NATURAL LANGUAGE QUERY PROCESSING USING SEMANTIC GRAMMAR

[1]Gauri Rao, [2] Chanchal Agarwal, [3] Snehal Chaudhry, [4] Nikita Kulkarni, , [5] Dr. S.H. Patil

[1] Lecturer department o f Computer Engineering BVUCOE, Pune

[2] Lecturer department o f Computer Engineering BVUCOE, Pune

[3] Lecturer department o f Information Technology BVUCOE Pune

[4] Lecturer department o f Computer Engineering BVUCOE Pune

[5] Professor and Head department o f Computer Engineering Bharati Vidyapeeth University College of Engineering, Pune, India

Abstract: The field of natural language processing (NLP) has seen a dramatic shift in both research direction and methodology in the past several years. In the past, most work in computational linguistics tended to focus on purely symbolic methods. Recently, more and more work is shifting toward hybrid methods that combine new empirical corpus-based methods, including the use of probabilistic and information theoretic techniques, with traditional symbolic methods. The main purpose of Natural Language Query Processing is for an English sentence to be interpreted by the computer and appropriate action taken. Asking questions to databases in natural language is a very convenient and easy method of data access, especially for casual users who do not understand complicated database query languages such as SQL.
This paper proposes the architecture for translating English Query into SQL using Semantic Grammar

1.0 INTRODUCTION

Natural language processing is becoming one of the most active areas in Human-computer Interaction. It is a branch of AI which includes Information Retrieval, Machine Translation and Language Analysis. The goal of NLP is to enable communication between people and computers without resorting to memorization of complex commands and procedures. In other words, NLP is a technique which can make the computer understand the languages naturally used by humans. While natural language may be the easiest symbol system for people to learn and use, it has proved to be the hardest for a computer to master.

Despite the challenges, natural language processing is widely regarded as a promising and critically important endeavor in the field of computer research. The general goal for most computational linguists is to instill the computer with the ability to understand and generate natural language so that eventually people can address their computers through text as though they were addressing another person. The applications that will be possible when NLP capabilities are fully realized are impressive computers would be able to process natural language, translating languages accurately and in real time, or extracting and summarizing information from a variety of data sources, depending on the users' requests.

2.0 RELATED WORK

The very first attempts at NLP database interfaces are just as old as any other NLP research. In fact database NLP may be one of the most important successes in NLP since it began. Asking questions to databases in natural language is a very convenient and easy method of data access, especially for casual users who do not understand complicated database query languages such as SQL. The success in this area is partly because of the real-world benefits that can come from database NLP systems, and partly because NLP works very well in a single-database domain. Databases usually provide small enough domains that ambiguity problems in natural language can be resolved successfully. Here are some examples of database NLP systems:

LUNAR (Woods, 1973) involved a system that answered questions about rock samples brought back from the moon. Two databases were used, the chemical analyses and the literature references. The

program used an Augmented Transition Network (ATN) parser and Woods' Procedural Semantics. The system was informally demonstrated at the Second Annual Lunar Science Conference in 1971. [1]

LIFER/LADDER was one of the first good database NLP systems. It was designed as a natural language interface to a database of information about US Navy ships. This system, as described in a paper by Hendrix (1978), used a semantic grammar to parse questions and query a distributed database. The LIFER/LADDER system could only support simple one-table queries or multiple table queries with easy join conditions. [4]

3.0 SYSTEM DESCRIPTION

A brief description of the system is as follows: Suppose we consider a database say ORACLE. Within this oracle database I have placed certain tables, which are properly normalized. Now if the user wishes to access the data from the table, he/she has to be technically proficient in the SQL language to make a query for the ORACLE database. Our system eliminates this part and enables the end user to access the tables in his/her language.

Let us take an example:

Suppose if we want to view information of a particular employee from EMP table then we are supposed to use the following query:

SELECT * FROM EMP WHERE e_name =’ABC’;

But a person, who doesn’t know ORACLE, will not be able to access the database unless he/she knows the syntax and semantics of firing a query to the database. But using NLP, this task of accessing the database will be much simpler. So the above query will be rewritten using NLP as:

Give the information of employee whose name is ABC.

