INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Design and Develop Amharic Language Interface to Database

Smegnew Asemie[1]*, **Kassahun Azezew**[1], **Birhanu Gardie**[1]

1 School of Computing and informatics, Mizan-Tepi University, Ethiopia

OPEN ACCESS

Received: 15.07.2021

Accepted: 04.11.2021

Published: 22.12.2021

Citation: Asemie S, Azezew K, Gardie B (2021) Design and Develop Amharic Language Interface to Database. Indian Journal of Science and Technology 14(44): 3254-3263. https://doi.org/10.17485/IJST/v14i44.1178

* **Corresponding author**.

simegnewasemie@gmail.com

Funding: None

Competing Interests: None

Copyright: © 2021 Asemie et al. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Background/Objectives: Information has become part of our existence and to access the information from database we need to be skilled with database query languages such as SQL. Hence in this study we propose Amharic Language Interface to Database (ALIDB). Here, the request is simple like asking a human to do so in a local language (Amharic). **Method**: So far, different techniques such as pattern matching, syntax based, semantic grammar based and Intermediate Representation Language systems have been used to develop NLIDB. Among these techniques, the study employed pattern matching and similarity checking for developing Amharic language text retrieval from the Database. **Findings**: The result of the experiment shows 91% accuracy. However, the scheme has no impact on Amharic temporal queries. Further development will be done on the algorithm that includes temporal queries in ALIDB. **Novelty:** Finally we identified 20 rules and thereby contributed a new pattern / algorithm for this language that converts Amharic sentence into a Structured Query Language (SQL) and fetch results from the Database.

Keywords: Amharic Language; Database interface; DBMS; NLIDB; NLP

1 Introduction

From the past decade, information has been an important character in our lives; mostly when it comes to decision-making, many people and organizations depend on it. In recent times, with the evolution of technologies, information can be accessed almost everywhere, at any time, by anybody, including those who do not necessarily have computer backgrounds[1]. One of the key foundations 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[2,3].

NLIDB deals with representation of user request to database in his/her native language[4,5]. NLIDB then maps the user request in standard SQL to retrieve desired results from the target database[6,7]. The purpose of this interface/system is to facilitate access by the user through hiding complexities of database query language syntax[8,9].

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[3]. Hence, the demand to have a user interface comes in that would facilitate diverse users to access data. The necessity to design and develop an interface in the native language will help the user without the knowledge of English language and SQL can easily use the system. NLIDB approaches the subject of making database applications accessible in natural languages as a solution that needs to deal with two components – linguistic component and database component[2],[10–13]. An expression of Intermediate Query Language (IQR) is produced when a component called The Linguistic Component decipher the natural language input. Then, the IQR produced by this component is passed to another component named Database Component so that finally a SQL statement is generated.

Amharic is an official working language of the Federal Democratic Republic of Ethiopia and it is the second most widely spoken Semitic language in the world next to Arabic[14]. Amharic is spoken by more than 100 million people in the country and it is also widely spoken in different countries like Eritrea, USA, Israel, Somalia and Djibouti[15].Currently, the language's script contains 34 base characters (called fidels) each of which falls in the basic form and six other forms known as orders. The total seven orders represent syllable combinations consisting of a consonant followed by a vowel. The 34 basic characters and their orders give 238 distinct symbols. In addition, there are forty other, which contain a special purpose usually representing labialization. The Amharic language differs from other Semitic languages such as Arabic and Hebrew in the fact that it is written starting at the left-hand side and proceeds to the right[16–18]. In Amharic, there is no Capital-Lower case distinction. The language has its punctuation marks and number system[17].

So, to make database applications easy to use for the users, we have proposed a model and an algorithm to convert Amharic sentence into SQL and eventually retrieve relevant text from the Relational Database. It can be used in HRM database with Employee, Department, and Employee on education table as a case study for developing a natural language query processing using a database system. The query can be input in Amharic sentence for retrieving relevant data from the database and display the results.

2 Related works

Himani Jain[7] developed a Hindi Language Interface to Database. Hindi Shallow Parser uses Shakti Standard Format for parsing a sentence. The system was developed in Java with MySQL as backend. For testing of the system, employee database is used containing Employee and Department tables. The system does not include linguistic components. It straight away connects user's keyword to database entity name and the result is displayed in Hindi language. It covers single and multiple column retrieval queries, conditional and joins queries.

According to Darshil Shah and D Vanusha[19], developing a natural language interface for the relational databases (RDBMS), in natural languages in the form of english sentences as inputs from the user and generate results in the form of SQL(Structured Query Language) queries, the interface makes use of Recurrent Neural Networks to translate the natural langulage input to the Structured Query language. The database is executed in the JSON format to reduce the query processing time when the database is huge.

