JIMMA UNIVERSITY

JIMMA INSTITUTE OF TECHNOLOGY

FACULTY OF COMPUTING AND INFORMATICS

Master's Thesis

Automatic Amharic Question Generation System Using Recurrent Neural Network

BY

Alelign Mekonen Bogale

A THESIS SUBMITTED TO JIMMA UNIVERSITY, SCHOOL OF POSTGRADUATE STUDIES FOR THE PARTIAL FULFILLMENT FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATION TECHNOLOGY.

Jimma, Ethiopia

June, 2022

JIMMA UNIVERSITY

JIMMA INSTITUTE OF TECHNOLOGY

FACULTY OF COMPUTING AND INFORMATICS

Automatic Amharic Question Generation System Using Recurrent Neural Network

By:

Alelign Mekonen Bogale

This is to certify that the thesis prepared by Alelign Mekonen, Automatic Amharic Question Generation System Using Recurrent Neural Network Submitted in partial fulfillment of the requirements for the Degree of Master of Science in Information Technology complies with the regulations of the University and meets the accepted standards concerning originality and quality.

Approved by Examining Committee:

Getachew Mamo (Ph.D.) _	____________________	__________________
Main Advisor	Signature	Date
Abaynew Guadie (MSc.) _	____________________	__________________
Co-Advisor	Signature	Date
Dawud Yimer (MSc.) ____	____________________	__________________
Internal Examiner	Signature	Date
Worku Jifara (Ph.D.) ____	[illegible]	__________________
External Examiner	Signature	Date

Declaration

This thesis is a summary of my original study findings. I, the undersigned, declare that this study has not been presented for a degree in any other university and that all sources of materials included in the thesis have been properly acknowledged.

_____Alelign Mekonen______

Name of student

____

Sign

____

Date

Acknowledgement

First and foremost, praises and thanks to the God, the Almighty, for His blessings throughout my research work to complete the research successfully.

I would like to express my deep and sincere gratitude to my research Advisor, Getachew Mamo (PhD) and Co-advisor Abaynew Guadie (MSc), for giving me the opportunity to do research and providing invaluable guidance throughout this research. Their dynamism, vision, sincerity and motivation have deeply inspired me. They have taught me the methodology to carry out the research and to present the research works as clearly as possible. It was a great privilege and honor to work and study under their guidance. I am extremely grateful for what they have offered me and their patience during the discussion I had with them on research work and thesis preparation.

My Special thanks is to my sisters Birtukan M, and Fikradis M., for their strong motivation to complete this research, and my friends, colleagues, and classmates for the deep interest shown to complete this thesis successfully.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ANN	Artificial Neural Network
AQG	Automatic Question Generation
BiGRU	Bidirectional Gated Recurrent Unit
BLEU	BiLingual Evaluation Understudy
CGC-QG	Clue Guided Copy Network for Question Generation
CNN	Convolutional Neural Network
GRU	Gated Recurrent Unit
K2Q	Keyword to Question
KG	Kindergarten
LSTM	Long Short-Term Memory
MCQ	Multiple Choice Question
METEOR	Metric for Evaluation of Translation with Explicit ORdering
NER	Named Entity Recognizer
NLP	Natural Language Processing
POS	Part-of-Speech
QA	Question Answering
QG	Question Generation
QGSTEC	Question Generation Shared Task and Evaluation Challenge
QP	Question Pattern
RNN	Recurrent Neural Network
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
SQuAD	Stanford Question Answering Dataset
SRL	Semantic Role Labeling
WIC	Walta Information Center

Abstract

A question is a linguistic term that is used to make a request for information and is essentially posed in order to meet informational needs. These days electronic documents written in different languages are available over the internet or any other storage media. These documents however do not contain enough questions. This is because manually preparing questions is very time-consuming and tedious task. The solution for such problem is an Automatic Question Generation System, which is a very challenging task in NLP that is designed to automatically create questions from sentences. In this study an attempt is made to design an Automatic Question Generation System from Amharic sentence using Recurrent Neural Network. To train the model, 60,023 sentence-question pair and/or sentence-question-answer triple dataset collected from the internet, with a data collection system that has been specifically designed for this task, is used. To make the system generate more than one question from a single sentence a beam search decoder is used. The study achieved an accuracy of 88.36% and 82.54% for the model trained with the sentence-question pair dataset and sentence-question-answer triple dataset respectively. The former model generated Amharic questions for the given Amharic sentence while the later generated both questions and answers for the given sentence.

Key words: Amharic Question, Factual Question Generation, Sequence-to-Sequence, encoder, decoder, Attention Mechanism, Beam Search Decoder.

Chapter One

1. Introduction

1.1. Background

Natural language processing (NLP) is an area of artificial intelligence and computer science concerned with the interaction of humans and computers [1, 2]. It is an area of research that discovers the way that computers can be used to understand and manipulate natural language text or speech to do some useful things. The main aim of natural language processing is to use natural languages as effectively as we humans do [3]. Machine translation, speech recognition, question generation etc. are among the applications of NLP [3]. An interesting challenge in the field of Natural Language Processing for preparing questions automatically is Automatic Question Generation (AQG).

A question is a linguistic expression used to make a request for information and basically asked so as to fulfill the informational needs. Questions are used to extract useful information from the text. That means questions are essential elements of learning. Questions have two categories based on how complex they are. These are shallow questions (who, what, when, where, which, how many/much and yes/no questions) which are factual, and deep questions (why, why not, what-if, what-if-not and how questions), which involves complex inference [4].

Whether shallow or deep, questions are used in several areas for example, teachers use questions to check the understanding of the student, and in interviews questions are asked to check the skills of the candidate, for entrance exams and in many other areas. More generally, questions are used in many applications such as student learning, medicine, security contexts etc. It is also an input to the question answering task [4]. Nevertheless, preparing questions manually is very tedious and time-consuming activity. In addition, even though there are massive amounts of educational data available both online and offline in electronic form, they do not have questions for every specific content in it.

So far, manual generation of questions from a text for creating practice exercises, tests, quizzes, etc. has consumed labor and the golden time of academicians and instructors.

But now adays the world is trying to create questions automatically through AQG. Automatic question generation (AQG) is the task of automatically creating questions from natural language texts that can be answered by a certain span of text within a given passage [5, 6, 7]. Automatic question generation task has been done for foreign languages like English, Arabic, Spanish, etc. Moreover, the researchers

[8] and [9] have worked on the local language. The former applied rule-based approach to generate Amharic Math Word Problem and Equations while the later used the same approach to generate factual Amharic questions.

The way that the Amharic question formed is very different from the way that other foreign Language's question is formed due to the different nature of those languages structure. In this study, we used recurrent neural network to generate Amharic question(s) from a sentence by training a sequence2sequence learning model with a large data set.

1.2. Motivation

As a working language of the Federal Democratic Republic of Ethiopia, Amharic is widely used in many parts of the country including Amhara, Southern Nations Nationalities and Peoples, South West Ethiopia Peoples regions. Amharic speakers whether individual or organization create educational Amharic documents. When these Authors write books for KG, primary and secondary school students, they need to add questions at the end of chapters manually to test the understanding of the readers and add answer for those questions at the end of books. Preparing questions manually for every specific content of documents is very laborious. There are also radio programs like Yiteyiku Yishelemu (ይጠየቁ ይሸለሙ) that uses factual questions. These radio programs authors collect factual Amharic sentences from books, internet, newspapers etc. and prepare factual questions from that sentence manually. These manual generation of questions takes much labor and time. This unnecessarily consumption of labor and time is the motivation behind conducting this study.

1.3. Statement of the Problem

Nowadays, the electronic educational document is being produced in a very high amount every day both online and offline. Part of those documents is written in the Amharic language. Even the students text books are made public electronically. These text books consist of miscellaneous exercise which are prepared manually by the authors. The students guide books also consists manually prepare questions whose answers are appended at the end of the book. Teachers also prepare quizzes, practice exercises and other questions to test students understanding. In addition, radio programs collect factual sentences from different domains like historical, sport, economical, language, social, political, current issues etc. from different sources like internet, books and newspapers. From those collected sentences, the authors create factual questions manually. The problem behind preparing questions manually is that: first, it takes time, second: it is difficult to prepare enough questions, and third: the learners or readers cannot get those manually prepared questions from the teachers unless it is for a test even though questions are the key linguistic expressions used to request information. Therefore, to make questions available easily for Amharic text, it is found better to make a study how questions can be generated from Amharic text automatically.

Lots of work has been done for other languages like English [4, 10, 11], Chinese [12], Portuguese [13] etc. so far. However, the construction of questions, the answering techniques, and the grammatical structure of the Amharic language is different from those other languages. Until now, the researcher at [9] has studied the generation of Amharic questions from historic documents by writing specific rules for generating who, where, when and how much/many kinds of factual questions. However, this rule can hardly work well as it is very challenging to extract many features using those specific rules. The researcher also missed the *what* kind of factual questions which is very commonly used question type in the real world. In Amharic, it is common to use the what /ምን question. For instance, from the sentence አበበ ከፍተኛ ዉጤት በማምጣቱ ዋንጫ ተሸለመ።, we can have the question አበበ ከፍተኛ ዉጤት በማምጣቱ ምን ተሸለመ? Moreover, the study assumed generation as a single question is generated from a single sentence, although more than one questions can be created from a single sentence.

So, in the proposed approach we studied the way questions can be generated automatically from Amharic text with recurrent neural network as it can learn more complex features than the hand-written specific rules. The proposed study also used the answer text to make the model generate questions with their answers.

1.4. Research Questions

1. Does answer features have an impact on the performance of Amharic question generation?
2. Which of the selected learning algorithms offer better performance?

1.5. Objectives

The general and specific objectives of this study are presented as follows:

1.5.1. General Objective

The general objective of this study is to investigate and design an automatic question generation system from Amharic sentences using recurrent neural network.

