity=10"

5.3 Performance evaluation

The performance of the proposed SQL query parser is evaluated with two set of queries. The query set 1 is the combination of ten different NLP queries and query set 2 is also the combination of ten set of queries. These two sets are used to evaluate the performance of the system. At first, query set 1 is given as input to the system and outputs are computed. Ten outputs of automatic SQL queries are compared with the original SQL query. Based on the matching with the original SQL query, accuracy, error rate and time are computed and plotted in figure 6, 7 and 8. From the figure 6, when the number of rules given in the rule base is increased, the accuracy is improved. For the rules of seven, the output accuracy is 80% if weightage is included in the ranking measure. If the ranking measure is not included the weightage measure, the accuracy of the system is 70%. From the figure, 7, error rate for ranking measure with sigmoid function is two which means that only two queries are not matched. When comparing the computation time, the ranking measure with sigmoid function takes average execution time of 14 sec.

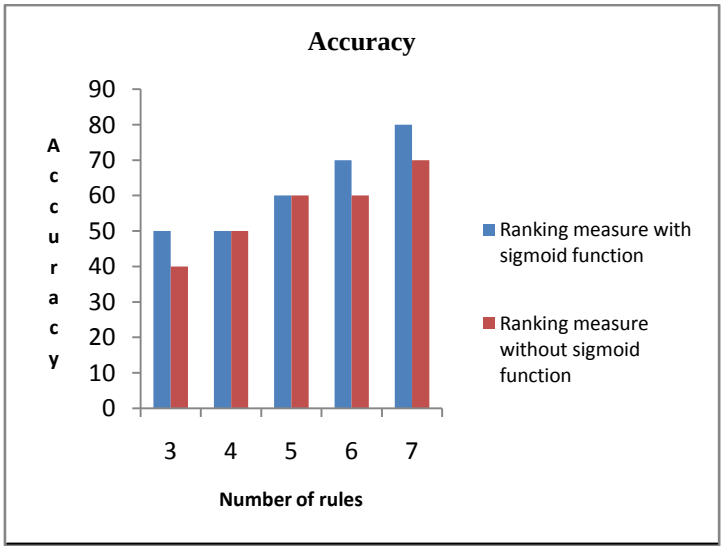

Fig 6. Accuracy graph for query set 1

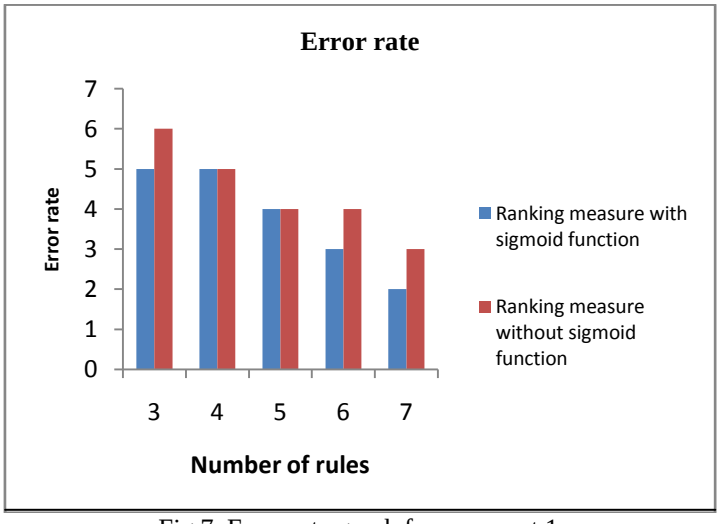

Fig 7. Error rate graph for query set 1

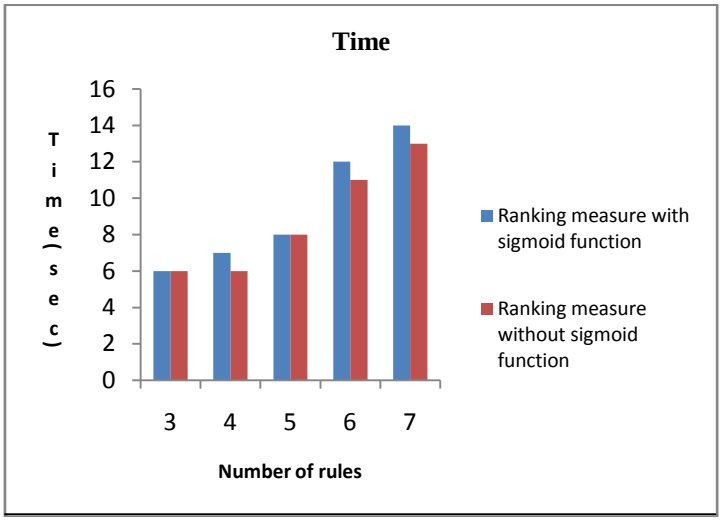

Fig 8. Computational time graph for query set 1

Similarly, the performance of the query set 2 is evaluated with accuracy, error rate and computation time and graphs are plotted in 9, 10 and 11. From the figure 6, output accuracy is 80% if weightage is included in the ranking measure. If the ranking measure is not included the weightage measure, accuracy of the system is 70%. When the number of rules given in the rule base is increased, error rate is decreased. For the rules of seven, output error rate is 2 if weightage is included in the ranking measure. If the ranking measure is not included the weightage measure, error rate of the system is 3. It means that the system wrongly given only 3 SQL queries. The average computation time of the ten queries given in the query set is computed and it is plotted in figure 11. From the figure 11, the ranking measure with sigmoid function takes average execution time of 12 sec for the rule base of having seven rules and ranking measure without sigmoid function takes average execution time of 11 sec for the rule base of having seven rules.