Avinash Agarwal[20] defines a method for semantic analysis of natural language queries for Natural Language Interface to Database (NLIDB) using domain ontology. For the experimentation of this, he proposed domain ontology for railway inquiry. The system is verified on a corpus of English language statements which is collected from various categories of users for the railway inquiry. Natural language Toolkit using python is implemented for pre-processing. The author also discussed various types of questions and answers.

Rachana et al.[12] developed a system for translating Hindi sentence in an audio clip form to an equivalent SQL query. The proposed technique makes use of python language with MySql as a database, NLTK (Natural Language Toolkit) library, Speech recognition library, CORE NLP library and PY audio library. The researchers use POS tagging for better accuracy.

Khaleel Al-Rababah and Safwan Shatnawi[21] propose an Arabic Natural Language Interface to Databases (ANLIDB) by applying Arabic morphological, ontological, and syntactical analyses. They implemented algorithms for mining significant single and multiple phrases from Arabic natural language inputs submitted to the database and then creating and executing SQL queries. In addition a lexicon from the database was created, and a simple part-of speech (PoS) was applied. They state that their system has high success rate in identifying relations, mapping of attributes, constructing and executing SQL statements.

Khaled Nasser ElSayed[21] proposed a system that translates natural Arabic language requests such as queries or imperative sentences into SQL commands to retrieve responses from the Quran DB. It performs parsing and minute morphological processes according to a sub set of Arabic context-free grammar rules to work as an interface between users and a Database.

Siyuan et al.[22] present a new system called neuron that facilitates natural language interaction with qeps to enhance its understanding. Neuron takes a sql query (which may include joins, aggregation, nesting and many other things) as input, executes it, and generates a simplified natural language-based description (both in text and voice form) based on the execution strategy deployed by the underlying RDBMS. Furthermore, it helps in understanding various features related to the qep through a natural language-based interactive framework.

Amardeep Kaur [2] presented the design and implementation of natural language interface to agricultural database in Punjabi language. The system uses MS Access as database. The system accepts input in specified pattern i.e. Table name, column name and condition query mapped manually. The author considers the limited words.

H. V. Jagdish et al. [23] developed NALIX system- a generative, interactive Natural Language Query Interface to an XML database. The system accepts an English language sentence as input, which can include aggregation, nesting, and value joins and many other things. The system can be classified as syntax based system. The conversion process has three steps: (a) generating parse tree, (b) validating parse tree, and (c) translating parse tree into an Xquery expression. It reformulates the input query to XQuery expression and translates it by a method of mapping grammatical proximity of natural language, and parses tokens to the nearest corresponding elements in the resulted XML. The system makes slight attempt to understand natural language.

Looking at the restrictions of the various NLIDB systems developed by various researchers in this ground, we have designed and developed a Natural Language Query Interface for Amharic Text Retrieval from Relational Database. So our system has improved the results that retrieve from the database by developing a new algorithm to efficiently map, based on the structure of Amharic language. The system uses human resource database with appropriate tables stored with data in Amharic natural language. Hence, the user formulates the query in Amharic sentence and the system can analyse such user's query and convert into Structured Query Language (SQL).

3 Proposed Work

3.1 System /Structure Architecture

Amharic languages interface for database accepts Amharic sentence as an input and generates SQL query. The generated queries then execute on the actual database and retrieve results and display to the user. The given input has been analyzed semantically based on the domain-dependent dictionary.

3.1.1 Query Pre-Processing

The pre-processing of the query includes: analyze the token, stop word removal, spelling checking, normalization, and stemming. The user token after going through pre-processing step gets converted into a base word which could be the table name, column name, condition word, or keyword.

3.1.2 Mapping of the User Query

The analysis of the language structure was necessary if we are to develop a properly functioning interface for Amharic speakers. The contents of the query are then categorized as the column name for state and column name for assortment. To this end, we have analyzed the key numeral of the position of the word in a known natural language query without any added language module like POS, parser, etc.

A dictionary is prepared to enable mapping NL (natural language) token into column name and table name. This lexicon encloses Amharic token vocabulary that communicates column name, table name, and provisional words with the corresponding significance of map words. Table: I denote example tokens word and suitable map words. Pre-processing of the input query includes: analyze the token, stop word removal, spelling checking, normalization, and stemming. The user token after going through pre-processing step gets converted into a base word which is whether it is the column name, table name, condition word, keyword etc.

Table 1. Table Handling Table

Token Word	**Mapped Word**
ሰራተኛ/መምህር	Employee
ለትምህርትየሄዱሰርተኞች	Employees_On_Study_Leave

The algorithm given below shows the sequence of instructions followed to identify column name, table name, or provisional words from the given Amharic Sentence.

Step1. Receive the query entered in the form of Amharic letters.

