JIMMA UNIVERSITY

JIMMA INSTITUTE OF TECHNOLOGY

SCHOOL OF GRADUATE STUDIES

FACULTY OF COMPUTING

DEVELOPMENT OF TIGRINYA LANGUAGE INTERFACE TO DATABASE USING RNN

BY:

Hagos Hailemaryam

A Thesis Submitted to School of Graduate Studies in Partial Fulfillment for the Degree of Master of Science in Information Technology

JANUARY, 2020

JIMMA, ETHIOPIA

JIMMA UNIVERSITY

JIMMA INSTITUTE OF TECHNOLOGY

SCHOOL OF GRADUATE STUDIES

FACULTY OF COMPUTING

DEVELOPMENT OF TIGRINYA LANGUAGE INTERFACE TO DATABASE USING RNN

BY:

Hagos Hailemaryam

ADVISOR:

1. Getachew Mamo(PhD)

CO-ADVISOR:

1. Mr. Teferi Kebebew (MSc.)

Jimma University

Jimma Institute of Technology

Faculty of Computing

INVESTIGATING THE POSSIBILITY OF DEVELOPING TIGRINYA LANGUAGE INTERFACE TO DATABASE

By: Hagos Hailemaryam

This is to certify that the thesis prepared by **Hagos Hailemaryam**, titled **Investigating the Possibility Of Developing Tigrinya Language Interface To Database,** Submitted in partial fulfillment of the requirements for the Degree of Master of Science in *Information Technology* compiles with the regulations of the University and meets the accepted standards with respect to originality and quality.

Approved by board of Examining Committee:

	Name	Signature
Dean, Faculty of computing:	Getachew Mamo(PhD)	[illegible]
Advisor:	Getachew Mamo (PhD)	[illegible]
External Examiner:	Kula Kekeba(PhD)	[illegible]
Internal Examiner:	Ephrem Tadesse	[illegible]

Jimma, Ethiopia

January 2020

DECLARATION

I declare that this thesis entitled "**Development of Tigrinya language interface to database using rnn**" has not been submitted as a requirement for the award of any type of work in Jimma University or elsewhere. The resources used while preparing this thesis are duly acknowledged.

Hagos Hailemaryam Hailenchael	[illegible]	28/05/2012.
Name	Signature	Date

ACKNOWLEDGNMENT

First and foremost I praise the Almighty God, and St. Marry for giving me health and patience in completing my thesis work.

Next, I would like to express my deep gratitude to my advisor Dr. Getachew Mamo, for his continuous support on this thesis. He motivated and recommended me to work on natural language interface to database and his guidance helped me in all the time of the research work.

In addition, I would like to express my deep gratitude to my co-advisor Mr. Teferi kebebew for his continuous support till the end of my research thesis. He was always with me in every minute to support me. He was working on showing me different directions to accomplish this research thesis.

Next, I would like to thank to my fellow Negassa Weyessa and Kebede Assefa for their support and sharing of ideas on my research work.

Next, I would like to thank my family (mom, dad, brothers and sisters) for their continuous support, motivation and patience.

Finally, I can’t call all, so I would like to express my sincere gratitude to all who had direct and indirect contributions on my research work.

Dear all, I have no words to express your immense contributions’.

DECLARATION I
LIST OF TABLES IX
LIST OF FIGURES X
ACRONYMS AND ABREVIATIONS XI
ABSTRACT XII
Chapter One 1
Introduction 1
1.1 Statement of the Problem 3
1.2 Objectives 5
1.2.1 General objective 5
1.2.1 Specific objectives 5
1.3 Scope and Limitations 5
1.4 Methodologies 6
1.4.1 Literature Reviews 6
1.4.2 Data Collections 6
1.4.3 Development Requirements 6
1.4.4 Neural machine translation approach 7
1.4.5 Evaluations 7
1.5 Research Organization 7
Chapter Two 8
Literature Review 8
2.1 Natural Language Interface to Databases 8
2.2 History of NLIDB 8
2.2.1 Lunar: 8
2.2.2 Chat-80 9
2.2.3 Team 9
2.3 Components of NLIDB 9
2.4 Approaches of NLIDB 10
2.4.1 Symbolic (Rule Based) Approach 10

2.4.2 Empirical (Corpus Based) Approach 10
2.4.3 Connectionist (neural network) Approach 10
2.5 Architectures 11
2.5.1 Pattern matching systems 11
2.5.2 Syntax-based systems 11
2.5.3 Semantic Grammar Systems 13
2.5.4 Intermediate Representation Languages 14
Chapter Three 16
Tigrinya Language 16
3.1 Introduction 16
3.2 The Tigrinya Alphabets 16
3.3 Word Classes 16
3.3.1 Noun 16
3.3.2 Adjective 17
3.3.3 Preposition 17
3.3.4 Adverb 17
3.3.5 Verb 18
3.4 Punctuation Marks 18
3.4.1 Preface Colon (፡-) 18
3.4.2 Full stop (።) 18
3.4.3 Colon (፤): 18
3.4.4 Comma (፣) 18
3.4.5 Question mark (?) 18
3.4.6 Exclamation mark (!) 19
3.5 Types of Phrases 19
3.5.1 Noun phrases 19
3.5.2 Verb phrases 19
3.5.3 Adjectival Phrases 19
3.5.4 Prepositional phrases 19
3.5.5 Adverbial phrases 19

3.6 Types of Sentences 19
3.6.1 Simple sentences 20
3.6.2 Compound sentences 20
3.6.3 Complex sentences 20
3.6.4 Compound-Complex sentences 20
CHAPTER FOUR 22
Related Works 22
4.1 Introduction 22
4.2 Information Retrieval 22
4.3.1 Text summarization 23
4.4 NLIDB for different languages 23
4.4.1 Amharic Language 23
4.4.2 Afaan Oromo Language 24
4.4.3 Arabic language 24
4.4.4 Hindi Language 25
4.4.5 Punjabi Language 25
4.4.6 English Language 25
Chapter Five 28
Designing and Developing of TLIDB 28
5.1 Introduction: 28
5.2 Types of Queries 29
5.3 Neural Network 31
5.3.1 Recurrent Neural Network 32
5.4 The General Architecture of TLIDB 33
5.4.1 Corpus: corpus 34
5.4.2 Input Query: 35
5.4.2 Tokenizer: 35
5.4.3 Cleaning: 35
5.4.4 Preparation of Dictionaries and specifying of the dimensions values: 35
5.4.5 Encoder-Decoder Model 35

5.4.5.1Word representation: 35
5.4.5.1.1 One-hot encoding 35
5.4.5.1.2 Word embedding 36
5.4.5.1.3 Word embedding layer 36
5.4.5.2 LSTM Model: 36
5.4.5.2.1 Encoder LSTM 36
5.4.5.2.2 Decoder LSTM 38
5.4.5.2.2.1 Training mode 38
5.4.5.2.2.2 Inference mode 40
5.4.6 Query executer: 41
5.3.7 Result displayer and manipulator 42
CHAPTER 6 43
Experiment 43
6.1 General pipelines 43
6.2 Database Design 43
6.3 Queries 44
6.4 Model Training 47
6.5 Evaluation Result 49
6.5.1 Loss 50
6.5.2 Accuracy 51
6.6 Discussion 52
6.7 Contribution of the study 53
CHAPTER 7 55
Conclusion and Recommendation 55
7.1 Conclusion 55
7.2 Recommendation 56
REFERENCES 57
APPENDIX: 61
Appendix A: Approval Paper one 61

Appendix B: Sample DB contents of Diseases table 62

Appendix C: Approval Paper two 63

Appendix D: Sample Dataset 64

Appendix E: Train Model 65

Appendix F: Encoder Model 66

Appendix G: Decoder Model 66

Appendix H: Summery of the model 67

LIST OF TABLES

Table 1: Summary of related works 26
Table 2: Sample table Names 29
Table 3: Samples attribute names of patients and employees 29
Table 4: keys 43
Table 5: Sample Queries 45

LIST OF FIGURES

Figure 1: Pattern matching... 11
Figure 2: Syntax based.. 12
Figure 3: Parse Tree 13
Figure 4: Semantic based.. 14
Figure 5: Semantically parsed Tree 14
Figure 6: Intermediate representation 15
Figure 7: RNN Architecture.. 32
Figure 8: General Architecture 34
Figure 9: Encoder.. 37
Figure 10: Training mode of decoder 39
Figure 11: Encoding input sequence into final state (thought vectors) 40
Figure 12: Decoder Lstm at t=1 41
Figure 13: Decoder Lstm at t=7 41
Figure 14: Persistent diagram 43
Figure 15: Output of the Prototype 46
Figure 16: Training and evaluation process... 49
Figure 17: loss model.. 50
Figure 18: Accuracy model... 51

ACRONYMS AND ABREVIATIONS

NLIDB: Natural Language Interface to Database

TLIDB: Tigrinya Language interface to database

SQL: Structured Query Language

Seq2seq: Sequence-to-sequence

LSTM: Long Short Term Memory

DB: Database

DBMS: Database Management System

ABSTRACT

Now a day, different organizations use database with local language written contents to manage their work. These databases have huge information in an electronic format, and to access and manipulate these information users expected to know the SQL. Also, using SQL to access and manipulate the information written in local language is very difficult and tedious. So, instead of knowing and using of the SQL, it is better to use natural language for users to access and manipulate the contents of the database. Because using natural language is simple and comfortable. And also, it is good to form conjunctions and negations query simply. Because of the simplicity and comfortablity of the natural language for ordinary users many researches have been carried out on natural language interfaces since 1970. Therefore, that's why Tigrinya language interface to database had been proposed.

The database contents have been accessed and manipulated using the developed Tigrinya language interface to database (TLIDB) prototype without the knowledge of SQL. To carry out this, first the input Tigrinya sentences have been translated into the corresponding SQL statements and further the SQL statements were executed in the database. The TLIDB was designed and developed using a robust and effective approach called neural machine translation. The encoder-decoder long short term memory was used for the translation of the input Tigrinya sentence to corresponding SQL statements. In the sequence to sequence problems the encoder-decoder long short-term memory is a good technique. Also, word embedding technique was used to estimate the similarity of words and to have a dense representation. This solved the sparse data problem with the traditional approaches.

The developed TLIDB prototype was evaluated on healthcare database that has patients, diseases and employees table. The record of diseases was prepared with health professionals. This prototype handles list query, conditional queries, aggregate functions, complex queries (join, union), update, delete and etc. To develop the prototype 6338 sentences were prepared with their corresponding SQL statements. The model was trained with 80% and tested with 20% of the dataset. This was done using the percentage split evaluation technique. Since, in percentage split evaluation technique the model has been evaluated with the data that were not included during the training. After the model had been evaluated, above 98.5% overall accuracy has been scored.

Keywords: TLIDB, Natural Language Interface to Database (NLIDB), Natural Language Processing

Chapter One
Introduction

Natural language processing is one of the fields of computer science and this used to study how machines can process the natural language to perform useful things. Obviously, the machines required an input and processing to perform a useful task. So, natural language processing is the studies how the machines can understand the natural language text, speech, image and etc to do its task. So, to achieve this task many NLP researchers aim to gather knowledge on how human being understand and use the natural language. So that, researchers should develop appropriate tools and techniques that can help machines to process the natural languages. Besides this study, there are also other prominent general purpose NLP applications such as: speech recognition, information retrieval, information extraction and etc. [1].

Now a day, there is huge information in an electronic format stored in the database. Retrieving and manipulating on that information are becoming more essential. Database is one of the major sources of information. To retrieve and making operations on those information, writing SQL statements is required. Since, computers can understand the SQL statements. However, everyone is not able to write SQL queries. Because, to write the SQL query, knowing the structure of the SQL query are required. Instead of writing SQL statements, using natural language to retrieve, update and delete data from database is simple. This stimulated the development of Natural Language Interface to Database Systems (NLIDB)[2].

Natural Language Interface to Database Systems is one of the natural language processing applications and this application contains two components. These are called, **linguistic** and **database** components. The **linguistic** component is used to translate the natural language queries to SQL queries and providing the natural language results retrieved from the database. The **database** component is used to perform the traditional database management functions [9, 10, 12].

In this study, a Tigrinya language interface to database (TLIDB) system was being proposed for the native Tigrinya users. This system enables users to retrieving and manipulating the information's in database using Tigrinya plaintexts. Since NLIDB does not need to learn the artificial communication (formal) languages and it required less training to users than other forms of interfaces [11, 13].

Sequence to sequence approach was used to develop the TLIDB prototype. Because this approach is robust and effective in translation problems from one-source to another-source. This means translating from one domain to another domain. Therefore, the Tigrinya natural Language interface to database is a translation problem which means a translation of the input Tigrinya sentences to their corresponding SQL statements. As a result, the neural machine translation approach was applied. Once the model was created using the neural machine translation approach, it can be loaded and used without the need to retrain again. Even though, this approach wants large datasets to train and test the model, it is very robust and effective [8].

It is clear that, words are symbols which represented objects in real world and are putting together into a sentence that obeys well specified grammar rule [9]. So, stating all words of the language with their corresponding SQL commands is very difficult. Even though, rule based approach wants small dataset to test the prototype, it is very difficult on translating the Tigrinya input sentences to SQL statements. Because, to translate that every words of Tigrinya language must be written with their corresponding SQL statements. So, this takes time and storage spaces. Obviously, no one is perfect with one language. So, this is very difficult to write every word of the Tigrinya language and it is too tedious to use rule based approach for Tigrinya language interface to database. That’s why the neural machine translation approach has been chosen for development of Tigrinya language interface to database prototype.

As Tigrinya users are with none SQL backgrounds’, they couldn’t access and manipulate the contents of the database using SQL. Even, Tigrinya users with SQL backgrounds’ got difficulty to formulate the complex queries to access the database. For example: when Tigrinya users want to formulate complex queries like conjunction and negations to access the database, they got difficulty. Since, it is difficult to formulate that for the users with SQL backgrounds’ and impossible to formulate that for the users with none SQL backgrounds’. For example as we have seen in endadenagle hospital in shire indaslasaie town, the employees got difficulty to formulate queries such as patients below 7 years or patients above 7 years or patients between 7 and 20 years and then like. Also, to formulate queries such as patients those who have only malaria or patients those who have not malaria and etc. because, they are none SQL backgrounds’. Therefore, accessing the database using Tigrinya language written form is simple and comfortable for them. That’s why we proposed the TLIDB that enables them to access the database using Tigrinya language. This can be accomplished by translating the Tigrinya sentences to corresponding SQL statements. The TLIDB is an opportunity for the native Tigrinya users. Because, they didn’t expect to learn the SQL, they can formulate queries such as conjunctions’ and negations’, simple and comfortable. The problem of TLIDB is it did not provide a message, when the input query is an incorrect.

1.1 Statement of the Problem

There are many database applications that have different language contents, have been developed so far. Obviously, database applications have their own language that enabled users to access and manipulate the contents. This language is called structured query language (SQL) and this language has its own structures. So, if users want to access and manipulate the database they should know and write the syntaxes and semantics of the SQL. Therefore, to access and manipulate the database users were expected to know or learn SQL. SQL is not simple, comfortable and flexible for users. Since, in the absence of one element it will not execute. For example if users missed table name in the SQL statement it doesn’t work. Users don’t have simple and comfortable alternatives that enabled them to access and manipulate the database. That’s why users use SQL. So, to access or manipulate the database contents users are expected to know the database tables, the relationship between the tables, the attributes used with each table and the actual value stored with in each tables. Therefore, this is very difficult for them.
So, Instead of using SQL statements, users should use their native language to access and manipulate the database contents. Because, using natural language plaintext is better than SQL. So, NLIDB enables users to access and manipulate the database using the natural language plaintexts. Natural language interface has been used for users to write using their native language. A natural language interface is considered as a main goal for a database query interface and many natural language interfaces have been developed for this goal [3, 4, 5]. Many researches were conducted for Amharic, Afaan Oromo, Hindi, English, Arabic and etc. However, to the best of the researchers’ knowledge, no work is done for Tigrinya that enables users to query the database using Tigrinya language. Since, a model that developed for other languages will not be applied to Tigrinya language and this happened because of the syntax and semantics variation of the languages. Even though, both Amharic and Tigrinya belong to the Semitic language family, they have different syntaxes and semantics. Therefore, a model that developed for Amharic language will not be applied to Tigrinya language.

Hence, Tigrinya users need Tigrinya language interface to database that enabled them to access and manipulate the database contents. For example if there is a healthcare database that contains a table called DISEASES(ሕማማት) and attributes called symptom(**ምልክት)** and type(**ዓይነት)**.

To get information about all diseases, the user expected to write the following statement.

Select * from ሕማማት;

ሓበሬታ ሕማማት ኣርእየኒ።, In English this translated as ‘show me, information of diseases’

ሓበሬታ ሕማማት ኣምፅኣለይ።;

ሓበሬታ ሕማማት ኣውጽኣለይ።,

ሕማማት ኣርእየኒ።,

To get symptoms of malaria, the user expected to write:

Select **ምልክት** from **ሕማማት** where **ዓይነት**=’ **ዓሶ**’;

ዓሶ ዓይነት ዝኮነ ሕማም ምልክት ኣርእየኒ።, In English this translated as’ show me the **symptom** of **malaria** type **disease**’

ናይ ሕማም ዓሶ ዓይነት ምልክታት ኣምፅኣለይ።

ናይ ሕማም ዓሶ ዓይነት ምልክታት ኣርእየኒ።,

ናይ ሕማም ዓሶ ዓይነት ምልክታት ኣውጽኣለይ።,

To update (replace) the name of patients who have the identifier 4 by “ሃይለ″**, the user expected to write:**

Update **ሕሙማት** set **ሽም**="**ሃይለ**" where **መለለይ="4";**

ባዓል 3 መለለይ ዝኮነ ሕሙም ሽሙ ብሃይለ ተክኣለይ። , in English this translated as replace the patients name by “ሃይለ” who have an identifier of 4.

ባዓል 3 መለለይ ዝኮነ ሕሙም ሽሙ ብሃይለ ተክእ።

Therefore, the TLIDB is good for the Tigrinya users to access and manipulate the database instead of writing SQL statements.TLIDB is easy to understand and allow users to write in Tigrinya plaintexts when compared with formulating SQL statements. And also TLIDB doesn’t require learning of the artificial languages. Again, as described above the natural language query is too flexible when we compare with the SQL. The prototype developed using neural machine translation technique and this enabled to handle the different variations of the linguistic queries. The example below shows the variation of the linguistic queries.

ሓበሬታ ሕማማት ኣርእየኒ።, (Select * from ሕማማት;)

ዓሶ ዓይነት ዝኮነ ሕማም ኣርእየኒ።,(select * from ሕማማት where ዓይነት="ዓሶ")

In this study, the following research questions have been addressed. Those are:

- ❖ How can be generated results from the database using Tigrinya statements without the knowledge of SQL?
- ❖ How can be manipulated the contents of the database with Tigrinya statements without the knowledge of SQL?
- ❖ How the performance of neural machine translation approach for translating Tigrinya input-sentences to SQL statements can be designed and evaluated?

1.2 Objectives

1.2.1 General objective

The objective of this study is to explore and develop Tigrinya language interface to database (TLIDB).

1.2.1 Specific objectives

To achieve the general objective, the following specific points have been performed:

- ❖ Different literatures were reviewed on NLIDB to understand better the NLIDB area and the approaches and techniques which are used for NLIDB
- ❖ A dataset was prepared
- ❖ TLIDB prototype was designed and developed
- ❖ Finally, the performance of the prototype was evaluated

1.3 Scope and Limitations

Here, the main emphasis that the research study focused on is described as follows:

- ❖ This research has been studied the **Select** SQL statements and the Data manipulation statements. The prototype should accept the input queries in Tigrinya language i.e. input must be Tigrinya plain texts, Tigrinya speech and non Tigrinya languages won't be considered.
- ❖ A new healthcare database has been designed. This minimizes the efforts and time of the researchers than using existing database. Because, to use the existing database requires permission from the managements of the healthcare organization and this takes time and efforts. And even, after long time process the managements could not be voluntary to provide the database.

- ❖ Ambiguity issues were not handled completely: because as the language is morphologically rich it is very difficult to completely avoid ambiguity issues.
- ❖ Tigrinya spelling checker, tagger, parser, morphology analyzer and etc were not included. Since there are no publicly available tools to integrate to the work.

1.4 Methodologies

Generally, in order to accomplish this research, the following methodologies have been used:

1.4.1 Literature Reviews: different kinds of literatures have been reviewed from different sources on natural language interface to database. This helped to get better understanding and to have a detailed knowledge on the various approaches and techniques of NLIDB. Also, this helped us to select an appropriate approach and technique to develop the Tigrinya language interface to database prototype. And again, this showed us in detail the advantage of natural language interface to database. Here, based on the popularity and time of development the history of natural language interface to database was discussed.

1.4.2 Data Collections: a new database was designed to evaluate this prototype. Because, requesting for an existing system from different organizations has many challenges. As a result, this wastes much time and efforts. Therefore, instead of requesting to an existing database from different organization, designing a new healthcare database is simple. That's why we have chosen to design a new database. And this healthcare database is created and filled an actual data. The health professionals' were requested to fill the database tables with Tigrinya texts. In Tigrinya language the actual values of the database can be noun or phrase. After, they filled actual values to the healthcare database, a Tigrinya language professionals' were also requested to prepare the corpus (Tigrinya sentences). The corpus (Tigrinya sentences) was prepared to train and evaluate the model. It is possible to see the evidence in the appendixes' **A** and **C** of this work.

1.4.3 Development Requirements:

Keras 2.2.4 deep learning python library was used to develop the prototype. This library used with Python 3.6.x and it uses tensorflow as its backend. Keras is a high level neural network API which is very useful in fast prototyping and ignores the detail implementations of back propagation (writing optimization procedures). This library is an open-source and has large community support. Also, this library supports both types of neural networks (i.e convolutional and recurrent) [42]. The prototype has been developed using python 3.6 and MySQL. Since, Python is easy to use, more powerful for natural language processing, recommended for a single developer and little number of modules. MySQL is more popular because it is free and open sources. That's why those tools have been chosen to use them.

Edraw max is one of the graphical representation tools which used to draw the graphical representations. It is very easy and simple to draw diagrams in Edraw Max. Therefore, this helps us to minimize our effort and saves our time.

1.4.4 Neural machine translation approach: This approach uses a deep neural network to translate from one sequence to another sequence and this makes it a novel approach in machine translation. To carry out an end to end translation, encoder and decoder components are required. The former was used to encode the input Tigrinya sentences by reading one word at a time and storing the hidden state into a vector representation. The decoder was used to visualize the final hidden states as a representation of the input Tigrinya sentence which summarizes all the information present in the complete Tigrinya input sentence [38].

