International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 05, May - 2017

Possibility of Amharic Query Processing in Database using Natural Language Interface

Smegnew Asemie[1],
[1]Mizan Tepi University,
School of computing and informatics,
Tepi, Ethiopia

Tefery kibebew[2]
[2]Jimma University,
School of Computing and Informatics,
Jimma, Ethiopia

Getachew Mamo[2],
[2]Jimma University,
School of Computing and Informatics,
Jimma, Ethiopia

***Abstract*— In the present computing world, computer based information technologies have been extensively used to help many organizations, private companies, academic and education institutions to manage their processes and information systems. A general information management system that is capable of managing several kinds of data, stored in the database is known as Database Management System (DBMS). Database Management System is a collection of interrelated data and set of programs to access those data. In this paper, we have designed and developed Amharic Language Interface for database, so that user can easily communicate with database without the knowledge of database language like SQL. So, in order to address this issue we have developed an algorithm to efficiently map Amharic language into Structured Query Language (SQL). We divided the algorithm into four parts an algorithm to handle query selection, an algorithm to handle conditional query, an algorithm to handle aggregation and grouping and ordering queries. The algorithm has been implemented in Java and tasted on Human Resource Management (HRM) database containing Employee, Department and Employee on education tables. The experimental result shows that 91% overall accuracy. However, the system shows better performance in single and multiple conditional query, grouping and ordering query and aggregation functions, it doesn't work with Amharic temporal queries. Further improvement will be done to include temporal queries in ANLIDB.**

Keywords— ***NLIDB, Amharic Language Interface to Database, Natural Language Processing (NLP)***

INTRODUCTION

Since a long time ago, information has been playing an important role in our lives; most people try to get the information they need before making a decision. One of the major sources of information is database. Database contains a collection of related data, stored in a systematic way to model the part of the world. In order to extract information from a database, one needs to formulate a query in such a way that the computer will understand and produce the desired output. However, not everybody is able to write such queries, especially those who lack a computer background [1].

A language is the primary means of communication used by humans. Natural Language Processing (NLP) is a technique which can make a computer to understand a natural language and easily communicate with a human being. In the context of Human Computer Interaction (HCI), there are many NLP applications such as Information Retrieval Systems, Information Extraction, Speech Recognition, Language Translator, Question Answering, (QA) Natural Language Interface to Database (NLIDB), and Dialog Systems [2].

NLIDB deals with representation of user request to database in his/her native language. NLIDB then maps the user request in standard SQL to retrieve desired results from the target database. The purpose of this interface/system is to facilitate access by the user through hiding complexities of database query language syntax. Thus the user writes his/her request similar to email message and submit to NLIDB system. The system then understands the request and translates it in accurate database query so that the precise results can be retrieved.

Amharic is the official working language of the Federal Democratic Republic of Ethiopia and thus has official status nationwide and the official or working language of several of the states/regions within the federal system, including Afar, Amhara, Somali, Gambela, Benishangule Gumuz, Oromia, Tigray, and the multi-ethnic Southern Nations, Nationalities and Peoples region. The language is spoken as a mother tongue by a large segment of the population in the northern and central regions of Ethiopia and as a second language by many others. It is the second most spoken Semitic language in the world next to Arabic [3]. One of the major differences between Amharic and Semitic languages like Arabic and Hebrew is that Amharic is written from left to right as of English [4].

So, to make database applications easy to use for these people, we have present a model and an algorithm to convert Amharic sentence into SQL and retrieve relevant text from a Relational Database. It has been use HRM database with Employee, Department, and Employee on education table as a case study for developing a natural language query processing from a database system. The query has been asked in Amharic sentence for retrieving relevant information from databases and display the results in the same.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 05, May - 2017

I. *Natural Language Interface to Database (NLIDB)*

One may find it intricate and frustrating to interact with a foreign person with no knowledge of English. Thus, a translator will have to come into the picture to allow one to communicate with the foreigner. Companies have related this problem to extracting data from a Database Management System (DBMS) such as MS Access, Oracle and others. A person with no knowledge of Structured Query Language (SQL) may find himself or herself handicapped to correspond with the database.

Natural Language Interface to Database (NLIDB) is a system that allows the user to access information stored in a database by typing requests expressed in some natural language. In the last few decades, many NLIDB systems have been developed through which users can interact with database in a more convenient and flexible way. Because of this, this application of NLP is still very widely used today [6]. Natural Language Interface has been a very interesting area of research since the past. The aim of Natural language Interface to Database is to provide an interface where a user can interact with database more easily using his/her natural language and access or retrieve his/her information [1]. Moreover, the NLIDB is a system that converts the query in native language into SQL.

