NATURAL LANGUAGE INTERFACE FOR STUDENT INFORMATION SYSTEM (NLSIS)

Amisha Shingala*, Rinku Chavda and Paresh Virparia

*Department of MCA, SVIT, Vasad, Gujarat, India
1G. H. Patel Computer Science Department, Sardar Patel University, Vallabh Vidyanagar, Gujarat, India

ABSTRACT

Any computer system that interacts with a user has a high utility value and ease. Natural Language Interface is a concept for making computer interface with a user who wants to retrieve the information from a computer database with its own language rather than learning a specialized language. The challenges in Natural language arise due to difficulty in correct interpretation, disambiguation and context resolution.

We present our work in designing and implementing the Natural Language Interface for querying Student Information System. It semantically parses the natural language question and built corresponding structured query from the database. The data which is to be retrieved can be in the form of a spreadsheet or database. The system is developed in Java using JDBC for Student Information System.

Keywords: Language, Database Interface, Structured Query Language, Data Retrieval

INTRODUCTION

The computer era has begun from last few years and the phase of bringing awareness into different walks of society regarding use and benefits of computer increased successfully. Not only industry but educational institute, service sectors, manufacturing sectors etc are using computers to store, process and update the information. The huge amount of data are stored in repository called database and in order to query or retrieve information from a database by general public, a Natural Language (English) will provide correct and precise information without knowing the depth of SQL query language. The idea of using Natural language with database prompted the development of new type of processing method called Natural Language Interface to Database.

Some of the earlier attempts in providing Natural Language Interface are given below:

2) LUNAR [13] that answered questions about rock samples back from the moon.
3) LIFER/LADER describe by Hendrix [9] was designed as a natural language interface to database about US Navy ships.
4) Warren el al’s CHAT-80[12] and Auxerre et al’s transform written English questions into prolog queries which are executed against Prolog database.

Information related to overview of various Natural Language Interfaces and the difference between Natural Language Interface and Question Answering could be found in reference [1]. The approach used in most of the recent systems is to handle the entire query as one entity for transformation to SQL either through conversion to intermediate representation or directly. We had overcome many limitations which were given by earlier researchers [11].

MODEL AND METHODOLOGY OF NATURAL LANGUAGE STUDENT INFORMATION SYSTEM (NLSIS)

The following goals were set for the proposed NLSIS:

- **Fault Tolerance** – An algorithm [2] is developed for tolerating spelling errors, which uses degree of phonetic match and degree of spelling match to correct misspelled words. Since the semantics of the query for purpose of conversion to SQL statement within a limited domain does not use the grammatical structure of sentence formation, it is implicitly tolerated. However, use of grossly, inappropriate words (particularly prepositions e.g. using “by” in place of “of”, etc and interrogative pronouns) may lead to wrong or failed interpretation of the user query.

- **Better Context Resolution and Disambiguation** – We first partition the query into intermediate language and then to SQL clause. A part of query with associate SQL along with domain-specific lexicon leads to an improved context resolution and disambiguation.

- **Multiple database tool support:** A tool is developed which can covert data from spreadsheet to database and database to relational database.

- **History Log Maintenance:** The query once asked, was stored in separate log which can be help in retrieving the same information again without processing whole algorithm.

Domain-Specific Ontology

Figure A present below ontology of Student Information System (SIS) domain.

*Corresponding author : am_bt@yahoo.com*
A. The Linguistic model consists of Lexical Analysis, Parser and Semantic representation as shown in figure B which is explained below:

Lexical Analysis: Here the natural language sentence is divided into smaller fragments called tokens. The elements in natural language query are words or special characters. This process is performed by following function:

i. Token analyzing function: It is used to generate the token which is treated as a single unit.

ii. Spell checker function: It makes sure that the user inputted query with correct word.

iii. Ambiguity reduction: It reduces ambiguity of sentence and simplified the task of parser by substituting multiple words or symbols with base word. E.g. Comma considered as AND, co-occurring word as single word etc. Also, we try to find out that the expected answer would be named entity or not. Figure C represent hierarchy of named entity [7]

Stanford Parser: The Stanford Parser is a probabilistic parser which uses the knowledge of language gained from hand-parsed sentences to try to produce the most likely analysis of new sentences as shown in figure D. This package is a Java implementation of probabilistic natural language parsers.