1.4.5 Evaluations: the prototype was evaluated automatically using keras accuracy metrics, on sample healthcare database. Because, this prototype is developed using keras deep learning python library. In keras the metrics such as precision and recall are removed starting from keras 2.0 and this prototype is developed using keras 2.4. That's why we use the keras accuracy metric to evaluate automatically.

$$Accuracy = \frac{amount\ of\ correct\ predicted\ SQL}{Total\ predicted\ SQL\ amount} * 100\%$$

1.5 Research Organization

The thesis report is organized as follows: chapter two introduces an overview of NLIDB and the different approaches used in NLIDB researches. Moreover, the components of NLIDB and the different architectures' are also discussed. Chapter three deals with the word categories, types of phrases, types of sentences and punction marks of Tigrinya language. Chapter four focused on the related works of other languages to apply the NLIDB. Chapter five emphasized on the general design and architecture of TLIDB. Chapter six discusses the developed prototype and it's evaluation. Finally, Chapter seven presents the conclusion and recommendation works of this thesis.

Chapter Two
Literature Review

2.1 Natural Language Interface to Databases

NLIDB is a system that used to translate a natural language query into SQL, manipulates and displays a result. NLIDB system is used for end user to query a database using their native languages and obtained result also in the same language as they have queried [14]. This also provides comfortablity for SQL professionals. NLIDB is proposed as a solution to the problem for accessing and manipulating information in a simple way and allowing any type of users to retrieve information from a database using natural language [15]. In the past many NLIDB systems have been developed to interact with database in a more convenient and flexible way. Due to this, NLIDB is still widely used today [16]. Because the NLIDB does not need learn artificial languages such as SQL, does not need training for non-experienced, good for formulating negations and conjunctions, it is simple and easy to use[7,11,12,13,14]. Different NLIDB researches are conducted using different techniques and approaches for different languages. In this chapter some systems, approaches and techniques of the NLIDB are described as follows:

2.2 History of NLIDB

Natural Language interfaces had appeared in the late sixties and early seventies. Many of these systems relied on pattern matching to directly mapping the user input to the database queries [19]. The usages of the databases were spreading during the 1970's and then this idea presented new challenges to the designers. One approach was to use natural language processing, which allowed the users to interactively query the data stored in database. The challenges of the design of the NLIDB are the understanding of wide variety of input sequences (length of the input), word variation (word synonymy), and user input variation (it may be ungrammatical input) and etc. Different NLIDB systems were developed from time to time. Based on their development period and popularity (acceptance), some existing systems are described as follows:

2.2.1 Lunar:

This was the first NLIDB and introduced in 1971. This system used to answer questions about samples of rocks brought back from the moon. Lunar uses one database for chemical analysis

and another database for literature references. Lunar was developed using syntax-based technique.
To process and retrieve the data of this system the geologists should learn the programming skill. So, this is a waste of time and cost to teach those [21]. Lunar was used augmented transition network parser and woods' procedural semantics. Lunar handled 78% of requests without errors and this performance was quite impressive. However, it had a limitation of linguistic capabilities [20]. The augmented transition network parsers were not useful and not efficient to extract information from ungrammatical sentences. Because, they are not handled well and not flexible [12].

2.2.2 Chat-80

This was developed in 1980's and took query in English language. This was implemented using prolog. It was developed using dialog based approach and semantic grammar system. According to [24], it was "remarkable, efficient and sophisticated system". It's database consisted of facts such as oceans, major seas, major rivers and major cities of 150 countries in the world and a small set of English language vocabulary that are enough for querying the database.

2.2.3 Team

This was developed in 1987 and took the query in English language. It was developed using semantic grammar system. It was designed to be easily configurable by database administrators with no knowledge of NLIDBs [25, 26].

2.3 Components of NLIDB

NLIDB has two components. Namely, Linguistic and database components [9, 10, 11,]. We described them in detail below:

The former component that means the linguistic component is responsible for processing and translating natural language input into a formal query (SQL) and generating a natural language response based on the results from the database search.

The later component that means the database component performs traditional Database Management system functions. Users can enter a natural language question using the linguistic component and the prototype translating the natural language statement into their corresponding SQL statement. Once the entered natural language question translated into a statement of formal query (SQL) language, the query can be processed by database management system in order to produce the required result (output). And then the output passed back to the linguistic component of the NLIDB.

2.4 Approaches of NLIDB

There are three approaches which are used to develop a natural language interface to database. Several researchers applied different approaches to deal with natural language interface to database [2, 16, 9]. Here, below are discussed in detail the three approaches of NLIDB.

2.4.1 Symbolic (Rule Based) Approach

Rule based approach sometimes called 'if—then' rules and those, takes actions when a condition is satisfied. Every natural language has their own grammar rules. To develop a natural language processing system, understandings of the grammatical rules of the specific language are required. It is obvious that, one sentence in natural language can be formulated from different words that obey the specified grammar rules' of the natural language. The knowledge of language is explicitly written in rules or other forms of representation. Rules are formed for every level of linguistic analysis and try to obtain the meaning of the language based on these rules. This approach is good for under resourced natural languages such as Amharic, Afaan Oromoo, Tigrinya and etc. Again, in this approach it is possible to set a number of rules, add and remove rules if necessary. However, it is very difficult and tedious to set all the rules' (knowledge's) of one natural language as even one person is not perfect with one natural language.

2.4.2 Empirical (Corpus Based) Approach

Empirical approaches are based on statistical analysis of the input data (text corpora). Researchers developed a number of techniques' to enable the analysis of text corpora [16]. According to the different researches empirical methods are used to develop robust and efficient natural language processing systems [17, 18] and this is before neural network. However, those techniques have the problems of sparse data and those needs another technique (smoothing techniques) to solve the problem.

2.4.3 Connectionist (neural network) Approach

This approach process information in a similar manner as human brain does. This approach contains different components such as input, neurons, weights, bias, activation functions and etc. The neurons can be interconnected to each other to process the information. The connection (relation) is represented the weight (the stored knowledge). This approach is a self-organizing and good in dealing with linguistic phenomena that are not well understood. Once we build a model using this approach, it can predict for a new data input. This approach needs a large dataset and if once we build the model we cannot restructure the data [8].

2.5 Architectures

This section describes the architectures adopted in the existing natural language interface to database (NLIDB) systems [2, 8, 9, 14, 16].

2.5.1 Pattern matching systems

Early NLIDBs uses pattern-matching techniques to answer the user's queries. This works on the bases that if the input matches one of the patterns then the system is able to build a query for the database. This technique is simple and easy to implement and no need of elaborate parsing and interpretation modules. But implementing pattern-matching in shallow would often lead to bad failures and **ELIZA** is an example of these systems. The general architecture of pattern matching systems of NLIDB is depicted below as follows:

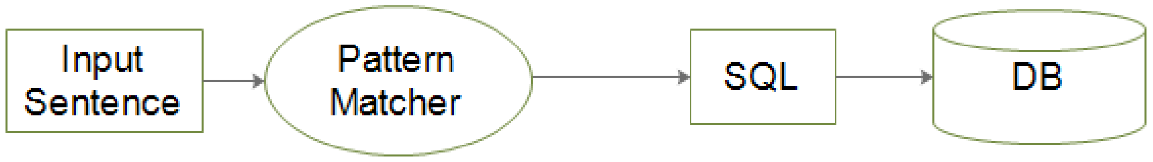


Figure 1: Pattern matching

For example: pattern "Capital"… <country>

Action: report Capital of row where CONTRY=<country> of each row.

So, if a user asks "what is the capital of Ethiopia?", the system can reply "Addisababa". This is based on matching a pattern.

2.5.2 Syntax-based systems

In this architecture, users query is first parsed and then the parsed tree is directly mapped to an expression in some database query language. The advantages' of syntax based technique is that they provide detailed information about the structure of a sentence. The parse tree contains a lot of information about the sentence structure. This shows us all the words and their part of speech and how these words can be grouped together to form a phrases, how these phrases can be grouped together to form a sentences. Having this information, we can map the semantic meanings to certain nodes in a parse tree. However, not all nodes should be mapped and some nodes have to be left just as they are without adding any meanings. And it is not always clear which nodes should be mapped and which should not. In addition of the above problems the same node in different parse trees is not necessarily going to be translated in all the trees. And another problem is a sentence may have multiple correct parse trees that can be translated to different queries. These different queries' may have different results. In syntax based technique it is difficult to directly map a parse tree into some general database query language, such as SQL (Structured Query Language). There are some systems that are developed using this architecture. Systems such as **LUNAR** and **PHILIQA** are developed using this architecture.

The general architecture of the syntax based NLIDB is depicted as follows:

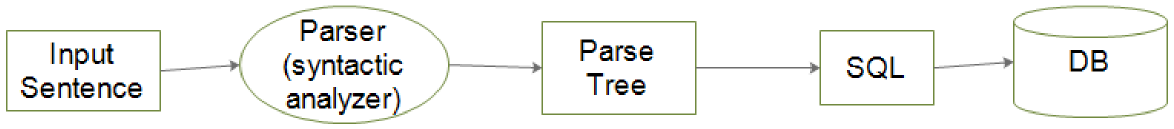


Figure 2: Syntax based

It is obvious that a sentence(**S)** can formulated from a noun phrase (**NP**) and verb phrase (**VP**). For example, "**This shop has a candy**". Thus, **S=NP+VP**

Therefore, from this sentence "**this shop"** is **NP** and **"has a candy"** is **VP.** Also, the **NP** has a determinant **(DT)** and a noun **(N).** From the **NP** the word **"this"** is **DT** and the word **"shop"** is a **N.** The same is true; the **VP** has verb **(V)** and **NP.** Thus**, "has" is** a **V** and **"a candy"** is **NP.** Again, the **NP** has a determinant and a noun**.** Therefore**, "a"** is a **DT** and **"candy"** is a **N.**

The syntactic analysis of the above sentence is described in figure 3 below[50]:

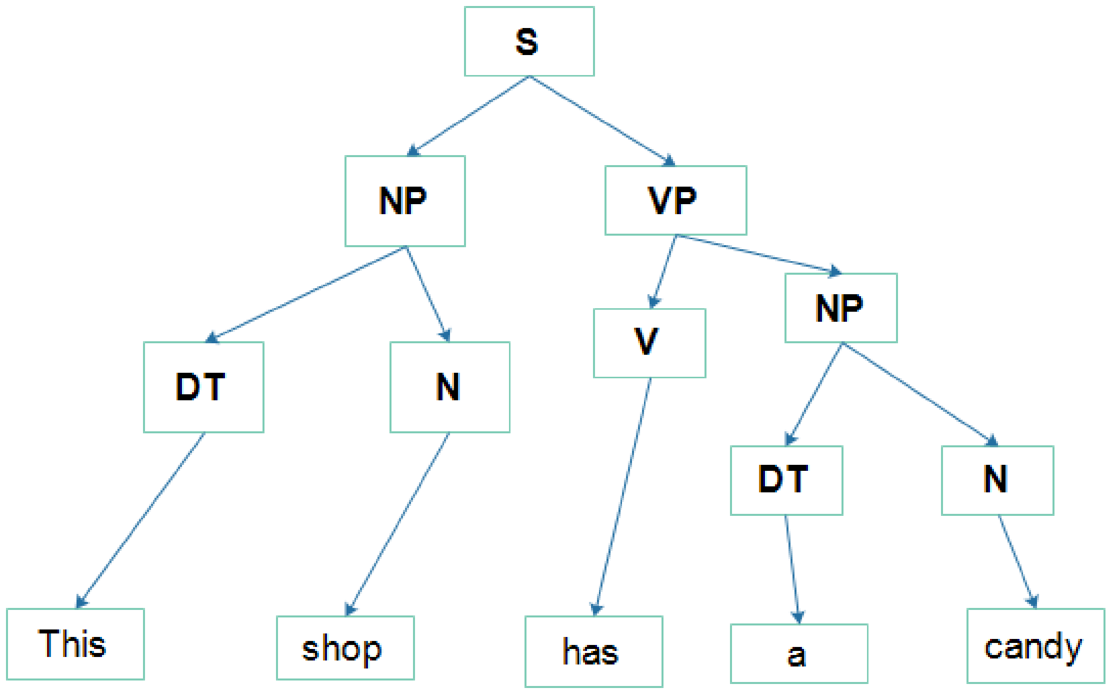


Figure 3: Parse Tree

2.5.3 Semantic Grammar Systems

A semantic grammar system is very similar to the syntax based system and the basic idea of a semantic grammar system is to simplify the parse tree by removing unnecessary nodes or by merging some nodes together. Therefore based on this idea the semantic grammar system can better reflect the semantic representation without having complex parse tree structures. Therefore, a node in the parse tree in a semantic grammar system does not necessarily relate to the general syntactic concepts. This technique also provides a special way for assigning a name to a certain node in the tree and thus resulting in less ambiguity compared to the syntax based technique. The main weakness of semantic grammar is that it requires some prior- knowledge of the elements in the domain and this 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. **LADDER, CHAT-80 and TEAM** are examples of a system that were developed using this architecture. The general architecture of the semantic grammar system of the NLIDB is depicted below as follows:

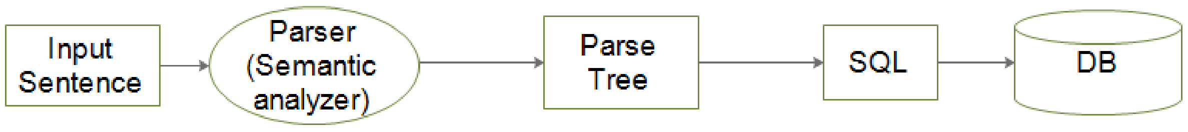


Figure 4: Semantic based

For example: “**abebe highly suffering from dental infection**”. From this sentence when we analysis the semantic meaning it is about the health condition. And from the health condition who is affected. And then, what part is affected and finally it’s severity. The semantic analysis of the above sentence is described in figure 5 below [51]:

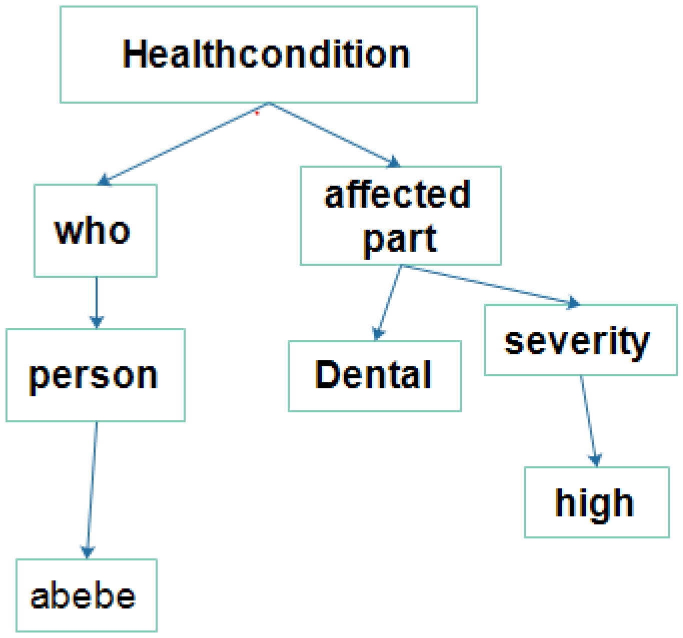


Figure 5: Semantically parsed Tree

2.5.4 Intermediate Representation Languages

This system was proposed, due to the drawback of syntax based technique that means in syntax based technique it is difficult to directly translating a sentence into a general database query languages. In this system the idea is first to map a sentence into a logical query language, and then further translate this logical query language into a general database query language, such as SQL. In the intermediate representation language approach, the system can be divided into two parts. The first part starts from a sentence up to the generation of a logical query and the second part starts from a logical query until the generation of a database query. The intermediate logical query is described in figure 6 below:

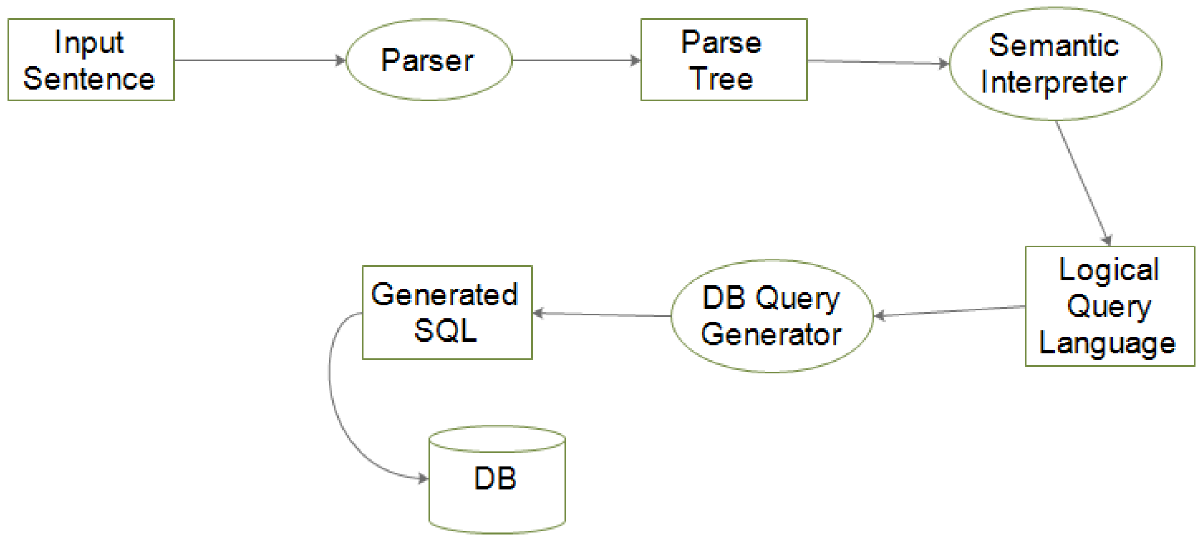


Figure 6: Intermediate representation

Generally, the development of natural language interface to database has many problems, such as robustness, domain-dependence, language-dependence and etc. Many researches were conducted on natural language interface to database by different researchers for different natural languages. However, still it is not successful. This is because of the complexities of the natural languages. As the natural languages have their own syntax and semantics, a model that developed for one natural language could not applied to another natural language.

There are a number of techniques and approaches that can be used to develop the natural language interface to database. Those, also have their own weakness and strength as we described above to develop the natural language interface to database. According to the literatures', neural machine translation is good on developing of a natural language interface to database.

Chapter Three
Tigrinya Language

3.1 Introduction

There are different Semitic languages families in Ethiopia. Amharic (አማርኛ), Tigrinya (ትግሪኛ), Ge’ez (ግእዝ) and Silt’e (ስልጤ) are examples of them [30]. As described in the above Tigrinya is one of the Ethio-Semitic languages and it is spoken in Tigray-Ethiopia and Eritrea. It is also the official language of Tigray regional government of Ethiopia and Eritrea. It is the second largest of Ethiopian Semitic languages next to Amharic language. According to the traditional work of [31], Tigrinya grammar has eight word classes. Those are noun, pronoun, conjunction, preposition, verb, adjective, exclamation mark and adverb. But now according to his recent work, the word classes of Tigrinya are reduced to five. Those are noun, adjective, verb, adverb and preposition. Since he did by putting conjunctions and prepositions together in one class and considers pronouns as subclass of nouns. Morphological analysis of Tigrinya is very complex because as the language is highly inflected from gender, person and number.

3.2 The Tigrinya Alphabets

Tigrinya language writing system has thirty seven characters and each character’s has seven forms. Those forms represent the combination of consonants and vowels [32]. The seven orders represent the different forms of the symbol. There are also characters that are pronounced the same but symbolically different, called homophones. For example, ሀ and ኀ, ጸ and ፀ are used interchangeably since there is no clear rule when to use. They have similar sounds. For example one can write ፀበል(tsebel) or ጸበል, ፀሓይ(sun) or ጸሓይ. Since, they have same meaning.

3.3 Word Classes

As we described before, Tigrinya words are categorized into five classes [30, 31, 32]. Those are noun class, adjective class, adverb class, preposition class and verb class.

3.3.1 Noun: nouns are names given for people, places, or other thing. In Tigrinya, noun can be concrete or abstract. Concrete nouns are used to represent things that can be touched and seen whereas abstract nouns are used to represent things that can’t touch and seen. For example **ህዝባዊነት (**populism**), እንስሳነት (**animalism**), ሕዉነት (**brotherhood**)** and etc. are examples of the abstract class and **ብዕራይ** (ox), **ላሕሚ** (cow), **ገመል** (camel) and etc. are examples of concrete class.

In Tigrinya, noun can be plural and singular. For example **ብዕራይ**(ox), **ላሕሚ**(cow), **ገመል**(camel), **ገረብ**(forest), **ተኽሊ**(plant), **መምህር**(teacher) and etc. are examples of singular nouns and **ኣብዑር**(oxen), **ኣላሕም**(cows), **ኣግማል** (camels), **ኣግራብ**(forests), **ተኽልታት**(plants), **መምህራን**(teachers) and etc. are examples of plural nouns.

There are also common nouns which are used to call a group of things that share common properties. For example **እንስሳት**(animals), **ህዝቢ**(population), **ተኽልታት**(plants) and etc. are examples of common nouns. There are also pronouns which used to represent the proper nouns such as, **ኣነ**(I), **(he)**, **ንሳ** (she), **ንሳቶም|ንሳተን** (they), **ንስኻ|ንስኺ|ንስኻትኩም** (you) and etc. When we bring to our work noun can represent table names, and actual values. However, the actual values of the database can be phrases not only noun. Here below, is an example to show us noun can represent tablenames and actual values. e.g. **ሕሙማት** (patients), **ሓካይም** (nurses), **ዓሶ** (malaria), **ኤችኣይቪ** (HIV) and etc.

3.3.2 Adjective: the main advantages of adjective is to give a clear explanation for a noun and to determine a noun itself. This means that, it can provide us more information about people, animals or things represented by nouns and pronouns. For example **ብዙሕ** (many), **ቀይሕ** (red), **ነዊሕ** (tall), **ቀጢን** (thin) and etc. are examples of adjectives. When we bring to our work, they used to represent the **በዝሒ**(count), **ማእከላይ**(average), **ዝለዓለ**(maximum), **ዝተሓተ**(minimum) and etc.

3.3.3 Preposition: they use to express relationship among things, person and events. They show the relationship between the noun or pronoun and other words in a sentence. Those use to describe the time when something happens or the way in which something is done. These are commonly followed by a noun phrase or pronoun. For example **ክሳብ|ናብ** (to), **ዉሽጢ**(in), **ልዕሊ**(on), **ድሕሪ**(after), **ምስ**(with), **ምክንያቱ**(because), **ወይድማ**(or), **ይኹንእምበር**(however) and etc. are examples of prepositions. When we bring to our work, it used to represent **ወይድማ** (or), **ን-ን** (and), **ልዕሊ** (greater), **ትሕቲ** (lower), and etc.