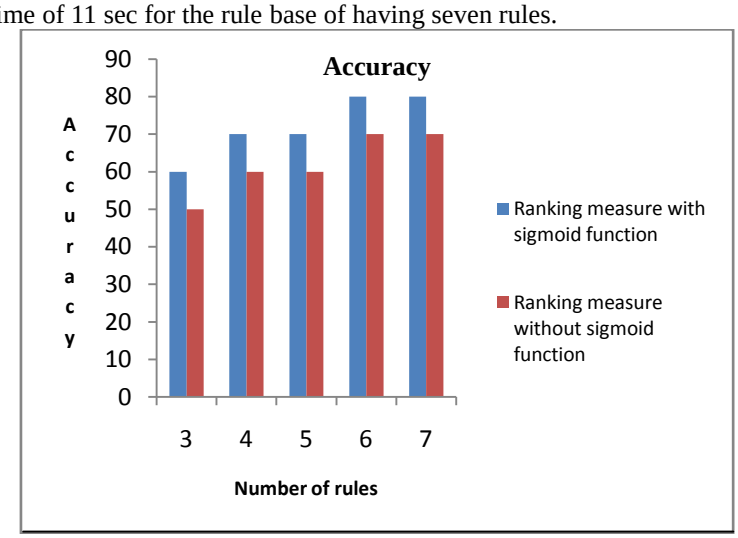

Fig 9. Accuracy graph for query set 2

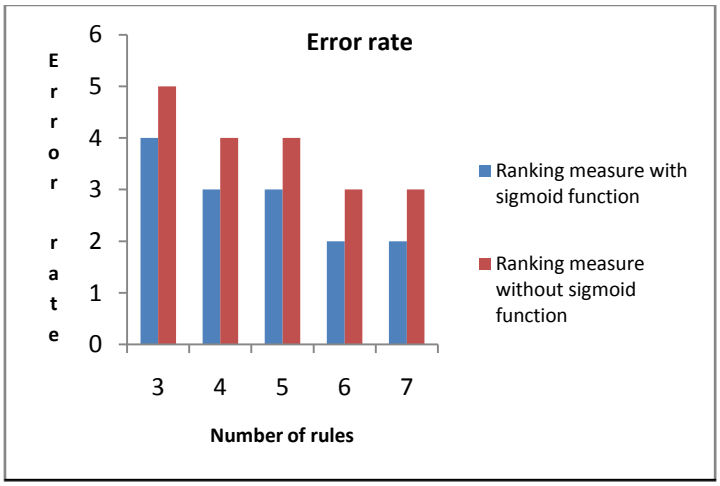

Fig 10. Error rate graph for query set 2

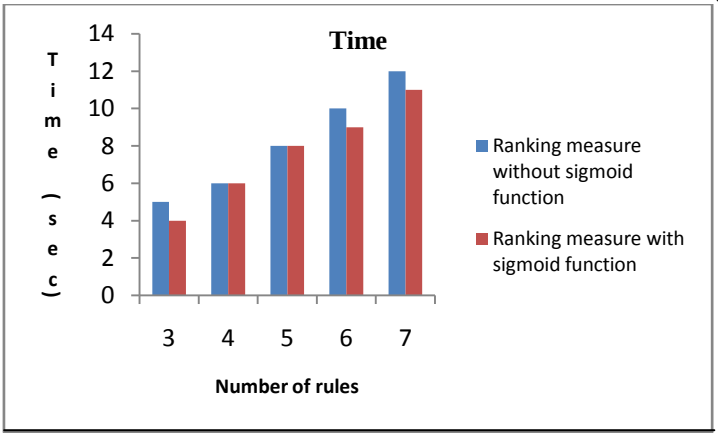

Figure 11. Computational time graph for query set 2

VI. CONCLUSION

We have presented an approach for automatic mapping of natural language questions into programming language using semantic tagging and sigmoid function-based ranking measure. The user input of NLP query was processed and the informative words were identified using stop word removal and stemming. Then, semantic words from the ontology were identified to generate candidate queries by making use of lattice structure. After the generation of candidate queries, an optimal SQL query was selected based on the ranking measure which takes into consideration of number of rules which follows the rule definition and reverse sigmoid function. The experimentation was performed with the database of having 10,000 data records individually in three tables with two set of queries. For both the query set, the proposed approach reaches the maximum accuracy of 80% with less computation task. The future work can be in the direction of improving the ranking measure to differentiate the optimal query from the remaining candidate queries.

REFERENCES

[1] Alessandra Giordani and Alessandro Moschitti, "Translating Questions to SQL Queries with Generative Parsers Discriminatively Reranked", in proceedings of International Conference on Computational Linguistics (Coling), Mumbai, India, 2012.