Step2.Tokenization the statement using white space and punctuation –Amharic such as? ። ፣ ፤ ፦

Table 2. Conditional Word Handling Table

Token Word	Mapped Word
ያነሰ//በታች	<
የበለጠ//በላይ	>
እናከዚያበታች//እናከዚያያነሰ//ያልበለጠ//እናከዚያያልበለጠ	<=
እናከዚያበላይ//እናከዚያየበለጠ//ያላነሰ//እናከዚያያላነሰ	>=
ያልሆነ|| ያልሆኑ	!=

Step3. Maping token on table handel table.
Step4. Maping token on the column value handel table.
Step5. Maping token on provisional word handel table.
Step6. Keep the index number of columns and table given name.

3.1.3 Sql Query Generation

For developing an algorithm for generating SQL query the natural language structure has been analyzed. Consequently for analyzing language requests and sentence types structured in different ways have been analyzed. We have categorized the user query into three parts called to query for choice of the entire table, query for choice of the convinced column from a given table, and query for choice of a specific row from a certain column or query using the where condition; the remaining database concept called aggregate function, grouping and ordering query discussed separately. Once we identify which one is the column name and which one is the table name the first and the second category is straightforward. That means there is no attribute and value relation in the sentence. The third category needs additional investigation to classify which column is for selection and which columns are for the condition from the given column identified on the sentence. We would analyze the syntactic structure of Amharic sentences for querying a sentence to retrieve a text from a database. The sentence or the question contains the name of a table, attribute, or value.

We have come up with 20 rules which are used in converting the query that was given in the Amharic natural language into SQL and the rules are derived from the index numbers resulted from the previous operation. The natural language input is divided into the following three parts based on the rules we have just laid out: table value, the column value with a key numeral fewer than the key numeral of table value, and column value with an index number superior to the key numeral of the table value. Having categorized the input query, it is determined that the column value that has a smaller amount of key numeral than the key numeral of table value is column name for the condition, and column name that has superior key numeral is column value for a selection. In this research conditional query has been analyzed by dividing only provisional query and many provisional queries. Some examples are given below to illustrate the process.

Example 1: Display the name of the employees who are hired in 2005. Querying this sentence in Amharic is ”**[በ፪፻፭] [ዓ/ም] [የተቀጠሩ] [ሰራተኞችን] [ስም] [አውጣ]**”. This structured looks “[in 2005] [year] [hired] [employee’s] [name] [display]”. From the given statement **ሰራተኞች**/ employee is a table value & **የተቀጠሩ**/hired, and **ስም**/names are column names. The token **የተቀጠሩ**/hired is a column name used for a state to grip the outcome to be displayed. From the entire given sentence, the column name has come before the table name and after the table name, and we have identification rules. From the given sentence a token recognized as a column name before a table name is considered as a column name used for a condition, and a token recognized as a column name after a table name is considered as a column name used for a selection statement. However, there is an exception on how to or where to extract the column from in the sentence. The next example illustrates this point.

Example 2: **[የአካዉንቲንግ] [ትምህርትክፍል] [ሰራተኞችን] [ደመወዛቸዉ] [ከ፲፲፲፻] [በላይ] [የሆኑትን] [ስም] [እና] [ጾታቸዉን] [አዉጣልኝ] ።** this means that select the name and sex of the employees who have got more than 10000 birrs and worked on the accounting department. And the structure looks ”[accounting] [department] [employees] [there salary] [10000] [more than] [have been] [name] [and] [there sex] [display]. This structure indicates that the column name **ደመወዛቸዉ**/salary is presented after a table name called **ሰራተኞችን**/ employees. So to handle this and such exception, we have checked the word after a table name called **የሆኑትን**/have_been. So if the word **የሆኑትን** or **የሆነዉን** is presented after the table value, column value after a table value and before **የሆኑትን** is considered as a column name used for a condition.

Finally, we have concluded that:-
IndexNumberOf(ColumnNameforCondition) < (less than)
IndexNumberOf(TableName) <(less than)
IndexNumberOf(ColumnNameforSelection).

As it was mentioned earlier, we have come up with 20 rules of communication, and some of them are listed below as a sample.

RULE #1: If the sentence doesn't contain the table name, the sentence is invalid for translation.

RULE #2: If the sentences include both the table value and column value, and if the column value is located subsequently to the table value and a token "**የሆኑትን**" is found after a column value in the sentence, the column feature value is a column name for conditions.

RULE #3: If the sentence contains both the table feature value and column feature value, and if the column feature value is located both before and after the table name, the column name placed before the table value is column name for condition (Column_NAMEc) and the column name placed after table value is column feature value for choice (Column_NAMEc).