3.3.4 Adverb: adverbs are used to express and modify verbs, adjectives, sentences, clauses and other adverbs [33]. All verb modifiers are usually express time, manner, and number. For example **ቀልጢፍካ (**quickly), **ብደምቢ (**deeply), **ብጥንቃቀ**(carefully**)** and etc. are examples of adverbs. When we bring to our work, it is used to represent **ጸምቢርካ (**jointly**), ሰሪዒካ (**orderly) and etc. Those are used to join and group the queries in our work.

3.3.5 Verb: verb used to describe subject's action or state within a sentence. Therefore, without a verb any sentences can't provide complete information. Most of the time, they are found at the end of the sentences. For example **በሊዑ** (ate), **ከይዱ** (went), **ሰትዩ** (drank), **ጎዩ** (runned), **ደኪሙ (**tired) and etc. are examples of verbs. When we bring to our work it is used to represent **ኣርኢ (**select**), ኣጥፍእ**(delete), ተክእ(**update**) and etc.

3.4 Punctuation Marks

Tigrigna has its own punctuation marks which are used to indicate boundaries of a sentence, list of words and the connection between sentences. There are many punction marks in Tigrinya. Some of the most commonly used Tigrigna language punctuation marks are listed below [30]:

3.4.1 Preface Colon (፦): colon is a punctuation mark which used to separate a small title from it's definition. We can write a small title and then we write a colon. After we written the colon we can write the definition of the small titles. Most of the time, the definition of the small title is written inline next to the colon.

3.4.2 Full stop (።): full stop is a punctuation mark which used to show the end of a sentence. Full stop is always written at the end of the sentences. We can use this punctuation mark to conveying meaningful information.

3.4.3 Colon (፤): colon is a punction mark which used to connect two sentences. Most of the time we use with complex and compound complex sentences to separate one sentence from another sentence.

3.4.4 Comma (፣): comma is a punctuation mark which used to separate a list of words from each other. Always before the end of the last word of the list we must use connectors. We can use this punction mark to separate a list of words.

3.4.5 Question mark (?): a question mark is a punctuation mark which used to indicate the end of a question. This must write at the end of the questions. we can use this punction mark to ask a question for others.

3.4.6 Exclamation mark (!): exclamation mark is a punction mark which used to show the end of a forceful speech. This also must write at the end of the forceful speech. We can use this punctuation mark to express our emotion to others.

3.5 Types of Phrases

A phrase in Tigrinya is a structure which is constructed from one or more words in Tigrinya language. A phrase in Tigrinya language may or may not convey a meaning. In Tigrinya language, there are five types of phrases which are described below [30, 31, 32]:

3.5.1 Noun phrases: it consists of **noun** (e.g.**ኣበበ)** or a **pronoun (ንሱ** (he), **ንሳ** (she), **ንስኻትኩም** (you) as a head. For example **ኣበበ መጺኡ፡፡** (abebe came), here, **ኣበበ** (abebe) is an example of simple noun phrases.

3.5.2 Verb phrases: it is composed of a verb as a head. For example **ኣበበ መጺኡ፡፡** (abebe came), Here, **መጺኡ** (came) is the verb phrase.

3.5.3 Adjectival Phrases: It is composed of an adjective as head, and other constituents such as complements, modifiers and specifiers. For example እቶም ብጣዕሚ ነዋሕቲ(these very tall) is an example of AdjP, Here, **እቶም(**those) are specifiers, **ብጣዕሚ(**very) is a modifier that used to modify adjective(**head**) called **ነዋሕቲ(**tall).

3.5.4 Prepositional phrases: it can be constructed from a preposition head and other constituents such as: nouns, noun phrases, verbs, verb phrases, etc. for example **ብላዕሊ ኣቀምጦ** (put on top) is an example of P. Here, **ብ(**on**)** is a **preposition** which combined with **ላዕሊ(**top**).**

3.5.5 Adverbial phrases: it can be constructed from one or more adverbs in the language. For example **ቀልጢፉ መጺኡ፡፡** ([he] came quickly) is an example of Adverbial Phrases, Here, **ቀልጢፉ (**quickly) is the **head**.

3.6 Types of Sentences

In Tigrinya language, a sentence is a group of words which conveys meaningful information. Tigrinya language has its own forms of sentence construction system and rules. Tigrinya sentences formulation follows the **subject-object-verb** (SOV) word orders. For example **ኣበበ እንጀራ በሊዑ፡፡** (abebe ate Injera.). Here, **ኣበበ(**abebe**)** is the **subject** (action performer),

እንጀራ(Injera**)** is the **object** (action receiver) and **በሊዑ(**ate**)** is the **verb** (action). Tigrinya language has four major types of sentences and those are described below [30]:

3.6.1 Simple sentences: This type of sentences has a subject and a verb. This type of sentences has independent clause. For example **ኣበበ እንጀራ በሊዑ።** (abebe ate Injera.). Or **ኣበበ በሊዑ።** (abebe ate.) are examples of simple sentences.

3.6.2 Compound sentences: This type of sentences has two or more than two independent clauses which are joined together using connectors' words. These connectors can be alternative or associative connectors. For example **እንተደሊኻ ተምሃር እንተደሊኻ ግደፎ::** (either you learn **or** leave it) is an example of the **alternative** connecter. This provides a choice either to learn or leave the education and both alternatives are connected using "OR". Again, **ኣበበ ነዊሕን ጽቡቅን እዩ።** (abebe is tall **and** handsome) is an example of **associative** connecter. The connecters which are used to connect these sentences are **ወይድማ** (or), **ን-ን** (and), **ምስ** (with) and etc.

3.6.3 Complex sentences: This type of sentences contains two or more clauses, and at a least one clause should be dependent on the other clause. For example **ሎምዓንቲ ድሕሪ ቤተ ክርስትያን ምስዓምና ዕዮ ገዛና ሰሪሕና** (we have done our homework, after we were prayed in church today). In this example the clause **ዕዮ ገዛና ሰሪሕና(**we have done our homework**)** is a dependent clause over the **ሎምዓንቲ ድሕሪ ቤተ ክርስትያን ምስዓምና(**after we were prayed in church today**)** which is an independent clause**.**

3.6.4 Compound-Complex sentences: this type of sentences consists of at least two independent and one or more dependent clauses. For example ኣደይ ናብ ዕዳጋ ክዳን ክትገዝእ ምስ ከደት፥ እንጀራ በሊዐ ናብ ቤት ትምህርቲ ከይደ። (when my mother went to the market to buy a cloth, I ate Injera and I went to school.) . In this example ኣደይ ናብ ዕዳጋ ክዳን ክትገዝእ ምስ ከደት (when my mother went to the market to buy a cloth) has two independent clauses. ኣደይ ናብ ዕዳጋ ምስ ከደት(when my mother went to the market) and ኣደይ ክዳን ክትገዝእ ምስ ከደት (when my mother went to buy a cloth). The example እንጀራ በሊዐ ናብ ቤት ትምህርቲ ከይደ።(I ate Injera and I went to school.) Are the two dependent clause on the independent clause and this has two dependent clauses. Namely, እንጀራ በሊዐ(I ate Injera) and ናብ ቤት ትምህርቲ ከይደ።(I went to school).

Generally, this chapter has been discussed about Tigrinya language because it is necessary to the work. In Tigrinya language interface to database the queries can be simple sentences, compound sentences, complex and compound-complex sentences. Those types of sentences may have different words those different words may have different word class such as noun, verb, adjective, preposition and adverbs etc. Also, Tigrinya words are highly inflected from gender; number, person and etc. Therefore, that’s why we discussed about Tigrinya language to make clear for the readers. Anyone can read this to understand about Tigrinya language. However, this is not enough to understand in detail about the language as Tigrinya is highly inflected and derivated language. For detail and better knowledge of Tigrinya language anyone can refer to Tigrinya books.

CHAPTER FOUR
Related Works

4.1 Introduction

A number of researches have been conducted on the natural language interface to database in different natural languages such as Amharic, Afaan Oromo, Arabic, Hindi, Punjabi and English. The researchers was used different techniques as discussed in chapter two, to develop the natural language interface to database for the above listed natural languages. According to the literature most of the researchers were focused on information retrieval. In this study, the approach and the functionality of the database (support of the complex queries of natural language) were considered as the baseline. This work considered the conducted researches on local languages and on some international languages. Here, below the different natural language interface to database of the above listed natural languages were discussed in the next sections:

4.2 Information Retrieval

Information retrieval defined as the process of finding information from a collection of documents. This aims on retrieving relevant documents that can satisfy the information need of users. However, still the IR system retrieves relevant and irrelevant documents or information's that can satisfy users or not. Therefore, to make the IR system to retrieve an exact answer to the user query, Question Answering system was good [35]. This system aim's to provide an exact answer to the user query. By providing an exact answer to the user query, it makes a user to save his/her time, effort and other resources. By having of the exact answer within a short time and less effort a user can satisfied.

There is one attempt on Tigrinya question answering system for factoid questions by [35]. The researcher developed the prototype using statistical approach on unstructured data. He prepared a question with it's corresponding answer and by calculating the similarity between the input sentence with in the corpus (prepared questions), then he makes to return the answer. He scored an overall accuracy of **87**%. However, in his work updating and deleting of the contents is impossible. Since, some facts may change from time to time. And also, when the size of corpus is very large it can take long time as it is developed using the statistical approach. This means that it may have a high computational time. Also, he considered only factoid questions. He recommended to develop a question answering system for complex factoid questions and none factoid questions. Again he recommended to develop a standard parallel corpora as the performance of the question answering system is depend on the parallel corpora. Finally, he recommended adding Tigrinya WordNet and Tigrinya spelling checker can also improve the performance.

4.3.1 Text summarization

Retrieving of some relevant part of a document (unstructured text) to the user is called text summarization. The data with in documents is very large and to read the whole is very difficult. Because, it takes time. So, instead of reading manually, applying automatic text summarizer is good [49].

There is one attempt in Tigrinya automatic text summarizer by [49]. The researcher used term frequency and topic modeling techniques to develop the prototype. The first technique used to add a sentence that has the most frequent words in the documents to the summary and the second identifies the topic word and it extracts the sentence that has the topic word from the documents to add to the summary. The data sources used 30 news articles from Aiga forum and dmtsi woyane Tigray websites to evaluate the prototype. The performance scored is an f-score of 46%. The accuracy scored is to lower and the summary result is subjective. The researcher recommends having a large prepared Tigrinya corpus and good Tigrinya stemmer can improve the performance of the summarizer. He recommends topic modeling is better than term frequency on selecting of the important sentences and it will be better if the title is supported by sub topic of the input document [49].

Generally, question answering system and text summarizations are carried out on unstructured data (documents). Those systems are impossible to apply to natural language interface to database. Because, natural language interface has their own functionality and it is one application area of NLP. For example: updating and deleting of information as information can modify from time to time. In general, it has it's own syntax. Therefore, natural language interface is required to enable users to access and manipulate the database contents using native language.

4.4 NLIDB for different languages

4.4.1 Amharic Language

Modeling and Designing Amharic Query System to Bilingual (English-Amharic) Databases has been developed by [2]. In her work she considered the natural language preprocessors, query mapper, query generator and query executers. She used a dictionary to develop the system. She evaluated the system from query success rate and user satisfaction and scored 76.5% and 65% respectively. She recommends the database to be extended to include more tables and attributes, extended the system to accept more complex query and the where clause to have Amharic values.

"Possibility of Amharic Query Processing in Database using Natural Language Interface" has been developed by [8]. He developed an algorithm to efficiently map Amharic language into Structured Query Language (SQL). He divides the algorithm into four parts an algorithm to handle query selection, conditional query, aggregation and grouping and ordering queries. He used rule based approach to develop the system. He has implemented the algorithm using java and tasted on Human Resource Management (HRM) database which containing Employee, Department and Employee on education tables. His experimental result shows that 91% overall accuracy. He recommends extending his work with temporal queries.

4.4.2 Afaan Oromo Language

"**Designing Natural Language Interface to Database for Afaan Oromo**" has been developed by [36]. His prototype accepts the Afaan Oromoo language plaintexts and translates into the corresponding SQL statements to further execute and retrieve the data stored in the database. He used the pattern-matching techniques and rule based approach to develop the system. He considered the preprocessing, spelling checker and auto-correction, checking word similarity, database element extraction and etc. His system handled a single query, list query, multiple conditional queries, aggregate and join query. He evaluated his system using 100 Afaan Oromoo queries on student database and scored an average performance of 92.3%. Finally, he recommended that applying neural network, machine learning and parsing techniques can improve the success rate of natural language interface to database.

4.4.3 Arabic language

An **Arabic Natural Language Interface System for a Database of the Holy Quran** has been developed by [6]. In his work he translated the natural Arabic requests such as question or imperative sentences into SQL statements (commands) to retrieve answers from a Quran DB. He performed parsing and little morphological processes according to a sub set of Arabic context-free grammar rules to work as an interface layer between users and database. He recommends the system to be extended to include more tables and attributes and to accept more complex query.

4.4.4 Hindi Language

A '**design and implementation of Hindi language interface to database'** has been developed by [7]. In his work he accepted the query in Hindi, tokenized them into tokens and mapped the Hindi words with their corresponding English words with the help of system dictionary maintained. After mapping the sentences, the sentence has been checked whether it is data **retrieval**, **update**, **insert** or **delete** type of sentences. This was done by analyzing the input Hindi sentence. After analyzing, the input Hindi sentence, table names, column names, condition and operators are searched in the dictionary. After mapping, SQL query is generated and executed on database to display the result set to the user.

4.4.5 Punjabi Language

A '**Punjabi language interface to database'** has been developed by [33]. He used agriculture domain which has farmer, crop and sale tables. This accepts query in Punjabi language that is translated into SQL query, by mapping the Punjabi language words, with their corresponding English words with the help of database maintained. Then, the query is executed. The architecture of this system has three phases. Tokenizing the query, Mapping and formation of SQL query and Execution of query. The researcher recommended that the existing system can be extended to : execute Queries that involves joining of tables and handle various clauses , made domain independent, handle more complex queries , Develop an interface for administrator and accepts queries only in a particular format. So to make the system to accept all the queries, they recommended to use Punjabi parser. They have been tested for three types of queries: Queries for selection of all the columns , Queries for selection of certain columns and Queries for selection of certain rows from certain columns.

4.4.6 English Language

A 'Pattern based Natural Language Interface to Database' has been developed by [34]. They developed a system for converting English language query into SQL query. They defined patterns for simple query, aggregate function, relational operator, short-circuit logical operator and join. They categorized the patterns into valid and invalid. Valid patterns are directly used to translate natural language query into SQL query whereas an invalid pattern assisted the query authoring service in generating options for user so that the query could be framed correctly. Their system taken an English language query as input recognizes pattern in the query, selects one of the above mentioned features of SQL based on the pattern, prepared an SQL statement, fired it on database and displayed the result. They have been recommended that their interface to include or aware of other features such as clauses (group by and order by), keywords used in predicate like BETWEEN, NOT IN, IN etc., nested query, varieties of join, union and much more.

Generally, the related works were developed using non robust approaches and their accuracy were calculated manually. Therefore, to write all the grammar rules of one natural language and to carry out the calculation, it was very tedious and not effective. Again, the queries of the natural language that enabled users to access the database were also limited as it did not include the functionality of the database. One natural language sentence can be described in different forms that have same meaning. So, stating all the rules for one natural language is very difficult. Again, those works were developed for non-Tigrinya language users. Therefore, this study overcomes those problems by extending the functionality of the database and by using the robust approach. This study considers complex queries and manipulation queries to enable the user to access and manipulate the database. Again, this study developed for Tigrinya language users. As this study was carried out using the existing tools and approaches, no algorithm was contributed rather; it solved the existing problem using existing tools and approaches. Therefore, the general contribution of this study was to enable Tigrinya language users to access and manipulate the database contents using Tigrinya plaintexts. Generally, below table showed the summary of the related works on NLIDB.

Table 1: Summary of related works

Author	**Language**	**Approach/ technique**	**Evaluation**	**Accuracy**	**Domain**	**Recommendation**
Tihitina	Amharic	Rule based	User satisfaction and query success rate	65% and 76.5%	Telecom	Include more tables, complex queries and where clause to include Amharic values

Smegnew	Amharic	Rule based	query success rate	91%	HRM	include manipulation and temporal queries
Shumet	Afaan Oromoo	Rule based	query success rate	92.3%	Student	Applying parsing, Machine learning and neural network
Khaled Nasser	Arabic	Parser	query success rate		Quran DB	Include more tables, attributes, and more complex queries
Ashish Kumar	Hindi	Rule based	query success rate		Independent	
Suket et al	Punjabi	Rule based	query success rate		agriculture	Include clause(join and union), Complex queries and parser
Niket Choudhary	English	Rule based	Semantic test(query success rate)		Library	Include clause(join and group by) and Predicates(IN, BETWEEN, NOT) and complex queries

Chapter Five
Designing and Developing of TLIDB

5.1 Introduction

A new healthcare database was designed to develop the Tigrinya language interface to database. Because, this can minimize the time and efforts of the researcher and also it can improve the knowledge of the researcher on database design. To use existing database permission is required from managements. So, to have permission it has many challenges and takes time. And even, after long time process the managements could not be voluntary to provide the database. That's why a new database is designed.

In designing of Tigrinya language interface to database prototype, the name of the database, tables, attributes and actual values are all in Tigrinya plaintexts. To translate (convert) the Tigrinya plaintexts into SQL statements, neural machine translation has been used. The prototype accepts the queries in Tigrinya plaintexts and translates it to SQL statements that can be execute in database either to display or to manipulate the contents of the database based on the Tigrinya input queries. The prototype translates the Tigrinya plaintexts into SQL statements sequence by sequence using the encoder-decoder model. The encoder accepts the Tigrinya plaintexts (input data) and converts into a vector of floats. The decoder accepts both the input vector and the target vector and converts into vectors of floats.

As there is no prepared Tigrinya dataset for Tigrinya natural language interface to database, a dataset was prepared from scratch by Tigrinya language professionals'. The dataset was prepared, first a sample database that has three tables with four or seven attribute were designed. After that, the healthcare database was created and filled records with health professionals. Then the Tigrinya sentence was prepared based on those tables, attributes and values with the help of Tigrinya language professionals and with a corresponding SQL.

To design and develop the Tigrinya language interface to database, a healthcare database that has three tables each with four or seven attributes were considered. Here below, are described sample table names in table 1 as follows:

Table 2: Sample table Names

Tablenames in Tigrinya	Their meaning in English
ሕሙማት	Patients
ሕማማት	Diseases
ሰራሕተኛታት	employees

The table holds the table names with their corresponding meaning in English. Again, here there is a table named table 2 that holds the sample attributes of the employees and patients table names. Most of the attributes are common for the two table names.

Table 3: Samples attribute names of patients and employees

Attribute names in Tigrinya	Meaning in English	data types
መለለይ	Id	Int
ሽም	Name	Nvarchar
ሽምኣቦ	Fname	Nvarchar
ዕድመ	Age	Int
ፆታ	Sex	Nvarchar
ዓይነት	Type	Nvarchar
ደሞዝ/መሃያ/ክፍሊት	Salary	Double

This table contains the sample attributes names with their meaning in English words and their data types.

5.2 Types of Queries

Users may formulate different queries using the above information stored in the tables to query the database. Generally, the types of queries are grouped into two. Namely, retrieval query and manipulation query. One may ask a simple query and others may ask a complex query. So in this work, some of the queries that can be formulated by users with their corresponding SQL statements are discussed.

- **ሓበሬታ ሕሙማት ኣርእየኒ። (select * from ሕሙማት;)**

This is a simple query that shows all the details' of the patients. This is the retrieval query. **ሕሙማት**(patients) is a table name and **ኣርእየኒ(**select) is the retrieval and **ሓበሬታ(**information**)** are stop words. A user can write this statement to view all the information of patients.

- **ሽም ሕሙማት ኣርእየኒ። (select ሽም from ሕሙማት;)**

This is also a simple query that shows the name of all patients. **ሽም** (name) is the attribute name, **ሕሙማት**(patients) is a table name and **ኣርእየኒ**(select) is the retrieval. A user can write this statement to view the name of all patients.

- **ሽም ሕሙማት ዓሶ ኣርእየኒ(select ሽም from ሕሙማት where ዓይነት=' ዓሶ')**

This query is more complex than the former queries. From this statement **ሽም(**name) is an attribute, **ሕሙማት**(patients) is table name, **ዓሶ**(malaria) is an actual value, **ኣርእየኒ(**select) are a retrieval. A user can write this statement to view the name of all patients of malaria.

- **ዓሶ ወይድማ ሳምባ መንቀርሳ ዓይነት ሕሙማት ሽም ኣርእየኒ። (select ሽም from ሕሙማት where ዓይነት=' ዓሶ' or ዓይነት=' ሳምባ መንቀርሳ ' ;)**

This query is more complex than all the former queries. From this statement, **ዓሶ**(malaria) **and ሳምባ መንቀርሳ** (tuberculosis) are actual values, **ወይድማ**(or) is a connecter, **ዓይነት**(type) is an attribute name that has two values, **ሕሙማት**(patients) is a table name, **ሽም**(name) is also an attribute name and **ኣርእየኒ**(select) is a retrieval query. The two actual values are connected using the word **ወይድማ(OR).** A user can write this statement to view the name of all patients' of malaria or tv.

- **ሽም ሕሙማትን ምልክታት ሕማማትን ጸምብር ኣርእየኒ። (Select ሕሙማት.ሽም, ሕማማት.ምልክት from ሕሙማት JOIN ሕማማት ON ሕሙማት.ዓይነት =ሕማማት.ዓይነት;)**

This query is also more complex than the former queries. From this statement **ሽም(name)** and **ምልክታት(symptoms)** are an attributes of the **ሕሙማት(patients)** and **ሕማማት(diseases)** table names**. ጸምብር(JOIN)** is the word that we use to merge the contents from both tables. **ኣርእየኒ(Select)** is shows as a retrieval query. The character '**ን'** at the end of the table names shows us to retrieve from both.

All the above listed queries are an example of retrieval query.

Others may formulate a manipulation queries as follows:

- **ባዓል 4 መለለይ ዝኮነ ሰራሕተኛ ሽሙ ብሃይለ ሽምኣብኡ ብግርማይ ተክእ/ተክኣለይ። (update ሰራሕተኛታት set ሽም=' ሃይለ', ሽምኣቦ=' ግርማይ' where መለለይ ='4' ;).**

In this statement, **ባዓል**(owner) is a stopword, **መለለይ**(id) is an attribute name,4 is an actual value of id, **ዝኮነ**(equal) is an equal operator, **ሰራሕተኛ**(employee) is not a real table name but it must be changed to the table name **ሰራሕተኛታት**(employees) because the table name in database is **ሰራሕተኛታት**(employees), **ሽሙ**(name of) is not a real attribute name but shows the name of the owner and it must translate to **ሽም**(name), **ሃይለ**(Haile) and **ግርማይ**(Girmay) are actual values, **ብ**(by) is a prefix in the two actual values, **ሽምኣብኡ**(name of father's) are not also areal attribute name but shows the father's name of owner and must translate to **ሽምኣቦ**(father's name) and **ተክእ/ተክኣለይ**(update) shows manipulation query.