Components of NLIDB

Computing scientists have divided the problem of natural language access to a database into two sub-components [1]: (a) Linguistic component and (b) Database Component. The Linguistic Component translates the natural language input to an expression of Intermediate Query Representation (IQR), which is subsequently passed to Database Component for generation of Structured Query Language (SQL) statement. The resulting SQL statement is then executed by relational database management system. The Linguistic Component consists of morphological analysis, query pre-processing & context resolution, lexical analysis, syntactical analysis and semantic analysis. On the other hand, Database Component consists of SQL query generation and SQL query execution.

Techniques Used for Developing NLIDB

Pattern-Matching Systems: - Pattern matching system is the earliest and the simplest techniques to implement natural language interface to database (NLIDB). These patterns and rules are fixed [7]. The rules states that if an input word or sentence is matched with the given pattern, the action has been taken. Those actions are also mention in the database [8]. The main advantage of pattern matching approach is that no elaborate parsing and modules of interpretation are required and the systems are very easy to implement. Also, pattern-matching systems often manage to come up with some reasonable answer, even if the input is out of the range of sentences the patterns were designed to handle [5].

Syntax-Based Systems: - In syntax based system user questions are analyzed syntactically i.e. it is parsed and the resulting syntactic tree is mapped to an expression in some database query language [1]. Syntax-based systems use a grammar that describes the possible syntactic structures of the user's questions. The main advantage of using syntax based approaches is that they provide detailed information about the structure of a sentence. A parse tree contains a lot of information about the sentence structure; starting from a single word and its part of speech, how words can be grouped together to form a phrase, how phrases can be grouped together to form more complex phrases, until a complete sentence is built. As Neelu Nihalani [9] present the problem of syntax based, unfortunately not all nodes should be mapped, some nodes have to be left just as they are without adding any semantic meanings. And it is not always clear which nodes should be mapped and which should not. The second problem is a sentence can have multiple correct parse trees, and if all are translated, they may lead to different query results. The last problem is that it is difficult for a syntax based approach to directly map a parse tree into some general database query language, such as SQL.

Semantic Grammar Systems: - In semantic grammar systems, the requests and responses is still done by parsing the input and mapping the parse tree to a database query. The difference, in this case, is that the grammar's categories do not necessarily correspond to syntactic concepts. The basic idea of a semantic grammar system is to simplify the parse tree as much as possible, by removing unnecessary nodes or combining some nodes together [9]. The main drawback of semantic grammar approach is that it requires some prior-knowledge of the elements in the domain, therefore making it difficult to port to other domains. In addition, a parse tree in a semantic grammar system has specific structures and unique node labels, which could hardly be useful for other applications.

Intermediate Representation Languages: - Due to the difficulties of directly translating a sentence into a general database query languages using a syntax based approach, the intermediate representation systems were proposed. The idea is to map a sentence into a logical query language first, and then further translate this logical query language into a general database query language, such as SQL [5]. In the intermediate representation language approach, the system can be divided into two parts. One part starts from a sentence up to the generation of a logical query. The other part starts from a logical query until the generation of a database query. In the part one, the use of logic query languages makes it possible to add reasoning capabilities to the system by embedding the reasoning part inside a logic statement. In addition, because the logic query languages are independent from the database, it can be ported to different database query languages as well as to other domains, such as expert systems and operating systems [9].

II. OVERVIEW OF THE PROPOSED SYSTEM

We have analyzed the language structure of Amharic to categorize column name for condition and column name for selection. To do so, we have analyzed the index number of the word place in a given NL requests without any additional language modules like parser, POS etc.

To map natural language tokens into table name and column name, we are prepared a dictionary. This dictionary contains

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 05, May - 2017

Amharic token words that express table name, column name, and conditional words with appropriate meaning of map words. Table: 1 shows sample token word and appropriate map word.

Table 1: sample column handling table

Token word	Mapped word
የመታወቂያ ቁጥር	Id
ጾታ	SEX
ያጠኑት የትምህርት አይነት| |የተመረቁበት የትምህርት አይነት	FILDE_OF_STUDY
የተቀጠሩበት ቀን| |ስራ የጀመሩበት ቀን	EMPLOYMENT_DATE
ትምህርት ክፍል| |ዲፓርትመንት	DEPARTMENT
ኮሌጅ	COLLEG
ደመወዝ| |ወርሀዊ ክፍያ	SALARY
ስም	NAME

The algorithm used for identifying table name, column name, or conditional words from the given natural language input is listed below.