Select (Column names), (COLUMN_NAMEs)
From table name
where Column_namec = Column_namec_value;

- **Identify the Table Name with Algorithm**

```
Loc = -1;
    Users Input = Input value;
    Set User_ input on Array = list Of Input;
    For (index= 0; index <= list Of Input.size();index ++) {
    IF (list Of Input(index) value equals with(Token Word)) {
      Location = index;
    break;
    }
    }
```

The algorithm learns the location, in the array, of the table value, and bases on these locations it identifies column value for state and column value for selection.

- **Identify Conditional Columns with Algorithm**

```
For (index = 0; index <TableLocation; index ++) {
    If (index value = columnName) {
    If (listOfInput.get (index + 1)!=TableName OR ColumnName OR Keywords) {
    ColumnName = listOfInput.get (index);
    ColumnValue = listOfInput.get (index +1 →TableName OR ColumnName OR Keyword exists); /* index assessment increment by table value or keyword found or column value */
    } If Else (listOfInput.GET (index + 1) = ColumnName OR TableName OR Keywords) {
    ColumnName = listOfInput.get (index);
    ColumnValue= listOfInput.get (index – 1 →tablename or Keyword found OR ColumnName OR index = -1);
    } Else { Where condition is not found on the query
    }
    }
    }
```

The algorithm steps handle both solo and numerous provisional query.

Example3: provide the Name, Sex, Educational level, and Salary data of the worker whose Educational level is a first degree & gender is MALE. **የትምህርት ደረጃቸዉ የመጀመሪያ ዲግሪ የሆነ እና ጾታቸው ወንድ የሆኑ ሰራተኞችን ስም፣ ጾታ፣ የትምህርት ደረጃ እና ደመወዝ አዉጣልኝ።**

From this particular example, a system identifies **ሰራተኞችን**/employee as table name and ""**የትምህርትደረጃቸዉ**/level of education, **ጾታቸው**/sex, **ስም**/name, and **ደመወዝ**/salary is a column name. The column name level and sex are found both before the table name and after the table name. So, according to RULE#3, the column value is set up before the table value used for a state, and the remaining is used for a selection. Finally, the query converted into

"SELECT NAME, SEX, LEVEL, SALARY
FROM Employee WHERE LEVEL = "**የመጀመሪያዲግሪ**" AND SEX='**ወንድ**'

- **Algorithm to Identify Columns for Selection**

The algorithm works by identifying the column value set up prior to the table feature value as a column feature value for clause and the column value found after the table value as a column name for choice.

```
for (index = Locations; index <= listOfInput.size(); index ++) {
IF (IndexValue = columnName) {
ColumnName= listOfInput.get (index);
}
}
```

This snippet of an algorithm checks whether the column feature value is set up subsequent to the table value have survived.

- **More Condition and Aggregate Function on Sql Concepts**

Ordered By: - This query is used to display the results in ascending or descending orders. In a natural language (Amharic), ordered by is presented on a sentence like **ቅደምተከተል**. This word found on the sentence after a table name is placed i.e.:

0 <= IndexNumberOf(TableName) <
IndexNumberOf (**ቅደምተከተል**).

The column name used for ordering is found on:

IndexNumberOf(**ቅደምተከተል**) – 1(IndexValue).

Based on this word we formulate the rule to handle the ordered queries. For example, **ደመወዛቸዉከ፳፬፩በታችየሆኑሰራተኛንበስ ምቅደምተከተላቸዉአውጣልኝ**; display all employees whose salary less than 2500 ordered by names. In this sentence array value of TableName is 5 and the array value of **ቅደምተከተል** is 7. So 0 <= 4 < 6. Based on this rule the query has been converted. The above natural (Amharic) language query converted into:

SELECT * FROM EMPLOYEE WHERE Salary <፳፬፩ ORDERED BY Name;

Group By: - This database query has been used to display the results in a group. To recognize the Group By query from the given input, we have checked the word **መደብ/መድበህ** from the given sentence. This keyword comes after a table name and before a table name according to the user request. Then based on the keyword **መደብ** we formulate the group by queries. For example, **ደመወዛቸዉ ከ፫፳፶ በታች የሆኑ ሰራተኛን በትምህርት ክፍላቸዉ መድበህ አውጣልኝ**; this means that display all employees whose salary is less than 3550 group by their departments. Then this natural language query has been converted into:

SELECT * FROM EMPLOYEE
WHERE Salary <፫፳፶
GROUP BY Department;

Count (): - This query is used for counting the result which fulfills the queries or the condition. To recognize this SQL function we find the word **ብዛት**/ Total from the given sentence. This keyword is found after the table name on the sentence.

SUM (): - This database query is used to add the column values based on the specification. To formulate the sum () query, we identified the word **ድምር**/Sum from the given sentence. This keyword is found after the table name and follows the Ordered by rules.

