IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-4, Issue-02 E-ISSN: 2347-2693

Natural Language Query Processing for Relational Database using EFFCN Algorithm

B.Sujatha[1*], S.Vishwanadha Raju[2]

[1]*Department of CSE, Osmania University, India*
[2*] *Department of CSE, College of Engineering JNTUH Jagityal, India*

www.ijcseonline.org

Received: 22/Jan/2016 Revised: 02/Feb/2016 Accepted: 14/Feb/2016 Published: 29/Feb/ 2016

***Abstract*—** This paper addresses the procedure to develope an interface to natural language database that is efficient and flexible to handle unrestricted natural language and interpret the request appropriately called as EFFCN, stands for EFFiciently Compliant Natural language interface to database. The Experimental set up is created by developing a database named as CPVBase. The database holds the tables instituted with the sample records of Customer, Product, Vendor and Invoice data. The database tables have foreign key references to the other tables epitomizing a relation database management system. This paper explains about various technical segments of the implementation of the EFFCN algorithm. The working procedure of the algorithm for the natural language statement transformation into SQL query is depicted. The EFFCN algorithm's precision and recall measures for the score of relevancy is obtained with the success rate of 84%. The PR curve shows the variation of precision and recall measures tested on discrete set of input queries.

***Keywords*—**Natural Language Query, First Order Logic, Structured Query, Precision, Recall, F1-measureNatural Language Query, First Order Logic, Structured Query, Precision, Recall, F1-measure

I. INTRODUCTION

Database systems are used since 1970s for the storing various kinds of data for different purposes such as commercial and personal needs. Though there are many types of architectures for database design like object oriented, object based, file based, hierarchical based and network based, the predominant designing of databases follow relational database architecture to store the data by using various types of storage devices. In relational databases, the data is stored using tables. The table contains set of rows and columns. Each column represent and attribute and each represents the instance of the data for a set of attributes. The data can be manipulated using various operators with fixed set of keywords by following a set syntax rules. By learning this structured query language one can extract the required data from the whole set of data, can also perform various operations such as update, manipulate and deletion of the data.

The Relational database management systems are more popular based on the characteristics like its robustness and flexibility, high performance, scalability, data security and protection and flexible data maintenance. Above all these advantages, it allows to index, perform aggregation, filtering and sorting can be done on the data using structured query language.

There are some disadvantages with relational databases. To perform operations on the data which is stored on databases, it is required to learn the structured query language. Hence , the naive user who knows only the natural language cannot directly access the required information from the databases. To come out from these limitations, it is required to design a tool which can understand the requirements of the naive user through natural language query, convert the natural language query into an equivalent structured language query. Then the obtained structural query is used to access the required information from the databases. This kind of tool ins termed as Natural Language Interface to Databases or NLIDB system. Thus, the NLIDB system take the input as natural language query and converts it into a structures language query and returns the desired information to the naive user.

The designing of a NLIDB system for various languages and for different underlying databases is attempted by various researchers since five decades. But, designing of an most suitable NLIDB systems with high accuracy, precision and recall is still an open research problem which need to be addressed. The various earlier developed NLIDB systems focused on particular databases. There is need of designing a generic NLIDB system which can address the robustness and scalability of the applications. It is required to attempt the problem of portability to customize a NLIDB system to a other language and to other set of datasets designed for various domains.

In this paper, it is focused on designing a NLIDB system using EFFECN algorithm, the procedure to develope an interface to natural language database that is efficient and flexible to handle unrestricted natural language and interpret the request appropriately. The procedure to implement the proposed system is initiated by constructing ontology. The proposed a system process the query using First Order Logic and the parsed query is converted into SQL query. The designed system maintains a high accuracy 84% for customer database.

II. RELATED WORK

There are many designing models are proposed in the literatures in the field of NLIDB such as pattern matching systems, syntax based systems, semantic based grammar systems and intermediate representation of languages system.

The pattern matching systems takes input as a set of rules and sample set of pattens. Based on the inputted word of sentence with natural language, it will be compared with the predefined patterns [1]. If there is a match between the input and predefined pattern then an action will be generated and these generated actions will be stored in the database. The response given to the user is based on the action generated. This kind of systems are limited to specific databases. The accuracy of the system is depend on the complexity of the patterns used to train and based on the set of rules used to train the system [2]. The NLIDB system SANVY is a good example for pattern- matching systems [3].

The syntax based systems takes the user query as input and parse the given input syntactically. The parse tree generated for the input query is overlapped with the one structured query of the database expressed using structured query language. LUNAR is a best example for syntax based NLIDB systems [4]. In these systems, the grammar rules are derived to match the various user questions with syntactic structures [5]. This system is used to answers the questions on rocks which were collected from the moon. With the corrections in the dictionary errors, the performance of the system has increased [8].