Step1.
Accept the query in Amharic

Step2.
Tokenize the sentence using white space and Amharic punctuation like ?,።፣፤፥

Step3.
Map tokens in the table handling table

Step4.
Map tokens in the column handling table

Step5.
Map tokens in conditional word handling table

Step6.
Save index number of column and table name

After all these common steps, in this finding, the first task is identifying the index number of the word that expressed the table name, column name and conditions from the given input. Based on this index number we have developed 20 rules to convert a natural language query (Amharic) into SQL. The developed rules are categorized the NL input into three parts: table name, the column name that have the index number less than the index number of table name, and the column name that have the index number greater than the index number of table name. From this category we have concluded that the column name having less index number than index number of table name is column name for condition, and column name having greater index number is column name for a selection. Moreover, researchers analyzed conditional query by separating single conditional query and multiple conditional query.

Example1: "[የሰራተኞችን] [ስም] [ዝርዝር] [አውጣልኝ]" and this structured looks "[Employee's] [Name] [List] [display]" respectively. From this structure "ሰራተኞች/Employee" is a Table Name, "ስም/ name" is a Column Name and the remaining words are no contribution on generating the query on this system. On this structure the column name ስም/ Name is found next to table name የሰራተኞችን/ Employee's. With this structure our algorithm conclude that the column name is column name for selection.

Example2: "[ጾታቸው] [ወንድ] [የሆኑ] [ሰራተኞችን] [ስም] [እና] [የመታወቂያ ቁጥራቸውን] [አውጣልኝ]". This structure looks "[their sex] [male] [been] [employees] [name] [and] [identification numbers] [display]". From this sentence ሰራተኞች/employee is a table name and the other ጾታ/sex, ስም/name, and የመታወቂያ_ቁጥር/idnumber are column name. Indeed, ጾታ/sex is a column name used for a condition to handle the retrieved results. The index number of the column name "SEX" is less than index number of the table name, and the remaining column name index number is greater than index number of table name (employee). In general, we have developed 20 rules for conversion, and from those let as show some of the rules.

Rule #1.
If the input sentence tokens map with (contains) only table name, the query is selection of the whole table.
*→"SELECT * FROM TABLE_NAME;"*

Rule#2
If the input sentence contains both table name and column names, and if the column name positioned next to table name, the column name is used for selection.
→SELECT (COLUMN_NAME), [COLUMN_NAME] FROM TABLE_NAME;

Rule#3.
If the sentence contains both table name and column names, and if the column name is positioned before table name, the column name is used for the condition.
*→SELECT * FROM TABLE_NAME WHERE COLUMN_NAME = COLUMN_VALUE;*

The algorithm to handle the rule is presented below.

Algorithm to Handle Rules
A natural language input NL contains X tokens (K_1, K_2, …, K_X) and this tokens stored on Array A. In our database table T there are N columns (C_1, C_2, …, C_N) with M rows (R_1, R_2, …, R_M). A NL contains column name C, table name T, a column value V, and conditional word W. Column name has further divided into column name for condition Cc and column name for selection Cs.

The Array separated into three groups A_1, A_2, and A_3. Array A_1 holds from the beginning of the array called A[0] to the array value that holds table name minus one A[T-1], A_2=A[T], and A3 contains from A[T+1] to the last. The column names found on A_1 are column name for condition (Where) and it contains attribute, sign, and value (column name, condition, and column value) i.e. WHERE Cc_1 W_1 V_1 AND/OR Cc_2 W_2 V_2 AND/OR … AND/OR Cc_N W_N V_N.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 05, May - 2017

Array A_2 contains table name T specified next to FROM keyword like FROM T. And finally, A3 contains column names used for selection positioned next to SELECT keyword, like SELECT Cs_1, Cs_2, …, Cs_N. A_3 also contains aggregate functions like SUM, AVG, MAX, MIN, and COUNT queries. Additional quires like group by and ordered by is included called Po. So the final output could be the combination of A_1, A_2, A_3 and Po and it looks SELECT Cs_1, Cs_2, …, Cs_N FROM T WHERE Cc_1 W_1 V_1 AND/OR Cc_2 W_2 V_2 AND/OR … AND/OR Cc_N W_N V_N Po;.

To do so, the first task is identifying the index number (A2) that hold table name.

Algorithm to handle A2

```
Table_location = -1;
User Input = Input;
Put the input on ArrayList = input_list;
For (intial= 0; intial <= input_list.size();intial ++) {
    IF (input_list.get(intial) contents equals
with(Token_word)) {
        Table_location = initial;
        Break;
    }
}
```

This algorithm acquires the array location of the table name, and based on this location we formulate column name for condition and column name for selection.