1.5.2. Specific Objectives

To achieve the general objective, the study has the following specific objectives:

- ✓ To study the formation of Amharic questions.
- ✓ To design the general architecture of the Automatic Amharic Question Generation system.
- ✓ To collect a corpus for model development purpose.
- ✓ To develop a prototype for the proposed system.
- ✓ To evaluate the performance of the designed system.

1.6. Scope and Limitation of the Study

The scope of this study is limited to Amharic language as we are very familiar with the language. The question types that are included in this study are factual questions, which are questions that do not involve complex inferences. This includes: who, what, when, where, which, and how many/much questions. The study used open domain data to design the automatic Amharic Question Generation.

In this study, we do not include the generation of questions that require complex inferences like why, why not, what-if, what-if-not and how. In addition, we do not include simplification of complex sentences, ambiguity resolution and anaphora resolution.

1.7. Significance of the Study

The result of the proposed system is a question or list of questions. Those questions can be used as training and testing data for different areas including question answering and even for further improvement of question generation. It can also be applied in the areas of education: in KG, primary and secondary schools as a great learning tool if it is likely to be properly implemented, as it is helpful for students to understand things about facts. Moreover, this study decreases the labor and time of teachers, book authors, online learning material publishers and radio program authors like Yiteyiku Yishelemu (ይጠየቁ ይሸለሙ) by generating questions within short time.

1.8. Organization of the Thesis

This work is organized into seven chapters. The second chapter covers the literature review on question generation. This chapter discusses the overview of QG and its general architecture components. In addition, it covers related works from the local and global levels. Similarly, the third chapter presents the Amharic language. The formation of questions in Amharic, construction of sentences and the punctuation marks used in Amharic is included in the chapter.

In the fourth chapter, the method used in this study is described in depth. The steps followed in design science are discussed in accordance with this study including the system architecture and components used in designing Automatic Question Generation. The fifth chapter discusses the design part of the Automatic Amharic Question Generation in detail. The architecture of the system and its components, as well as what and how each component of the architecture functions, are thoroughly discussed. The sixth chapter, Implementation and Evaluation, explain the experiments carried out and the results obtained through the use of evaluation techniques. Finally, the final chapter discusses the conclusions reached based on the results of the studies, as well as recommendations offered to other future investigations.

Chapter Two

2. Literature Review

2.1. Question and Its Types

A question is a linguistic term that is used to make a request for information and is essentially posed in order to meet the informational wants. To elicit meaningful information from the text, questions are utilized. That is to say, questions are fundamental components of learning. Questions are divided into two categories based on their complexity [4, 14]. These are factual shallow questions (who, what, when, where, which, how many/much, and yes/no questions) and deep questions (why, why not, what-if, what-if-not, and how questions).

Sentence: በ2013 ዓ.ም በጦርነት ምክንያት የአስር ሺህ ሰዎች ህይዎት አለፈ።

Factual questions	Deep question
በ2013 ዓ.ም በጦርነት ምክንያት የስንት ሰዎች ህይዎት አለፈ?	ባለፈው ዓመት አስር ሺህ ሰዎች ለምን ሞቱ?
መቼ በጦርነት ምክንያት የአስር ሺህ ሰዎች ህይዎት አለፈ?	Assuming that it is 2014 E.C when the question is asked.

As shown above, factual questions can easily be answered from the given sentence. That means there is a span of text from the sentence that can answer the shallow questions. The text spans "የአስር ሺህ" and "2013 ዓ.ም" are the answers for the above two factual questions respectively from top to bottom. On the other hand, deep questions need some extra information to answer the question. So, to answer the above deep question, one should know that "ባለፈው ዓመት" is a year that is 1-current year, and ህይዎት አለፈ/passed away has the same meaning with ሞቱ.

There are also other kinds of questions that have different structure and way of answering including Gap-filling and multiple choice.

Gap-filling questions

A gap-fill is a practice activity in which one must substitute words that are missing from a text. These words are selected and eliminated to practice a certain linguistic concept. Gap-fills are often used to practice specific language points, for example items of grammar and vocabulary, and features of written texts such as conjunctions, or to make learners extract facts [15]. For instance:

A sentence: አርባ ልጆች የተባሉት በአንደኛው የዓለም ጦርነት ወቅት የቱርክ መንግስት ባደረገው ጭፍጨፋ ወላጆቻቸውን ያጡት ናቸው።

Gap fill question: ______ የተባሉት በአንደኛው የዓለም ጦርነት ወቅት ______ ባደረገው ጭፍጨፋ _________ ያጡት ናቸው።

In the above gap filling question, the respondent is expected to provide the text spans "አርባ ልጆች", "የቱርክ መንግስት", and "ወላጆቻቸውን" for the given spaces from left to right respectively.

Multiple choice questions (MCQ)

Multiple choice question (MCQ) is a WH-type question with multiple distractor and one right answer. Multiple choice questions are created in three steps: word/phrase extraction, distractors selection, and question construction [16]. The similarities between the key and the distractors makes the MCQs complex or shallow. The more similar they are, the more information you will need to distinguish between them and select the correct one. As a result, if we have an acceptable measure of similarity for ideas, we can attempt to create MCQs with predicted difficulty [17]. For example: from the Amharic sentence "የሐረር ጀጎል ግንብ ከ1998 ዓ.ም ጀምሮ በዩኔስኮ የዓለም ቅርስ ዝርዝር ውስጥ ይገኛል።", one can construct a multiple-choice question as:

የሐረር ጀጎል ግንብ ከ____ ጀምሮ በዩኔስኮ የዓለም ቅርስ ዝርዝር ውስጥ ይገኛል።

ሀ. 1990 ዓ. ም ለ. 1994 ዓ. ም ሐ. 1998 ዓ. ም መ. 2000 ዓ. ም

From the above multiple-choice question, choice "ሐ" is the correct answer based on the given sentence and the rest are distractors. One promising example of the educational use of NLP methods is the automated generation of multiple-choice questions (MCQs). Multiple-choice question tests are widely utilized, regarded as extremely beneficial, and effective in measuring students' knowledge; yet, manually writing those questions is costly [18]. MCQs may be constructed using a rule-based technique and assessed using several criteria such as grammatical correctness, meaningfulness, and suitable WH word usage [19].

2.2. What is Question Generation?

Automatic question generation from natural language text is a tough issue that has received a lot of attention in natural language processing [12]. Automatic question generation generates questions by taking the input text, which has the potential value in education [4]. It is being considered an essential component of the learning environments (— to generate questions for reading comprehension materials [6]), help systems, information seeking systems, and many other applications [20]. Question Generation involves two tasks: content selection (the text selected for question generation) and question formation (transformations of the content to get the question) [21]. It is an important technique that can improve the training of question answering [11]. That means as the reverse task of question answering, question generation also has the potential for providing a large-scale corpus of question-answer pairs [7].

2.3. General Architecture of Question Generation System

Question generation consists of Input Pre-processing, Input Processing and Question Generator as diagrammatically shown in Figure 2.3.1 [11, 22]. Sentence preprocessing is the initial step after a dataset collection process. It is where the data can be normalized, free from unnecessary special characters, and tokenized so that they become ready to go to the sentence processing component of QG. The sentence processing component then convert the input sequence into vector form in the embedding layer so that the encoder of the processing component read it and produces a data structure that summarizes the input sequence which is the context vector. This vector is used as an input for the decoder of the processing component. The decoder is trained to predict the next word based on the previously predicted words and the context vector. So, based on the general idea of studies [11, 22], the very basic architecture of the neural automatic question generation can be generalized as shown below.

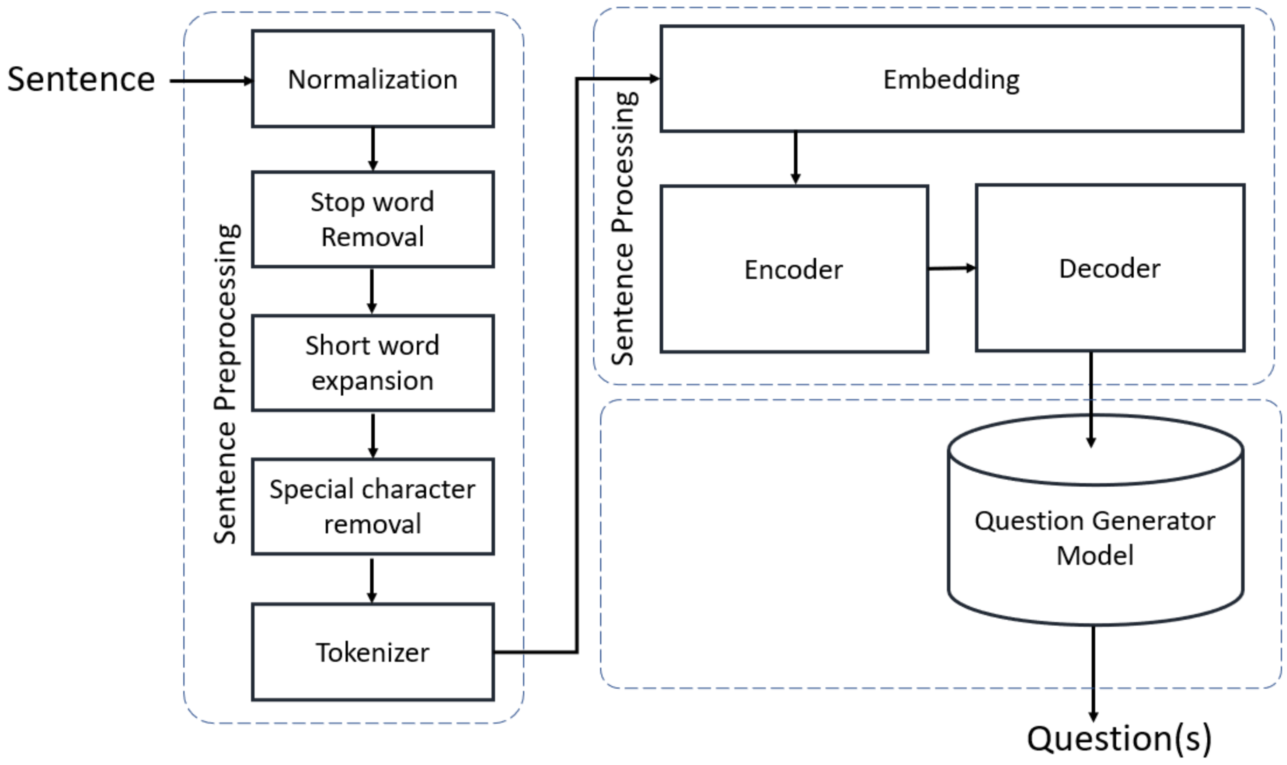


Figure 2.1 General Architecture of Automatic Question Generation System

2.3.1. Sentence Preprocessing

The first step after collecting necessary data leads to Preprocessing as the raw data by itself cannot be an input to the model that is being developed. Input preprocessing is the way of preparing the data into a more analyzable and comprehensive way of presenting it to the machine learning process.