Users may write this to update the name and father's name of a unique person based on his/her id. This is an example of the manipulation type query. This is the simple query in manipulation because the conditional clause has one attribute with it's value. When it has two or more than two it increases it's complexities. Also, the attributes used for the update clause is two. This makes also simple query. When it has three or more than three attributes, it increase it's complexities. Again, when it has one attribute for update and one attribute with it's value for conditional clause, it is the simplest query in manipulation.

- **ባዓል 4 መለለይ ዝኮነ ሕሙም ኣጥፍእ። (DELETE FROM ሕሙማት WHERE መለለይ="4";)**

From this query the word **ባዓል** (**owner**) shows the owner of the value, 4(four) is the actual value of the attribute **መለለይ** (**id**) which used to identify the required one. **ዝኮነ(has)** is an operator that indicates the patient has the **4 መለለይ** (**id**). **ሕሙም** (**patient**) which is a singular name of the table name patients. **ኣጥፍእ(DELETE)** shows manipulation query.

5.3 Neural Network

Neural network have input, hidden and output layers. The input layer is used to feed the input value to the network and the output layer is used to provide the output. The hidden layers are used to provide the discrimination necessary to be able to separate our training data. The output of a neuron is function of the weighted sum of the inputs plus the bias. The function of the entire neural network is simply the computation of the outputs of all the neuron in the neural network.

An activation function is applied to the weighted sum of the inputs of neurons to produce the output. The advantage of neural network is, it has the ability to learn and generalize [44].

$$Output\ of\ NN = \sum_{i=1}^{n} X_i W_i + b$$

In the above formula X_i represents the input vectors, W_i the weight vectors and b represents bias.

5.3.1 Recurrent Neural Network

This is a type of artificial neural network designed to recognize patterns in sequences of data. And this is the most powerful and useful type of neural network. Figure 1 shows us the architecture of the recurrent neural network.

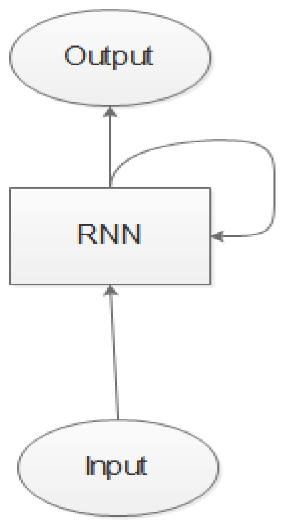


Figure 7: RNN Architecture

In recurrent neural network information are feeds in loop. This type of neural network takes as its input just the current input example it see and also what it has perceived previously in time. Therefore, the decision a recurrent neural network reached at time step t-1 affects the decision it will reach one moment later at time step t.

The type of recurrent neural network that applied to develop the prototype is called LSTM. The LSTM have both hidden state and cell state. LSTM used cell state to store and update the information in LSTM. Vanilla has only hidden state and it used the hidden state to store the information. In along sequence to sequence prediction LSTM has higher performance than vanilla [48]. This mean in long sequence prediction LSTM is better than vanilla.

The LSTM has three gates called input gate, forget gate and output gate. The following equation shows us how the LSTM gates work [46].

$$input\ gate = sigmoid(W_i[h_{t-1}, x_t\ ] + bi)$$

$$forget\ gate = sigmoid(W_f[h_{t-1}, x_t\ ] + bf)$$

$$output\ gate = sigmoid(W_o[h_{t-1}, x_t\ ] + bo)$$

W_x represents the weights of each gate, h_{t-1} represents the vector of the hidden states at t-1, x_t represents the input vectors at time t and b_x represents the bias at each gates. Sigmoid function is an activation function that used to show either the gates allows everything to pass through it or to block. The output of the sigmoid is **0** or **1**. **0** means the gate blocks everything and **1** means the gate allows everything to pass through it.

5.4 The General Architecture of TLIDB

The system was trained by fetching input data from a file that contains Tigrinya sentences with the corresponding SQL statements which was annotated by Tigrinya and SQL professionals. The input data contains both Tigrinya sentences and SQL statements. This input data was splitted into input Tigrinya sentences and SQL statements (target data). After that, the encoder-decoder model was used to translate from input-Tigrinya sentences to their corresponding SQL statements. The input Tigrinya sentences are translated to their corresponding SQL statements sequence-by-sequence. Because, the natural language interface to database is a translation problem, from input Tigrinya sentences to SQL statements. Therefore, the encoder-decoder model is good in sequence-to-sequence translation problems [37].

This model has three layers. Namely input layer, hidden layer and output layer. There is an interconnection between each layer. In this work the output layer uses an activation function called 'softmax' to display the outputs. The softmax activation function is used to classify multiple inputs into multiple outputs. The following formula shows how the probability distribution can be calculated using the softmax activation function.

$$p = \frac{e^i}{\sum_{i=1}^{n} e^i}$$

In the above formula e represents mathematical constant, i represents inputs and n represents the vocabulary size. Based on these values the SQL statement (sequences) can be predicted and for the translated (predicted) SQL statements by the encoder-decoder model, we have executed on mysql database. Finally, when the predicted query is retrieval, an output can display to the user and when the query is manipulation, changes can be made in the database contents.

The general architecture of Tigrinya language interface can be applied in any domain. Here, figure 2 shows the system architecture starting from the dataset upto the final result which displayed or manipulated from/in the database. The functions of each component are also described as follows:

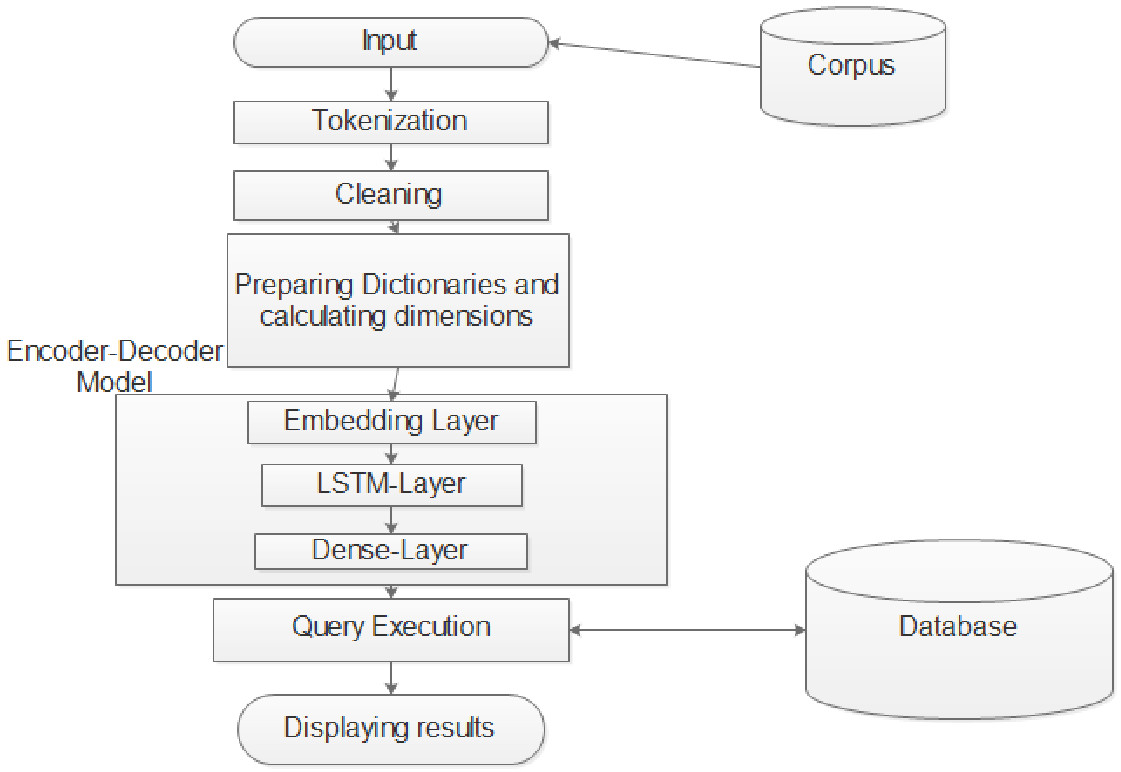


Figure 8: General Architecture

5.4.1 Corpus: corpus is a collection of texts. This corpus has a Tigrinya sentences with their corresponding SQL statements separated by the word <**BEGIN**>. The Tigrinya sentences are the input data and the SQL statements are the output (target) data. This corpus holds both the training and testing data of our model. The input Tigrinya sentences are prepared by Tigrinya language professionals'.

5.4.2 Input Query:

The input is Tigrinya plaintext that is used to access and manipulate the contents of the database. In our work, this is used to accept a retrieval and manipulation queries.

5.4.2 Tokenizer:

Tokenizer is uses to split the input Tigrinya statement into character and words. To split into words the tokenizer uses spaces. We have been applied splitting into words after cleaning and into character before cleaning. Therefore, we did not consider that to include in the architecture below cleaning. for example in the above Tigrinya statement □□□(owner) is a none relevant word, □□□□ (id) and □□(name) are attribute names, **6** and □□□□ (teklay) are an actual values □ is a prefix, □□□(=) is an operator **ሕሙም**(patient) represents a table name but not a real table name which is found in the database.

5.4.3 Cleaning:

Here, data cleaning is carried out. This means that the punction marks in input Tigrinya sentences must be removed to split the sentences into words. After this, tokenization of the input Tigrinya sentence can carry out.

5.4.4 Preparation of Dictionaries and specifying of the dimensions values:

On this vocabularies of both the source and the target have been prepared. And then, four dictionaries have been prepared for both input Tigrinya sentence and the SQL statements. The two dictionaries have been prepared for the input Tigrinya sentence which has words and their indexes' and vice versa. The two dictionaries also have been prepared for the target statements which has words and their indexes and vice versa. Here also, the dimensions for the encoder and decoder architecture have been specified and calculated from the given input data (corpus). And also, generating of the data in batch has been carried out.

5.4.5 Encoder-Decoder Model

5.4.5.1Word representation:

Here, below word representation techniques have been discussed as follows:

5.4.5.1.1 One-hot encoding: this encoding mechanism is used to encode the words into a one-hot vector. The one hot vector represents words in size equal with the vocabulary size of the input data and it is a sequence of one and zero. If the word is available it assigns one, if not it assigns all sequences zero. Those vectors are high dimensional and sparse. Therefore, this is not used to create a low dimensionality model.

5.4.5.1.2 Word embedding: is also used to convert words into their corresponding vectors. Those techniques are used to create a low dimensionality model. Since, they are a low dimensional and dense. Low dimensional means it is possible to specify the dimensionality size and it is not the vocabulary size as in one-hot vectors. And dense means all the components (features) of the vectors contains none zero numbers unlike one-hot vectors. For example word2vector is used to convert input words into vectors (sequence of real numbers).

5.4.5.1.3 Word embedding layer: Word embedding layer of keras python library was used to convert words into their corresponding vector representations [39]. Word embedding layer was developed as part of deep learning in keras. This generates a random value to the first and iteratively learns the representation of all words. This layer is used to build a low dimensionality model and a dense representation like the embedding techniques such as word2vector. Therefore, this layer was used to translate (embed) the dataset to the corresponding vector representations and this vector representation is fed to the LSTM layer. That's why we were used this layer to represent words.

5.4.5.2 LSTM Model:

This model is a seq2seq recurrent neural network model which uses to translate one sequence to another sequence. In old type of recurrent neural network, it was difficult to translate a variable length input into a variable length output. However, with the use of recurrent neural network which is the LSTM, the problem is solved. The LSTM has the capability to store past information and this makes it a very powerful in sequence prediction problems [40]. As described in the above, the translation of input Tigrinya sentences to their corresponding SQL statements is a sequence problem. Therefore, the LSTM model is used to translate the input Tigrinya sentences into their corresponding SQL statements.

5.4.5.2.1 Encoder LSTM

The encoder LSTM reads Tigrinya input sequences and summarizes information as hidden state and cell state vectors called internal state vectors. The encoder reads Tigrinya input sentence one word after the other from the sequences. For example in the following Tigrinya input sentences: **ሽም ሕሙማት ዓሶ ኣርእየኒ።.** This sentence has a sequence of 4 words. First the encoder reads the word **ሽም,** then **ሽም ሕሙማት,** then **ሽም ሕሙማት ዓሶ** and finally reads **ሽም**

ሕሙማት ዓሰ ኣርእዩኒ. This has four (4) time steps (sequence length) to read the complete Tigrinya input sentence. As described in figure 3 the encoder reads all sequence one after the other until all the sequences is read. If the Tigrinya input data has a sequence of length n, the encoder starts reading from zero up to n. Zero is the initial-state that the encoder starts reading and n is the final step of the sequence.

As described in figure 3 the input to the encoders are: X_1, X_2, X_3, ……$\mathbf{X_n}$ and the encoder state vectors are: H_0C_0, H_1C_1, H_2C_2,……..H_nC_n.
The outputs (predicted) of the encoder's are: Y_1, Y2,….…Y_n. and we should discard them.
From the above example of Tigrinya sentence:
The inputs to the encoder are: X_1=**ሽም**, X_2=**ሕሙማት,** X_3=**ዓሰ, X_4=ኣርእዩኒ.**
The encoder internal state vectors are: H_0C_0=0, H_1C_1=**ሽም,** H_2C_2=**ሽም ሕሙማት,** H_3C_3=**ሽም ሕሙማት ዓሰ,** H_4C_4=**ሽም ሕሙማት ዓሰ ኣርእዩኒ.**
The encoder sets the initial vectors to zero (i.e H_0C_0 is set to zero as the encoder has not started to read the input) and the final state vector(thought vector) is the final state that encode(summarize) the complete input sequences[37].

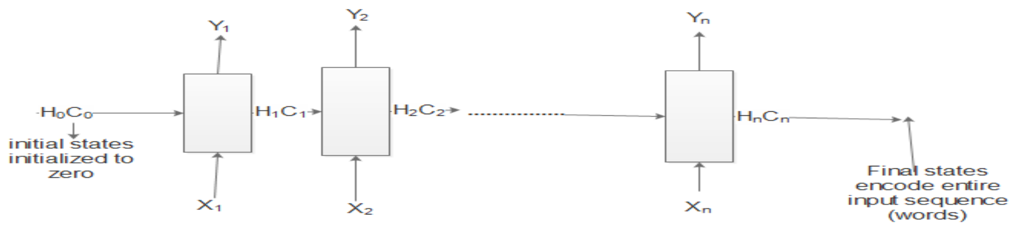

Y_1
Y_2
Y_n
H_0C_0
initial states initialized to zero
H_1C_1
H_2C_2
H_nC_n
X_1
X_2
X_n
Final states encode entire input sequence (words)

Figure 9: Encoder

From the above diagram the variables shows us**:** X_n=represents input data
$\mathbf{H_nC_n}$=represents state vectors (hidden state and cell state respectively)
Yn=represents output (predictions) of the encoder at each time step (iteration)
The role of the internal states at each time step is to remember what the LSTM has learned (read) till the current time step. For example the two vectors H_2C_2 will remember what the LSTM network has learned till the time step two (2). At time step two the LSTM has learned **ሽም ሕሙማት.** Generally, the summary of information till time step two is stored in vectors H_2 and C_2. In the above input Tigrinya sentence the final state is H_4C_4. This contains the summary of

the complete (entire) input sentences which is **ሽም ሕሙማት ዓሶ ኣርእየኒ**. This step are also called last time step and also called "**thought vectors**" as it summarizes the input sequences as vectors. "The size of the internal state vectors (both hidden state and cell state) is equal to the number of neurons used in LSTM cell"[37].

Yn is a probability distribution over the entire vocabulary of Tigrinya input sentence which generated by softmax activation function. So, every Yn is a vector of size input Tigrinya sentences words (all-Tigrinya words or vocabulary-size) which representing probability distribution. The output of the encoder (Yi) must be discarded as we have started to generate the output (predict), when the entire Tigrinya input sentence read.

Generally, the input Tigrinya sentence (sequence) was read word by word and preserves the internal states of the LSTM network generated after the last time step of H_nC_n. The two vectors (states) called H_nC_n are called as the encoding of the input Tigrinya sequence, as they encode (summarize) the entire input in a vector form. Outputs of the encoder (Yn) are discarded at each time step because we start to generate the output once the entire input Tigrinya sequence (complete sentence) read. During the training and inference phases, the encoder LSTM has same role.

5.4.5.2.2 Decoder LSTM

The training and inference phases of the Decoder LSTM has described below.

5.4.5.2.2.1 Training mode

The goal of the training phase is to train the decoder to display the target sentence of the corresponding input Tigrinya sentence. In the target sequence two tokens must be added one for starting of the sequence and one for ending of the sequence. Adding '<BEGIN>' as the starting token and '<END>' as ending token in the target (output) sequence (simply SQL statement). For example <BEGIN> SELECT **ሽም** FROM **ሕሙማት** WHERE **ዓይነት**=' **ዓሶ**'; <END>

The above SQL statement is one of the output sequences (target sequence) of the input sequence "**ሽም ሕሙማት ዓሶ ኣርእየኒ።**"

Figure 4 shows us how a decoder LSTM training on[37].

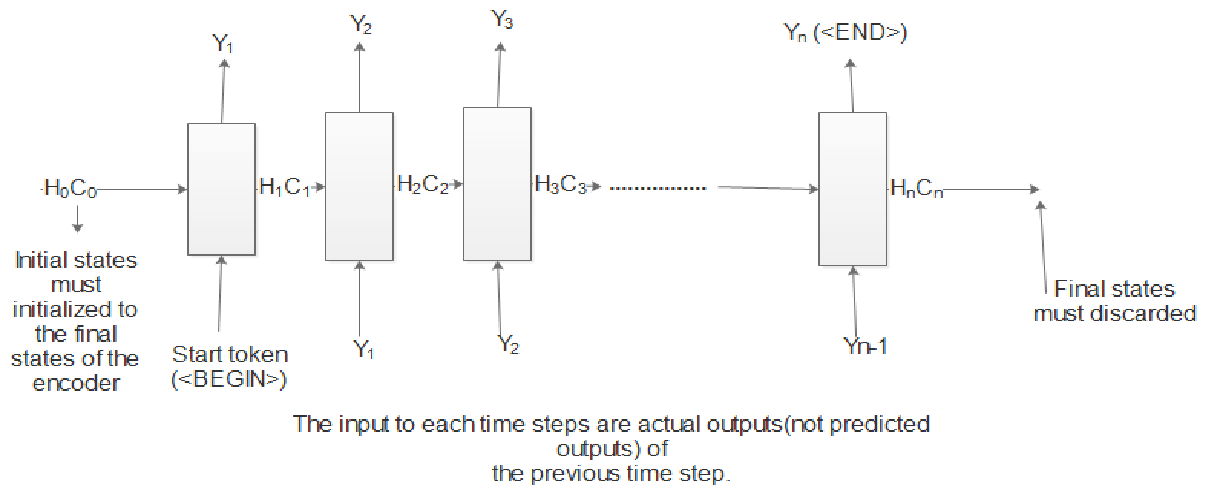

Y₁
Y₂
Y₃
Yₙ (<END>)
H₀C₀
H₁C₁
H₂C₂
H₃C₃
HₙCₙ
Initial states must initialized to the final states of the encoder
Start token (<BEGIN>)
Y₁
Y₂
Yn-1
Final states must discarded
The input to each time steps are actual outputs(not predicted outputs) of the previous time step.

Figure 10: Training mode of decoder

From the above output sequence example and the diagram:

The start token is: **<BEGIN>**

Y_1=SELECT, **Y_2=ሽም, Y_3=FROM**, **Y_4=ሐሙማት,Y_5=Where, Y_6=ዓይነት=’ ዓሶ’;** ,

The final token: Y7=**<END>**

The initial states of the decoder (H_0C_0) are set to the final states of the encoder. Based on the information summarized by the encoder the decoder learns (trains) to generate the output sequence. The start token (<BEGIN>) given in the first time step makes the decoder to start generating the next actual word of the SQL statement (output sequence). Next, at each time step the actual output word are fed as an input to the decoder until the last word of SQL statement. This means that the input at every time step is provided as an actual output not predicted output from the previous time step. In the last time step the decoder trains to predict the “<END>” token to indicate the end of the SQL statement. The decoder conditioned to predict the next word in a given sequence (context, which is the thought vector). This calculates the probability of the next word using the following formula [47]:

$$p(y/x) = \frac{p(x \text{ and } y)}{p(x)}$$

In the above formula y represents the next word and x represents the previous words (thought vectors). At the last, from the predicted outputs of each time step the loss is calculated. And then, the errors are back propagated through time in order to update the parameters (weights) of the network.

5.4.5.2.2.2 Inference mode

As described above the encoder LSTM plays the same role of reading input Tigrinya sentence and generating the final states H_n,C_n. And then, having these final states the decoder has to predict the entire SQL statement [37]. For example

Input Tigrinya sentence: **ሽም ሕሙማት ዓሶ ኣርእዩኒ።**

Output SQL statement: SELECT **ሽም** FROM **ሕሙማት** WHERE **ዓይነት=' ዓሶ';**

Step 1: encode input Tigrinya sentence into thought vectors

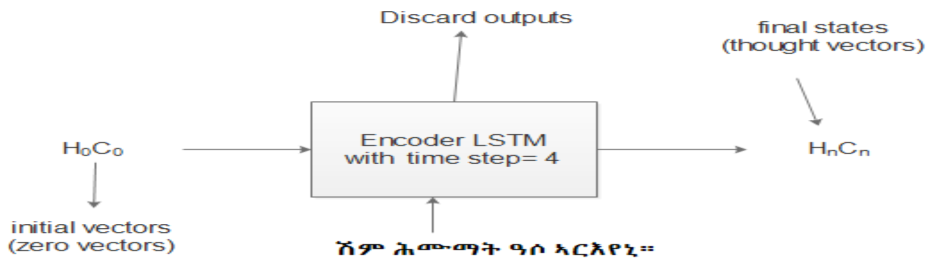


Figure 11: Encoding input sequence into final state (thought vectors)

Step 2: Start generating the SQL statement (target sequence) in a loop word by word

The initial states of the decoder must set to the final states of the encoder. Then the decoder takes input word by word. The first input token is fed to the decoder at a time. The states at that time is fed as an input states to the next time and the output at that time is also fed as an input in the next time until it gets the word <END>. Here, below is showed the decoder process at time one and last time.

For example at time step (t) =1

The states at t=1(H_1,C_1) will fed as input states at t=2 and same is true the output at t=1(SELECT) is also fed as input at t=2.