MAX (), MIN (), AVG (): - This database query is used to select the maximum, minimum, and average of the value approximately. To formulate the query, we have identified the word **ከፍተኛ** for maximum, **ዝቅተኛ** for minimum, and **አማካኝ** for average from the given Amharic sentences.

- **Algorithm to Handle Aggregate Function**

```
For (index = 0; index <= listOfInput.size(); index ++) {
    If (index value = ColumnName) {
    If (listOfInput.get(index + 1) contains ("ድምር")) {
    Query = SUM ();
    The column to be add= listOfInput.get (index);
    index += 1;
    } Else If (listOfInput.get (index – 1) contains ("አማካኝ")) {
    Query = AVG ();
    The column to be calculated = listOfInput.get (index);
    }Ese If (listOfInput.get (index – 1) contains ("ከፍተኛ")) {
    Query = MAX ();
    The column to be compared = listOfInput.get (index);
    }Else If (listOfInput.get (index – 1) contains ("ዝቅተኛ")) {
```

```
    Query = MIN ();
    The column to be compared = listOfInput.get (index);
    }Else {
    The query does not contain an aggregate function
    }
    }
```

These are the types of queries that are processed when the user wants to perform some calculations on the retrieved results or to compare the results.

Let us say, for instance, a user requires to find the total number and the average salary of all the employees who work in the mathematics department and whose name starts with the letter "**አ**". This can be paraphrased in Amharic as "**ስማቸው በአ የሚጀምሩ የሂሳብ ትምሕርት ክፍል ሠራተኞችን ብዛት እና አማካኝ ደመወዝ አውጣልኝ።**". According to RULE#18 if the sentence contains the keyword "**የሚጀምሩ/የሚጨርሱ**", the query includes LIKE in the where condition. In the same fashion, on RULE#10 and RULE#11, if the sentence contains the word "**ብዛት**", the query includes the COUNT () function and if the keyword is "**አማካኝ**" the query includes AVG () function. Therefore, according to RULE#18, RULE#10, and RULE11 the query converted into "SELECT count (*) AS TOTAL, AVG (SALARY) AS AVGSALARY FROM department INNER JOIN employee ON employee.DEP_ID = department.DEP_ID WHERE NAME Like "**አ**%" AND DEP_NAME ='**ሂሳብ**';"

- **Algorithm to Handle Like, Between, Group By, and Ordered By Queries**

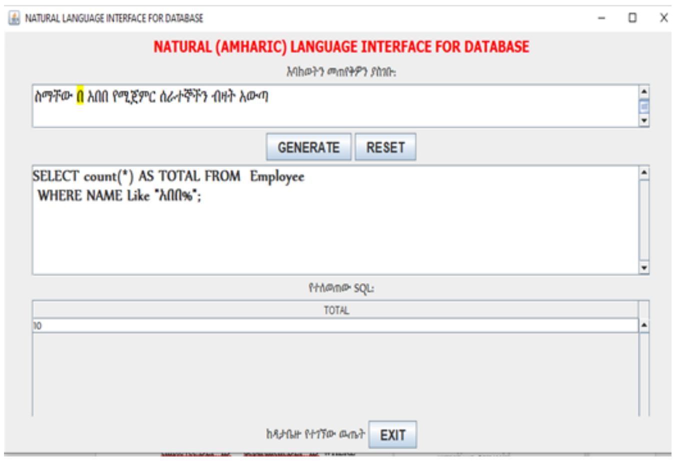
```
For (index = 0; index <= listOfInput.size(); index ++) {
    If (index value = ColumnName) {
    IF (listOfInput.get (index + 2) contains ("እስከ")) {
    The Condition Sign = "BETWEEN";
    First value= listOfInput.get (index + 1);
    Second value= listOfInput.get (index + 3);
    index += 2;
    } Else If (listOfInput.get (index + 2) contains ("የሚጀምር" OR "የሚጨርስ")) {
    The Condition Sign = "LIKE";
    The value = listOfInput.get (index + 1); index += 2;
    } Else If (listOfInput.get (index + 1) contains ("ቅደምተከተል")) {
    Query = "ORDERED BY";
    The column to be compared = input_list.get (index);
    } Else If (listOfInput.get (index + 1) contains ("መደብ")) {
    Query = GROUP BY;
    The column to be group = input_list.get (index);
    } Else {
    The query does not contain an aggregate function
    }
    }
```

For example **ስማቸውበአበበየሚጀምርሰራተኞችንብዛትአውጣ።** (display the number of employees whose name begins with Abebe). From specific example, the tokens "**ብዛት**" is recognizes as a token that indicates the queries are for counting and the column featurename uses for counting is set up immediately before the token **ብዛት**. Therefore, this queries can be transformed into SQL as follows:

```
SELECT count(*) AS TOTAL FROM Employee
WHERE NAME Like 'አበበ%%';
```

Sql Query Execution

After the user's query is processed and the SQL query is generated, the next process is executing the query into the database. For executing the query there should be a connection between the application program and the database. Based on this connection the query is sent to and executed on the database.


