DB Mapping and IQML Generation for SQL Queries Generated from Natural Language Queries

Manisha[1] Sona Malhotra[2]
[1,2]Department of Computer Science & Engineering
[1,2]UIET, Kurukshetra University India

Abstract— Natural language processing has always been an area of great interest to the research scholars and NLIDB (Natural language interface for databases) is that field which uses the techniques defined in NLP to the best possible level to create an interface for humans which interacts with database like an SQL expert does. An advanced XML based technique is defined which convert the natural language queries into SQL queries. An interface (IQML based DBMapper) is defined which inputs the user query and then parses the query generating an IQML which gets converted into a SQL query with the help of DBMapper which is finally executed and then the intended result is displayed. The proposed research methodology is programmed in java language due to its ease and quick XML generation and parsing. The conversion into IQML makes the query exportable into many other language independent formats like JSON and can be used for different purposes.

Key words: XML (Extensible Markup Language), DTD (Document Type Definition), RDBMS, SQL

I. INTRODUCTION

Information plays a vital role and it is increasing day by day so, to store and manage this huge amount of data the need of databases arises. To retrieve information from database one should know database languages such as Structured Query Language (SQL). Persons who have no knowledge of database language may find it difficult to access database. In recent days, there is a great increase in the need of non-expert users to query database in natural language and as everyone is not able to write SQL queries efficiently. Many researchers have actually turned in to use Natural Language (NL) i.e. Hindi, English, French, Tamil, Arabic etc. instead of using any database language [1]. The concept of using NLP give away the evolvement of a new type of processing method known as Natural Language Interface to Database systems (NLIDB) which is a computer human interface. These are the interfaces between computers and humans where users have no requirement to learn any other formal language, they can give query in their native language discarding the burden on user to learn SQL. This is a system in which user access information stored in databases by typing request in natural language. It makes accessing of data from database very easy for a person who has no knowledge of query language. All of the applications in the system requires information to be retrieved from the repository who actually needs to be aware of database languages like SQL. A NLIDB system will help in many ways. By using such systems anyone can collect information from the database. Furthermore, it may change our thinking about the information in a database. For this system human have to type a question in natural language suppose English so as to get the information from data base. User need not learn a new database language, can frame a query in natural/native language, reducing the burden on user [2].

II. RELATED WORK

Lukas Blunschi et al. [3] in 2012, introduced a new system named as SODA system (Search over Data Warehouse). This system links the needs of analysts and technical complexities of the current data warehouse. This enables a Google-like search experience for the users in this, keywords were taken in the form of queries consisting of business needs and then an executable SQL query was generated automatically. The experiments with both synthetic data as well as with a large data warehouse of a global player in the financial services industry showed that the generated queries had high precision and recall compared to the manually written gold standard queries. One of the strengths of SODA was that it could disambiguate the meaning of word by taking into account join and inheritance relationships among the matching tables. Moreover, SODA allowed mitigating inconsistencies in the schema or data as well as the data quality issues by updating the respective metadata graph or by the extending graph pattern matching algorithm.

Yingzhong Xu et al. [4] in 2013, QMapper was developed which maps SQL queries into SQL queries by utilization of query rewriting rules and cost based Map Reduce flow evaluation on the basis of column statistics. Evaluation demonstrated that when the correctness was assured, QMapper had improved the performance for upto 42% in terms of the execution time taken which had extended the rewriter to support more rules and enhance the estimator to involve the parallelism of multiple SQL blocks, data compression and the difference of cost effect between Map and Reduce to provide a more fine-grained cost model.

Rakesh Kumar et al. [5] in 2014, performed a comparison between SQL and SQL on basis of their architecture, advantages, disadvantages and other features. The author concluded that the primary reason for moving data between SQL stores as well as Hadoop was taken so as to take the advantage of massive storage and the processing capabilities to process quantities of data larger than one could hope to cope with in SQL alone.

Prasunkanti Ghosh et al. [6], in 2014, created a natural language processing tool that integrated speech and natural language. The author worked so as to enable the communication between the user and computer without resorting memorization of complex commands.

Bajwa et al. [7] in 2016, presented a natural language query based framework to generate SPARQL/RDF queries for NoSQL databases. In this research, a challenging task was to map natural language queries to map to SPARQL queries. This framework accepted input natural language query in English from user and then NL queries were translated into SPARQL/RDF queries. The results of the experiments with the designed framework reflected importance of such approach used for automated generation of SPARQL/RDF queries for NoSQL databases.