Normalization, stop word removal, short word expansion, special characters removal, and Tokenization are some of the preprocessing steps in Natural Language Processing.

Normalization

In the Amharic language, there are different characters that have the same sound but a different structural appearance. When these characters (Fidels) are used in the construction of Amharic words, they share their properties with the words too. That is, the one word in meaning can be written in two or more ways. For instance, the Amharic words ሀምሌ፣ ሐምሌ፣ ኀምሌ are similar words used to represent the month July. Therefore, normalizing these words lets us use one of them depending on our normalizer. These help us to minimize the dictionary size as there are a number of similar sounded but different structured Amharic characters or Fidels like: [ሀ፣ሐ፣ኀ], [ጸ፣ፀ], [ሠ፣ሰ], [አ፣ዐ], [ሆ፣ሖ፣ኆ], [ጾ፣ፆ], [ሦ፣ሶ], and [ኦ፣ዖ].

Short Word Expansion

The Amharic words ት/ት, ጽ/ቤት, and ጠ/ፍ/ቤት are some examples of short words to represent ትምህርት, ጽሕፈት ቤት, and ጠቅላይ ፍርድ ቤት respectively. We usually use "/" while writing Amharic words in short form. When we use both short and expanded word forms interchangeably in our corpus, these words are considered as different words while representing them in a vector form. So, to avoid such problem, short word expansion is made based on the predefined short words [6].

Stop Word Removal

Words that appear in many sentences frequently like "እና", "ነው", "ናቸው", etc. are stop words [6]. Removing these words depends upon the type of task. For instance, these words are not considered practical to distinguish one sentence from another. On the other hand, these words are treated as being very useful in generating questions.

Tokenization

To get keywords or tokens easily we use tokenization. The first step of input processing begins by taking the vector form of individual terms or tokens. So, to get these terms it is very crucial to chop up the input by using different demarcations like white spaces. For instance, the Amharic sentence "ኢትዮጲያ የሰዎች መገኛ ነች" can be tokenized as 'ኢትዮጲያ', 'የሰዎች', 'መገኛ', 'ነች'.

2.3.2. Sentence Processing

Sentence processing most often consists of the construction of sentence representation, encoding the sentence tokens to produce meaningful machine-understandable value, and decoding the encoded value to predict the possible question words. Sentence processing helps the machine to easily represent words numerically, comprehend sentences by encoding features, and generate question. The inputs to sentence processing are represented in the form of vectors to perform encoding and decoding. This vector representation is referred to as word embeddings, encoding, or vectorizing. It helps to convert symbolic representations into meaningful numbers that capture underlying semantic relations between the symbols [23].

Embeddings are trained on large datasets, saved, and then used for solving other tasks. Such embeddings are named pre-trained embeddings [24]. There are a number of pretrained embeddings for different languages including Word2vec, GloVe, and FastText.

Word2vec is a language modeling technique that employs a neural network to learn correlations in a huge dataset such as a text. The model may discover semantically related terms or suggest additional words for a partial text once the word modeling has been trained. Word2vec associates each different word with a set of numbers known as vectors, as the name implies. Using a basic mathematical function or cosine similarity between the vectors Word2vec reflects the level of semantic similarity between the words represented by those mathematical functions or vectors [25].

GloVe [26] learns features in an unsupervised manner to represent words as vectors. It trains the algorithm using word-to-word co-occurrence statistics in a corpus. The cooccurrence statistical representations generated to highlight the linear substructures of the word vector space. The word representation is also used to compress the input into a low-dimensional vector. It has two parts: an encoder and a decoder. The encoder is responsible for creating the low-dimensional embedding that helps the machine understand the raw text. The decoder rebuilds the original input by inverting the vectorized representation.

FastText is a word2vec model with a pre-trained resource for more than 157 languages throughout the world, including Amharic, which was trained using Common Crawl and Wikipedia. These models were trained with a Continuous Bag of words with position-weights of 300 dimensions, character n-grams of length 5, a window of size 5, and 10 negatives [27]. CBOW (Continuous Bag of Words) is a model for predicting the likelihood of a target word in a given context. A context could be a single word or a collection of words [28].

Encoder

Encoding is the process of converting data into the desired format. In the context of neural network sequences of the words are turned in to a hidden state by the encoder. Recurrent neural networks are stacked to create the encoder. This layer is used because its structure allows the model to understand the sequences' context and temporal dependencies. The hidden state is the latest state of the Recurrent Neural Network timestep, which is the output of the encoder [29].

Decoder

Decoding is the process of converting a coded message into a language that can be understood. The decoder's job in the deep learning model is to turn the hidden vector into the output sequence, which is a question in the case of the question generation model. RNN layers and a dense layer build the decoder to predict the word [29]. One of the main advantages here is that the input and output sequences can have different lengths. This opens up the possibility of some extremely fascinating applications, such as video captioning, question answering and question generation.

2.3.3. Question Generator

Question Generator attempts to generate natural language questions from given content (knowledge base triples, tables, texts, or images), with the generated questions requiring answers from the contents [30, 31]. Deep learning methods require a huge quantity of data to construct a data-driven model for a specific problem domain. The reason is that when the data volume is small, deep learning algorithms often perform poorly. Having a huge sentence-question pair dataset has a direct performance impact on the question generation model [32].

2.4. Deep Learning and Artificial Neural Networks

Artificial Intelligence is a general field that incorporates the subgroup of machine learning called deep learning [33]. The fundamental problem in machine learning and deep learning is to meaningfully learn valuable representations of the input data and transform the data into representations that help us get closer to the target output.

To learn the representations from data and extract valuable representations, deep learning uses the successive layers [33]. That is why sometimes, deep learning is also known as layered or hierarchical representation learning which is an evolution of machine learning that makes increasingly complex hierarchical models intended to mimic thought processes in the human brain over the simple machine learning models [34].

Modern deep learning often includes a number of successive layers to automatically learn representations from the experience of sample datasets unlike other machine learning models. Those layered representations are learned through learnable models that are known as neural networks [33]. When a set of processing units are assembled in a closely interconnected network, they offer an astonishingly rich structure showing some features of the biological neural network found in the brain of human beings. This kind of structure is called an artificial neural network (ANN) [35].

The deep learning model learns by keeping the specification of what each layer does with its incoming data in the weights of each layer. These weights represent the input data numerically. That in a very simple context means, deep learning finds a set of values for the weights of all the layers in a network which will map sample inputs to their associated targets correctly [33].

Deep neural networks perform input to target representation by mapping through a deep sequence of simple data changes or layers and learning these data transformations from the exposure of examples. When the deep neural networks are learning, we need to determine how the predicted output is far from the original target value, to maintain the output of a neural network.

The task of determining the gap between the actual and predicted value is the responsibility of the loss function of the neural network. The loss function computes the network's performance by calculating the distance score between the predictions and the true target of the example data. To minimize the achieved loss using the score as a feedback signal, deep learning propagates back to adjust the values of the weights in a direction that will lower the loss score. To make this adjustment, the optimizer implements the Backpropagation algorithm which is the essential algorithm in deep learning [33].

Each neuron, the learning units in the neural network, then receives several inputs, takes a weighted sum, passes it through an activation function, and responds with a predicted output [36]. The neurons extract features based on the weight values of the input data. When these weighted inputs are summed together, they produce a function that represents probability values from 0 to 1, and negative infinity to infinity [37]. This function, most of the time, includes a constant value called bias to adjust the output based on the weighted sum of the inputs to the neuron [38].

Algorithms implemented in the neural network have multidimensional nature which let them show good performance [35] although they require high cost, CPU utilization, and physical memory [39]. Neural networks are used in many areas including speech recognition, question answering, and question generation. They determine the error of the network and then adjust the network to minimize the faults [39].

2.4.1. Layers

A layer is the highest-level building unit in deep learning. A layer is a container that accepts weighted input, changes it using a collection of primarily non-linear functions, and then delivers the transformed values as output to the next layer. In a network, the initial and last levels are referred to as input and output layers, respectively, while the layers in between are referred to as hidden layers. It is also a data-processing module that accepts one or more tensors as input and returns one or more tensors as output [33]. A tensor is either a container for numeric data or a multidimensional vector input matrix. Diverse layers, as well as tensor formats, are ideal for different data processing. We may create helpful data transformation pipelines by clipping suitable layers together. Hidden layers, learn compressed representations of the original input with less neurons than the input layer.