NATURAL LANGUAGE INTERFACE FOR DATABASE
NATURAL (AMHARIC) LANGUAGE INTERFACE FOR DATABASE
እባክዎትን መጠየቅዎን ያስገቡ:
GENERATE
RESET
የተለወጠው SQL:
ከዳታቤዙ የተገኘው ውጤት
EXIT

ስማቸው በ አበበ የሚጀምር ሰራተኞችን ብዛት አውጣ

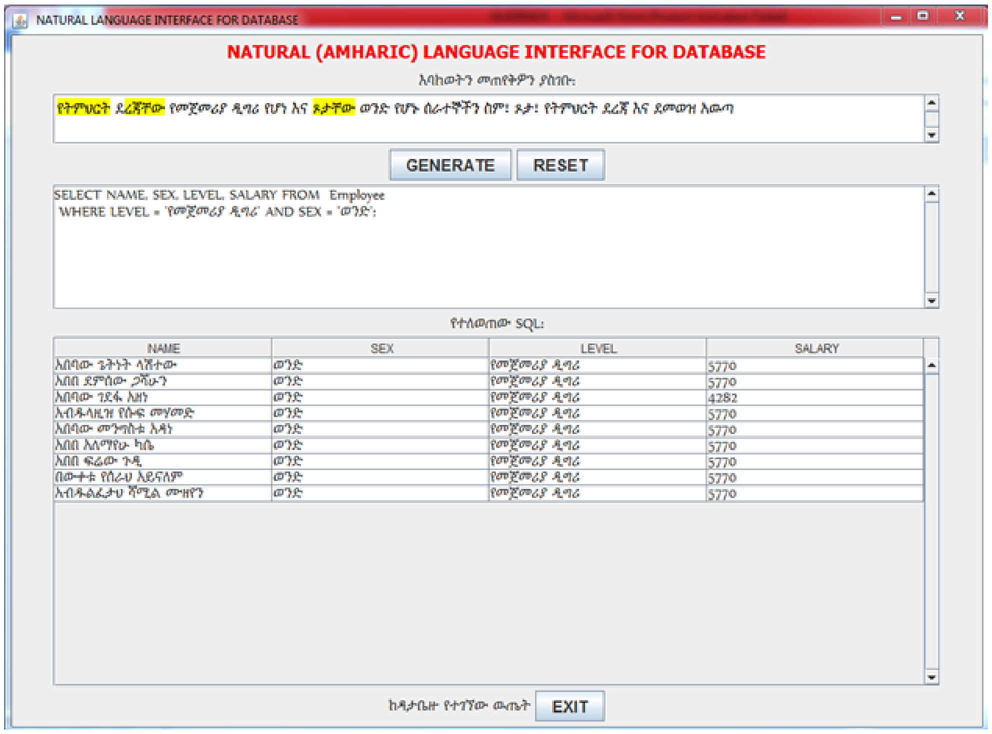
```
SELECT count(*) AS TOTAL FROM  Employee
 WHERE NAME Like "አበበ%";
```

TOTAL
10

Fig 1. Example of Query for Aggregate Function

4 Experiment and Result

The results that are used to determine the efficiency of the system are the outcome of experiments conducted by using the contents of a certain HRM database. Several queries are collected from users without any prior skills on how to perform advanced queries on a given database. We have used this collection as a sample to test the performance of the new ALIDB prototype.


NATURAL LANGUAGE INTERFACE FOR DATABASE
NATURAL (AMHARIC) LANGUAGE INTERFACE FOR DATABASE
እባክዎትን መጠየቅዎን ያስገቡ:
GENERATE
RESET
የተለወጠው SQL:
ከዳታቤዙ የተገኘው ውጤት
EXIT

የትምህርት ደረጃቸው የመጀመሪያ ዲግሪ የሆነ እና ጾታቸው ወንድ የሆኑ ሰራተኞችን ስም፣ ጾታ፣ የትምህርት ደረጃ እና ደመወዝ አውጣ

```
SELECT NAME, SEX, LEVEL, SALARY FROM  Employee
 WHERE LEVEL = 'የመጀመሪያ ዲግሪ' AND SEX = 'ወንድ';
```