Li et. al in [8], presented a NaLIR which is an interface to query relational databases. NaLIR accepted a logically complex English language sentence as natural language query. The query was first translated into a SQL query, which included aggregation, nesting, and various types of joins and then evaluated against an RDBMS. In this demonstration, the authors showed that NaLIR, while far from being able to pass the Turing test, was perfectly usable in practice, and could efficiently handle even quite complex queries in a variety of application domains.

III. PROPOSED METHODOLOGY

NLIDB has always been an interesting field of research for NLP scholars. The tables, attributes, conditions and sometimes mixed expressions from the natural language query need to be identified. So, an interactive technique have to be designed, which updates the user with keywords interpretations and allows the user to make an amendment to his input query if requires. Each word in NLQ have to be mapped to a database term. This can only be done by getting words to their basic form through stemming process. This can further be facilitated by matching not just the words but also the synonyms of these words generated by using wordnet tool. A DTD (Document type definition) is defined to validate the IQML generated, which declares the tags, their existence in xml doc and their validity to understand the XML tags defined therein. The different components to perform the NLIDB operation are explained below to elaborate the entire process.

A. *Dataset Used – YELP*

Yelp.com is a website for the review of the crowd for local business and it is also a social networking site, users of this site is majorly present in metropolitan cities. The pages of the site is devoted individually such as schools, theatres, restaurants where its users can submit their review about the service or product offered to them, rating system consists of one to five stars. Businesses can also update their contact information, hours of working or special deals.