In the semantic grammar system, the parse is simplified by eliminating unimportant nodes or by combining two or more nodes into one node. The complexity of structured query can be reduced in semantic grammar system. Semantic grammar systems are more simpler when compared with syntax based systems. But these systems need to be trained with a prior knowledge of the various elements of a domain. PLANES and LADDER are the good examples for Semantic grammars systems [6,7].

In many NLIDB systems, the natural language query is transformed into an intermediate logical query. The logical query is represented using a meaningful representative language such as first logic language or Boyce codd normal form. This kind of representative languages, represents the meaning of the users queries in high order level of concepts. These concepts are independent from the structure of the database. This representative query is then transformed into an expression in the structured query language which can extract the relevant data from the databases.

In the intermediate representation of natural language systems, the natural language query is inputted to the system. This query is processed for syntax rules using a parser. Based on the set of syntax rules of a natural language, it generates a parse tree. By using the semantic rules of semantic interpreter module, the generated parse tree is translated into an intermediate logic query. In the semantics rule, left hand side of the syntax rule contains the logic expression of the constituent where as right-hand side of the syntax rule is a function of the logic expressions of the constituents. The logic expressions represents the words which are corresponds to lexicon. To get the required information from the database, the logic query is to be transformed into a structured query which is supported by the underlying Database Management System. MASQUE/SQL is an example of intermediate representation language systems [7].

By using semantic grammar techniques which interleaves semantic and syntactic processing in distributed databases, LADDER system is used to parse natural language questions to database understandable queries [7]. The another NLIDB system implemented using the language called Prolog was CHAT-80. This system transforms the natural language inputted English queries into Prolog expressions. These Prolog expressions are evaluated using the Prolog database. ROBOT which was a prototype of a NLIDB system named INTELLECT which was a commercial natural language interface to database systems [9]. ASK is the another NLIDB system which allows the users to train the system with new words and concepts while inter actioning with the system. By using the system, it is possible to make interactions with various external sources such as external databases, chating, Facebook, twitter, email programs and many other applications.

Generic Interactive Natural Language Interface to Databases (GINLIDB) was designed by the using UML and developed using Visual Basic.NET. The system was a generic system and it works for underlying suitable database and knowledge base [10]. SynTactic Analysis using Reversible Transformations (START) is also another Natural Language System. It was the first Web-based question answering system. It was available online and continuously operating till now [11]. It utilizes various language Dependant functions such as parsing, semantic analysis, word sense dis-ambiguous, natural language annotation for

appropriate information segmentation and presentation for the user [12].

JUPITER was a NLIDB system to know the weather information worldwide. The user can pose a question to the system in their native language to forecast the weather information over the telephone. The Oracle Structured Query Language SQL can be learned by the students using the NLIDB system called SQL-Tutor. If the student asked the new questions by typing at terminal then also, the SQL-Tutor can answer the question by using the existing knowledge [13]. KUQA system divides the query based on possible answer and after that it uses NLP techniques and also WorldNet to identify the answers which suitable to its corresponding category. But, this system can not handle any linguistic information [11]. QuALiM another NLIDB system designed based on complex syntactic structure which were based on certain syntactic description question patterns [11].

III. EFFCN ALGORITHM

The Interface of the system takes a natural language user generated query. The query internally is divided into parts based on the occurrence of the preposition or verb phase obtained after preprocessing for POS tagging, the ontology built assists in mapping to the appropriate columns of the corresponding table and gives the response as the structured representation that is the subset of the database table. The algorithm presented in the figure 3.1 partitions the query natural language query partitioning.

A. *Query Partitioning*

```
if (Preposition exists and is before verb and relative
pronoun)
{
Split at preposition into two parts Left and Right
Right: Scan for table names and conditions
Left: Scan for columns and associate these columns with
first found table in the right part
}
else if((verb exists and is before prep and relative
pronouns)OR (relative pronoun exists))
{
Split at Verb or Relative Pronoun
Left: Scan for table name and columns for that table
Right: Scan for table and associate column in the
condition with the last found table
}
else
{
Do not split query
Scan for table and columns and associate columns with the
found table
}
```

Fig 3.1: Pseudo code for splitting query

The splitting of query is done based on the place of occurrence of the verb and prepositions in the query in connection with the question words such as who, what, where etc. Depending on these locations, the logic stated in the figure 3.1 applied gives the respective column and table references from the CPVBase.