NAME	SEX	LEVEL	SALARY
አበባው ጌትነት ላሽተው	ወንድ	የመጀመሪያ ዲግሪ	5770
አበበ ደምሰው ጋሻሁን	ወንድ	የመጀመሪያ ዲግሪ	5770
አበባው ገደፋ አዘነ	ወንድ	የመጀመሪያ ዲግሪ	4282
አብዱላዚዝ የሱፍ መሃመድ	ወንድ	የመጀመሪያ ዲግሪ	5770
አበባው መንግስቱ አዳነ	ወንድ	የመጀመሪያ ዲግሪ	5770
አበበ አለማየሁ ካሴ	ወንድ	የመጀመሪያ ዲግሪ	5770
አበበ ፍሬው ጉዲ	ወንድ	የመጀመሪያ ዲግሪ	5770
በውቀቱ የሰራህ አይናለም	ወንድ	የመጀመሪያ ዲግሪ	5770
አብዱልፈታህ ሻሚል ሙዘየን	ወንድ	የመጀመሪያ ዲግሪ	5770

Fig 2. Example of Query for Multiple Conditions

There are some unique aspects of the Amharic language that should be taken into consideration when evaluating the performance of this prototype. Some of the special and difficult aspects of the language the system is designed to handle and some of its features are given below.

• Recognizing letters with similar pronunciation but different forms such as "ሐ, ሀ, and ኀ" or "ጸ and ፀ", or "አ and ዐ", or "ው and ዉ", "ሰ and ሠ" the system is designed to navigate through and understand these ambiguous characters.

• Relaxation of grammar: - Although this is not a particularly unique feature of Amharic, the ALIDB is designed in such a way that it understands the different configurations of a query with the same meaning. For example, the system can recognize the following two-sentence forms of the single query as "ከ፩፮፻ብርበላይደመወዝያላቸዉ or ደመወዛቸዉከ፩፮፻ብርበላይየሆነ"

• Query submission: - for instance, if a user wants to retrieve information pertaining to a certain column, they are expected to include the column name during the request.

• Ease of use: - There is no special requirement needed to use this system. Users can just go ahead and enter their queries as if they are talking to any other person. There is no need to learn relatively complex query languages like SQL except to learn to follow the rules.

The overall accuracy of ALIDB is calculated by dividing the entire amount of accurate query by the whole quantity of inputted queries. This is done in order to measure its accuracy in terms of precision percentage with two classes that distinguish the results of a query as Correct Queries and Incorrect Queries.

$$The\ overall\ accuracy\ of\ Correct\ Query = \frac{Total\ number\ of\ correct\ query}{Total\ number\ of\ inputed\ query}$$

The results listed in table 3 depict that the overall accuracy score of the system is 91%. And this implies that the validity and reliability of the ALIDB system are indeed very high indicating strong and successful features uses and procedures.

Table 3. Total Accuracy of the System

Type of Query	Total Query	Correct Query	Accuracy
Query Selection	30	30	100%
Query Single condition	30	28	93.33%
Query Multiple conditions	30	25	83.33%
Query Aggregation, grouping & ordering	30	26	86.67%
Whole Value	120	109	91%

5 Conclusion

As is the case with any other system that is designed to be an interface between a natural language and a database, we have also set out to make it possible for the Amharic speaking part of the world to have access to the data store in the database using their language. Our target, as NLIDB system developers, was to help users with little or zero knowledge of how database query languages work to use regular Amharic sentences and to query the database for the information they need. Now they can perform the operations without the need to learn a new foreign language like English.

We were able to carry out tests to estimate the performance of the system and we have proposed by using it on a database named Academic Employee. We have seen the results produced by the ALIDB system that has been created for Amharic speakers by us. We have analyzed its validity and reliability by using some common standard metrics which are devised to evaluate the performance of such systems. The ALIDB measured using the correct and incorrect precession values scored 91% overall accuracy.

As the results are promising with some improvements, this system could be the key to remove the barriers of database access for many Ethiopians. Although the system performs the majority of database queries, it is not yet able to perform temporal queries. Future researches will have to focus on improving this aspect of the system.

References

1) Bais H, Machkour M, Koutti L. An independent-domain natural language interface for multimodel databases. *International Journal of Computational Intelligence Studies*. 2019;8(3):206. Available from: https://www.inderscience.com/info/inarticle.php?artid=102547.
2) Kaur A. PLID-Punjabi Language Interface to Database. 2010. Available from: http://tudr.thapar.edu:8080/jspui/bitstream/10266/1106/1/aman_final.pdf.
3) Asemie S, Mamo G, Kababa T. Possibility of Amharic Query Processing in Database using Natural Language Interface. *IJERT* . 2017;06. Available from: https://www.ijert.org/possibility-of-amharic-query-processing-in-database-using-natural-language-interface.
4) Kumar A, Kumar AR, Harshitha P, Mahadevaswamy, Sachin DN. Providing Natural Language Interface To Database Using Artificial Intelligence". *International Journal of Scientific & Technology Research*. 2019;8(10). Available from: https://www.ijstr.org/final-print/oct2019/Providing-Natural-Language-Interface-To-Database-Using-Artificial-Intelligence-.pdf.