Algorithm to handle A1

```
FOR (initial = 0; initial < Table_Location; initial ++) {
    IF (Array value of initial = column name) {
        IF (input_list.get (initial + 1)!=TableName OR
ColumnName OR Keywords) {
            The column name = input_list.get (initial);
            The column value = input_list.get (initial +1
→TableName OR ColumnName OR Keyword exists); /*
initial value increased by one until table name or column
name or keyword exists*/
        } ELSE IF (input_list.get (initial + 1) =TableName
OR ColumnName OR Keywords) {
        The column name = input_list.get (initial);
        The column value= input_list.get (initial – 1 →
TableName OR ColumnName OR Keyword exists OR
initial = -1);
        } ELSE {
        //
        }
    }
}
```

This algorithm handle both single and multiple conditional query.

Example3: give me all information of the employee whose name is Abebe Abate Dessi and sex is male. [ስሙ] [አበበ] [አባቱ] [ደሴ] [የሚባል] [እና] [ጾታው] [ወንድ] [የሆነ] [ሰራተኛን] [ሙሉ] [መረጃውን] [አውጣልኝ]. This structure looks: - [name] [Abebe] [Abate] [Dessi] [called] [and] [sex] [male] [been] [employee] [all] [information] [give me].

For this particular example the column name 'Name' is found at the beginning of the index and next three indexes are the column value of the first index. Based on the algorithm, once we identify the index number of column name, the column value found either the right side or the left side of the column name. So, for this particular example the index number of column value is index number of column name plus one and index number has increase and append the value on the column value until the index value is a keyword.

Algorithm to handle A3

In the above algorithm, the column name found before the table name is a column name for condition. However, the column name fond next to the table name is a column name for selection.

```
FOR (initial = Table_Location; initial <= input_list.size();
initial ++) {
    IF (Array value of initial = column name) {
        //The column used for selection
        The column name= input_list.get (initial);
    }
}
```

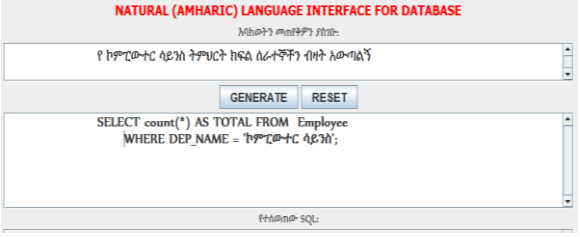
This algorithm check the column name found after the table name has exist.

Aggregate function

Aggregate function in Amharic language recognized as ከፍተኛ, ዝቅተኛ, አማካኝ, ድምር, ብዛት which means MAXIMUM, MINIMUM, AVERAGE, SUM, and COUNT respectively. In other case ከፍተኛ/MAXIMUM can be expressed as ትልቅ or የተሻለ, ዝቅተኛ/MINIMUM expressed as ትንሽ or አነስተኛ, and ብዛት/COUNT also can expressed as ቁጥር.

For example: የ ኮምፒውተር ሳይንስ ትምህርት ክፍል ሰራተኞችን **ብዛት** አውጣልኝ። which means that "display the number of employees found in computer since department. According to A1, from the given example, the column name ትምህርት ክፍል/department is used as conditional columns. The word **ብዛት**/number recognized as an aggregate word called COUNT. This particular aggregation word mainly found immediately next to table name. So, the query is converted like

Figure 1: Sample output for aggregate function



SELECT COUNT (*) FROM Employee WHERE Department = "ኮምፒውተር ሳይንስ";

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 05, May - 2017

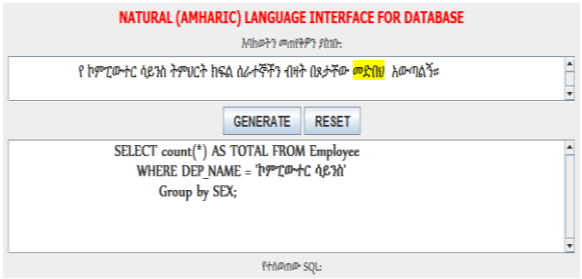
Groping and ordering Queries

In a natural language (Amharic) sentences grouping and ordering queries are recognized by ቅደም ተከተል Ordered by and መደብ/መድበህ Group by tokens.