Both the SQL statement and NLP statement to access the EMP table would result in the same output the only difference being, a normal person who doesn’t know anything about SQL can easily access the ORACLE database.

4.0 SCOPE OF THE SYSTEM

The scope of the proposed system is as follows:

- To work with any RDBMS one should know the syntax of the commands of that particular database software (Microsoft SQL, Oracle, etc.).
- Here the Natural language processing is done on English i.e. the input statements have to be in English.
- Input from the user is taken in the form of questions (wh- form like what, who, where, etc).
- A limited Data Dictionary is used where all possible words related to a particular system will be included. The Data Dictionary of the system must be regularly updated with words that are specific to the particular system
- Ambiguity among the words will be taken care of while processing the natural language.
- All the names in the input natural language statement have to be in double quotes.
- Data Dictionary used will be: - EMP, DEPT and PROJECT

The tables used are shown below:-

Employee	departmen	project
id	id	id
name	name	name
salary	location	empid
edptid	capacity	

Fig 1 Tables used in the System

4.1 System Requirements

4.1.1 Hardware Requirements

P1 (Pentium I) or higher processor
RAM – minimum 64MB RAM
HDD - minimum 500MB of free Hard-disk space
VGA Display Adapter – with higher resolution

4.1.2 Software Requirements

Operating System – WINDOWS 98 or above
Database – SQL SERVER 2000 or ACCESS
Java, MS Visual C++ compiler 5.0/6.0

5.0 SYSTEM ARCHITECTURE

Generally NLP has following steps:-

- Morphological Analysis: Individual words are analyzed into their components and non word tokens such as punctuation are separated from the words.
- Syntactic Analysis: Linear sequences of words are transformed into structures that show how the words relate to each other.
- Semantic Analysis: The structures created by the syntactic analyzer are assigned meanings.
- Discourse integration: The meaning of an individual sentence may depend on the sentences that precede it and may influence the meanings of the sentences that follow it.
- Pragmatic Analysis: The structure representing what was said is reinterpreted to determine what was actually meant

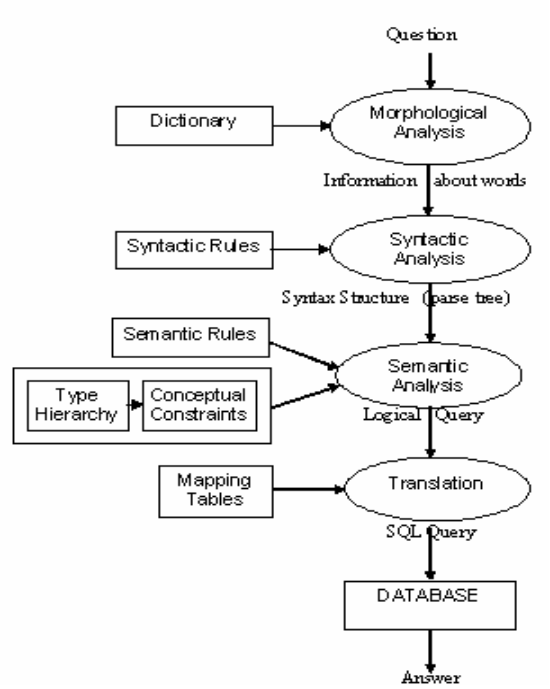

Fig.2 Stages in Natural Language Interpretation Process

The system includes the following modules:

- GUI: Designing the front end or the user interface where the user will enter the query in Natural Language.
- Parsing: Derives the Semantics of the Natural Query given by the user and parses it in its technical form.
- Query Generation: After the successful parsing of the statement given by the user, the system generates a query against the user statement in SQL and further gives it to the back end database.
- Data Collection: This module collects the output of the SQL statement and places it in the User Interface Screen as a result form.