B. *Paradigm for Joining of Tables*

1. *Shortest route algorithm could be used to find the join conditions but since the 4 tables are joined in a path like formation the tables given ids and sorted. Customer = 1, invoice = 2, product = 3, vendor = 4.*

2. *If the query is 'show all products purchased by customer'. From the query only two tables can be inferred: product and customer. The ids are 3 and 1. These are sorted to 1, 3 and the missing table 2: invoice is also joined.*

3. *Tables are given aliases c,i,p,v and these aliases are used to specify the columns in the conditions and the columns to be displayed*

Fig 3.2: Conditions for Joining of tables

The criterion for joining the related tables is accomplished by giving a numeric integer identity for the CPVBase tables. If the natural language query involves multiple tables references then the tables are selected in the numeric order obtained using the table Id's as quoted in the figure 3.2.

C. *Paradigm for Selecting Columns*

(*) *if column_list is empty and "show, give, tell, find, which, what, whose" is used then all columns are selected provided other wise if column_list is not empty then the columns in the column_list are displayed*

(*) *if on the other hand "how, count" is used in the query then count(*) is shown*

Fig 3.3: Conditions for Column Selection

The norm for column selection is presented in the figure 3.3. The decision of the requirement of specific columns or all the columns from a table is retrieved from the column_list element.

IV. RESULTS AND ANALYSIS

The EFFCN system performance is measured in terms of retrieval efficacy using the information retrieval system metrics known as precision and recall. The attainment of relevant information by the user as per the natural language query in English gives the retrieval efficacy. The precision is the measure of retrieved results that are relevant to the

need, evaluated using the fraction of relevant documents retrieved to the total number of documents retrieved.

Mathematically it can be expressed as

$$Precision = \frac{CorrectlyAnsweredQueries}{AnsweredQueries} \quad (4.1)$$

Recall measures the relevant results retrieved as per the user statement. That is the fraction of relevant documents retrieved to the total number of relevant documents present in the system.

$$Recall = \frac{CorrectlyAnsweredQueries}{TotalNumberofQueries} \quad (4.2)$$

based on the precision and recall measurements, the system was tested for a random of 100 queries, giving the result tabulated displayed in table 4.1

Total Queries : 100
Answered Queries: 97
Unanswered Queries: 3
Correct Results: 84
Wrong Results: 13
Precision: 84/97 = 86.5%=0.86
Recall: 84/100 = 84.0%=0.84

Table 4.1 Implementation Results

The results shows that the system offers a precision and a recall rates 86.55 and 84% probability of generating correct responses to the user queries. This proves the effective and optimal working of the system. The result is determined by taking portion of queries and is obtained as presented in the table 4.1.

The table 4.2 contains system data generated by testing using different amount of queries and obtained the counts of correctly answered queries (correct_q), wrongly answered queries(wrong_g) and unanswered queries due to improper mapping or no corresponding record (unans_q) with their respective measures of precision and recall. The precision and recall is decreasing if irrelevant queries are observed. Thus the precision and recall can be well defined if the data search is acquired with maximum relevant terms in the query.

No. of queries	Correct_q	Wrong_ q	Unans_q	Precision	Recall
20	18	2	0	0.9	0.9
40	35	3	2	0.92	0.875
60	52	6	2	0.89	0.866
80	70	9	2	0.897	0.875
100	84	13	3	0.86	0.84

Table 4.2: Precision and Recall for varying number of Queries

A. *PR CURVE*

The data of table 4.2 is plotted on a two dimensional graph and a Precision-Recall Curve is obtained shown in the figure 4.1.

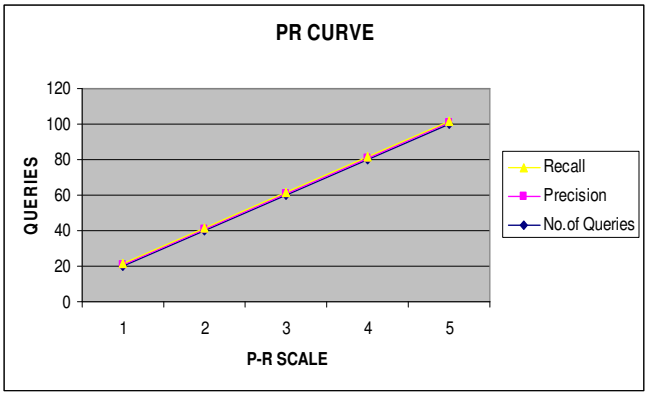

Fig: 4.1 PR Graph

The X-axis holds the Precision and Recall Values with respect to the number of queries. The Precision and Recall are constant showing the desirable and obtained relevancies are near approximately. Another graph plotted in the figure 4.2 shows the Precision and recall variations.