5) Affolter K, Stockinger K, Bernstein A. A comparative survey of recent natural language interfaces for databases. *The VLDB Journal*. 2019;28(5):793–819. Available from: https://dx.doi.org/10.1007/s00778-019-00567-8.
6) Manaris B. Natural Language Processing: A Human-Computer Interaction Perspective. In: Advances in Computers;vol. 47. Elsevier. 1998;p. 1–55. Available from: https://www.semanticscholar.org/paper/Natural-Language-Processing%3A-A-Human-Computer-Manaris/99573a980c0af0c38f2c47c95e8475594387a1bd.
7) Hosu I, Iacob R, Brad F, Ruseti S, Rebedea T. Natural Language Interface for Databases Using a Dual-Encoder Model", in proc. ICoCL. Santa Fe, New Mexico, USA. 2018. Available from: https://www.aclweb.org/anthology/C18-1043/.
8) Yuan C, Ryan PB, Ta C, Guo Y, Li Z, Hardin J, et al. Criteria2Query: a natural language interface to clinical databases for cohort definition. *Journal of the American Medical Informatics Association*. 2019;26(4):294–305. Available from: https://dx.doi.org/10.1093/jamia/ocy178.
9) Poetra DA, Widagdo TE, Azizah F. NLIDB for Query with Temporal Aspect. *International Conference on Data and Software Engineering*. 2019. Available from: https://ieeexplore.ieee.org/document/9092618.
10) Ramesh D, Sanampudi SK. Telugu Language Interface to Databases. *International Journal of Advanced Research in Computer and Communication EngineeringVol*. 2013;2(7). Available from: https://www.ijarcce.com/upload/2013/july/72-o-Suresh%20Kumar-telugu%20language%20interface%20to%20databases.pdf.
11) Wang W, Tian Y, Wang H, Ku WS. A Natural Language Interface for Database: Achieving Transfer-learnability Using Adversarial Method for Question Understanding. *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 2020.
12) Dubey R, Kawale T, Choudhary T, Narawade V. Hindi Language Interface to Database. *ITM Web of Conferences*. 2020;32:01007. Available from: https://dx.doi.org/10.1051/itmconf/20203201007.
13) Abass Y, Zahoor S, Irfan. Common Database Interface with NLP. *International Journal of Computer Science and Mobile Computing*. 2017;6(6). Available from: https://www.ijcsmc.com/docs/papers/June2017/V6I6201733.pdf.
14) Tefera YA. ACASN-Automatic Construction of Amharic Semantic Networks from Unstructured Text Using Amharic WordNet. 2010. Available from: https://www.aclweb.org/anthology/W14-0123/.
15) Belay B, Habtegebrial T, Liwicki M, Belay G, Stricker D. A Blended Attention-CTC Network Architecture for Amharic Text-image Recognition. *Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods*. 2021;p. 978–989.
16) Teshome Y. Sentence Level Opinion Mining. Debreberhan University, DebreBerhan, Ethiopia. 2019. Available from: http://etd.dbu.edu.et/bitstream/handle/123456789/348/yodit%20teshome%40%40%40.pdf?sequence=1&isAllowed=y.
17) Gashaw I, Shashirekha HL. ML Approaches for Amharic Parts-of-speech Tagging. In: Proc. of ICON-2018. 2018;p. 69–74.
18) Tegegnie AKT. HANTC-Hierarchical Amharic News Text Classification. 2010. Available from: http://etd.aau.edu.et/handle/123456789/14540.
19) Shah D, Vanusha D. Optimizing Natural Language Interface for Relational Database. *International Journal of Engineering and Advanced Technology (IJEAT)*. 2019;08:4.
20) K DOG, Avinash JA. Semantic Analysis of Natural Language Queries Using Domain Ontology for Information Access from Database. *I J Intell Syst Appl*. 2013;p. 81–90.
21) Al-Rababah K, Shatnawi S. An Arabic Language Interface to Databases Using a Morphologically-Based Lexicon, Language Indicators and Pos Tagging. *International Journal of Multimedia and Image Processing*. 2012;2(1/2):87–95. Available from: https://dx.doi.org/10.20533/ijmip.2042.4647.2012.0011.
22) Liu S, Bhowmick SS, Zhang W, Wang S, Huang W, Joty S, et al. NEURON: Qery Optimization Meets Natural Language Processing For Augmenting Database Education. .
23) Li HVJ, Yang H. NaLIX : an Interactive Natural Language Interface for Querying XML," in in the proceedings of the SIGMOD, 2005. 2005.