Different layers alter their inputs in different ways, and some layers are more suited for certain tasks than others. A convolutional layer, for example, is commonly utilized in models that cope with picture data. Recurrent layers are used in models that operate with time-series data, and fully connected layers, as the name implies, connect each input to each output within their layer [40].

2.4.2. Activation Functions

These functions are intended to highlight key data values in their input. [41] It is a function that is used to obtain the node's or neuron's output. It's also referred to as the Transfer Function. The two types of activation functions are Linear and non-linear activation functions [42]. However, in actuality, three primary kinds of neurons are utilized, introducing nonlinearities in their calculations. Sigmoid, Tanh, and ReLu are the three of them.

Sigmoid

This is particularly useful in models where the probability must be predicted between 0 and 1 [38]. The sigmoid function has the potential to stall a neural network during training. As a result, it's mostly used at the output layer or in binary classification applications [34].

Tanh (Hyperbolic Tangent Function)

Tanh is a shifted variant of the sigmoid function [38], with a range of -1 to 1. This function facilitates training for the next layer easier and faster by centering the data to have a zero mean by utilizing the activations that come from the hidden layers. One drawback of both Sigmoid and Tanh is that if our weighted sum input is either extremely large or extremely small, the gradient, also known as the derivative or slope of this function, becomes very small and approaches zero, delaying gradient descent [43].

ReLu (Rectified Linear Unit)

ReLu takes less time to compute than the other activation functions. As a result, it is frequently used as the default activation function. In ReLu, the gradient decline does not become stuck. One downside of this activation function is that when the weighted sum input is negative, the derivative is equal to zero. The issue is referred to as the fading ReLu. If the network weights always result in negative inputs to a ReLu neuron, that neuron will not effectively contribute to network training [43].

Softmax

In deep learning, the softmax layer is utilized for multi-class classification. It makes use of a probability distribution by making sure that the total of all outputs is near to or equal to one. A single entry with a weighted vector near to one is a strong prediction. While the remaining entries near 0 are considered weak predictions, they might be labeled in a variety of ways. This layer provides a more accurate picture of our forecast confidence. The output of a neuron in a softmax layer is dependent on the outputs of all the other neurons in the layer, unlike in other types of layers [43].

2.4.3. Loss functions and Optimizers

A loss function is a way to see how effectively our algorithm models our dataset. The purpose of loss functions is to compute the quantity that a model should seek to minimize during training. Our loss function will produce a greater value if our forecasts are completely wrong. If it's good, it'll give us a lower number. Based on the loss function then optimizers improve the learning process by determining how the network is updated [44].

Optimizers are techniques for modifying the parameters of a neural network, such as weights and learning rate, in order to reduce losses [45]. The most widely used adaptive learning rate algorithms are AdaGrad, RMSProp, and Adam.

AdaGrad (Adaptive Gradient Algorithm)

For each parameter, the AdaGrad optimizer updates the learning rate. While AdaGrad works well for basic convex functions, it is not intended to navigate the complicated error surfaces of deep networks [33].

Root Mean Squared Propagation, or RMSProp

RMSProp is a gradient descent variant that adjusts the step size for each parameter by utilizing a decaying average of partial gradients. Rather than using the entire set of gradients, this optimizer utilizes a fixed-size window over the gradients computed at each step [33].

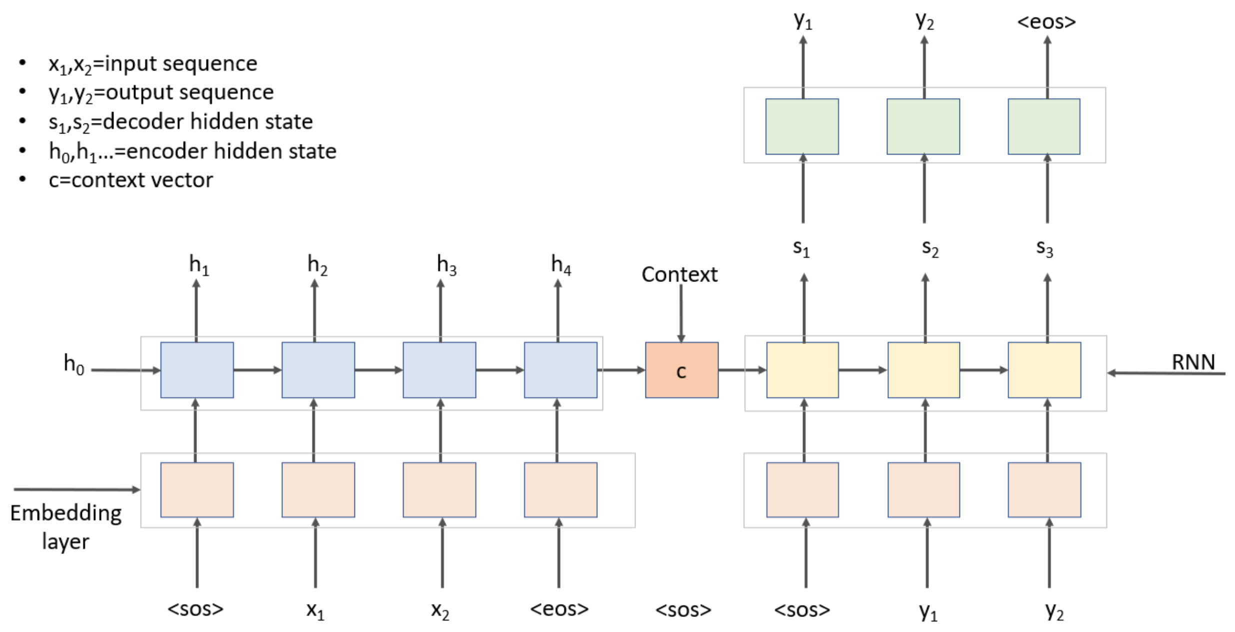
Adam

The Adam optimizer is notable for its corrective measures and ability to more effectively integrate zero initialization bias, which is a drawback of RMSProp, as well as the key concepts underpinning RMSProp with momentum [33].

2.4.4. Sequence to sequence

At the very beginning of our education, the teacher made us call the individual alphabets repeatedly like: say ha/ሀ, hu/ሁ, hi/ሂ, … Another time he teaches us how to construct simple words from those characters by combining them. And finally, we become able to construct sentences by combining learned words. That is how we humans learn our language. Likewise, to make the machine generate sequences from another sequence, we use a way of learning named sequence-to-sequence learning (Seq2Seq). It is concerned with training models that convert sequences from one domain (for example, phrases in Amharic) to sequences from another domain (e.g., questions created from the same sentences or other language sentence from the same sentence) [46].

The most prevalent sequence-to-sequence learning model is the encoder-decoder model, which uses RNN to encode the source sequence (input) into a single vector, and decode this vector to the target sequence [47]. The one that encode the source sequence in to a single vector is the encoder. The single vector is the abstract representation of the whole input sequence. This vector is commonly called as context vector [47]. This is equivalent to our document preparation. When we prepare documents (encode in this case) we follow many steps each requiring our effort. Then after completing everything in the document, we write an abstract that can provide the whole idea of the document from the beginning to end in short. Likewise, after processing the input, the encoder put the overall process in a single vector which is the context vector. This vector is used to initialize another RNN which is the decoder that learns to produce the target output [47].



• x_1,x_2=input sequence
• y_1,y_2=output sequence
• s_1,s_2=decoder hidden state
• h_0,h_1…=encoder hidden state
• c=context vector
y_1 y_2 <eos>
s_1 s_2 s_3
h_1 h_2 h_3 h_4
Context
h_0
c
RNN
Embedding layer
<sos> x_1 x_2 <eos> <sos> <sos> y_1 y_2

Figure 2.2 RNN in the task of translation

Figure 2.2. shows an example of encoder-decoder model where the input/source sequence “x_1, x_2” is passed over the embedding layer and then input into the encoder. <sos> and <eos> indicates a *start of sequence* and *end of sequence* respectively. At each time step, the encoder RNN receives both the embedding of the current word and the hidden state from the previous time step, and it produces a new hidden state. The hidden state used as a vector representation of the input “x_1, x_2” so far. The RNN, which is used generally here, can be any of the recurrent architectures including LSTM and GRU.

The final hidden state is used as the context vector once the final input has been passed into the RNN through the embedding layer. That is the entire input sequences representation in the form of vectors. This vector is used to initialize the decoder to start decoding to get the target output sequence “y_1, y_2” The decoder then generates sequence parts (y_1, y_2) one after the other, one for each time-step. As a first step the decoder uses the <sos> token for the first input y_1 and for the rest inputs the actual or the predicted output by the decoder which is teacher forcing. In the decoder, we need to go from the hidden state to an actual word, therefore at each time-step we use the hidden state, s_t to predict, by passing it through a Linear layer, what we think is the next word in the sequence. This decoding process then ends when the <eos> token is reached.

2.4.5. Attention Mechanisms

When the length of the input sequence becomes longer and longer, the performance of the encoder-decoder network lowers rapidly [48]. Back to the previous example, if the teacher tries to make students call many different alphabets and ask them to repeat, it becomes very difficult for them to call correctly as they forget the earliest ones. Encoder-decoder models have such kind of drawbacks as they use only the last state of the encoder, that is as a context vector [49]. As a solution to such problem Bahdanau [50] and Luong Attention [51] proposed a mechanism called Attention Mechanism.

In the area of neural networks, attention is the process of focusing on relevant parts of the input sequences to solve the problem of long-range dependency problem. The question here is how the focusing/relevant input sequence parts are selected, and how they are used in the decoder to help it decode and produce output with greater performance. According to Bahdanau [50] and Luong [51] Attention, the relevancy of the input sequences is determined by calculating the alignment score although the way they calculate it is different.