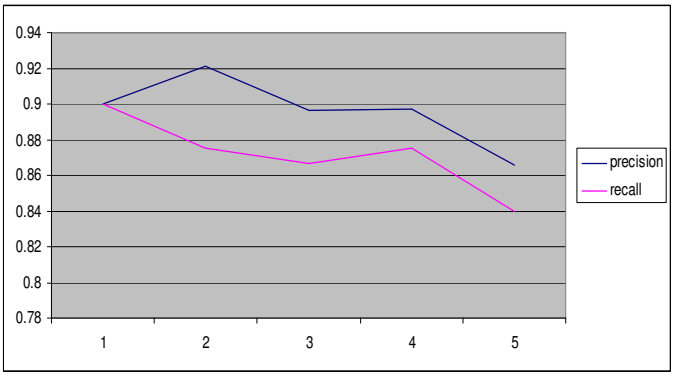

Figure 4.2 Precision- Recall Graph

The graph plotted in the figure 4.2 represents the relation between Precision and Recall with respect to the EFFCN system. The curvy edges in the graph change in the precision with the minor effect in the recall. The precision and recall is decreasing if irrelevant queries are observed. Thus the precision and recall can be well defined if the data search is acquired with maximum relevant terms in the query.

The NLIDB system based on authorizing queries has a success rate of 96.8% but with a drawback of having low precision compared to which the EFFCN system gives a success rate of 84% and high precision 86.5%. The other NLIDB system based on semantic parsing accomplished a success rate of 70%.

V. CONCLUSIONS AND FUTURE SCOPE

The research aimed at developing an interface that eases the work of the naive user to formulate a database request and generate appropriate responses. The system vitally uses the ontology constructs, Parsing rules and FOL logic to extract the requisite information in forming a standard database Query. The system is flexible and can be adapted to any of the Database management systems or a relational database management system. EFFCN is a domain independent and highly portable system. It uses the semantics and syntactic knowledge to generate the correct match of the input statement's SQL query. Using the power of ontology and enhanced parsing mechanisms to filter query up to a refined level where it incorporates needed information as per the user. Compared to which the EFFCN system gives a success rate of 84% and high precision of 86.5%.

The NLIDB system future growth is directed towards improving the success rate by applying concepts of neural networks, machine learning parsing techniques and the use of SQL standard aggregate functions such as average, min and max along with the operator precedence concepts. The analysis of the system from the perspective of abbreviations and the temporal queries also needs careful interpretation along with the complex restrictions of FOL logic.

REFERENCES

[1] Mrs. Neelu Nihalani, Dr. Sanjay Silakari and Dr. Mahesh Motwani, "Natural Language Interface for Database: A Brief Review", IJCSI International Journal of Computer Science Issues, vol. 8, no. 2, pp. 600-608, Mar. 2011.

[2] T. Johnson, "Natural Language Computing-The Commercial Applications", The Knowledge Engineering Review, vol. 1, no. 3, pp. 11-23, 1984.

[3] Androutsopoulos, G.D. Ritchie and P. Thanisch, "Natural Language Interface to Databases-An Introduction", Department of Computer Science, University of Edinburgh, King"s Buildings, Mayfield Road, Edinburgh EH9 3JZ, Scotland, U.K. , Mar. 1995.

[4] W.A. Woods, R.M. Kaplan and B.N. Webber, "The Lunar Sciences Natural Language Information System: Final Report", BBN Report 2378, Bolt Beranek and Newman Inc., Cambridge, Massachusetts, 1972.

[5] C.R. Perrault and B.J. Grosz, "Natural Language Interfaces", Exploring Artificial Intelligence, Morgan Kaufmann Publishers Inc., San Mateo, California, 1988, pp. 133-172.

[6] G. Hendrix, E. Sacerdoti, D. Sagalowicz, and J. Slocum, "Developing a Natural Language Interface to Complex Data", ACM Transactions on Database Systems, pp. 105-147, 1978.

[7] W. Woods, "An experimental parsing system for transition network grammars in Natural Language Processing", Algorithmic Press, New York, USA, 1973.

[8] L.R.Harris,"Experience with INTELLECT: Artificial Intelligence Technology Transfer", The AI Magazine, pp. 43-50, 1984.

[9] Faraj A. El-Mouadib, Zakaria S. Zubi, Ahmed A. Almagrous and Irdess S. El-Feghi, "Generic Interactive Natural Language Interface to Databases (GINLIDB)", International Journal of Computers, vol. 3, no. 3, 2009.

[10] "START Natural Language Question Answering system Online.

[11] M. Joshi, R. A. Akerkar, "Algorithms to improve performance of Natural Language Interface", International Journal of Computer Science & Applications, vol. 5, no. 2, pp. 52-68, 2008.

[12] Seymour Knowles and Tanja Mitrovic, "A Natural Language Interface For SQL-Tutor", Nov. 5, 1999.

[13] D.L. Waltz, "An English Language Question Answering System for a Large Relational Database", Communications of the ACM, pp. 526-539, 1978.