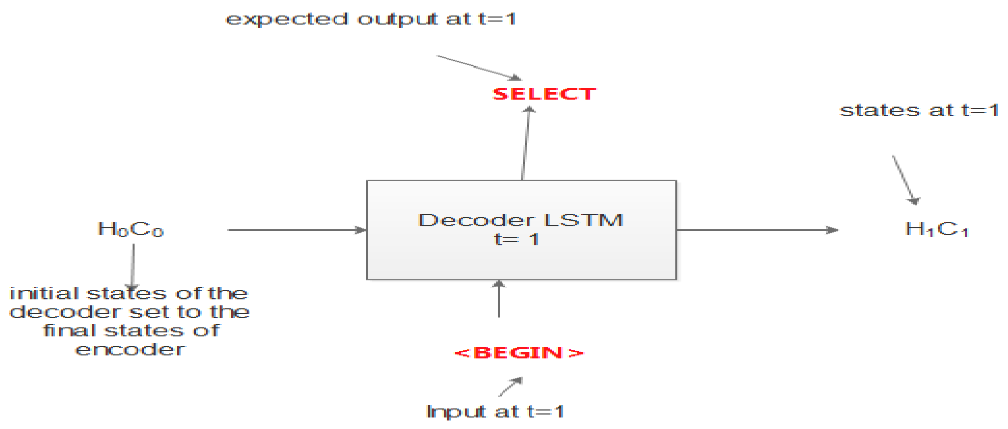


Figure 12: Decoder Lstm at t=1

At time step (t)=7

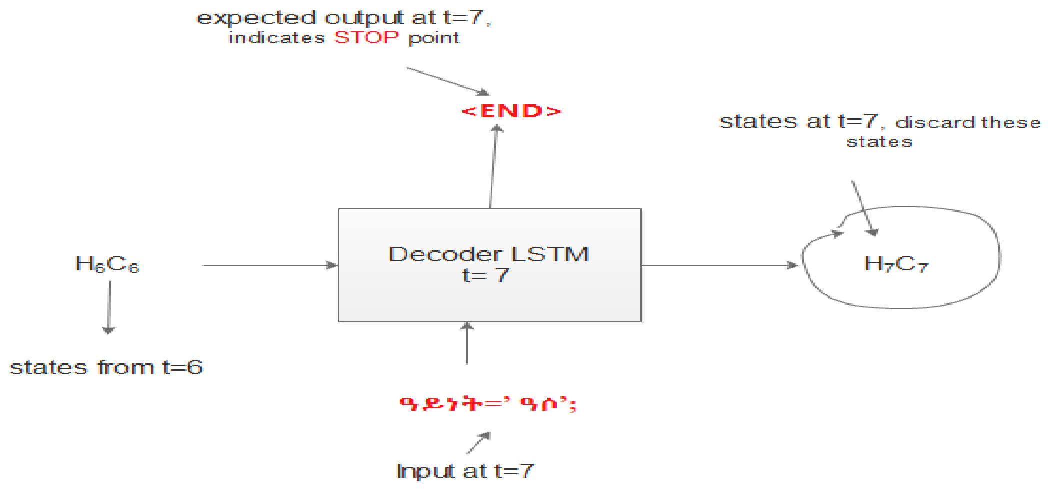

expected output at t=7, indicates STOP point
<END>
states at t=7, discard these states
H_6C_6
Decoder LSTM t= 7
H_7C_7
states from t=6
ዓይነት=' ዓሶ';
Input at t=7

Figure 13: Decoder Lstm at t=7

The input at t=7(**ዓይነት=' ዓሶ';**) is the predicted output from t=6.

The predicted output at t=7 which is (<**END**>) indicates a stopping point and the states at t=7(H_7,C_7) must discarded because we stop. Then finally it generates "**SELECT ሽም FROM ሕሙማት WHERE ዓይነት=' ዓሶ';" .**

5.4.6 Query executer: this used to execute the generated (predicted) SQL statements on the database. As described in section of the query types, the queries can be retrieval or manipulation of contents in the database. Therefore, this used to execute for all types of sql queries generated from Tigrinya plaintexts.

5.3.7 Result displayer and manipulator: this used to display the fetched result to the user based on the matching of the retrieval queries. If the queries are retrieval this used to display the actual contents. Also, if the queries are manipulation this makes changes in the database and nothing can display.

CHAPTER 6
Experiment

6.1 General pipelines

The following pipelines were followed to carry out the experiment:

First, a dataset was prepared with the help of linguistic professional's and health professionals. Second, the functional RNN model was defined, because this model is used in multiple inputs and multiple outputs. Third, the rmsprob (Root Mean Square Propagation) optimizer was selected because it adjusts the learning rate automatically and the learning rate of this optimizer is any real number greater than or equal to zero. This optimizer is faster and well known. Fourth, the categorical cross-entropy loss function was selected to estimate the loss (error) of the model during training. Categorical cross-entropy loss function is used to estimate the loss during multi-class classification. Finally, we trained and evaluated the model.

6.2 Database Design

A sample database was designed in healthcare system. In the below persistent diagram of the healthcare database we use English language to write the name of the table and attributes for the sake of clarity but their real meaning which are written in the database is described keys on table 8. The edraw max software does not accept the Tigrinya language plaintexts. That's why we use English to draw the persistent diagram.

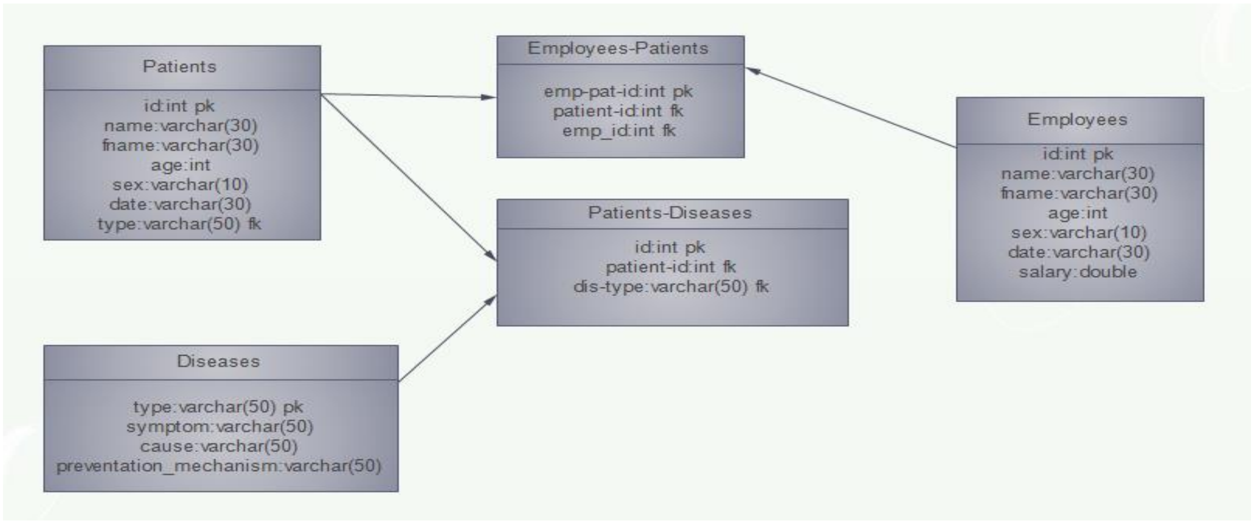


Figure 14: Persistent diagram

Table 4: keys

English	Tigrinya
Patients	ሕሙማት
Employees	ሰራሕተኛታት
Diseases	ሕማማት
Id	መለለይ
Name	ሽም
Fname	ሽምኣቦ
Age	ዕድመ
Sex	ፆታ
Salary	ደሞዝ
Type	ዓይነት
Symptom	ምልክት
Cause	ጠንቂ
Preventation-mechanism	መከላኸሊ_መንገዲ
Patient-id	መለለይ_ሕሙማት
Emp-id	መለለይ_ሰራሕተኛታት
Dis-type	ዓይነት_ሕማማት

The Tigrinya words are used to create the sample database as in many organizations use Tigrinya language plaintexts to create their database.

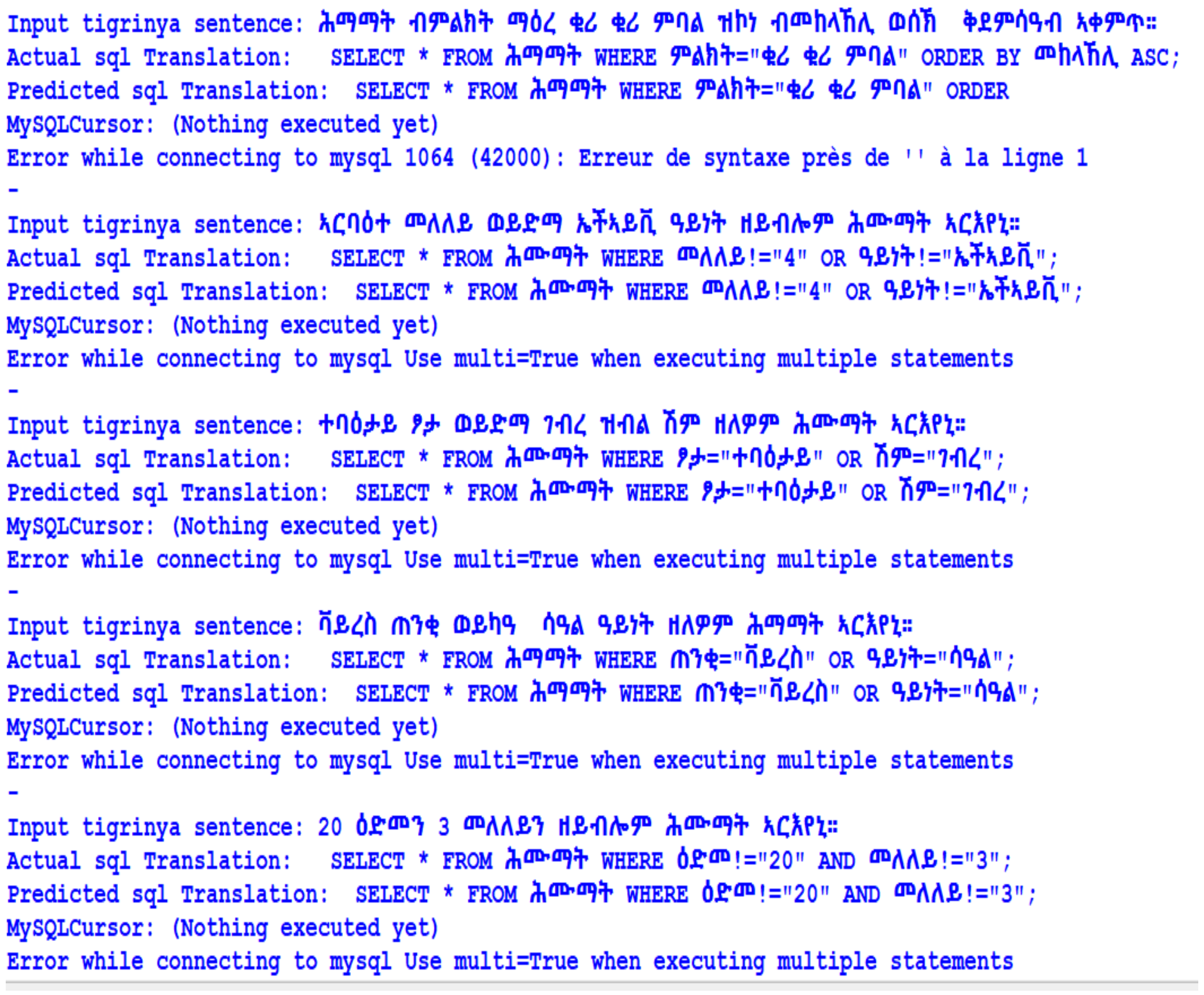
6.3 Queries

As discussed in chapter five different queries may be formulated. These queries may be retrieval or manipulation. The queries were grouped based on their query type's retrieval (select) and manipulation (update and delete). Also, the retrieval queries were grouped into different subgroups based on their complexities and others are sub grouped based on their specific commands. Table 4 shows the type of queries, their subgroups, Tigrinya queries of each subgroup and their corresponding SQL statements. This table presents a sample of Tigrinya queries for each subgroup and their corresponding SQL statements. This shows how the dataset is prepared and how much the language is rich in variations of words.

Table 5: Sample Queries

Query_Types	Subgroup of the Queries	Tigrinya Queries	Corresponding SQL statements
Select	Select queries	ሓበሬታ ሕሙማት ኣርእየኒ	select * from ሕሙማት;
		ሽምን ሽምኣቦን ሕሙማት ኣርእየኒ	select ሽም, ሽምኣቦ from ሕሙማት;
	Conditional queries	ምልክታት ሕማም ዓሶ ኣርእየኒ	select ምልክት from diseases where ዓይነት="ዓሶ";
		ዓሶ ወይድማ ቲቪ ዓይነት ሕሙማት ሽም ኣርእየኒ	select ሽም from ሕሙማት where ዓይነት="ዓሶ" or ዓይነት ="ቲቪ";
	Aggregation	በዝሒ ዓይነት ሕማማት ኣርእየኒ	select count(ዓይነት) from ሕማማት;
	Group/order by and asc/desc	ሕሙማት ብመለለይ ማዕረ 2ን ትሕቲኡን ብዕድመ ቅነሳ ጉጅለ ኣቀምጥ/ኣቀምጠለይ	select * from ሕሙማት where መለለይ<="2" group by ዕድመ desc;
	Join with it's type (inner,left,right) and union	ሽም ሕሙማትን ምልክታት ሕማማትን ጸምብር ኣርእየኒ	Select ሕሙማት.ሽም, ሕማማት.ምልክት from ሕሙማት join ሕማማት ON ሕሙማት.ዓይነት =ሕማማት.ዓይነት;
		ሽም ሕሙማትን ዓይነት ሕማማትን ሕብረት ኣርእየኒ	select ሽም from ሕሙማት UNION select ዓይነት from ሕማማት;
	Between/Not and In/Not	ኣብ መንጎ 2ን 10ን መለለይ ዘለዉ ሕሙማት ኣርእየኒ	select * from ሕሙማት where መለለይ BETWEEN "2" and "10";

Manipulation	**Update**	ባዓል 3 መለለይ ዝኮነ ሕሙም ሽሙ ብሃይለ ተክእ/ተክኣለይ	update ሕሙማት set ሽም="ሃይለ" where መለለይ="4";
		ባዓል 3 መለለይ ዝኮነ ሕሙም ሽሙ ብሃይለ ሽምኣብኡ ብግርማይ ተክእ	update ሕሙማት set ሽም="ሃይለ", ሽምኣቦ="ግርማይ" where መለለይ="3";
	Delete	ባዓል 4 መለለይ ዝኮነ ሕሙም ኣጥፍእ	delete from ሕሙማት where መለለይ="4";



```
Input tigrinya sentence: ሕማማት ብምልክት ማዕረ ቁሪ ቁሪ ምባል ዝኮነ ብመከላኸሊ ወሰኽ  ቅደምሰዓብ ኣቀምጥ።
Actual sql Translation:   SELECT * FROM ሕማማት WHERE ምልክት="ቁሪ ቁሪ ምባል" ORDER BY መከላኸሊ ASC;
Predicted sql Translation:  SELECT * FROM ሕማማት WHERE ምልክት="ቁሪ ቁሪ ምባል" ORDER
MySQLCursor: (Nothing executed yet)
Error while connecting to mysql 1064 (42000): Erreur de syntaxe près de '' à la ligne 1
-
Input tigrinya sentence: ኣርባዕተ መለለይ ወይድማ ኤችኣይቪ ዓይነት ዘይብሎም ሕሙማት ኣርእየኒ።
Actual sql Translation:   SELECT * FROM ሕሙማት WHERE መለለይ!="4" OR ዓይነት!="ኤችኣይቪ";
Predicted sql Translation:  SELECT * FROM ሕሙማት WHERE መለለይ!="4" OR ዓይነት!="ኤችኣይቪ";
MySQLCursor: (Nothing executed yet)
Error while connecting to mysql Use multi=True when executing multiple statements
-
Input tigrinya sentence: ተባዕታይ ፆታ ወይድማ ገብረ ዝብል ሽም ዘለዎም ሕሙማት ኣርእየኒ።
Actual sql Translation:   SELECT * FROM ሕሙማት WHERE ፆታ="ተባዕታይ" OR ሽም="ገብረ";
Predicted sql Translation:  SELECT * FROM ሕሙማት WHERE ፆታ="ተባዕታይ" OR ሽም="ገብረ";
MySQLCursor: (Nothing executed yet)
Error while connecting to mysql Use multi=True when executing multiple statements
-
Input tigrinya sentence: ቫይረስ ጠንቁ ወይካዓ  ሳዓል ዓይነት ዘለዎም ሕማማት ኣርእየኒ።
Actual sql Translation:   SELECT * FROM ሕማማት WHERE ጠንቁ="ቫይረስ" OR ዓይነት="ሳዓል";
Predicted sql Translation:  SELECT * FROM ሕማማት WHERE ጠንቁ="ቫይረስ" OR ዓይነት="ሳዓል";
MySQLCursor: (Nothing executed yet)
Error while connecting to mysql Use multi=True when executing multiple statements
-
Input tigrinya sentence: 20 ዕድመን 3 መለለይን ዘይብሎም ሕሙማት ኣርእየኒ።
Actual sql Translation:   SELECT * FROM ሕሙማት WHERE ዕድመ!="20" AND መለለይ!="3";
Predicted sql Translation:  SELECT * FROM ሕሙማት WHERE ዕድመ!="20" AND መለለይ!="3";
MySQLCursor: (Nothing executed yet)
Error while connecting to mysql Use multi=True when executing multiple statements
```

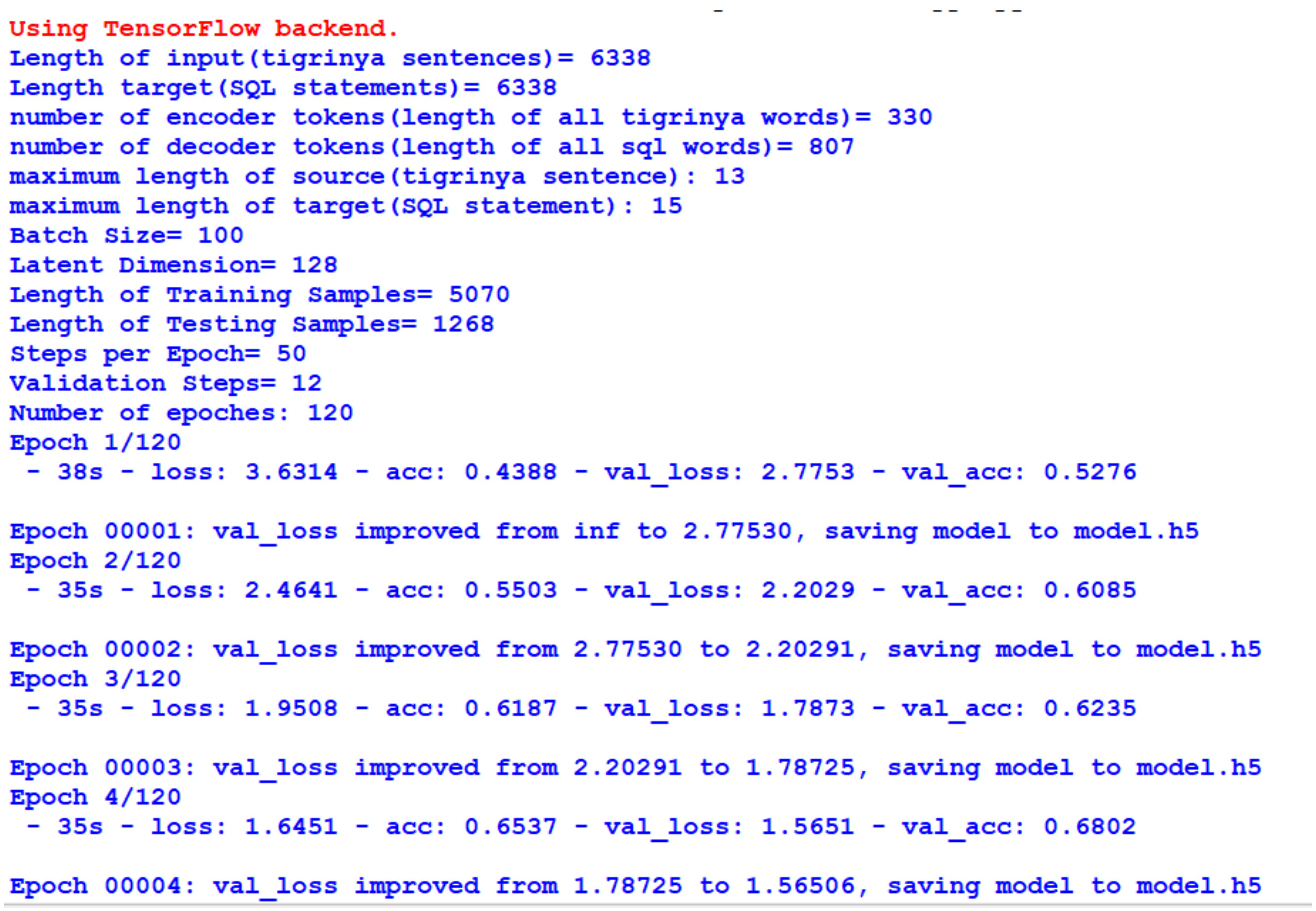
Figure 15: Output of the Prototype

6.4 Model Training

The corpus was prepared by separating the input Tigrinya sentences and the corresponding SQL statements using the word <BEGIN>. And then the file that contains the dataset was opened and separating the corpus into input and target data using the separator (i.e <BEGIN>). The maximum length of the input Tigrinya sentences has been calculated and the longest sentence has 13 tokens. And again the maximum length of the target SQL statements has been calculated and the longest SQL statement has 15 tokens. All the input and target sequences are sorted. And then the number of encoder tokens and the number of decoder tokens have been calculated i.e the length of all the Tigrinya words and the length of all SQL words which are 330 and 807 respectively.

After that the dataset was shuffled using the skylearn python library in order to mix it. After the data has been shuffled, the dataset splitted 80 %(5070) for training and 20%(1268) for testing data by using percentage split method of the skylearn python library. After that a batch size of 100 was provided to generate the batches of the training data using generate batch function. The encoder input data takes the index of the input Tigrinya words in an array of the batch size with the maximum length of the input Tigrinya sentences. This means that it takes the indexes of input Tigrinya words in a two dimensional array of 100 by 13. The decoder input data takes the indexes of the SQL words in an array of the batch size with the maximum length of the target (SQL) statements. This means that it takes the indexes of the SQL words in a two dimensional array of 100 by 15. The decoder target data takes the indexes of the SQL words in a an array of the batch size, maximum length of the SQL statement and the total number of decoder tokens(SQL words) and if the word is present it assigns one(1), if not it assigns zero(0). This means that it takes the indexes of the target tokens in a three dimensional array of [100, 15, 807]. This excludes the start token (i.e <BEGIN>) and offset by one time step. And then the function returns the input data (encoder input data and decoder input data) and the output data(decoder target data). This means that it returns the result of [[[100, 13],[100, 15]],[100, 15,807]].