1) Yelp_training_Set_business

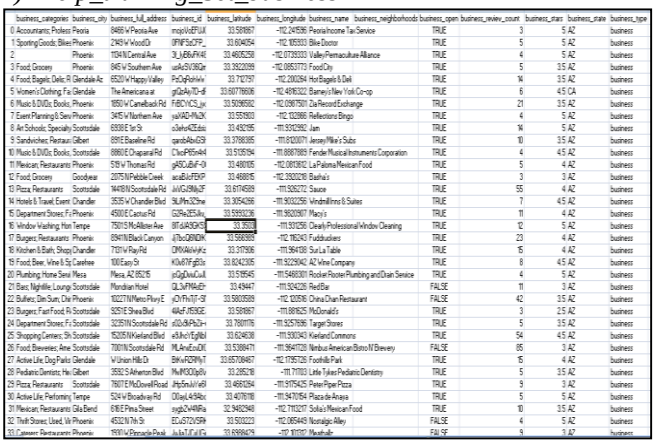

Fig. 1: Dataset Screenshot

2) Yelp_training_Set_User

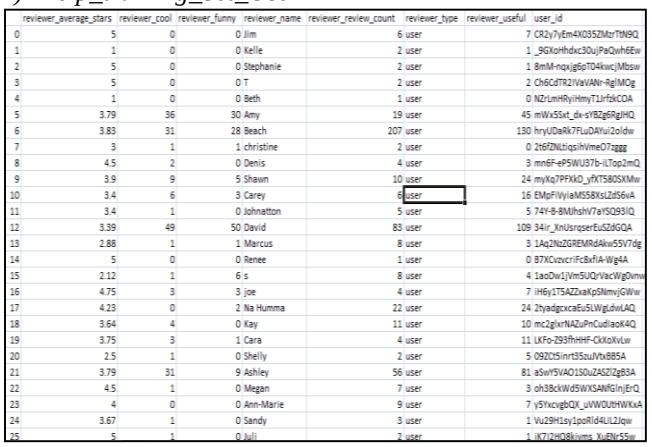

Fig. 2: Dataset Screenshot

3) Yelp_training_set_Flattened

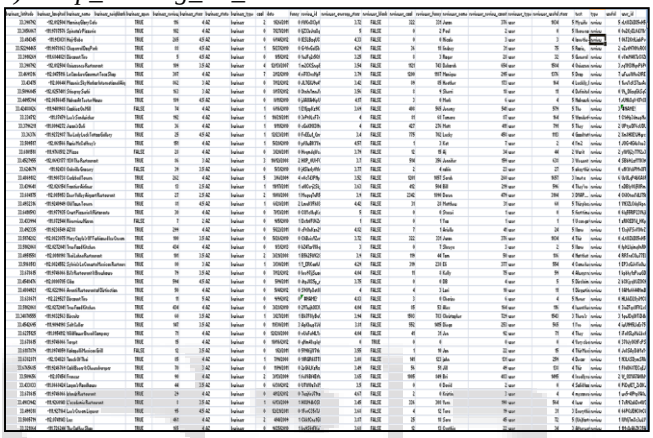

Fig. 3: Dataset Screenshot

4) Yelp_training_set_reviews

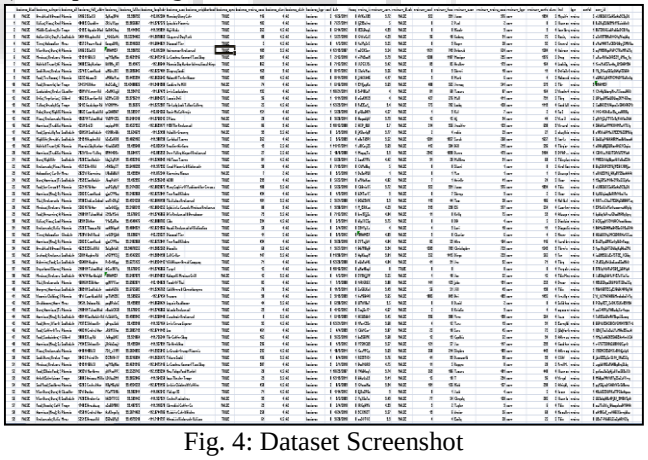

Fig. 4: Dataset Screenshot

B. *Stanford Parser*

"I am doing my work and no one should disturb".

- I/PRP
- am/VBP
- doing/VBG
- my/PRP
- work/NN
- and/CC
- no/DT
- one/NN
- should/MD
- uselessly/RB
- disturb/V

The above example shows the working of Stanford parser. It parses the sentence and returns the grammar type of each word, thus helps to understand the NOUN words which can possibly form the table names. The list of noun words are generated and also add to it the synonyms of these words generated by WORDNET.

C. Wordnet

WordNet is a large lexical database of English language. Wordnet includes verbs, nouns, adverb and adjectives but it ignores preposition, determiners, connectors and some function words. Words are then grouped into different sets of cognitive synonyms which are named as synsets, all these words then express different concept. These synsets are actually interlinked by means of conceptual-semantic and lexical relations. Resulting in a network of meaningful related words and concepts which will then be navigated with the help of a browser. WordNet's structure makes it a useful tool for computational linguistics and natural language processing.

D. DB-Mapper to generate IQML code

FIND DETAILS OF ALL EMPLOYEES WHO BELONG TO DELHI AND SALARY MORE THAN 40000

```
<?xml version="1.0" encoding="UTF-8" standalone = "YES"?>
<QUERY>
<Table name="employee">
<Column name="*"/>
<condition>
<txt>city='Delhi'</txt>
<next>and</next>
</condition>
<condition>
<txt> sal>40000 </txt>
<next></next>
</condition>
</Table>
</QUERY>
```