The Stanford dependencies provide a representation of grammatical relations between words in a sentence for any user who wants to extract textual relationships [3]. The dependency obtained from Stanford parser can be mapped directly to graphical representation in which words in a sentence are nodes in the graph and grammatical relationships are edge labels [6]. In our system, we use the POS tagger and Typed Dependency for an inputted query or sentences.

Semantic Representation: We had used two types of lexicon semantic representation in the form of grammar for terminal words, non terminal words and terminal symbols.

<table>
<thead>
<tr>
<th>Terminal Words</th>
<th>Semantics/Base Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>girls, girl, female, madam</td>
<td>Female</td>
</tr>
<tr>
<td>Percentage above 70</td>
<td>Distinction</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NonTerminal Words</th>
<th>Semantics/Base Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greater than or equal to</td>
<td>&gt;=</td>
</tr>
<tr>
<td>Less, lower or lesser</td>
<td>&lt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Terminal Symbols</th>
<th>Semantics/Base Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greater, larger, bigger, biggest</td>
<td>Greater</td>
</tr>
<tr>
<td>Equal to, Like, equals, same</td>
<td>Equal</td>
</tr>
</tbody>
</table>

B. SQL Constructing Component

This component consists of SQL generator, Database Adaptor and SQL execution.

SQL Generator: The task is to map element of the Natural Language sentence into an actual element of SQL. We consider
only SELECT as a statement to view the data in various forms. We used BNF form of SQL grammar [10]

\[
\text{<select query>} ::= \text{<select>} \ [\text{UNION} \ [\text{ALL} \ | \ \text{DISTINCT} \ ] \ \text{<select query>}]
\]

\[
\text{<select>} ::= \text{SELECT} \ [\text{ALL} \ | \ \text{DISTINCT}] \ \text{<select list>} \ \text{<from clause>}
\]

\[
\text{<where clause>} \ ::= \text{WHERE} \ <\text{search conditions}>\]

\[
\text{<search condition>} \ ::= \text{logical predicate composed of logical conditions with AND and OR operator. The <aggregate functions> like max, min, sum, average and count transforms set of rows into scalar statement. The <condition> is a expression in form } X \ Y \text{ where } X \text{ and } Y \text{ are the set of values representing column, aggregate function, constant or NULL and R can be operated like } \{<,>,<=,>=,!=,==}\.
\]

Database Adaptor: The data repository can be in the form of a spreadsheet or database. Our algorithm converts these data into MySQL database and retrieve the relevant information or query and join query. Figure E shown implemented version of the system.

SQL Executor: The task is to map the SQL generated query to Database Adaptor and retrieve the relevant information or answer by connecting to the appropriate database tool.

3. Algorithm for NLSIS

(i) Read statement or user query S

(ii) Search S from History Log H

If found, call query generate algorithm and retrieve the query Q

Execute the Query Q and Display the answer Ans.

(iii) If S not successful, perform all the steps from step iv.

(iv) For each word W from S do

If W base word then

Add W to symbol table ST

End if

End for

(v) For each Wi from ST do

Add Wi to Parse tree

End for

(vi) Display POS tagger and Typed dependency and get relationship between words PDR

(vii) For each PDR do

If metadata = PDR then

Add table name, attribute and condition to OUT

End if

End for

(viii) For each OUT do

Generate the SQL query Q1

Call Data conversion algorithm

Execute SQL query Q

End for

(ix) Display SQL query Ans.

RESULTS and DISCUSSION

The software has been subjected to test with a number of volunteers phrasing the queries differently. In around 70% cases, the system could correctly interpret and process the query. Currently, the system fails in some cases to handle very complex queries coupled with fuzzy terms. We had paraphrased different queries in the form of simple query, condition query, order by query and join query. Figure E shown implemented version of the system.
CONCLUSION AND FUTURE WORK

The degree of correct interpretation and processing of Natural Language Queries relies on the extent of exhaustiveness of domain-specific lexicon. Despite of some limitations, the proposed system provides a robust tool to handle Natural Language Queries allowing fault-tolerance to some degree. The work can be further extended by providing multilingual (Gujarati or Hindi language) query to SQL conversion.

REFERENCES