A latent dimension of 128 and an epoch of 120 were provided. After that the encoder-decoder model was created and then a functional API model was defined with encoder-decoder data (the encoder input, decoder input and decoder output). The shape (dimension) of these three arrays is equal as the above arrays respectively. The encoder input takes in the place of the indexes of all input Tigrinya words the embedding matrix of 330 by 128. This means that the matrix of the total input Tigrinya words by the latent dimension. The decoder input and decoder output takes in the place of the indexes of all SQL words the embedding matrix of 807 by 128. This means that the matrix of the total SQL words by the latent dimension. And then the defined functional API model was compiled with rmsprob optimizer, categorical cross entropy loss functions and with the accuracy metrics of keras library. After that the created model was trained with the fit_generator() of the keras python library by calling the function. The model fitted with the parameters such as the batches of the training data, number of epochs, steps per epoch, batches of validation data, validation steps, callbacks and verbose 2. Finally, the model was evaluated with evaluate_generator() keras python library and scored an overall accuracy above 98.5% on the testing data. Figure 10 shows how the model can be trained and its evaluation results.



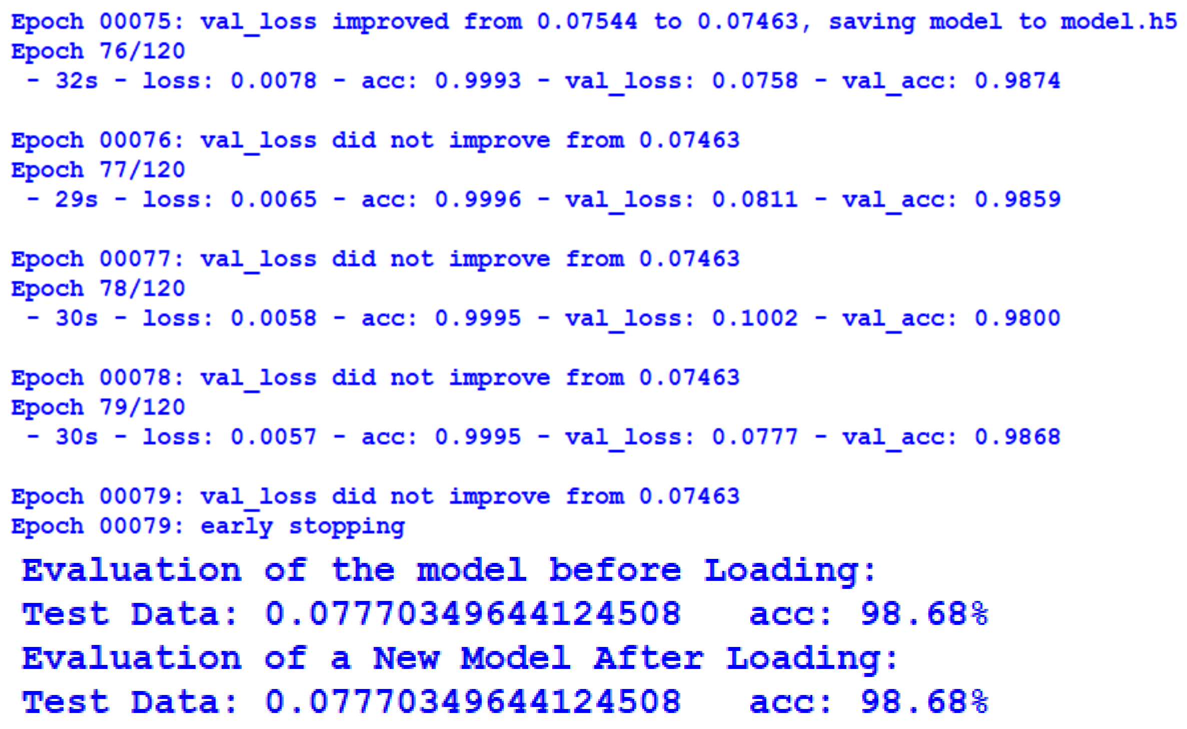
```
Using TensorFlow backend.
Length of input(tigrinya sentences)= 6338
Length target(SQL statements)= 6338
number of encoder tokens(length of all tigrinya words)= 330
number of decoder tokens(length of all sql words)= 807
maximum length of source(tigrinya sentence): 13
maximum length of target(SQL statement): 15
Batch Size= 100
Latent Dimension= 128
Length of Training Samples= 5070
Length of Testing Samples= 1268
Steps per Epoch= 50
Validation Steps= 12
Number of epoches: 120
Epoch 1/120
 - 38s - loss: 3.6314 - acc: 0.4388 - val_loss: 2.7753 - val_acc: 0.5276

Epoch 00001: val_loss improved from inf to 2.77530, saving model to model.h5
Epoch 2/120
 - 35s - loss: 2.4641 - acc: 0.5503 - val_loss: 2.2029 - val_acc: 0.6085

Epoch 00002: val_loss improved from 2.77530 to 2.20291, saving model to model.h5
Epoch 3/120
 - 35s - loss: 1.9508 - acc: 0.6187 - val_loss: 1.7873 - val_acc: 0.6235

Epoch 00003: val_loss improved from 2.20291 to 1.78725, saving model to model.h5
Epoch 4/120
 - 35s - loss: 1.6451 - acc: 0.6537 - val_loss: 1.5651 - val_acc: 0.6802

Epoch 00004: val_loss improved from 1.78725 to 1.56506, saving model to model.h5
```



```
Epoch 00075: val_loss improved from 0.07544 to 0.07463, saving model to model.h5
Epoch 76/120
 - 32s - loss: 0.0078 - acc: 0.9993 - val_loss: 0.0758 - val_acc: 0.9874

Epoch 00076: val_loss did not improve from 0.07463
Epoch 77/120
 - 29s - loss: 0.0065 - acc: 0.9996 - val_loss: 0.0811 - val_acc: 0.9859

Epoch 00077: val_loss did not improve from 0.07463
Epoch 78/120
 - 30s - loss: 0.0058 - acc: 0.9995 - val_loss: 0.1002 - val_acc: 0.9800

Epoch 00078: val_loss did not improve from 0.07463
Epoch 79/120
 - 30s - loss: 0.0057 - acc: 0.9995 - val_loss: 0.0777 - val_acc: 0.9868

Epoch 00079: val_loss did not improve from 0.07463
Epoch 00079: early stopping
 Evaluation of the model before Loading:
 Test Data: 0.07770349644124508   acc: 98.68%
 Evaluation of a New Model After Loading:
 Test Data: 0.07770349644124508   acc: 98.68%
```

Figure 16: Training and evaluation process

6.5 Evaluation Result

Keras uses batches. That's why metrics such as precision, recall and f-measure were removed from keras model [43]. Accuracy classification, which measured automatically, was used to evaluate the prototype. To train and evaluate the prototype 6,338 Tigrinya sentences (queries) were prepared by Tigrinya language professionals'. Those sentences have their own corresponding SQL statements. Therefore, the dataset that holds the Tigrinya plaintexts must provide as an input to the prototype. The prototype should understand the queries and provides a response based on the queries. The dataset has splitted to train and testing using percentage split of skylearn python library and it was provided 80% for training and 20% for testing (validating) the model. This means that 5070 sentences of the dataset are used for training the model and 1268 sentences are used to test the model. To test the model, the percentage split evaluation technique was used. Percentage split technique is good to test the model because the evaluation carried out by the test data that are not included during training. That's why the percentage split evaluation technique has been chosen.

During the training of the model the loss, accuracy, validation-loss and validation-accuracy are calculated automatically at each epoch (iteration).

6.5.1 Loss

The loss function is also called objective function which used to calculate the loss value [41]. The losses on the training and validating data are showed in figure 11. We use the categorical cross entropy loss function to calculate the loss value.

$$Loss = -\sum_{k=1}^{k} T_k Log y_k$$

In the above equation T_k represents the actual target data (SQL statement) and y_k represented the predicted probability of the target. This is calculated after the softmax activation function transformed the output to a vector. The minus (-) sign is used to improve the performance of the model.

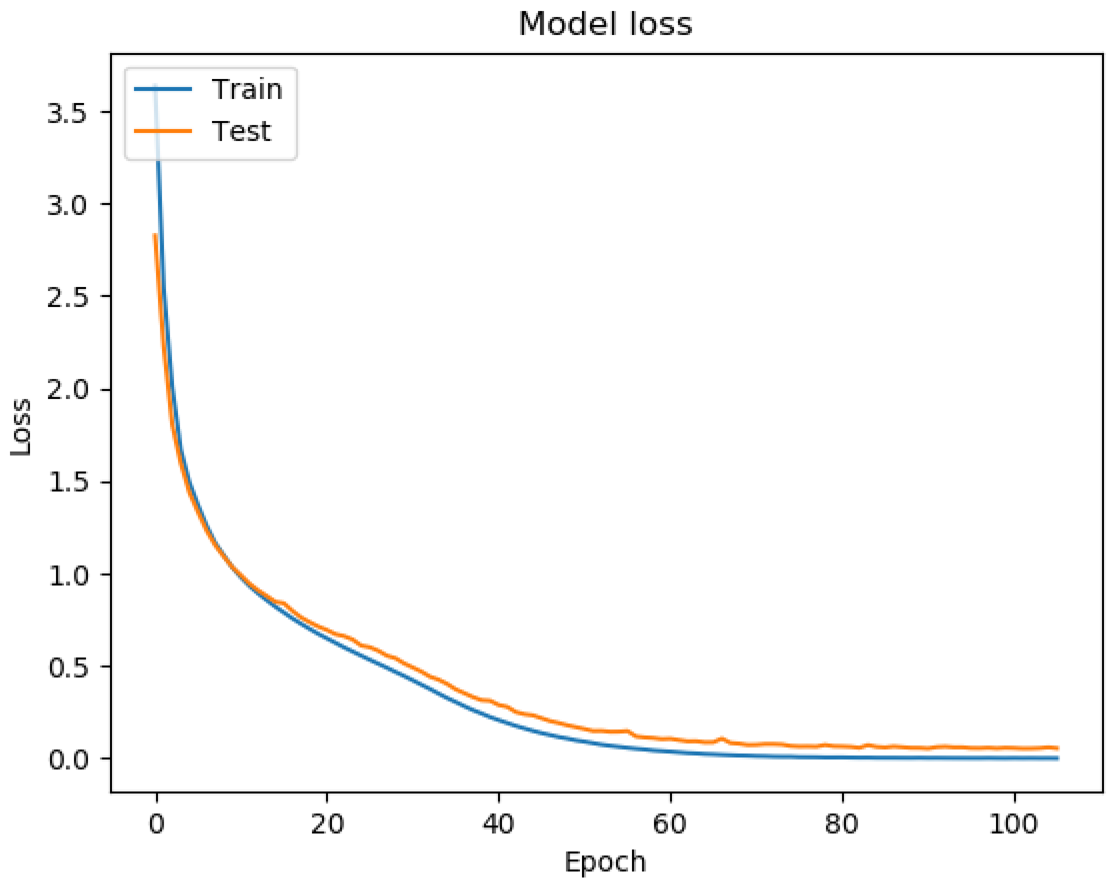


Figure 17: loss model

Figure 11 shows as the number of epoch is increasing, the loss value converges to zero. As the first few epochs both lines are on the same line. This means both have same loss value. And in the remaining many epochs the line of the validation data showed above the training data. This shows the loss value of the validation data is higher than the loss value of the training data.

6.5.2 Accuracy

The accuracy of the prototype is calculated by using the keras accuracy metrics that uses the following formula:

$$Accuracy = \frac{amount\ of\ correct\ predicted\ SQL}{Total\ predicted\ SQL\ amount} * 100\%$$

The accuracy on the training and validation data is showed in figure 12. This shows how the translation from the input Tigrinya sentences to their corresponding SQL statements is accurate. If the input Tigrinya sentence is accurately translated to SQL statement, the DBMS can execute the SQL statement on the database. So, the accuracy of the Tigrinya language interface to database prototype can depend on the accuracy of the translator model.

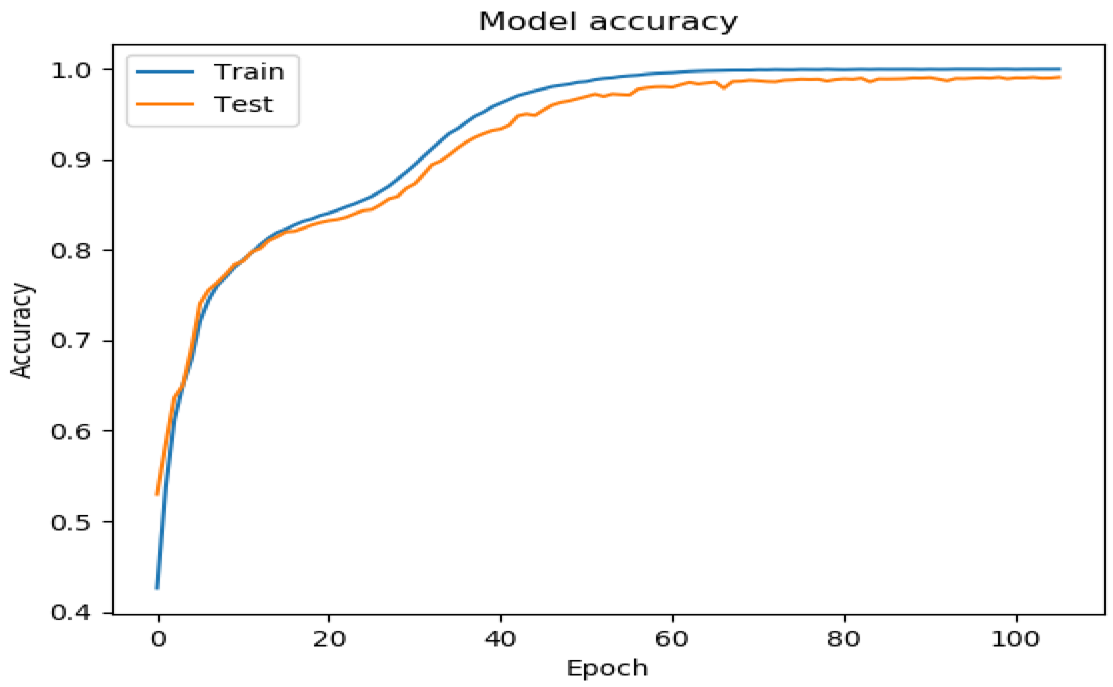


Figure 18: Accuracy model

Figure 12 shows as the number of epoch is increasing, the accuracy value converges to one (increasing). As the first few epochs both lines are on the same line. This means both have same small accuracy value. And in the remaining many epochs the line of the validation data showed below the line of the training data. This shows the accuracy value of the validation data is smaller than the accuracy value of the training data. An overall accuracy was scored above **98.5%**. This is remarkable compared with the accuracy of the related works on other natural language interface to database prototypes which were developed by different researchers for different local languages. However, only the accuracy score was compared nothing else as the languages have their own syntaxes and semantics.

6.6 Discussion

The Tigrinya language interfaces to database prototype enabled users to write a query that can be retrieval and manipulation of the contents of the database. As described above a dataset of 6338 sentences was prepared to train and test the model. To translate the Tigrinya sentences to the corresponding SQL statements, neural machine translation approach was used. This approach wants a large set of data and a long period of time to train. If the dataset is very large it may take long time to train. However, the advantage is once trained the model we can use that to predict a new data which means no need to train again. Again, the model is used to know which word is less important and which word is more important. This means throwing stopwords of the input Tigrinya language don't expected. Here also, word embedding technique was used to represent the similarity of words. It gives similar value for semantically similar words. Therefore, this technique helped us to build a low dimensionality model that can preserve previous states. When the model is trained in large dataset, it can have a good accuracy on predicting the SQL statements. However, still there are no prepared Tigrinya language datasets for NLIDB. So, the dataset was prepared from scratch with the help of Tigrinya language professionals' to train and test model. Therefore, the accuracy scored is good in the translator model as the dataset is small. The accuracy of Tigrinya language interface to database prototype is extremely dependent on the accuracy of the translator model. So, it is possible to conclude that the accuracy of the translator is equal with the accuracy of the Tigrinya language interface to database prototype. The executer can execute the predicted (generated) SQL statement and either to display an output to the user or to make changes to the database based on the queries. Therefore, to accomplish that a new database has been designed and tested. The prototype is evaluated using keras accuracy metric automatically and an accuracy of above 98.5% has been scored. This showed the neural machine translation approach is a good approach on the translation of Tigrinya sentence into their corresponding SQL sequence by sequence. This accuracy showed the developed prototype is good when we compare with the related works. If the model will be trained with large dataset, it will have better performance. This means, the performance of the model is depend on the size of the dataset. And again, 128 latent dimensions and 120 numbers of epochs have been given for the model as there is no standard, randomly given by try and error. As both the number of latent dimension and number of epochs are increasing and decreasing together the performance of the model gets degraded. When the number of latent dimension and the number of epochs are constant and increased respectively, the performance of the model increases. Therefore, the performance of the model is depending on the size of the dataset, number of latent dimension and the number of epochs. To prevent over and under fitting, early stop and model check point has been used. A val-loss has been taken as a monitor and on its minimum mode. A good or accurate model has an accuracy of 100% on both training and testing data. Therefore, the developed model has accuracy nearest to that. The prototype enables Tigrinya users to retrieve and manipulate the contents of the database without the knowledge of the SQL. This means that using the Tigrinya language interface. The prototype has been taken the input Tigrinya sentence and it has been translated into the SQL and further the SQL executed in the database. During this time the native Tigrinya users did not have any knowledge about the SQL and the database design. Instead, they have been expected to write Tigrinya text and the text can internally processed by the system. This means it doesn't mean that, to use Tigrinya texts instead of SQL rather the input Tigrinya can be translated to SQL and further the SQL can execute on the database. The developed prototype really achieved that. To conclude, that the developed prototype enables users to retrieve and manipulate the contents of the database without the knowledge of SQL. And additionally, the neural machine translation approach is appropriate on the translation of the input Tigrinya sentence into SQL.

6.7 Contribution of the study

The general contribution of this study was:

- ❖ To enable Tigrinya users to retrieve and manipulate the contents of the database using Tigrinya plaintexts'.

The specific contribution of this research study was:

- ❖ This work contributes a general architecture for Tigrinya language interface to database (TLIDB) that developed using neural machine translation approach. Therefore, anyone can use this architecture to work on TLIDB for different domains with different database schema and the approach makes the natural language interface to database efficient and robust.

- And another, specific contribution is a prepared dataset that has Tigrinya input sentences with their corresponding SQL statements that would use for Tigrinya language interface to database. Therefore, anyone can use this dataset to work on Tigrinya language interface to database.
- After the prediction has been carried out a rule has been bootstrapped to differentiate the queries either it is retrieval or manipulation

CHAPTER 7
Conclusion and Recommendation

7.1 Conclusion

Now a day, there are a number of databases that are developed in Tigrinya language contents. However, accessing of them using Tigrinya natural language plaintext is not easy for ordinary users. Therefore, this work was proposed Tigrinya language interface to database (TLIDB).

To accomplish this work different literature were reviewed in detail to understand the methods and techniques that used for developing natural language interface to database. A dataset was prepared and this contained Tigrinya language sentences with their corresponding SQL statements. And then TLIDB architecture was designed and developed. For the translation of Tigrinya plaintexts into SQL statements, a model was created and trained. Finally, the developed prototype was evaluated on three tables of healthcare domain.

The prototype accepts queries (dataset) and cleans the input Tigrinya sentences. The dataset has Tigrinya sentences with corresponding SQL statements. And then the dataset is splitting into two. Namely, training and testing data. 80% of the dataset is used for training and 20% for testing. Then, the model trained with 80% of the data and tested on the 20% of the dataset. For the testing dataset the model predicts the SQL statements for the input Tigrinya texts. And then, the predicted SQL statements executed on the database.

If the queries are retrieval it provides the information's from the database in a structured form. If the queries are manipulation, changes are made in the database. Finally, the model was evaluated on how it is accurate in the translation or mapping of the input Tigrinya sentences into the corresponding SQL statements. The accuracy and loss of both the training and testing data was showed in graph. the queries were grouped into two based on their query type as retrieval and manipulation. The model was evaluated and an overall accuracy above 98.5% was scored.

Therefore, if this system is fully trained with very large dataset it will be used as an effective querying interface between Tigrinya language and database. So, users can simply retrieve the contents of the database; manipulate the contents of the database using their natural language plaintexts. This does not need learning of the formal language (i.e SQL) syntaxes and semantics. Also, users do not need training as in form based systems. Again, it is very important for some questions such as negation and conjunctions. Because representing the negations and conjunctions using Tigrinya language is very easy than the SQL statements and also it is very difficult to represent in form based system.

7.2 Recommendation

Because of the limitations of time and resources, the features that were not included in our works either that would improve, the accuracy or the functionality of the system were suggested below as future works:

- ❖ The prototype performed some SQL commands such as the retrieval and manipulation and then it should include all the SQL commands to fully include the functionality of the database. For example it should include the creation of tables, altering of tables and inserting contents into database.
- ❖ The prototype also included simple nested SQL statements and then it should include more complex nested SQL statements. Since, we included a union query that uses the simple nested select query. So, it should include more complex queries.
- ❖ The prototype did not include Tigrinya stemmer, Tigrinya spelling checker, tagger, parser and etc. And then, if it includes Tigrinya stemmer, Tigrinya spelling checker, tagger, parser and etc the machine can be improved the understanding of Tigrinya language.
- ❖ As the prototype was developed using neural machine translation approach, it needs a large dataset. So, the dataset used is too small and it should increase the dataset to have the prototype better performance.
- ❖ The prototype has been developed for a single domain and it does not include auto completion. And then making domain independent and including the auto completion can improve the performance of the model.
- ❖ Applying attention based recurrent neural network to NLIDB would be improved the performance of the prototype. As it was developed, with capability to validate the model using long sequence that are not included during training.

Therefore, the above stated ideas can be considered as a future work.