Table 1: Query.iqml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE QUERY [
<!ELEMENT QUERY(TABLE+, CONDITION+)>
<!ELEMENT TABLE ( COLUMN+)>
<!ELEMENT COLUMN (#PCDATA)>
<!ELEMENT CONDITION (TXT)>
<!ELEMENT TXT (#PCDATA)>
<!ELEMENT NEXT (#PCDATA)>
<!ATTLIST TABLE NAME CDATA #REQUIRED>
<!ATTLIST COLUMN NAME CDATA #REQUIRED>
]>
```

Table 2: Query.dtd

The DB Mapper maps the input query tokens with the database table list and attributes list to form the IQML file which is created as per the DTD defined for that. The conditions are also interpreted and set into IQML code as per SQL format.

E. Query-Gen to convert IQML to SQL query

Select * from employee where city='Delhi' and sal>40000

The Query-Gen module converts the IQML generated by DBMapper into SQL query by simply using an XML parser to read the IQML. The query is presented to the end user to allow any amendment in input query and then executed to get the result displayed.

F. Algorithm

The entire process of NLIDB architecture based on DBMapping and IQML generation to generate SQL queries from NL queries can be summarized in an algorithm given below:

- Step 1: Read into Q the natural language query.
- Step 2: V_i = [split(S_i) -> where S_i is ith sentence in $Q]_{i=1->n}$
- Step 3: Perform filtering and stemming V_i=filter(V_i).
- Step 4: L_i= Parser(V_i) Now L_i is the list of tokens after Stanford parsing. So this list contains the words which are nouns. This will be mapped to database terms to perform the DBMapping.
- Step 5: Next define the IQML for these tokens by mapping tokens to relations, field names and extracting the conditions from the tokens generated above. Store this IQML as query.iqml.

G. Convertor Algorithm

- Step 1: Take an empty string sql="select %1 From %2", condn =" where %3 ", tname="", attr="", CN="".
- Step 2: Repeat following steps for each table tag T in IQML code.
  1) Tname=tname+","+T.
  2) Repeat following steps for each column tag C under T.
     - Attr=attr + (","+T+"."+ C)
  3) Repeat following steps for each condition tag Cn under T.
     - CN=CN+ Cn + (Next=="and")? " and ": " or "
- Step 3: Replace %1 with tname, %2 with Attr and %3 with CN.
- Step 4: Add at end of sql any join condition if exists for the tables listed above.
- Step 5: Execute the query SQL generated above to get the desired result.

IV. RESULTS ANALYSIS

DB	Sqlizer		IQML based Db-Mapper	
	Parse Time	% solved	Parse Time	% solved
YELP	0.85s	78.1%	0.87s	80%

Table 3: Results Analysis

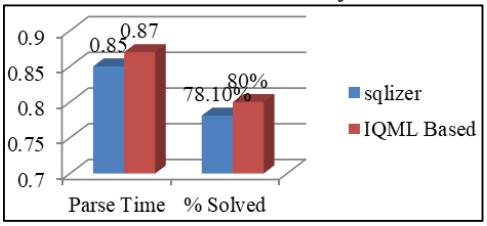

Fig. 5: Result graph

In the above graph, it is clearly evident that the proposed technique has lesser parse time and greater efficiency in terms of percentage solved, as compared to the base research by Yaghmazadeh [9]. The base paper [9] demonstrated the efficiency of their algorithm in comparison to their base research by Li[8]. The comparison there also was based on the same parameters but it worked on YELP

databases and therefore the comparison was done on the basis of YELP database.

V. CONCLUSION

Every application has its data in the form of RDBMS. These datasets are accessed by applications and the results are presented to the end users. But, sometimes the user needs to interact with the database for some results which are not directly available in the application. Then, he has to have the knowledge of SQL. To solve this issue, an advanced XML based technique is defined which will convert the natural language queries into SQL queries. The research defines an interface (IQML based DBMapper) which inputs the user query and then parses the query thereby generating an IQML from that and then converts it into a SQL query using a DBMapper that is finally executed to display the intended result. The base research by Yaghmazadeh[9] created a technique called SQLizer to work on same database. He compared his results with the base research by Li et.al[8] and demonstrated his effectiveness based on two parameters – percentage solved and parse time. The results generated and demonstrated via Charts shown above clearly convey the efficiency of our technique against the base research. In future, methods to export IQML into other commonly used formats to increase the acceptability of the language can be defined. The other improvement can be to include the other DBA provisions as a part of the interface. Converting all DBA preferences through natural language queries will certainly be a challenge.

REFERENCES

[1] Joginder Singh, Pratiba Verma and Navneet Kaur, "Natural Language Interface to Database-An Introduction", Imperial Journal of Interdisciplinary Research (IJIR), Vol. 2, Issue 1, 2016.
[2] Manika Tyagi, "Natural Language Interface to Databases: A Survey", International Journal of Science and Research (IJSR), Vol. 3, Issue 5, 2014.
[3] Lukas Blunschi, Claudio Jossen and Donald Kossman in "SODA: generating SQL for business users", 38th International conference on very large databases, Vol. 5 No. 10, August 21-31, 2012 Istanbul, Turkey.
[4] YingzhongXu and Songlin Hu, "QMapper: A Tool for SQL Optimization on RDBMS Using Query Rewriting", ACM WWW pp. 211-212, 2013.
[5] Rakesh Kumar, Neha Gupta, Shilpi Charu, Somya Bansal and Kusum Yadav, "Comparison of SQL with SQL", International Journal for Research in Technological Studies, Vol. 1, Issue 9, pp. 28-30, 2014.
[6] PrasunKanti Ghosh, Saparja Dey and Subharta Sengupta in "Automatic SQL formation from Natural Language Query" International conference on micro-electronics, circuits and systems, 2014.
[7] S. Bajwa, F. Razzaq, A. H. S. Bukhari and R. Amin, "Using Machine Learning to Generate SPARQL Queries from NL for NoSQL Database", Sindh Univ. Res. Jour. (Sci. Ser.) Vol. 48, No. 2, pp. 233-240, 2016.
[8] Fei Li and H. V. Jagadish, "NaLIR: An Interactive Natural Language Interface for Querying Relational Databases", ACM 2014.
[9] Navid Yaghmazadeh Yuepeng Wang Isil Dillig and Thomas Dillig, "Type and Content Driven Synthesis of SQL Queries from Natural Language", IEEE 2017.

