ABSTRACT

This project aims at developing a system which will accept English query from user and convert it into SQL. This helps novice user who can easily get required contents without knowing any complex details of SQL languages. We can store huge amount of data in databases but casual user who doesn’t have any technical background not able to access data. So this paper proposes system that will convert English statement given by user to all possible intermediate queries so that user can select appropriate intermediate query and then system will generate SQL query from intermediate one. Finally system will fire SQL query on database and gives output to user. When an interpretation error occurs, users often get stuck and cannot recover due to a lack of guidance from the system. To solve this problem, we present a natural language query recommendation framework.

KEYWORDS: NLP, SQL, Morphological, Syntactic, Discourse, Pragmatic

INTRODUCTION

NLP is a form of human to computer interaction where the elements of human language are formalized so that Computer can perform value adding tasks based on that interactions. The word Natural Language means language used by human beings in their day to day life what they write, what they read. Everyone is more comfortable with Natural Language than other computer commands. So now lots of research is going on in field of Natural Language. It is more difficult for any machine to understand Natural Language. That’s why we have to do parsing so that we can convert any natural language statement into computer commands because understanding semantics of any natural language statement is more difficult for machine as there are so many statements having similar meaning.

Natural Language Processing holds great promise for making computer interfaces that are easier to use for people, since people will (hopefully) be able to interact with the computer in their own language, rather than learn a specialized language of computer commands.

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. **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.

**SYSTEM DESCRIPTION**

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. Some word sequences may be rejected if they violate the language’s rules for how words may be combined.

For example An English semantic analyzer would reject the sentence “Boy the go the to store ”.

**Semantic Analysis**

The structures created by the syntactic analyzer are assigned meanings. In other word mapping is made between syntactic structure and objects in the task domain. Structures for which no such mapping is possible may be rejected.

For example In most universes the sentence “Colorless green ideas sleep furiously” would be rejected as semantically anomalous.

**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.

For example The word “it” in the sentence “John wanted it” depends on the prior discourse context, while the word “John” may influence the meaning of later sentence (such as “he always had”).

**Pragmatic Analysis**

The structure representing what was said is reinterpreted to determine what was actually meant.

For example The sentence “Do you know what time it is?” should be interpreted as request to be told the time.

The boundaries between these five phases are often very fuzzy. The phases are sometimes performed in sequence; One may need to appeal for assistance to another. For example part of process of performing the syntactic analysis of the sentence “Is the glass jar peanut butter?” is deciding how to form two noun phrases out of four nouns at the end of the sentence.

We consider a database SQL Server 2005. We have placed 3 tables in this SQL Server database. Novice users can not access contents of databases as they don’t have knowledge of SQL language. That’s why we proposed system which will enable user to access contents of databases using simple English language. Suppose we want salary of an employee whose name is “Saharsha” then we have to form a SQL query: SELECT salary FROM Employee WHERE
emp_name="Saharsha";

For a novice user it is not possible to form SQL query so using our system he/she can simply asks a question like “What is salary of employee Saharsha?”

In our daily life we always use a WH question that’s why proposed system easily interprets WH questions and generates its relevant intermediate query. In addition with WH questions our system works with In Which, Total Number Of, On Which type questions.

PROPOSED SYSTEM

When user opens system he/she has to establish connection to database and then he/she can fire queries to database. User can asks queries to database in ‘How many’, ‘Total number of’, ‘In which’ format in addition with WH formats. Our system also provides facility to update tables in database. User can insert values into tables and can also delete values from table. Our system generates number of intermediate queries depending on semantics of user entered English statement. User have to select one of intermediate query which is more relevant to user’s intended query. Then system will generate its appropriate SQL query. Our system also works fine with JOIN. User can retrieve data from two or more columns also.

Firstly system accepts English statement from user then system tokenized that statement and removes unwanted words. After that it identifies synonyms of column names and table names then replace synonyms with actual names. System places tokens in 4 parts depending on criteria words and then properly placing that parts generates one or more intermediate statements. This is one part of the system which only generates intermediate query. System simplify decision making task by relying on user for selection of intermediate query. This also helps to system to give proper output to the user and user can also easily recover from mistakes.