6.0 SEMANTIC GRAMMAR

There are two main parts of a semantic grammar. The first is a **lexicon** that stores all the possible words that the grammar is aware of. A simple entry in the lexicon might look like this:-

(customer -> customer patron member)

(customers -> customers patrons members)

On the left-hand side of the '->' the word 'customer' defines a symbol that can be used in the grammar. When 'customer' is used in the grammar, it refers to the English words on the right-hand side of the '->', that is 'customer', 'patron' or 'member'. Similarly, the plurals of these words are shown in the next line. Only single English words, or terminal symbols, can appear in the lexicon.

The other part of the semantic grammar involves **rules** to combine the terminal symbols in the lexicon to form phrases or sentences in a specific way.

For example, the rule

(ATT_TAPE_CUSTOMER -> RENTED by customer ATT_NUMBER)

Demonstrates how the TAPE.CUSTOMER attribute can be referred to in English. On the left-hand side is the non-terminal representation of the phrase.

This can be used in other rules to refer to this phrase. On the right hand side is a combination of non-terminal symbols from other rules (in UPPER CASE) and from the lexicon (in lower case). RENTED represents another rule for synonyms of the verb 'rented', such as taken out, borrowed etc. Note that RENTED is not in the lexicon because 'taken out' is a two-word phrase, and so must be defined as a rule. The words 'by' and 'customer' appear in the lexicon, and the ATT_NUMBER symbol refers to a number attribute. Now the above rule can be used to parse the phrase "borrowed by member number 22", or "taken out by customer 14" etc.

7.0 OVERVIEW OF PROCESS

Firstly, the English input (in the form of a list) is parsed by the semantic grammar, then a post-processor matches table and attribute names and joins up tables if the query involves more than one table. After that the post-processor can construct the resulting SQL query and output it.

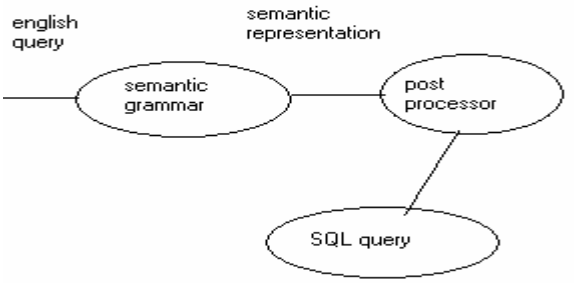

Fig.3 Database Semantics

The question "What is employee name with salary greater than 2000" is processed by the system as given below.

The English text is converted into a list, so that the system can process it. The sentence becomes:

(WHAT IS EMPLOYEE NAME WITH SALARY GREATER THAN 2000)

Now this list is parsed by the semantic grammar. The parser attempts to find all possible parses of the

sentence, using the semantic grammar. This sentence only has one possible parse, so only one semantic representation is returned:
((SELECT NAME! FROM (EMP (WHERE (SALARY >= 2000)))))
The semantics already look similar to an SQL query. The brackets in the semantics indicate how the phrases from the sentence relate to each other. For example, the (EMP (WHERE (SALARY>= 2000))) section shows that the 'born in or after 1920' condition relates to the director. The 'NAME!' symbol, after SELECT, is a special representation for both the first and last names.
This semantic representation of the sentence is now taken by a post-processor to transform it into an SQL query. The 'NAME!' symbol is expanded to FNAME and LNAME and these are identified as attributes in the EMP table.The DEPT attribute is also matched to the EMP table because the WHERE clause directly refers to the EMP table. After processing the semantics, there is only one table needed for this query, so only the EMP table is included in the FROM clause, and there is no need to add join conditions.
The aim of a natural language interface is to facilitate the user to computer in a natural way. For this purpose, we have designed a domain specific dictionary to keep the synonyms of the columns and tables Names. The addition of synonyms makes it possible for the user to write a sentence in different natural ways. We call this dictionary as semantic dictionary, because there is no syntactic information for tokens.