2.5. Question Generation Approaches

2.5.1. Traditional Approach

In this approach, wide coverage of NLP techniques is used in order to achieve the accuracy of the questions generated. These systems first generate training data and test data through a semantic model. Some systems of this type generate rules for each type of questions such as the semantic classes who, when, where and why type questions. “Who” rules look for names that are the class of nouns. “When” rules consist of time expressions only. “Where” rules mostly look for locations. These systems require learning rules from training data and are mostly used in reading and answering comprehension questions. A system called Quarc used some heuristic rules that look for sematic and lexical hints to identify the question class [52, 53].

This approach uses text patterns instead of complex processing involved in other competing approaches. Many of the question generation systems automatically learn the text patterns from passages rather than using linguistic knowledge or tools such as named entity, WordNet [54], ontologies [55], etc. for retrieving answers. For example, the question “who is the founder of Addis Ababa?” will look for the pattern “who is <person name > of <location>?” and the answer is like “<person name> is the founder of Addis Ababa”. Most of the patterns matching QG systems use surface text patterns [56, 57].

The question-generation task has gotten more attention from the natural language processing field in recent years, notably following the first QGSTEC [52]. This challenge includes two parts: the first is question generation from paragraphs, and the second, which got so much attention, is question generation from single sentences. As a result of this task, numerous researchers devised various approaches to overcome these issues. The study [53] created a system for the second task that first translates the complicated input text into a few simple sentences and then classifies each sentence for the appropriate type of question based on the sentence’s subject, verb, object, and preposition. This study did not make use of semantic data.

There are a number of approaches to the question generation problem. The study at [54] succeeds in creating multiple-choice questions from electronic educational texts, as well as semantically relevant distractors. The study used WordNet, as well as a shallow parser, automated term extraction, and word sense disambiguation. Then, it generated questions from declarative phrases by applying transformational rules to them. Another study [55] looked into the problem of vocabulary assessment. These tests are often time-consuming processes that involve hand-written development and can be handled subjectively. The study [55] then created a system that generates vocabulary evaluation questions automatically. These questions are divided into six categories and can take various formats, such as word bank or multiple-choice. It also checked the validity of created questions. The work at [56] which used keyword modeling, was one of the syntax-based methods addressing the QG problem. The study created factoids and definitional questions by using NER, parse trees, and identifying important phrases in a document.

Heilman and Smith [57] also addressed the QG problem on factual information in syntactically complicated reading materials. They proposed an algorithm in this method to extract simple, correct factual statements from semantic and syntactic perspectives. These researchers [57] conducted another investigation that used a template-based method. Their aim is to produce a large number of questions and then rank them. They developed some hand-written procedures to accomplish a syntactic translation of declarative sentences into questions. In addition, for ranking purposes, they utilized a logistic regression model.

The study at [58] proposed a system for transforming keywords into questions (K2Q). It takes into account both query history and user input. K2Q creates a list of refinement words as well as a set of candidate questions. The user can select a preferred question or a refining term. The algorithm then creates another set of potential questions with refining words until the user discovers the desired question or exits.

Engaging readers while reading a news item is one application of question generation [59]. According to Becker et al [59], by posing several questions in an article, a reader attempts to answer those questions and so concentrates more on that issue. Having this idea, the researchers [59] produced quiz questions based on online materials for self-motivated learners to learn about new topics. Their main issue was deciding which section of a sentence to ask about, a process known as gap selection. According to their findings, all semantic jobs might be ideal candidates for these gaps. They were successful in developing a cloze (fill-in-the-blank) question generator.

Another researcher [60] developed a system for generating comparable questions in news articles. The algorithm used consists of two parts: offline and online. A database of comparable question templates was constructed offline, and the algorithm selects a related template based on the article content online. Finally, the algorithm populates the template with article entities in such a way that the comparison of the entities appears logical.

Semantic role labels are also used in the area of question generation. Researcher [61] used a semantic method to develop questions of various types and depths for self-study. They used semantic role labels to generate both questions and answers from a given text.

The study at [62] developed a system that generates questions without having the text's extensive semantic information. The main idea is to benefit from a low-dimensional ontology for document parts. They then crowdsource a list of promising query templates that correspond to that representation. Finally, the results are ranked according to their relevance to the original input. Questions can also be generated from paragraphs using predefined rules and templates. This [63] study proposed a system that generates comprehensive questions automatically from paragraphs based on the assumption that each body of the text is related to a topic of interest. The study created 265 templates and 350 rules to produce questions. These questions were then ranked, to determine their relevance, using community-based question answering systems.

2.5.2. Machine Learning Approach

This method converts input text into a structured collection of features that are then used to generate questions. The input text is first divided into sentences and information about tokens is obtained. That means the unstructured text is transformed into structured data. And that obtained information used as features of the sentence. The features include: POS tags, Named Entity tags, word classes, supper sense tags and multiword expressions. Then after, the system builds its set of rules for each type of questions and the users do not have to manage it [64].

2.5.3. Deep Learning Approach

Deep learning is a machine learning approach that trains computers to accomplish things that humans do naturally. Deep learning is the process through which a computer model learns to execute different tasks directly from pictures, text, or voice, etc. Deep learning models can achieve state-of-the-art performance, sometimes outperforming humans. Models are trained using a massive amount of data and neural network architectures with lots of layers. Deep learning approaches employ neural network architectures, and that is why deep learning models are sometimes referred to as deep neural networks. The term "deep" generally represents the number of hidden layers in a neural network [65].

With the success of deep learning approaches in a variety of natural language processing applications, numerous new research has been conducted to benefit from these techniques. It has demonstrated encouraging results in machine translation, text summarization, question answering, and reading comprehension. The study at [66] published a paper on the question creation challenge that employs

Neural Network ideas. They solve this issue by transforming knowledge graph facts into questions. As a result, they used a Recurrent Neural Network architecture to generate a factual question-answer corpus and demonstrated that their system outperforms the proposed template-based baseline.

Over the last several years, there has been a lot of effort placed towards integrating the natural language and computer vision communities. Some examples include image caption generation, video transcription, and answering questions about an image. Furthermore, this paper [67] introduced the visual question generation task, in which the system's purpose is to generate a question given an image.

After the introduction of the attention mechanism the study at [68] proposed a method that addresses both the QG and QA problems. They used an attention-based encoder-decoder model that is dependent on the target answer and takes both the passage and the target answer as input. The query understanding is then used to capture further relationships between the target answer and the passage. They also utilized a policy gradient learning algorithm during training to address the bias exposure problem.

Another study [69] used an attentional encoder-decoder architecture, which uses the answer location in the input sentence, as well as lexical features and the passage, as inputs to the encoder. The study used the part-of-speech (POS) and named entity (NER) tags as lexical characteristics.

Recurrent neural networks, namely Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), are used in sequence-to-sequence models.

Recurrent Neural Network (RNN)

RNN can loop back and get information or we can say it can predict the information. But it has the problem of long-term dependency. That means as the sequences length increases, it cannot predict what will happen well. All recurrent neural networks take the form of a series of repeating neural network modules. This repeating module in ordinary RNNs will have a relatively basic structure, such as a single tanh layer. However, LSTM addresses this long-term issue (see next section).

Long Short-Term Memory (LSTM)

LSTM is a kind of RNN capable of learning long-term dependencies. Because they can remember information for extended periods of time, LSTMs are intended to avoid the problem of long-term dependency. The cell state is the key to LSTMs, and the LSTM has the power to delete or add information to the cell state, which is carefully regulated by structures called gates. They are built with a sigmoid neural net layer and pointwise multiplication. The sigmoid layer produces integers ranging from zero to one, indicating how much of each component should be allowed through [70, 65]. A value of zero indicates "let nothing through," whereas a value of one indicates "allow everything through."

Three of these gates are present in an LSTM to protect and control the cell state. The first step in LSTM is to select what information will be discarded from the cell state. This choice is determined via a sigmoid layer known as the forget gate. The next step is to decide what additional information will be stored in the cell state. This is divided into two sections. First, a sigmoid layer known as the "input gate layer" determines which values to update. A tanh layer is then used to generate a vector of new candidate values that could be added to the state. In the following step, we'll combine these two to make a state update. Finally, we must decide what we will produce. This output will be filtered, but it will be depending on our cell state. First, run a sigmoid layer to determine which bits of the cell state will be output. Then, pass the cell state through (to force the values to be between -1 and 1) and multiply it by the sigmoid gate output [65].

Gated Recurrent Unit (GRU)

The Gated Recurrent Unit is a new gating mechanism introduced in 2014, it is a newer generation of RNN. GRU is similar to LSTM and has shown that it performs better on smaller datasets. Unlike LSTM, GRU has only two gates, a reset gate and an update gate and they lack an output gate. GRU's got itself free of the cell state and instead uses the hidden state to transfer information [70].

The function of the Update gate is similar to forget gate and input gate of LSTM, it decides what information to keep, add and let go. The Reset gate determines how much previous information to discard. The GRU has fewer operations compared to LSTM and hence they can be trained much faster than LSTMs [65, 70].

2.6. Evaluation of Question Generation Systems

Evaluation is a critical phase in the development of question generation systems. Evaluation helps improve performance by pointing out weaknesses and identifying new tasks for which generation systems can be used. There are two methods of evaluation methodologies: intrinsic and extrinsic evaluation methodology [71].

Intrinsic evaluation methods

Intrinsic evaluation approaches assess a system's performance by evaluating its output in isolation, either against a reference corpus or by eliciting human evaluations of quality[71, 72]. This involves measuring the output's grammaticality and fluency. The predominant intrinsic evaluation methodologies are human evaluation and automatic evaluation. In order to evaluate the quality of a generated question, the human evaluation method uses human judgments, while the automatic evaluation uses algorithms that automatically calculates a score (e.g., by checking the similarity between the generated sentence and a set of reference sentences). Examples of automatic evaluation includes: BLEU [73], METEOR [74], ROUGE [75], Precision, Recall, F1, and Accuracy.