For example: የ ኮምፒውተር ስይንስ ትምህርት ክፍል ሰራተኞችን ብዛት በጾታቸው መድበህ አውጣልኝ። (display the number of employees found in computer science department grouping by sex). From this particular example the token መደብ recognized as grouping queries and the column name used for grouping is found immediately before the token መደብ. So this query converted like

SELECT COUNT (*) FROM Employee WHERE Department = "ኮምፒውተር ስይንስ"
Group by Sex;

Figure 2: sample output for group by query



III. EXPERIMENT AND RESULT

The experiment are conducted by taking HRM database as a case study. We have collected sample queries from the novice users who have no an expert knowledge of database to evaluate the developed prototype. This prototype designed in such a way that to handle complexity of Amharic language natures from different dimension.

- Language character recognition: - the user can type the query without too much care of ambiguous and multiple form characters such as "ሐ, ሀ, ኀ, ጸ, ፀ, አ, ዐ, ው, ዉ, ሰ, ሠ …" since, the system can understand such characters in various alternatives.
- Relaxation of grammar: - The system can also understand queries of the user constructed in the form of various expressions or structured in various grammatical forms. For instance, the system can recognize the following two sentence forms of single query as "ከ ፬፭፻ ብር በላይ ደመወዝ ያላቸዉ or ደመወዛቸዉ ከ ፬፭፻ ብር በላይ የሆነ".
- Language Understanding: - during the request the user expected to include the column name on the query. For instance in a certain query 'ሴት' express the sex is female but for our system would not understand and the user should reframe the query as "ጾታቸው ሴት የሆኑ".
- Simplification: - The user can type the query without a need for complexity of expression and without the strong knowledge of SQL.

The accuracy of the system is measured in term of precision percentage with two classes that identifies query response as: Correct Queries and Incorrect Queries. To do so we have calculated the overall accuracy of correct query through a division of the total number of correct queries as generated by the system, by the total number of imputed query.

Overall accuracy of Correct Query =

$$\frac{Total\ number\ of\ correct\ query}{Total\ number\ of\ inputed\ query}$$

Henceforth, the results as shown on table 2 revealed that the system we have developed has an overall accuracy of 91% which implies that the systems validity and reliability is very high, an indicator of its strong and success full feature use and operation.

Table 2: Overall accuracy of the system.

Total query		Correct queries	Incorrect queries	Accuracy
Selection	**30**	**30**	**0**	**100%**
Single condition	**30**	**28**	**2**	**93.33%**
Multiple condition	**30**	**25**	**5**	**83.33%**
Aggregation, grouping & ordering	**30**	**26**	**4**	**86.67%**
Total	**120**	**109**	**11**	**91%**

IV. CONCLUSION

Based on its aim and objective this study has tried to develop an Amharic language interface to database system. The system targeted users who have no knowledge or skill of a database system, and who have no a good command of English language. Based on this to intervene into and solve difficulties of such database users, we have designed and developed a user interface of a database system which enable users to execute their query in a natural language, Amharic in our case, to generate and retrieve data as per their queries from the database.

To evaluate the system we have calculated the systems' performance, using academic employee database, and have analyzed their validity and reliability of NLIDB system we have created. The accuracy of the system is measured by precession in two values called correct and incorrect, and the system registers 91% of overall accuracy. Even though, the system shows a promising result, it doesn't include temporal query. Further researches will be needed to include temporal queries in ANLIDB.

V. REFERENCE

[1] A. Kaur, "PUNJABI LANGUAGE INTERFACE TO Database," THAPAR UNIVERSITY, 2010.
[2] B. Manaris, "Natural Language Processing : A Human – Computer Interaction Perspective," vol. 47. Academic Press, New York, pp. 1–55, 1998.
[3] Y. A. Alelgn Tefera, "Automatic Construction of Amharic Semantic Networks From Unstructured Text Using Amharic WordNet," 2010.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 05, May - 2017

[4] A. K. T. TEGEGNIE, "HIERARCHICAL AMHARIC NEWS TEXT CLASSIFICATION," Addis Ababa UNIVERSITY, 2010.
[5] A. Kumar and K. S. Vaisla, "Natural Language Interface to Databases: Development Techniques," *Elixir Comp. Sci. Engg*, no. November, 2015.
[6] Y. Chandra, "Natural Language Interfaces to Databases," UNIVERSITY OF NORTH TEXAS, 2006.
[7] J. Patel and J. Dave, "A Survey: Natural Language Interface to Databases," *Int. J. Adv. Eng. Res. Dev.*, 2015.
[8] A. R. Sontakke and P. A. Pimpalkar, "A Review Paper on Hindi Language Graphical User Interface to Relational Database using NLP," *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 3, no. 10, pp. 3393–3397, 2014.
[9] N. Nihalani, S. Silakari, and M. Motwani, "Natural language Interface for Database: A Brief review," *IJCSI Int. J. Comput. Sci. Issues*, vol. 8, no. 2, pp. 600–608, 2011.