[2] Majdi Owda, Zuhair Bandar, Keeley Crockett, "Conversation-Based Natural Language Interface to Relational Databases", in proceedings of IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, pp. 363-367, Silicon Valley, CA, 2007.

[3] Ion Androutsopoulos, Graeme D. Ritchie, Peter Thanisch, "Natural language interfaces to databases - an introduction", Natural Language Engineering, vol. 1, no. 1, pp. 29-81, 1995.

[4] Esther Kaufmann, Abraham Bernstein, "Evaluating the Usability of Natural Language Query Languages and Interfaces to Semantic Web Knowledge Bases", Web Semantics:Science, Services and Agents on the World Wide Web, Vol. 8, no. 4, pp. 377–393, November 2010.

[5] Yunyao Li, Ishan Chaudhuri, Huahai Yang, Satinder Singh, H. V. Jagadish, "Enabling Domain-Awareness for a Generic Natural Language Interface", in proceedings of AAAI, pp. 833-07, 2007.

[6] Camille Pradel, Ollivier Haemmerl´e, and Nathalie Hernandez, "Natural Language Query Interpretation into SPARQL Using Patterns", In proceedinsg of ISWC, Sydney, Australia, October 2013.

[7] Vaclav Snasel, Pavel Moravec, Jaroslav Pokorny, "WordNet Ontology Based Model for Web Retrieval", in proceedings of international Workshop on Challenges in Web Information Retrieval and Integration, pp. 220 - 225, 2005.

[8] M. Jose A. Zarate, R. Rodolfo A. Pazos, Alexander Gelbukh and O. Joaquin Perez, "Improving the Customization of Natural Language Interface to Databases Using an Ontology", Computational Science and Its Applications – ICCSA, Vol. 4705, pp 424-435, 2007.