According to the study [71] made on 37 question generation papers from 2013-2018, among the papers that prefer intrinsic evaluation methods, 45% used human evaluation, 32% used automatic evaluation and 23% used both human and automatic evaluation.

Extrinsic evaluation methods

Extrinsic approaches assess system performance by analyzing the system's output in relation to its capacity to do the task for which it was designed [71].

2.7. Related Work

2.7.1. Automatic Amharic Question Generation

So far two researchers conducted research on the Automatic Generation of Questions from Amharic sentence. The first [8] study is non-factual question generation using a template-based approach. Its aim is to Automatically generate Amharic Math Word problem and equations. The study collected 154 Amharic math word problems from the school worksheets and text books. It used a manually Tagged corpus from WIC. The study then applies preprocessing tasks like segmentation, tokenization, and PoS tagging based on the tagged corpus of WIC. After the preprocessing step, the study then forms a templet which consists of placeholders for the math word problem, the type of problem and the equation. To solve the equation and store the solution in the database with necessary measures, the study used the subcomponent that the researcher called Equation Solver. The study's system evaluation was conducted based on the performance of the word problem, the relatedness between the human math word problem and the system generated word problem, and how solvable is the generated problem. The system shown 93.84% overall performance, 90.24% relatedness, and 90% accuracy respectively for the above-mentioned experiments.

The second study [9] was a factual Amharic question generation using a rule-based approach. This study collected about 300 simple Amharic historic sentences from the internet, newspapers, Amharic history books etc. These sentences passed through preprocessing step as usual including normalization, special character removal, segmentation, and tokenization. After preprocessing is an answer key selection using a NER which was trained by 16,130 tagged sentences and 4,032 sentences with untagged tokens. The NER shown 82.0% accuracy. Sentences containing more than 30 words are ignored. After that the study apply transformation rules in a way if the answer key is tagged as person, location related, time related, and ordinal or cardinal numbers, the answer key is replaced by who, where, when and how much/many interrogation words. In addition, an Amharic full stop/። is replaced by a question mark(**?**). The study also applies a post processing tasks like de-segmentation and de- tokenization to ensure proper formatting and punctuation usage. Finally, the study shown an overall accuracy of 86.4%.

2.7.2. Question Generation for the English Language

Discourse connectives are essential for making the text coherent. They link two clauses or sentences that show discourse relationships such as temporal, causal, contrast, elaboration, result, and so on. It has been demonstrated that discourse relations can be used to create questions. The work at [21] analyzes the senses of connectives that help in QG and proposes a method that uses this analysis to generate questions of the type why, when, give an example, and yes/no. It employs an end-to-end QG system that accepts a text as input and outputs all of the questions generated by the selected discourse connectives.

This research examined four subordinating conjunctions: since, when, because, and although, as well as three adverbials: for example, for instance, and as a result. The researchers investigated discourse connectives in relation to their target arguments and question categories. The sense of the discourse connective, according to this study, determines the question type. As an example: the connective "since" can indicate a temporal, causal, or temporal + causal relationship in a sentence. In the presence of terms such as time, year, begin, finish, date, month, and so on, the sentence demonstrates a temporal relationship. When the relationship is temporal, the question type is when, and when the relationship is causal, the question type is why.

Following the identification of question types, the target argument is identified. A clause or a sentence can be the target argument (structural or anaphoric) for a discourse connective. The system is then manually assessed using syntactic modifications to produce questions. Two graduate students with greater English proficiency performed the evaluation. On a scale of 1 to 4, evaluators were asked to rate the questions. Finally, the overall rating of the system is 5.8 out of 8.

Another study [20] developed a rule-based question-generation system for both simple and complex sentences. Its goal is to create factoids and descriptive answerable questions. The researchers gathered

290 simple and complex sentences from which 718 different possible questions were created manually with an overall agreement of 80% between two groups of annotators.

As a first step, various text sentence fragments known as segments identify clauses from those segments, and segments are split from the original text. A clause of the sentence is the section that comprises a noun phrase followed by a verb phrase. After each segment is tokenized, POS taggers and parsers are used to determine the parts of speech of the words as well as the chunked output of a segment. Using the specified criteria, each segment is checked for a clause. If a segment is not identified as a clause, however, a question is formed by substituting a chunk.

Based on the nature of the sentences, the researchers made two sets of rules: Question Specific Generation Rules and Question Specific Disambiguation Rules. The first rule is used to identify the positions where interrogative words (who, what, when, where...) will be placed, and the second rule is used to avoid ambiguity between the interrogative words (for example, between what and whom: If the first noun in the chunk is identified as 'PERSON' by the NER tagger, then "whom" will be used instead of "what" at the time of question generation).

Finally, the researchers used a rating method (acceptable, ambiguous, and incorrect) to evaluate the system-generated questions. Based on this rating method, the generated questions were checked by two sets of human evaluators. Finally, 80% agreement was achieved.

In complicated sentences, relative pronouns or relative adverbs connect or introduce the relative clause that is contained within the matrix clause. Examples of these in English include who, whom, which, where, when, how, and why. Relative pronouns and relative adverbs convey accurate information about the syntactic connection between sentence components. Khullar, Payal, et al [76] used a syntax-based approach to construct multiple natural language questions from difficult English sentences that comprised relative pronouns and relative adverbs. They used three sets of rules to construct questions automatically by using wh-pronouns and wh-adverbs. Each of the three rule sets has a total of 10 rules that are supported by linguistic principles, and each relative pronoun or relative adverb in the sentence is first checked for a set of requirements before being input into the rules. The relative pronoun and relative adverb, as well as the kind of relative sentence, are used to generate questions (restrictive or unrestrictive).

The first rule set analyzes the sentence for the existence of modals and auxiliary verbs, as well as the tense and aspect of the root verb. The do-insertion before Noun Phrase or aux/modal inversion is conducted based on this information. The second rule set detects relative pronoun dependency, as relative pronouns can occasionally operate as the subject of the relative clause's verb. The third rule set analyzes the clause verb's tense as well as its number agreement. The researchers then evaluated the system using 300 sentences from the collection of sentences with relative clauses, where four independent human evaluators gave scores to questions generated by the system. Evaluators assign a 3 to questions that are syntactically well-formed and natural, a 2 to questions that contain a few syntactic faults, and a 1 to questions that are syntactically unacceptable. Similarly, for semantic correctness, raters provide a score of 3 when the questions are semantically right, a score of 2 when they have an unusual meaning, and a score of 1 when they are semantically unacceptable. Finally, the researchers received a 9.44 out of 10 overall rating.

The study at [77] presents an interactive way to generate factual questions from unstructured text. This work converts input text into a structured collection of features that are then used to generate questions. It learns how to generate questions from a set of sentence-question combinations and improves its ability through user input. Its learning mechanism is based on a combination of reinforcement learning and supervised learning machine learning techniques. The learning process starts with an initial set of pairings made up of declarative statements and assigned questions, and it gradually learns how to transform sentences into questions. The process is also enhanced by user feedback on previously generated questions.

The researchers used a data-driven approach, which includes extracting features from text and selecting the best class of questions based on these features. Each sentence class is assigned a set of possible questions, which are chosen based on feature similarity. Their system is divided into three components. The first module is a text preprocessing component that yields features.

The second module explores the database for possible transformation rules that can be used to generate questions. The matching criteria are determined by the similarity of features obtained by the first module. The final module assesses the quality of the questions generated by each rule. The researchers next evaluated the quality of the generated questions using the dataset of question generation shared task and evaluation challenge (QGSTEC). It contains 81 sentences and 180 desired questions. Questions generated by participants in the question generation challenge were evaluated by human evaluators on a scale of one to four from various viewpoints (correctness, ambiguity, diversity). The average correctness of all questions and the number of generated questions were used to calculate the number of correct questions.

One of the tasks performed by natural language processing is question answering. Nonetheless, creating a question-answering system needs a large amount of data. The study at [78] attempted to explore how to generate questions from given passages using neural networks, with three goals in mind: the training data should require little or no human effort and should reflect commonly-asked question intentions; the questions should be generated based on natural language passages and should be of high quality; and the generated questions should be useful for question answering tasks. The training data comes from community-question-answer websites (e.g., Yahoo Answers and Quora).

The question generator in this work contains four components: Question Pattern (QP) Mining, Question Pattern Prediction, Question Topic Selection, and Question Ranking. These components are used to extract frequently-asked question patterns from a large set of questions, which predicts top-N question patterns using a retrieval-based method or a generation-based method, to select a phrase from the question topic based on a predicted question pattern, and to rank all generated questions using a number of features, respectively.

This study suggested two approaches to question generation: one is a retrieval-based method based on convolution neural networks (CNN), and the other is a generation-based method based on recurrent neural networks (RNN). The retrieval-based method accepts a passage and a question pattern as input and returns vector representations of both. In addition, the generation-based method is based on sequence-to-sequence model. To rank produced question candidates, the study also used features that include question pattern prediction score, question topic selection score, QA matching score, word overlap between question and passage, and question pattern frequency. Finally, the researchers include the QA pair generation work into an end-to-end QA task and achieve significant improvements.

The study at [22] used a fully data-driven approach for the generation of questions as a sequence-to-sequence learning problem that directly maps a sentence from a text passage to a question. The researchers used different encoders for encoding the sentence and the paragraph. They made the experiment based on the processed SQuAD dataset.

The model is trained using sentence question pairs from this dataset. The dataset contains 536 articles and about 100k questions regarding them. The researchers used Stanford CoreNLP for pre-processing, including tokenization and sentence splitting, before lower-casing the full dataset. They select the sentence containing the answer and use it as the input sentence using the offset of the answer to each question. In both the encoder and decoder, the LSTM hidden unit size is set to 600, and the number of LSTM layers is set to 2.