To build an appropriate interface for inputting database semantics information we need to develop a framework for this information. This will allow us to identify the different areas needed and easily obtain the information from the user. The first thing we need to know about the database is what the tables and attributes represent in the real world. The second thing is how they relate to each other. Finally we need to know the relationships from attributes to the tables in which they are contained. Attributes that are foreign keys should not be considered in this phase, because they link tables to other tables, and the appropriate phrases should have been covered by the table-table relationships.
Translating from one language to another always requires some kind of paring. Parsing is the process of identifying structure in data. It determines whether the input data has some pre-determined structure and respond accordingly.
Parsing requires a set of grammars to be defined. These grammars are set of rules, which define how the language is structured. Rules are specific pattern of data, which appears in the input. These rules can further reference to other rules, which are called sub rules. Rules can also be recursive if they somehow refer back to themselves. The rules consist of products, which are the different ways in which the rule can be satisfied. The actual data is referred to as terminals. Parser may also contain actions. To translate natural language queries, system needs to have a parser, which contains all these grammar rules. The parser also has to know the table names, column names, any relationships between tables and related words for the table and column names. Only the most common grammar is included in the parser.

8.0 ALGORITHM

- Tokenization (scanning)
- Split the Query in tokens
- Give order number to each token identified
- Split Query and extract patterns
  - Look for sentence connectors/criteria words
  - Break Query on the basis of connector/criteria tokens.
  - Use criteria tokens to specify condition in query.
  - Find attributes and values after criteria token
- Map value for identified attribute and corresponding table
- Replace synonyms with proper attribute names
- Get intermediate form of Query
- Transform it into SQL

9.0 RESULTS

The graphical User Interface is as shown below. The user has to first login and then connect to the database. A database setting is required to access the database after getting the information about host name, database name, user name and the password.
Once the database is setup, English Query can translate very complex English queries to SQL with the capability of searching multiple tables and multiple fields.
Steps followed to get the result:-

- Type the natural language query in the dialogue box given.
- Click on “Proceed” button.
- The system will ask the user for the expected meaning. In case of ambiguities the user has to select the desired query.
- Click on “Generate SQL” button.
- The system will generate SQL query.
- Click on “Run Query”.

- The result of the query will be displayed in the Output box.