REFERENCES

1. Gobinda G. Chowdhury, Natural Language Processing, Dept. of Computer and Information Sciences, University of Strathclyde, Glasgow G1 1XH, UK
2. Tihitina Petros, Modeling and Designing Amharic Query System to Bilingual (English-Amharic) Databases, department of computer science, Addisababa Univeristy, Ethiopia, April 2014
3. I. androutso poulos et al, Natural language interfaces to databases – an introduction. Natural Language Engineering, 1(1):29-81, 1995
4. A.-M. Popescu et al, towards a theory of natural language interfaces to databases, In IUI, pages 149-157, 2003
5. Y. Li et al, Nalix: an interactive natural language interface for querying xml, In sigmod Conference, pages 900-902, 2005
6. Khaled Nasser, An Arabic Natural Language Interface System for a Database of the Holy Quran ,Computer Science Department, Umm AlQura University, (IJARAI) International Journal of Advanced Research in Artificial Intelligence, Vol. 4, No.7, 2015
7. Ashish Kumar, design and implementation of Hindi language interface to database, international journal of current engineering and scientific research (ijcesr), ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-4, ISSUE-10, 2017
8. Smegnew Asemie, Possibility of Amharic Query Processing in Database using Natural Language Interface, International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 6 Issue 05, May - 2017
9. B.Sujatha et al, A Study of the Various Architectures for Natural Language Interface to DBs, International Journal of Computer Science and Network (IJCSN),Volume 1, Issue 4, www.ijcsn.org, ISSN 2277-5420, Hyderabad,India, August 2012
10. Ana-Maria Popescu et al, "Towards a Theory of Natural Language Interfaces to Database", University of Washington, Computer Science Seattle, WA 98195, USA
11. Deepshikha1 et al., International Journal of Advanced Research in Computer Science and Software Engineering 5(4), April- 2015, pp. 890-895
12. Joginder et al., Natural Language Interface to Database-An Introduction, Imperial Journal of Interdisciplinary Research (IJIR), Vol.2, Issue-1 , 2016
13. Suket et al, Punjabi language interface to database: a brief review, Punjab Technical University, June 2013

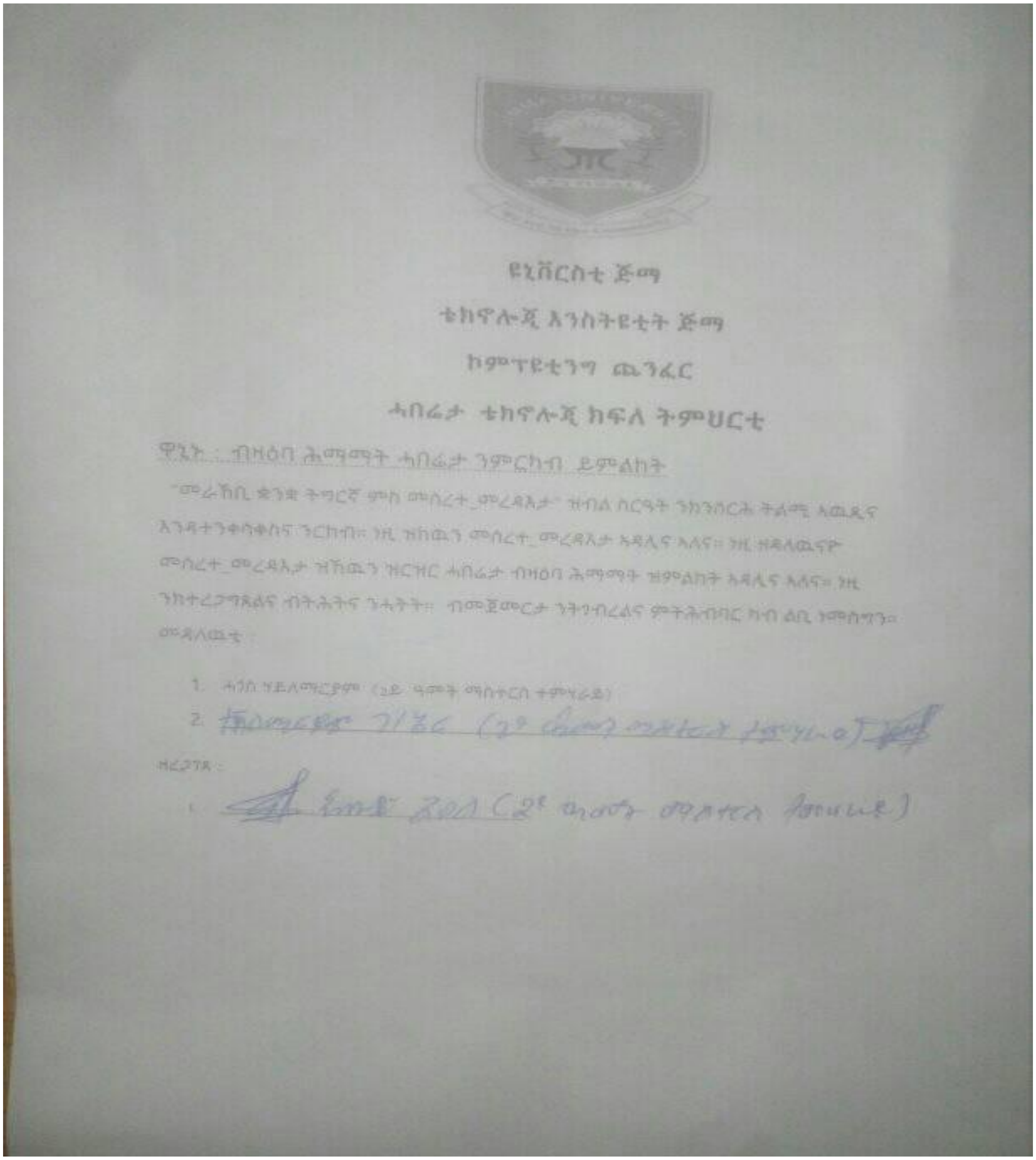
14. Ashish Kumar et al, Natural Language Interface to Databases: Development Techniques/ Elixir Comp. Sci. & Engg. 58 (2013) 14724-14727
15. Mrs. Neelu Nihalani et al, Natural language Interface to Database using Semantic Matching, International Journal of Computer Application, Vol. 31, no.11, Oct. 2011 ISSN: 0975 – 8887.
16. Mrs. Neelu Nihalani et al, Natural language Interface for Database: A Brief review, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 2, March 2011 ISSN (Online): 1694-0814.
17. Charniak E. 1993, Statistical Language Learning, MIT Press.
18. Church K., Mercer R., Introduction to the special issue on computational linguistics using large corpora, Computational Linguistics, 19 (1), pp. 1-24, 1993
19. Abrahams P. et al. The LISP 2 Programming Language and System, in proceedings of FJCC, No. 29, USA, 1966, pp. 661– 676.
20. Woods, W., An experimental parsing system for transition network grammars. In Natural language Processing, R. Rustin, Ed., Algorithmic Press, New York. 2, USA, Pages 105 – 147, 1973
21. Suket et al, Punjabi language interface to database: a brief review, Punjab Technical University, June 2013
22. G. Hendrix, E. et al, Developing a natural language interface to complex data, ACM Transactions on Database Systems, Volume 3, No. 2, USA, 1978, pp. 105 – 147.
23. D.L. Waltz., An English Language Question Answering System for a Large Relational Database, Communications of the ACM, 21(7): (July 1978), pp 526– 539
24. Amble et al, A Natural Language Bus Route Oracle. 6th Applied Natural Language Processing Conference, Seattle, Washington, USA
25. B.J. Grosz et al, TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces, Artificial Intelligence, 32: (1987), pp 173–243
26. B.J. Grosz, TEAM: A Transportable Natural-Language Interface System, In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, (1983), pp 39–45
27. Ana-Maria Popescu et al, Modern Natural Language Interfaces to Databases: Composing Statistical Parsing with Semantic Tractability, COLING (2004).

28. M. R. Joshi et al, Algorithms to Improve Performance of Natural Language Interface, International Journal of Computer Science and Applications, vol. 5, No. 2, pp. 52-68, 2010.
29. Jen-Yuan Yea et al, Visitor Service Collection and Information Management National Museum of Natural Science Taichung 40453, Dept. Of Operation, Taiwan one lab Technology ltd. Taipei city 11494, Taiwan 2015.
30. ዳንኤል ተክሉ (ዶክተር), ዘመናዊ ስዋስው ቋንቋ ትግርኛ , 5ይ ሕታም
31. Teffera, Tsehaye (1979). Reference Grammar of Tigrigna. Washington, Dc, USA: Georgetown University.
32. Tewelde, T. (2002). A Modern Grammar of Tigrigna. Savonarola, Roma: Tipografia U. Dettivia G.
33. Suket et al, Punjabi language interface to database: a brief review, Punjab Technical University, June 2013
34. Niket Choudhary, Sonal Gore, Pattern based approach for Natural Language Interface to Database, Journal of Engineering Research and Application ISSN : 2248-9622, Vol. 5, Issue 1(Part 2), January 2015, pp.105-110
35. Kibrom Haftu, Tigrigna question answering system for factoid questions, Thesis, Computer Science, AAU, 17 JUNE 2016.
36. Shumet Walelign, Designing Natural Language Interface to Database for Afaan Oromoo, school of computing, Jimma institute of technology, Jimma University, November, 2018
37. https://towardsdatascience.com/word-level-english-to-marthi-neural-machine-translation-using-seq2seq-encoder-decoder-lstm-model-1a913f2dc4a7,8/25/2019
38. Ruchit Agrawal, towards efficient Neural Machine Translation for Indian Languages, International Institute of Information Technology, Hyderabad (Deemed to be University) Hyderabad - 500 032, INDIA, October 2017
39. https://www.quora.com/What-is-word-embedding-in-deep-learning, Jaron Collis, AI PhD, Upvoted by Trideep Rath, MS Master of Science in Computer Science & Natural Language Processing, Arizona State University, Updated Apr 20, 2017 ,9/18/2019,1:40PM
40. https://www.kdnuggets.com/2018/11/keras-long-short-term-memory-lstm-model-predict-stock-prices.html,9/18/2019,2:58PM
41. https://keras.io/losses/,9/19/2019,2:01PM

42. Kishan Athrey, Tutorial on Keras, cap 6412 - advanced computer vision, spring 2018
43. https://datascience.stackexchange.com/questions/45165/how-to-get-accuracy-f1-precision-and-recall-for-a-keras-model,9/24/2019, 10:14 AM
44. Vincent cheung, kevin cannons, an introduction to neural networks, electrical and computer engineering, university of Manitoba, Canada
45. https://brilliant.org/wiki/recurrent-neural-network,10/11/2019,6:01PM
46. https://medium.com/@divyanshu132/lstm-and-its-equations-5ee9246d04af,10/12/2019,2:40PM
47. https://www.onlinemathlearning.com/Conditional Probability.html,12/28/2019,3:24PM
48. https://stats.stackexchange.com, 12/28/2019,4:47PM
49. Guesh Amiha Birhanu, Automatic Text Summarizer for Tigrinya Language, school of information science, college of natural science, AAU, Ethiopia, February 2017.
50. https://www.tutorialspoint.com/natural_language_processing/natural_language_processing_syntactic_analysis.htm,1/17/2020,10:45AM
51. Lee M. Christensen, ONYX: A System for the Semantic Analysis of Clinical Text, Department of Biomedical Informatics, University of Pittsburgh and Utah, Pittsburgh, PA 15214, USA.

APPENDIX:

Appendix A: Approval Paper one



ዩኒቨርስቲ ጅማ

ቴክኖሎጂ እንስትዩቲት ጅማ

ኮምፒዩቲንግ ጨንፈር

ሓበሬታ ቴክኖሎጂ ክፍሊ ትምህርቲ

ዋኒን ፡ ብዛዕባ ሕማማት ሓበሬታ ንምርካብ ይምልከት

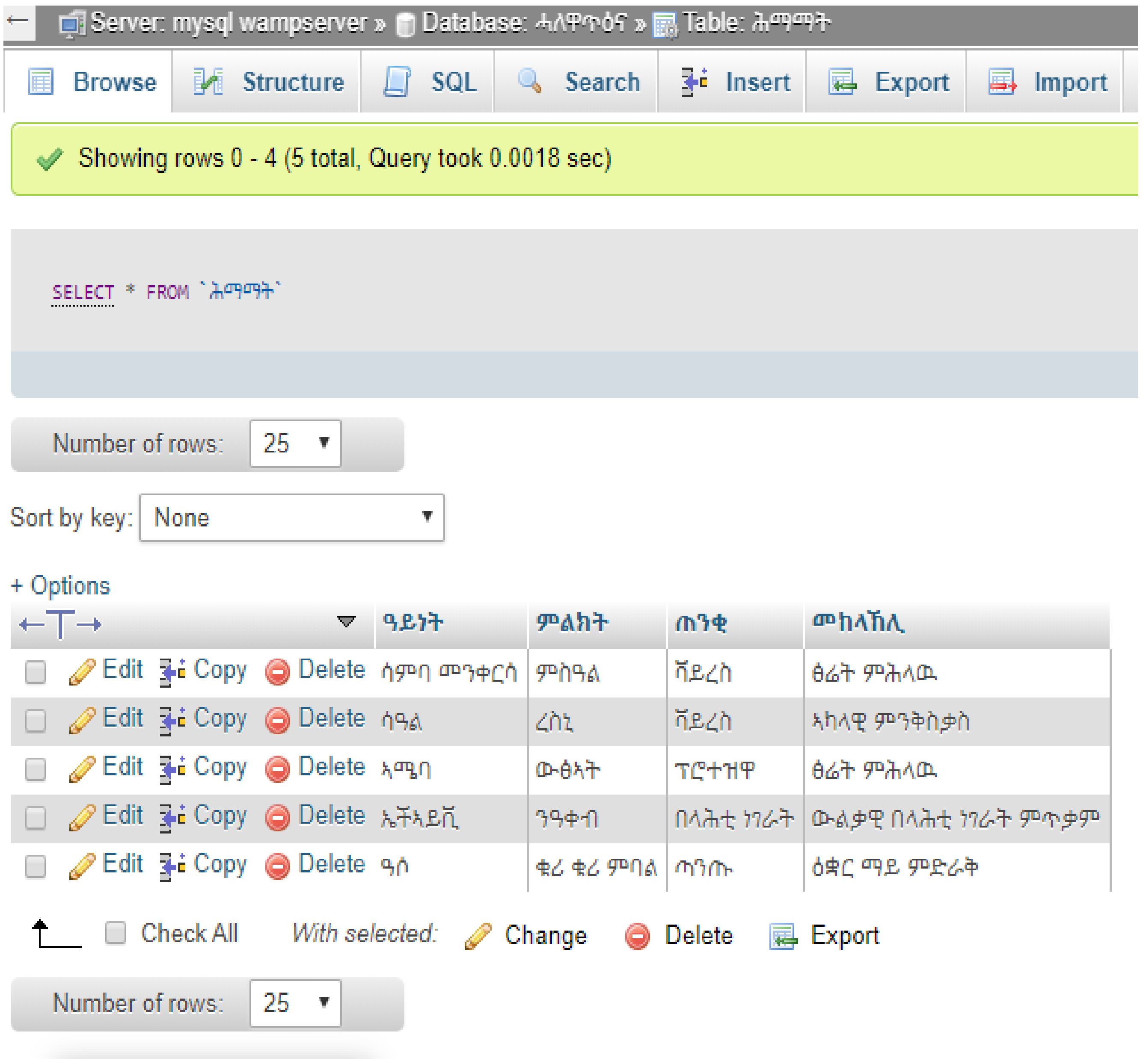
"መራኸቢ ቋንቋ ትግርኛ ምስ መሰረተ_ምረዳእታ" ዝብል ስርዓት ንክንሰርሕ ትልሚ ኣዉጺና እንዳተንቀሳቐስና ንርከብ። ነዚ ዝኸዉን መሰረተ_መረዳእታ ኣዳሊና ኣለና። ነዚ ዘዳለዉናዮ መሰረተ_መረዳእታ ዝኸዉን ዝርዝር ሓበሬታ ብዛዕባ ሕማማት ዘምልክት ኣዳሊና ኣለና። ነዚ ንክተረጋግጹልና ብትሕትና ንሓትት። ብመጀመርታ ንትገብረልና ምትሕብባር ካብ ልቢ ነመስግን።
መጻለዉቲ

1. ሓጎስ ሃይለማርያም (2ይ ዓመት ማስተርስ ተምሃራይ)
2. [illegible] ገ/ሄር (2ይ ዓመት ማስተርስ ተምሃራይ) [illegible]

ዘረጋገጸ ፡

1. [illegible] [illegible] (2ይ ዓመት ማስተርስ [illegible])

Appendix B: Sample DB contents of Diseases table



Showing rows 0 - 4 (5 total, Query took 0.0018 sec)

```
SELECT * FROM `ሕማማት`
```

Number of rows: 25

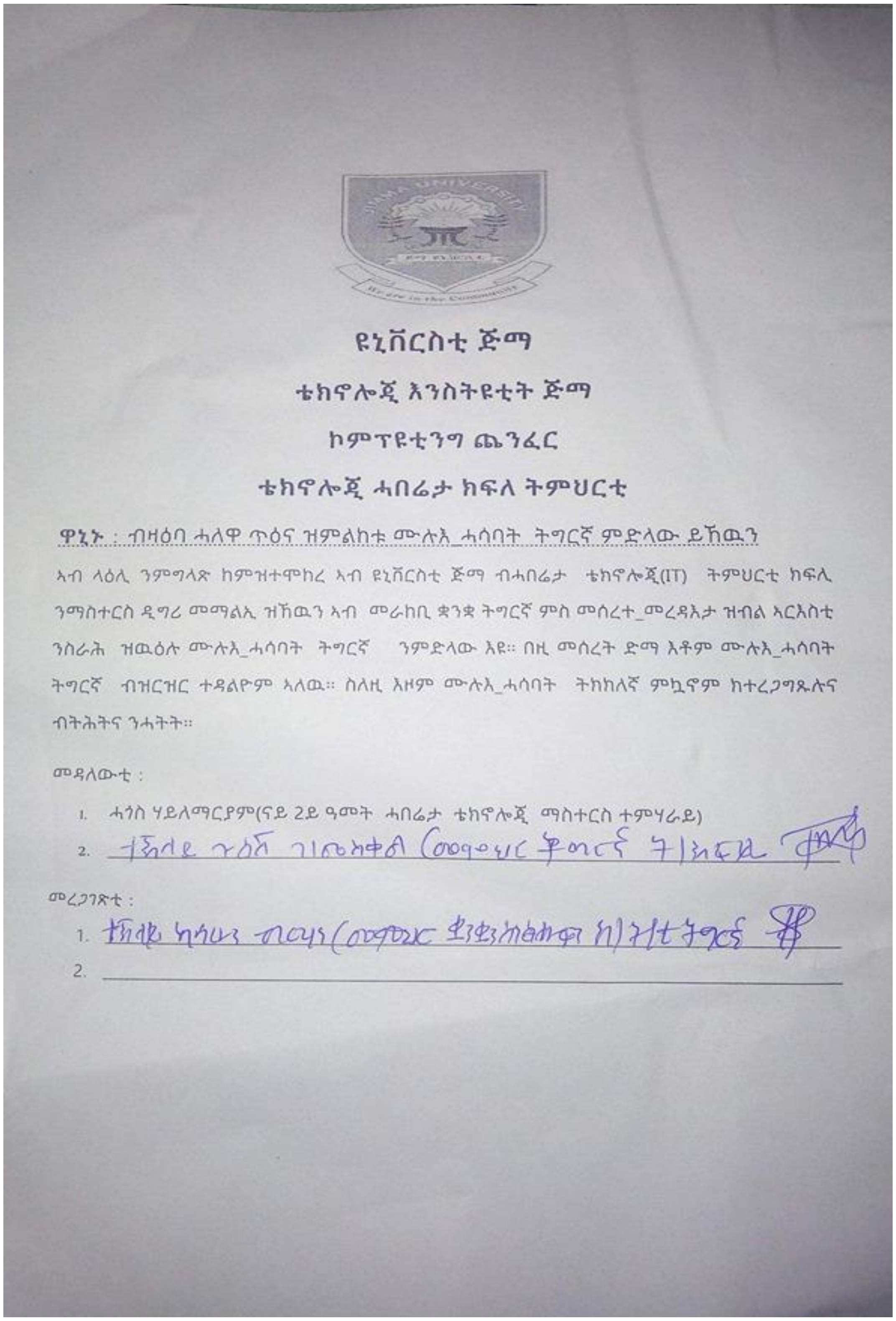
Sort by key: None

+ Options

	ዓይነት	ምልክት	ጠንቂ	መከላኸሊ
Edit Copy Delete	ሳምባ መንቀርሳ	ምስዓል	ቫይረስ	ፅሬት ምሕላዉ
Edit Copy Delete	ሰዓል	ረስኒ	ቫይረስ	ኣካላዊ ምንቅስቓስ
Edit Copy Delete	ኣሜባ	ውፅኣት	ፕሮተዝዋ	ፅሬት ምሕላዉ
Edit Copy Delete	ኤችኣይቪ	ንዓቀብ	በላሕቲ ነገራት	ውልቃዊ በላሕቲ ነገራት ምጥቃም
Edit Copy Delete	ዓሶ	ቁረ ቁረ ምባል	ጣንጡ	ዕቋር ማይ ምድራቅ

Number of rows: 25

Appendix C: Approval Paper two



ዩኒቨርስቲ ጅማ

ቴክኖሎጂ እንስትዩቲት ጅማ

ኮምፒዩቲንግ ጨንፈር

ቴክኖሎጂ ሓበሬታ ክፍለ ትምህርቲ

ዋኒኑ : ብዛዕባ ሓለዋ ጥዕና ዝምልከቱ ሙሉእ_ሓሳባት ትግርኛ ምድላው ይኸዉን

ኣብ ላዕሊ ንምግላጽ ከምዝተሞከረ ኣብ ዩኒቨርስቲ ጅማ ብሓበሬታ ቴክኖሎጂ(IT) ትምህርቲ ክፍሊ ንማስተርስ ዲግሪ መማልኢ ዝኸዉን ኣብ መራከቢ ቋንቋ ትግርኛ ምስ መሰረተ_መረዳእታ ዝብል ኣርእስቲ ንስራሕ ዝዉዕሉ ሙሉእ_ሓሳባት ትግርኛ ንምድላው እዩ። በዚ መሰረት ድማ እቶም ሙሉእ_ሓሳባት ትግርኛ ብዝርዝር ተዳልዮም ኣለዉ። ስለዚ እዞም ሙሉእ_ሓሳባት ትክክለኛ ምኳኖም ከተረጋግጹሉና ብትሕትና ንሓትት።