Finally, the model that only encodes sentence-level information achieves the best performance across all metrics, and the researchers concluded that adding pre-trained embedding generally helps, though encoding the paragraph causes the performance to drop slightly, which makes sense because, in addition to useful information, the paragraph contains many noises.

The neural encoder-decoder model was also used in the work at [10] to generate meaningful and diverse questions from natural language sentences. The encoder reads the input text and the answer location to create an answer-aware input representation, which is then sent to the decoder to form an answer-focused question. The answer span in the input sentence is denoted by the answer position characteristic, which is required to generate answer-relevant questions.

To aid in sentence encoding, lexical characteristics such as part-of-speech tags and named-entity tags are included. Furthermore, the attention mechanism in the decoder creates a response to the sentence's particular question. The encoder is built with Gated Recurrent Unit. This encoder is designed to read inputs. The word vector, lexical feature embedding vectors, and answer location indication embedding vectors are concatenated as the encoder's input. The researchers used the answer location feature to pinpoint the target answer in order to generate a question with regard to a specific answer in a sentence. The BIO tagging technique is used to denote the location of a target answer, with tag B denoting the start of an answer, the tag I continuing the response, and tag O denoting words that do not form part of an answer.

The attention-based GRU decoder examines the preceding word-embedding and context vector and decodes the sentence and answer information to produce questions. Furthermore, the researchers attempted to overcome the problem of rare and unknown terms by using the pointing mechanism to copy rare words from the source sentence. When decoding a word in this method, the copy switch takes the current decoder state and context vector as input and generates the likelihood of copying a word from the source sentence. The attention probability is then used to choose which word to duplicate. Finally, the evaluation of this system shows a BLEU-4 score of 13.29, which is found to be a 2.05 BLEU improvement.

The study at [11] also designed Clue Guided Copy Network for Question Generation (CGC-QG), which is a sequence-to-sequence generative model with a copying mechanism that utilizes a variety of novel components and strategies to improve question generation performance. CGC-QG employs a multi-task labeling technique to decide whether a question word should be copied from the input material or generated instead, allowing the model to learn the precise boundaries between copying and generation.

The passage encoder used in this study uses as input the prediction provided by a clue word predictor, which aids in determining if each word in the input passage is a hint to be copied into the target question. To decrease complexity, clue prediction and question generation with multi-task learning are trained together. The encoder is Gated Recurrent Unit that takes as input word embeddings, answer location indicators, lexical and frequency features of words, and the output of the clue word predictor. Another GRU with a copying mechanism is used in the decoder to generate question words sequentially depending on the encoded input passage representation and previously decoded words.

The researcher utilized a dataset including 120K questions and their related answers extracted from SQuAD, as well as documents containing CNN news articles. Finally, the researchers obtained BLEU-4, ROUGE-L, and METEOR results of 17.55, 44.53, and 21.24, respectively, while the comparable prior state-of-the-art findings from other methodologies are 16.17, 44.24, and 19.67.

A paragraph often contains much more context than a sentence. To generate questions from a paragraph or sentence, this [5] study used a sequence-to-sequence attention model with a maxout pointer mechanism and a gated self-attention encoder. The researchers used a Recurrent Neural Network (RNN) to encode the data. In this case, a gated self-attention mechanism is used to aggregate information from the whole passage and incorporate intra-passage dependence in order to improve the encoded passage-answer representation at each time-step. It consists of three steps: accepting encoded passage-answer representation as input, matching against itself to compute self-matching representation, and merging the input and self-matching representation.

2.7.3. Question Generation for the Chinese Language

This study [12] generated questions from key sentences using a template-based method. The good questions are then known by ranking the questions using a multi-feature neural network model. The methodology for this study consists of three steps: identifying key sentences of text using an adapted TextRank, generating questions using rule-based templates, and then the questions are ranked using a multi-feature neural network model.

An adapted TextRank, which converts text to a graph and calculates the score of each sentence node, is used to find essential sentences from text or paragraphs. When a sentence's score exceeds the predefined level, the sentence is selected. The input text was divided into sentences using different terminators (e.g., question mark, full stop, exclamation mark...). Then divide sentences into words. Each term is evaluated using Term Frequency–Inverse Document Frequency. A sentence can therefore be represented by a word vector. Then, the similarity score is calculated.

The researchers used rule-based templates to generate targeted questions from the parser tree. To select the top key questions, a multi-feature neural ranking model is created in the question ranking step.

The generated questions and text are used to import around twelve features (such as the number of tokens in the question and answer, the number of named entities in the question, the type of question, and the score of key sentences). Human evaluation is also used. In human evaluation, all generated questions are evaluated by humans, and each question is judged by two individuals (as excellent, borderline, or bad), giving an average human evaluation score of 2.73, whereas the baseline scores are 1.79, 1.95, and 2.12.

2.7.4. Question Generation for the Portuguese Language

The paper [79] proposes three different approaches to generate factual questions in Portuguese. The first does a syntax-based analysis on a given text using data from Part-of-Named Entity Recognition (NER) and Speech tagging (POS). The second method uses Semantic Role Labeling (SRL) to do semantic analysis on the sentences, while the third approach uses Dependency Parsing to extract the underlying dependencies inside phrases.

To integrate the findings of POS tagging with NER, the researchers use a syntax-based method. After that, the entities are detected using NER and matched with the POS pattern. The researchers searched for matches to their guidelines using regular expressions. When a match is found, it indicates that the statement may include one or more questionable facts.

In order to validate the result, the researchers designed a survey with 15 questions generated automatically from four Portuguese books. Five questions were created using the syntax-based method, five using the semantic-based method, and five using the dependency-based method. After generating questions from the Portuguese texts, the selection criterion was to choose five questions for each of the approaches. Furthermore, these questions serve as illustrations of the kind of questions that may be generated for each approach. Given the emphasis on factual questions, instructors were asked to rate the generated questions based on objectivity, grammaticality, answerability, and question length. On a scale of one to five (lowest to highest), each of these criteria was scored, with the generator achieving an average agreement of 3.48, 3.04, and 3.45 out of five for syntax-based, semantic-based, and dependency-based approaches, respectively.

Summary

In this chapter we made extensive literature review. The concept of question and question types including factual questions, deep questions, gap fill questions and multiple questions were explained. The general architecture of question generation systems with its components were also discussed. In addition, the different activation functions, loss functions, optimizers and attention mechanism that are being used in deep learning are discussed. Moreover, this chapter included the review of question generation approaches which are traditional, machine learning-based and deep learning-based approaches. Finally, we made a review of question generation studies for different languages including Amharic, English, Chinese and Portuguese which are related to our work. The table wise description of related works is shown in Table 2.7.1 below.

Table 2.7.1 Summery of related works

Author	Title	Problem	Method/tool	Result
Andinet Assefa [8]	Automatic Generation of Amharic Math Word Problem and Equation	Absence of automatic math word problems and equations generator	A rule-based approach is used in which sets of hand-written rule sets are utilized to generate math word problems and equations.	90% accuracy
Getaneh Damtie [9]	Automatic Amharic Factual Question Generation from Historic Text Using Rule Based Approach	Question generation	Sets of rules are used to convert the Amharic sentence to factual question.	86.4% accuracy

Manish Agarwal et al [21]	Automatic Question Generation using Discourse Cues	The sense of the discourse connective influences the question-type.	Rule-based approach is used to identify question type and target argument. Then syntactic transformations are made to generate questions	The overall rating of the system is 5.8 out of 8 for five Wikipedia articles and the total number of questions generated for this dataset are 150.
Rubel Das et al [20]	A Rule based Question Generation Framework to deal with Simple and Complex Sentences	Previous rule-based question generating systems were designed for basic sentences, but what about complex sentences in the same approach?	Used two different sets of rules based on the natures of the sentences: Question Specific Generation Rules and Question Specific Disambiguation Rules.	Used rating scheme (acceptable, ambiguous and incorrect) for evaluating the system-generated questions. Finally, 80% of agreement was achieved for two sets of human evaluators.
Payal Khullar et al [76]	Automatic Question Generation using Relative Pronouns and Adverbs	Exploiting wh-pronouns and wh-adverbs is necessary to generate natural language questions using syntax-based system.	Rule based approach using three sets of rules: the first checks the tense and aspect of root verb, the second rule identifies the dependency of relative pronoun and the third checks the tense of relative clause.	Finally, the researchers obtained an overall rating of 9.44 out of 10
Hai-Tao Zheng et al [12]	A Novel Framework for Automatic	Previous question generation works does not rank the	Identifying the key sentences of text with an adapted TextRank, constructing questions	An average human evaluation score of 2.73 based on two humans' evaluation

	Chinese Question Generation Based on Multi-Feature Neural Network Model	generated questions from sentences.	according to rule-based templates, and ranking questions with a multi-feature neural network model.	while the baseline score is 1.79, 1.95, and 2.12.
Miroslav Blstak and Viera Rozinajov [77]	Machine Learning Approach to the Process of Question Generation	The previous works on question generation were using handcrafted rules, which is time consuming and hard to make changes.	Combination of machine learning techniques: reinforcement learning and supervised learning	-
Nan Duan et al [78]	Question Generation for Question Answering	The previous works on generating questions from sentences using rule based or machine learning approaches does not generate questions from passages.	Neural networks: CNN & RNN	-
Qingyu Zhou et al [7]	Neural Question Generation from Text: A	Previous works on question generation does not generate answer aware questions.	Neural encoder decoder model using Gated Recurrent Unit	BLEU-4 score of 13.29

	Preliminary Study			
Yao Zha et al [5]	Paragraph-level Neural Question Generation with Maxout Pointer and Gated Self-attention Networks	Repeated occurrence of words in the input sequence tends to cause repetitions in output sequence, especially when the input sequence is long.	Recurrent Neural Network with maxout pointer	BLEU-4 score of 16.3
Bang Liu et al [11]	Learning to Generate Questions by learning what not to Generate	A strategy should be designed to identify whether a question word come from the input passage or be generated instead.	Sequence-to-sequence generative model with copying mechanism	BLEU-4 score of 17.55

2.8. The Amharic Language

2.8.1. Introduction

Amharic/አማርኛ is the working language of the Federal Democratic Republic of Ethiopia, and as such has official status across the country. Many federal states, most notably Amhara and the multi-ethnic Southern Nations, Nationalities, and Peoples area, as well as the South West Ethiopia Peoples region, use it as their official or working language. It has been the working language of the government, the military, and the Ethiopian Orthodox Church throughout modern history. Outside of Ethiopia, Amharic is spoken by immigrants (most notably in Egypt, Israel, and Sweden) as well as Eritrean deportees. It is written in a writing system called Fidel or Abugida, which was developed from the now-extinct Ge'ez language [80]. Though Ethiopia has several languages (including Tigrinya, Amharic, Affan Oromo, and others), Amharic is the most popular and commonly used[81].