[9] Niculae Stratica, Leila Kosseim, Bipin C. Desai, "Using semantic templates for a natural language interface to the CINDI virtual library", Data & Knowledge Engineering, Vol. 55, no. 1, pp. 4–19, October 2005.

[10] Axel-Cyrille Ngonga, Lorenz Bühmann, Christina Unger, Jens Lehmann, Daniel Gerber, "Sorry, I don't speak SPARQL – Translating SPARQL Queries into Natural Language", in Proceedings of the 22nd international conference on World Wide Web, pp. 977-988, 2013.

[11] Alessandra Giordani, Alessandro Moschitti, "Corpora for Automatically Learning to Map Natural Language Questions into SQL Queries", in Proceedings of the Seventh International Conference on Language Resources and Evaluation, 2010.

[12] Michael Würsch, Giacomo Ghezzi, Gerald Reif, and Harald C. Gall, "Supporting Developers with Natural Language Queries", in proceedings of CM/IEEE 32nd International Conference on Software Engineering, vol. 1, pp. 165 - 174, Cape Town, 2010.

[13] Alessandra Giordani and Alessandro Moschitti, "Semantic Mapping Between Natural Language Questions and SQL Queries via Syntactic Pairing", Natural Language Processing and Information Systems, Vol. 5723, pp 207-221, 2010.

[14] Giordani, A. and Moschitti, A. "Syntactic structural kernels for natural language interfaces to databases", in Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 391–406, Berlin, Heidelberg. Springer-Verlag, 2009.

[15] Giordani, A. and Moschitti, A, "Generating SQL queries using natural language syntactic dependencies and metadata", In NLDB, pp. 164–170, 2012.

[16] Minock, M., Olofsson, P., and Naslund, A., "Towards building robust natural language interfaces to databases", in Proceedings of the 13th international conference on Natural Language and Information Systems, Berlin, Heidelberg, 2008.

[17] M. J. Minock, "A Phrasal Approach to Natural Language Interfaces over Databases", Natural Language Processing and Information Systems, Vol. 3513, pp 333-336, 2005.

[18] I. Androutsopoulos, G. Ritchie, and P. Thanisch, "MASQUE/SQL – An Efficient and Portable Natural Language Query Interface for Relational Databases", Proceedings of the 6th international conference on Industrial and engineering applications of artificial intelligence and expert systems, Gordon & Breach Science Publishers, Edinburgh, Scotland, pp. 327-330, 1993.

[19] P. Reis, N. Mamede, and J. Matias, "Edite – A Natural Language Interface to Databases: a New Dimension for an Old Approach", Proceeding of the Fourth International Conference on Information and Communication Technology in Tourism, Edinburgh, Scotland, 1997.

[20] Yannis Ioannidis, "From databases to natural language:The unusual direction", Natural Language and Information Systems, vol. 5039, pp 12-16, 2008.

[21] Zarate, A., Pazos, R., Gelbukh, A., Padrón, I., "A Portable Natural Language Interface for Diverse Databases Using Ontologies", in Proceedings of the 4th international conference on Computational linguistics and intelligent text processing, vol. 2588, Springer, Heidelberg, pp. 494-505 , 2003.

[22] J. R. Mendez, E. L. Iglesias, F. Fdez-Riverola, F. Diaz, J. M. Corchado, "Tokenising, Stemming and Stopword Removal on Anti-spam Filtering Domain", Current Topics in Artificial Intelligence, Vol. 4177, pp 449-458, 2006.

[23] Lovins, J.B., "Development of a stemming algorithm", Mechanical Translation and Computational Linguistics, Vol. 11, pp. 22-31, 1968.

[24] Toutanova, K., Klein, D., Manning, C.D., Yoram Singer, Y., "Feature-rich part-of-speech tagging with a cyclic dependency network", in Proceedings of HLT-NAACL, pp. 252-259, 2003.

[25] Lindsey Kuper, Ryan R. Newton, "LVars: Lattice-based Data Structures for Deterministic Parallelism", in Proceedings of the 2nd ACM SIGPLAN workshop on Functional high-performance computing, pp. 71-84, 2013.