After user selects intermediate query system’s GenerateSQL module takes it as input and finds out 3 main keywords i.e. Select keyword, From keyword, Where keyword. Select keyword contains attributes that user wants to retrieve. From keyword contains table name from which user wants to retrieve attributes. From keyword can also contain more than one table then system has to generate query using JOIN as there is relationship between tables. Where keyword contains criteria which helps to retrieve specific contents by placing condition. Then GenerateSQL formats all these keywords in specific format and using different conditions that means formatting From keyword is different when there is only one table and different in case of two or more tables where we have to use JOIN. Then it places these keywords in standard SQL query and generate SQL.
SYSTEM MODULES

There are 5 modules in this project. They are -

- User Interface
- Process Query
- Generate Intermediate
- Formation of SQL
- Formation of Output

User Interface

This module contains user interface in which user can enter query. User Interface is designed such that any novice user can easily submit query and also can view intermediate form of SQL query which helps user to easily recover from errors.

Process Query

This module takes user query which is in natural language then it performs tokenization on that query and divides query into parts depending on criteria words or expressions.

Generate Intermediate

This module generates simple English statements which is formed using tokens that are derived from previous module. This module may generate more than one statement if system finds more than one semantic of user’s English query. So that user can select statement which is more relevant to his/her intended query.

Formation of SQL

This module takes intermediate query selected by user as input and generates SQL query by interpreting meaning of intermediate.

Formation of Output

This module collects result after that SQL query is fired on database and displays result to user in formatted manner.

ALGORITHM

- Process Query
  - Divide Query in tokens.
  - Remove punctuation marks.
  - Do initializations.
- Generate Intermediate
  - Divide Query into parts using criteria words.
  - Identify column attributes and table names from user Query and remove unwanted words.
  - Replace synonyms of column attributes and table names in Query with it’s actual names.
  - Arrange parts in proper sequence.
- Formation of SQL Query
  - Take intermediate Query as input
o Identify/ derive 3 things from Query:-

- Select keyword: - These are attributes which user wants to retrieve.
- From keyword: - This is the table name from which user want to retrieve data.
- Where keyword: - This is condition specified in query.
  o Replace select keyword with actual table attributes.
  o If there is only one from keyword then Replace it with actual table name.
else
form following sequence
table name1 JOIN table name2 ON attribute1(primary key of table1) = attribute2(attribute in table2 which is foreign key of table1)
  o Replace where keyword’s attribute with actual table attribute and concatenate ‘=’ following with value specified by user.
  o Form standard template of SELECT Query and substitute above keywords i.e. select keyword, from keyword, and where keyword in their appropriate place.

OVERVIEW OF PROCESS

The question "What is name and salary of employee with id 1" is processed by the system as given below.
Now we consider some examples and see how system handles them. Assume our database contains two tables department and employee in normalized form. Firstly we will take following example:-

How many employees works in department Management?

Then by processing above English query system generates intermediate query i.e.

What is employee count of department with name Management

Generate SQL modules takes above query as an input and firstly finds out all attributes and table names then by interpreting meaning identify relation between tables and form query using JOIN condition. Output of GenerateSQL module for above query is as follows :-

Select count(empid) from emp JOIN dept ON edeptid = deptid where deptname = 'Management'

CONCLUSIONS

Use of Natural Language brings ease for any human being. This system helps user to easily retrieve data from database using simple English language. This system works fine with JOIN condition. This system also responds to complex eries. We can add more synonyms for column names and table names so that system is able to handle more queries. This system also provides some recommendations so that it is helpful for user. In future we can add some strong recommendation framework in this system so that user have to take less efforts.

This system uses static database so if we want to add any other table in database we also have to add grammar to handle queries for that table as grammar is hard coded but we can also remove this problem by constructing a dynamic framework in which user can dynamically add new tables and remove older ones. In this architecture we have to generate grammar dynamically which can be future enhancement for this system.

REFERENCES

3. Natural Language Query Recommendation in Conversation Systems James Shaw , Shimei Pan IBM T.J. Watson Research Center 19 Skyline Drive Hawthorne, NY 10532