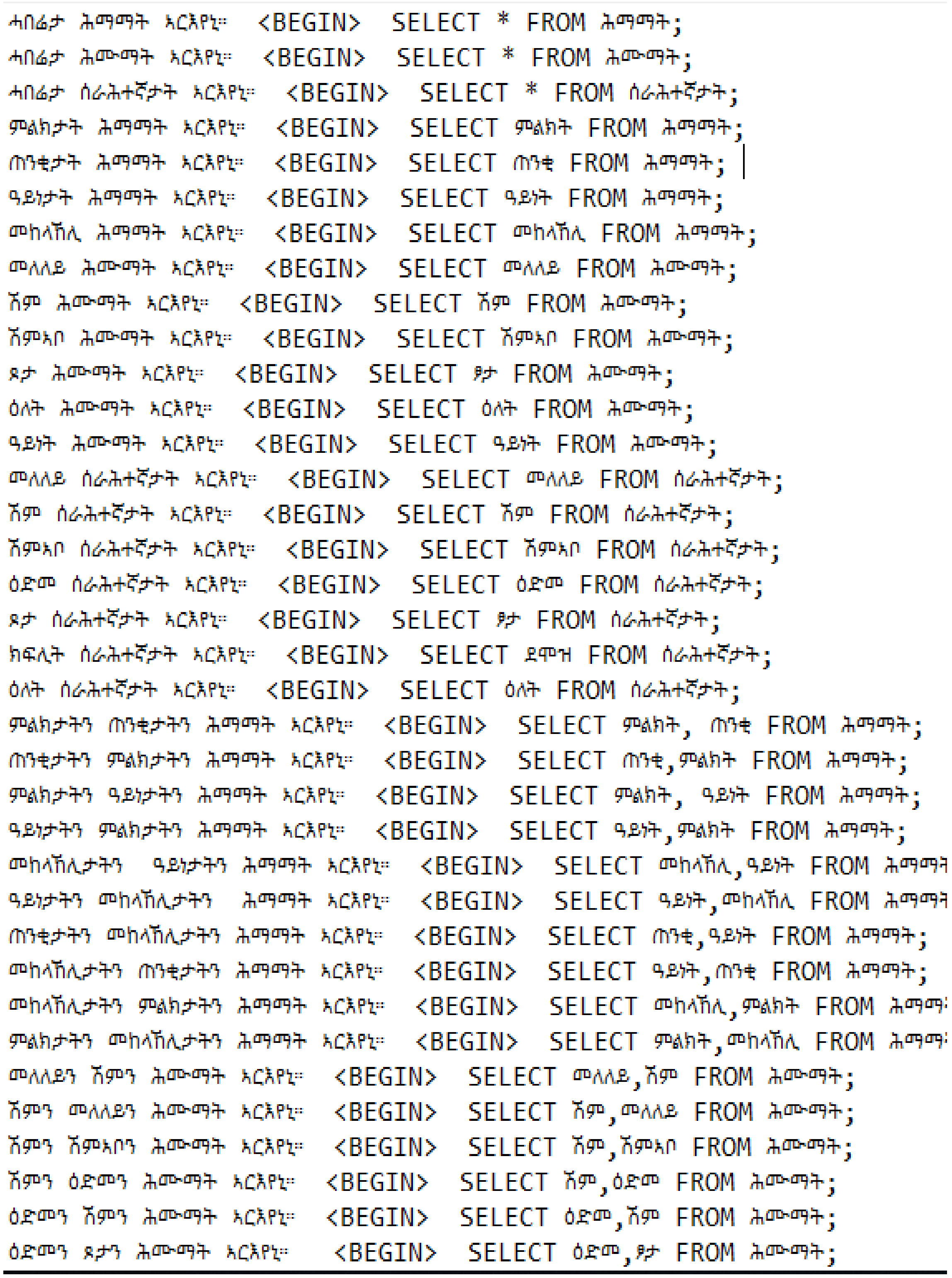
መዳለውቲ :

1. ሓጎስ ሃይለማርያም(ናይ 2ይ ዓመት ሓበሬታ ቴክኖሎጂ ማስተርስ ተምሃራይ)
2. ተኽላይ ጉሰስ ገብረእቑብ (መምህር [illegible]) [illegible]

መረጋገጽቲ :

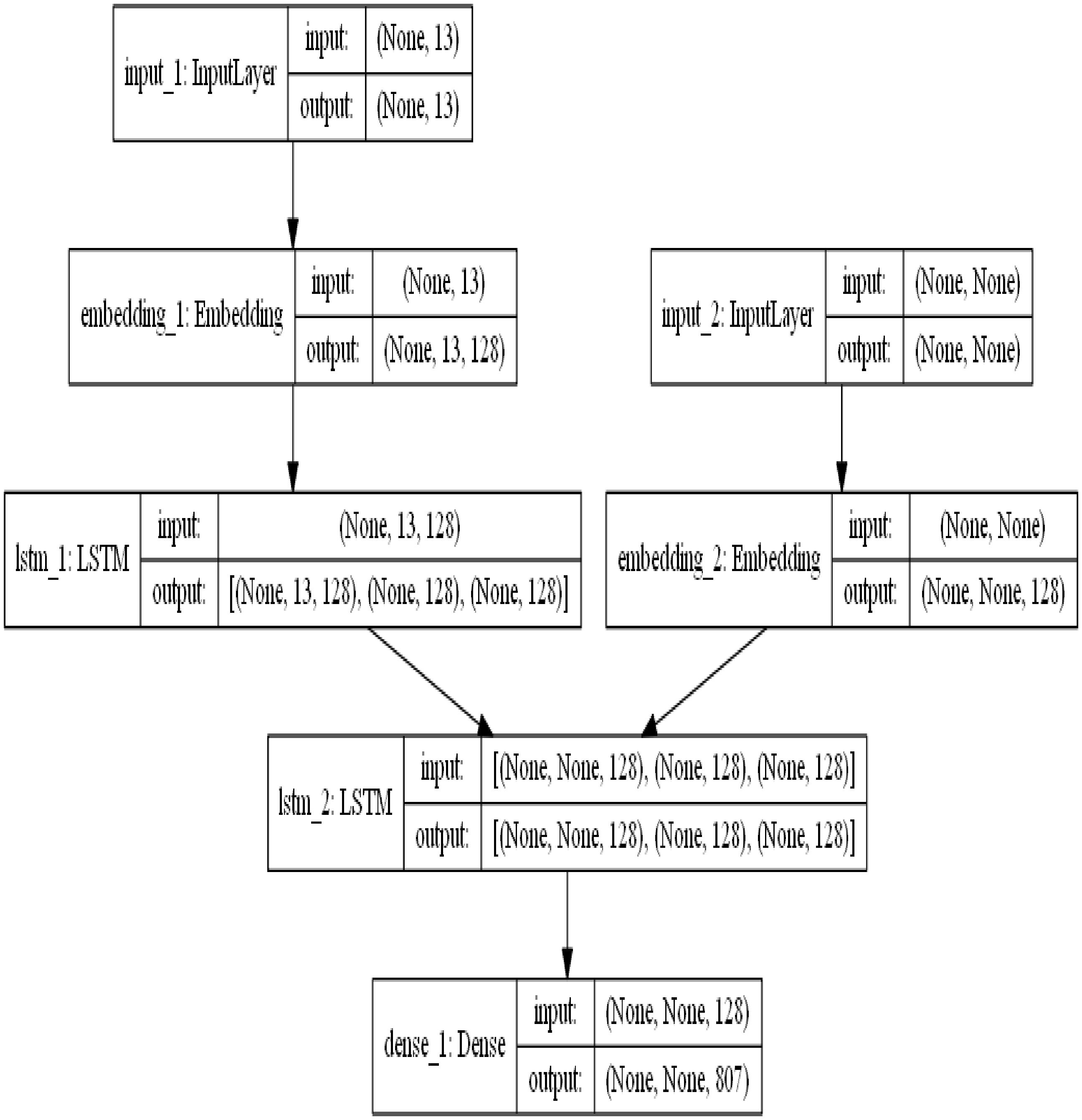
1. ተኽላይ ካሳሁን [illegible] (መምህር [illegible]) [illegible]
2.

Appendix D: Sample Dataset

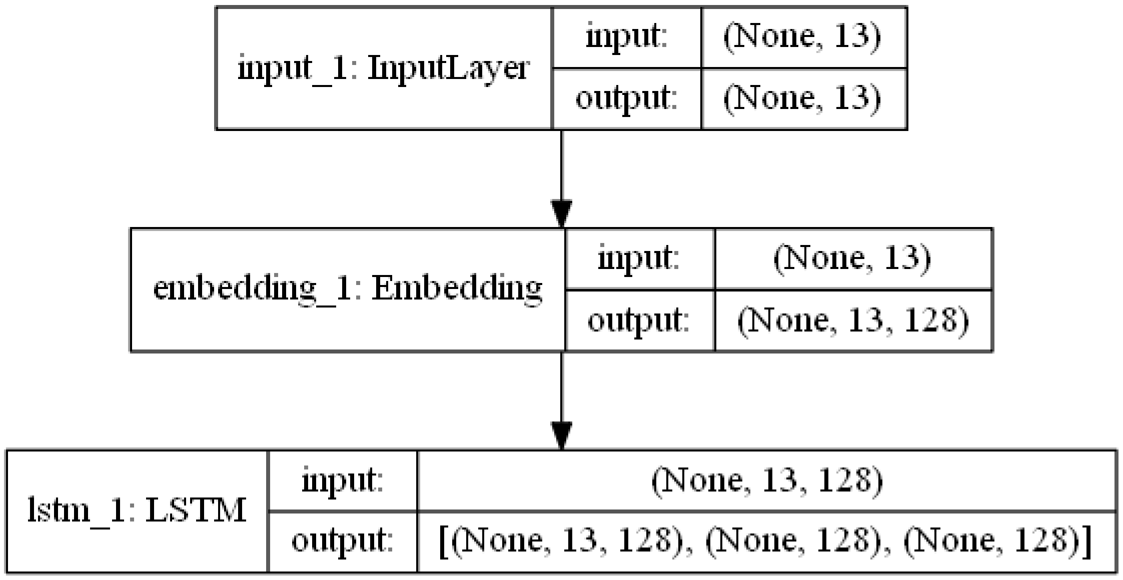


```
ሓበሬታ ሕማማት ኣርእየኒ።  <BEGIN>  SELECT * FROM ሕማማት;
ሓበሬታ ሕሙማት ኣርእየኒ።  <BEGIN>  SELECT * FROM ሕሙማት;
ሓበሬታ ሰራሕተኛታት ኣርእየኒ።  <BEGIN>  SELECT * FROM ሰራሕተኛታት;
ምልክታት ሕማማት ኣርእየኒ።  <BEGIN>  SELECT ምልክት FROM ሕማማት;
ጠንቂታት ሕማማት ኣርእየኒ።  <BEGIN>  SELECT ጠንቂ FROM ሕማማት;
ዓይነታት ሕማማት ኣርእየኒ።  <BEGIN>  SELECT ዓይነት FROM ሕማማት;
መከላኸሊ ሕማማት ኣርእየኒ።  <BEGIN>  SELECT መከላኸሊ FROM ሕማማት;
መለለይ ሕሙማት ኣርእየኒ።  <BEGIN>  SELECT መለለይ FROM ሕሙማት;
ሽም ሕሙማት ኣርእየኒ።  <BEGIN>  SELECT ሽም FROM ሕሙማት;
ሽምኣቦ ሕሙማት ኣርእየኒ።  <BEGIN>  SELECT ሽምኣቦ FROM ሕሙማት;
ጾታ ሕሙማት ኣርእየኒ።  <BEGIN>  SELECT ፆታ FROM ሕሙማት;
ዕለት ሕሙማት ኣርእየኒ።  <BEGIN>  SELECT ዕለት FROM ሕሙማት;
ዓይነት ሕሙማት ኣርእየኒ።  <BEGIN>  SELECT ዓይነት FROM ሕሙማት;
መለለይ ሰራሕተኛታት ኣርእየኒ።  <BEGIN>  SELECT መለለይ FROM ሰራሕተኛታት;
ሽም ሰራሕተኛታት ኣርእየኒ።  <BEGIN>  SELECT ሽም FROM ሰራሕተኛታት;
ሽምኣቦ ሰራሕተኛታት ኣርእየኒ።  <BEGIN>  SELECT ሽምኣቦ FROM ሰራሕተኛታት;
ዕድመ ሰራሕተኛታት ኣርእየኒ።  <BEGIN>  SELECT ዕድመ FROM ሰራሕተኛታት;
ጾታ ሰራሕተኛታት ኣርእየኒ።  <BEGIN>  SELECT ፆታ FROM ሰራሕተኛታት;
ክፍሊት ሰራሕተኛታት ኣርእየኒ።  <BEGIN>  SELECT ደሞዝ FROM ሰራሕተኛታት;
ዕለት ሰራሕተኛታት ኣርእየኒ።  <BEGIN>  SELECT ዕለት FROM ሰራሕተኛታት;
ምልክታትን ጠንቂታትን ሕማማት ኣርእየኒ።  <BEGIN>  SELECT ምልክት, ጠንቂ FROM ሕማማት;
ጠንቂታትን ምልክታትን ሕማማት ኣርእየኒ።  <BEGIN>  SELECT ጠንቂ,ምልክት FROM ሕማማት;
ምልክታትን ዓይነታትን ሕማማት ኣርእየኒ።  <BEGIN>  SELECT ምልክት, ዓይነት FROM ሕማማት;
ዓይነታትን ምልክታትን ሕማማት ኣርእየኒ።  <BEGIN>  SELECT ዓይነት,ምልክት FROM ሕማማት;
መከላኸሊታትን  ዓይነታትን ሕማማት ኣርእየኒ።  <BEGIN>  SELECT መከላኸሊ,ዓይነት FROM ሕማማ
ዓይነታትን መከላኸሊታትን  ሕማማት ኣርእየኒ።  <BEGIN>  SELECT ዓይነት,መከላኸሊ FROM ሕማማ
ጠንቂታትን መከላኸሊታትን ሕማማት ኣርእየኒ።  <BEGIN>  SELECT ጠንቂ,ዓይነት FROM ሕማማት;
መከላኸሊታትን ጠንቂታትን ሕማማት ኣርእየኒ።  <BEGIN>  SELECT ዓይነት,ጠንቂ FROM ሕማማት;
መከላኸሊታትን ምልክታትን ሕማማት ኣርእየኒ።  <BEGIN>  SELECT መከላኸሊ,ምልክት FROM ሕማማ
ምልክታትን መከላኸሊታትን ሕማማት ኣርእየኒ።  <BEGIN>  SELECT ምልክት,መከላኸሊ FROM ሕማማ
መለለይን ሽምን ሕሙማት ኣርእየኒ።  <BEGIN>  SELECT መለለይ,ሽም FROM ሕሙማት;
ሽምን መለለይን ሕሙማት ኣርእየኒ።  <BEGIN>  SELECT ሽም,መለለይ FROM ሕሙማት;
ሽምን ሽምኣቦን ሕሙማት ኣርእየኒ።  <BEGIN>  SELECT ሽም,ሽምኣቦ FROM ሕሙማት;
ሽምን ዕድመን ሕሙማት ኣርእየኒ።  <BEGIN>  SELECT ሽም,ዕድመ FROM ሕሙማት;
ዕድመን ሽምን ሕሙማት ኣርእየኒ።  <BEGIN>  SELECT ዕድመ,ሽም FROM ሕሙማት;
ዕድመን ጾታን ሕሙማት ኣርእየኒ።   <BEGIN>  SELECT ዕድመ,ፆታ FROM ሕሙማት;
```

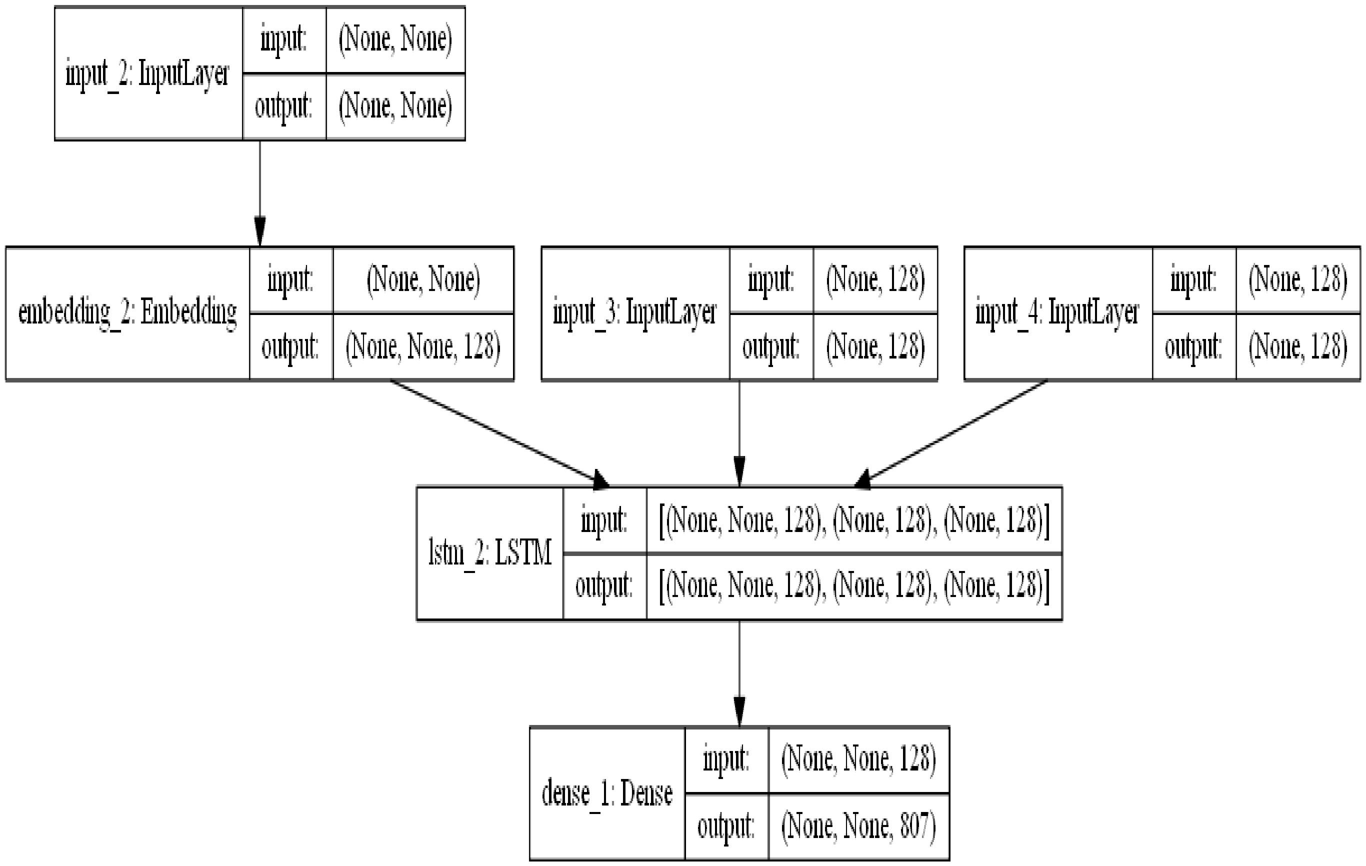
Appendix E: Train Model



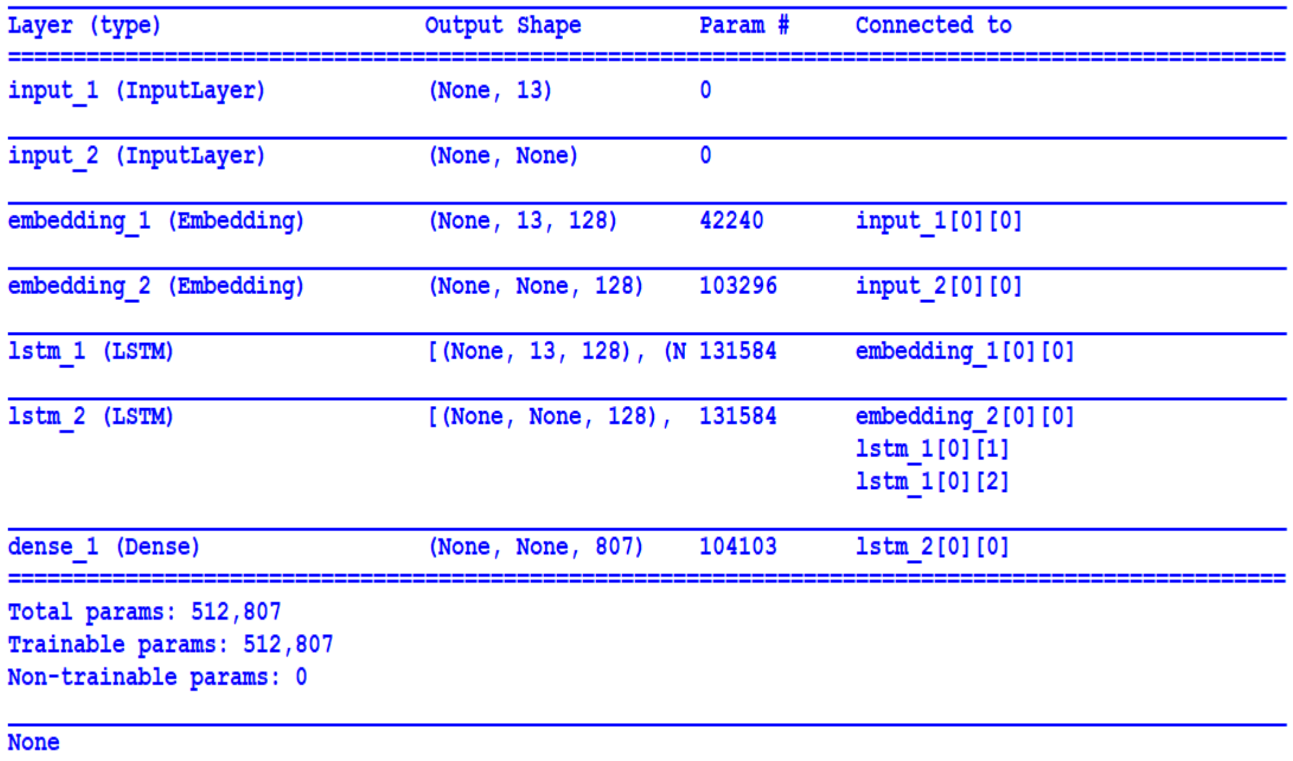
Appendix F: Encoder Model



Appendix G: Decoder Model



Appendix H: Summery of the model



```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 13)           0
__________________________________________________________________________________________________
input_2 (InputLayer)            (None, None)         0
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 13, 128)      42240       input_1[0][0]
__________________________________________________________________________________________________
embedding_2 (Embedding)         (None, None, 128)    103296      input_2[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   [(None, 13, 128), (N 131584      embedding_1[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   [(None, None, 128),  131584      embedding_2[0][0]
                                                                 lstm_1[0][1]
                                                                 lstm_1[0][2]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, None, 807)    104103      lstm_2[0][0]
==================================================================================================
Total params: 512,807
Trainable params: 512,807
Non-trainable params: 0
__________________________________________________________________________________________________
None
```