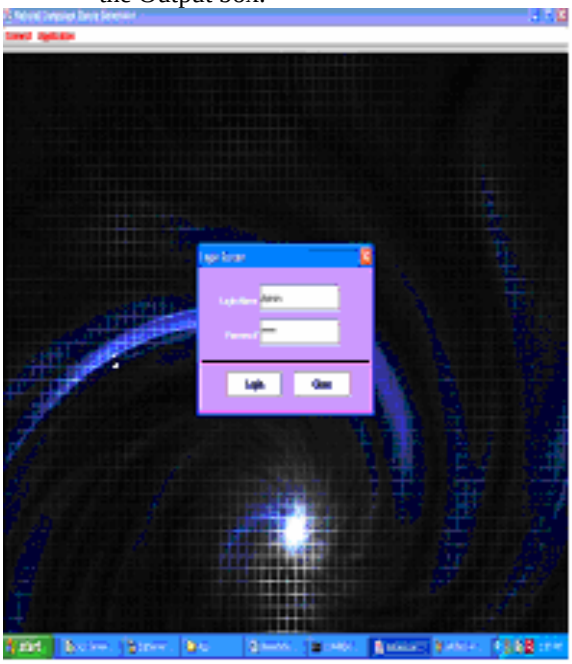

Fig .4 Login screen

Following example demonstrate SQL statement translated by English Query:-

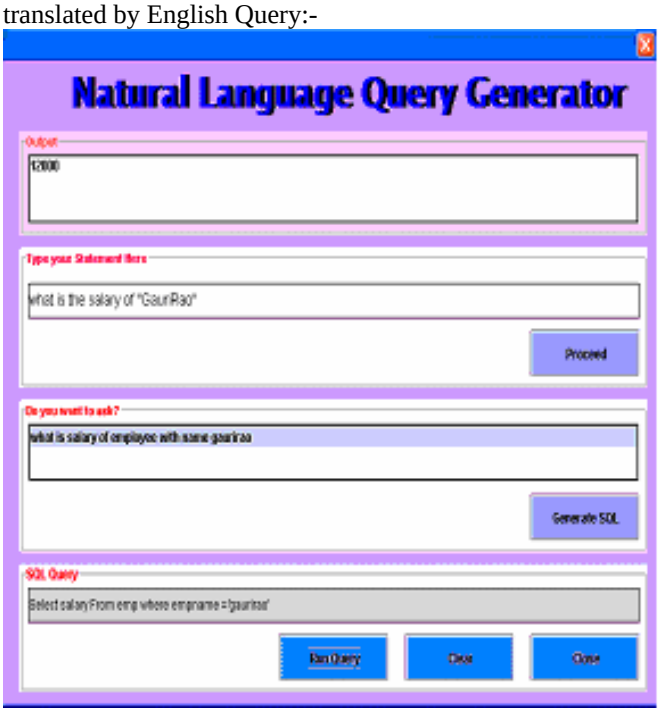

Fig.5 Snapshot of query: What is the salary of "Gauri"

English Query: - What is the salary of "GauriRao"
Meaning of Query:- salary of employee with name gauri rao
SQL Query:- Select salary From emp where empname = "gaurirao"

Output :- 12000

10.0 FUTURE ENHANCEMENT

More new grammar can be added to the parser to increase effectiveness.
Adding a thesaurus is another suggestion, which could help automating the related words for table and column names.
With the help of a thesaurus, the user input can be pre- processed to substitute related words with table or column names and also remove unwanted words.
So far, this system considers *selection* and a few simple aggregations. The next step of the research is, to accommodate more complex queries.

11.0 CONCLUSION

Natural Language Processing can bring powerful enhancements to virtually any computer program interface. This system is currently capable of handling simple queries with standard join conditions. Because not all forms of SQL queries are supported, further development would be required before the system can be used within NLDBI. Alternatives for integrating a database NLP component into the NLDBI were considered and assessed.

12.0 References

[1] Huangi,Guiang Zangi, Phillip C-Y Sheu "A Natural Language database Interface based on probabilistic context free grammar", IEEE International workshop on Semantic Computing and Systems 2008
[2] Akama, S. (Ed.) *Logic, language and computation,* Kulwer Academic publishers, pp. 7-11, 1997.
[3] ELF Software CO. *Natural-Language Database Interfaces from ELF Software Co,* cited November 1999, available from Internet: http://hometown.aol.com/elfsoft/
[4] Hendrix, G.G., Sacerdoti, E.D., Sagalowicz, D., Slocum, J. "Developing a natural language interface to complex data", in *ACM Transactions on database systems,* 3(2), pp. 105-147, 1978.
[5] Joseph, S.W., Aleliunas, R. "A knowledge-based subsystem for a natural language interface to a database that predicts and explains query failures", in *IEEE CH,* pp. 80-87, 1991.
[6] Mitrovic, A. *A knowledge-based teaching system for SQL,* University of Canterbury, 1998. Moore, J.D. "Discourse generation for instructional applications: making computer tutors more like humans", in *Proceedings AI-ED,* pp.36-42, 1995.
[7] Suh, K.S., Perkins, W.C., "The effects of a system echo in a restricted natural language database interface for novice users", in *IEEE System sciences,* 4, pp. 594-599, 1994.
[8] Whenhua, W., Dilts, D.M. "Integrating diverse CIM data bases: the role of natural language interface", in *IEEE Transactions on systems, man, and cybernetics,* 22(6), pp. 1331-1347, 1992.
[9] Dan Klein, Christopher D. Manning: Corpus-Based Induction of Syntactic Structure: Models of Dependency and Constituency. ACL 2004: 478-485.