The Amharic language has been declared to have the following word categories: ስም (noun), ግስ (verb), ቅፅል (adjective), ተውሳከግስ (Adverb), መስተዋድድ (preposition), and ተውላጠ ስም (pronoun) [82].

ስም/Noun

Nouns in Amharic can be masculine or feminine. Suffixes are used to indicate whether a word is masculine or feminine. Some nouns have both masculine and feminine genders, but others only have one. The feminine gender is used to denote both femaleness and smallness. For example, ቤቱ ትንሽ ነች። Plurals are formed by adding ዎች or ኦች whether the word ends with a vowel or consonant. For example: በሬዎች, እናቶች, ሰዎች, ልጆች, ፈረሶች, ቁራዎች e.t.c.

ግስ/Verb

Amharic verbs are words that inflect person, number, gender, mood, voice, and polarity using roots and affixes. Verbs are in agreement with their subjects. The use of verb agreement with objects is optional. In Amharic, verbs are usually put at the conclusion of a phrase.

ተውሳከግስ/Adverb

An adverb can be used to modify a verb by adding an extra idea to the sentence. The Amharic adverbs are limited in number and include እንደገና፣ ዛሬ፣ ትላንት፣ገና፣ ቶሎ፣ …

ቅፅል/Adjective

The adjective is any word that modifies a noun or an adverb, which really comes before a noun (Example ጎበዝ ልጅ፣ በጣም ጎበዝ). Another specific property of adjectives is, when pluralized, it repeats the previous letter of the last letter for the word (e.g., ትንሽ፡ ትንንሽ).

መስተዋድድ/Preposition

A preposition is a word which can be placed before a noun and perform adverbial operations related to place, time, cause and so on; which can't accept any suffix or prefix; and which is never used to create a new word. It includes ወደ፣ እንደ፣ ስለ and ከ.

ተውላጠ ስም/Pronoun

This category further can be divided as deictic specifier, which includes እሱ፣ እኔ ፣ አንተ፣ አንቺ፣ እነሱ; quantitative specifier, which includes ጥቂት፣ አንዳንዱ፣ አንዱ; and possession specifier such as የአንተ፣ የኔ and የሱ

2.8.2. Amharic Sentences

A sentence in Amharic can be a statement that is used to declare, explain, or discuss an issue; an interrogative sentence that can be used for questioning; exclamatory and imperative [82]. Sentences are made up of noun phrase and verb phrase combinations. The noun phrase and the verb phrase are further divided into different particles such as other sub noun phrase and verb phrase, noun, adjectives and so on. The interrogative sentences also have the same structure with little modifications and an introduction of question particles (interrogative words). For example: For the sentence: ልጁ ገበያ ሄደ።, the question can be ልጁ የት ሄደ?

The verb goes at the end of the sentence in subject/object/verb (SOV) order where the word order in English is SVO. For instance, in the Amharic sentence "ልጁ ደብተር ገዛ።", the word "ልጁ" is the subject, the word "ደብተር" is the object of the sentence, and "ገዛ" is the verb.

Amharic words contain lots of prefixes and suffixes as a result the sentences can be short in a number of words. For example, a single Amharic word "አስመጣችላቸው።" is a sentence which describes "She made something come for them".

For interrogative sentences, if it is a yes or no question, the sentence order stays the same but the intonation is that of a question, and no question word is used. If not, the question word is placed right before the verb. For instance, from the Amharic sentence "ጌታሁን ደብዳቤ ጻፈ።", the yes/no question can be ጌታሁን ደብዳቤ ጻፈ? This question is similar to the sentence except for the intonation.

2.8.3. Amharic Punctuation Marks and Numerals

The Amharic documents collected must be pre-processed before proceeding to the other components of the question generation system. Sentences utilize different punctuation marks for separating one from the other [83]. In Amharic, there are different punctuation marks used for different purposes. For example, sentence indexing will be done with the help of the Amharic full stop (።) for separating different sentences. So, the system should understand this punctuation mark indicates the end of a sentence. In the old writing systems, a colon (two dots ፡) has been used to separate two words. There are lots of articles (e.g., some Amharic articles on Wikipedia) that are written using a colon as a separator between words. These days the two dots are replaced with whitespace. In addition, ነጠላ ሰረዝ (፣ or ፤) is also used to separate lists of ideas just like the comma in English.

Similarly, numerals have a greater impact on Amharic question generation systems. Since numbers are stored in different formats, some kind of standardization should be applied to help the question generation system. Ethiopic and Arabic numbers should be normalized to the same standard to make a sentence suitable for the generator component. In Amharic, numbers can be represented using Arabic symbols. It has also its own number representations, Ethiopic number representations. Similarly, numbers can be represented in a word alphanumerically which is a representation of numbers in words [82].

2.8.4. Amharic Question Formation

A question is a language term that is used to make an informative request [4]. Questions may be asked to determine what we do not know or to ensure that we do know something. In Amharic, we use adverbs to ask questions about which we have no information, and additional words or tones to indicate that the sentences are interrogative [84].

አበበ መቼ መጣ? When did Abebe come?

አበበ መጣ ወይ? Did Abebe come?

አበበ መጣ? Did Abebe come?

The first example shows that the interrogator does not know when Abebe is coming so will add the adverb when (መቼ). In the second example, the interrogator knows that Abebe is coming but wants to be sure. In this sentence the last word ወይ indicates the sentence is interrogative and it can be used to be sure about something. It is placed at the end of the interrogative sentence.

The third sentence has no interrogative word. The only thing that indicates it is a question is the question mark and the sound (tone) of the interrogator and the way the sentence is read [85]

Questions might be raised about some action or condition, about the performer of an action, about the agent that performs the action or time and place, about the cause of the action or aim of the action, how the action is performed or techniques used to perform the action, and so on [85]. All of these types of questions might be within a single sentence.

In every language, questions are formed using interrogative phrases and a question mark (?), which is inserted at the ending of the question. The question mark is placed at the end of the sentence to indicate that it is a question. For instance: ቤቱን ማን አሳየህ? The interrogative words in Amharic are placed near the end of the sentence most of the time. There should be extra information to determine the question type besides the interrogative words such as question focuses or complex grammatical structure analysis of the sentence [86].

In English, the interrogative words (WH words) who, what, where, when, why, how … are used to construct a question. There are a number of interrogative words in Amharic that will help in constructing a question [86]. The well-known Amharic factoid interrogative words are ማን, ምን, የት, ስንት, መቼ. However, depending on the answer, the interrogation words can be combined with prepositions, suffices and prefixes like የማን, እነማን, ማንን, እነማንን, የማንን, ከማንኛው, ማንኛው, ማንኛይቱ, ማንኛዋ, ማንኛቸው, በማን, የቱ, በየት, የቷ, የቲቱ, የቶቹ, ወዴት, ከየትኛው, የየት, የትኛው, የትኛዋ, የትኞቹ, ከየት, እስከየት, ወደየት, የየትኛው, በየትኞቹ, የምን, ምንህን, ምንሽን, ምናችሁን, ምኔ, ምኑን, ስለምን, እንደምን, በምን, እስከምን, ከምን, ወደምን, ምንጊዜ, ምንያህል, በመቸ, እስከመቸ, ለመቸ, ስንቱ, ከስንት etc.

For example,

Sentence 1: ልጁ የወንድሙን ጃኬት ለበሰ። Question: ልጁ የማንን ጃኬት ለበሰ? Answer 1: የወንድሙን

Sentence 2: አስቴር መፃፍህን ስጠኝ አለች። Question: አስቴር ምንህን ስጠኝ አለች? Answer 2: መፃፍህን

Sentence 3: አስተማሪው ስልካችሁን አጥፉ አለ። Question: ምናችሁን? Answer 3: ስልካችሁን

For the first sentence's question the answer is "የወንድሙን" and the interrogation word is "የማንን". In this interrogation word, "የ" and "ን" are taken from the prefix and suffice of the answer. In the second sentence's question the interrogation word is "ምንህን". The last two letters from this interrogation word are taken from the answer "መፃፍህን". In the third example also, the interrogation word takes the suffice "ኣችሁን" from the answer text "ስልካችሁን".

Most of the time if the interrogation word for a certain question consists of suffices or prefixes or prepositions, it is true that the answer part contains what is combined with the interrogation word.

Sentence	Question	Answer
አማፂ ሃይሎች ለመዋጋት ተዘጋጁ።	አማፂ ሃይሎች **ለ**ምን ተዘጋጁ?	**ለ**መዋጋት
እሱ የሰባት ዓመት ልጅ አለው።	እሱ **የ**ስንት ዓመት ልጅ አለው?	**የ**ሰባት
የእኛ ቤተሰብ ቁጥር ስምንት ነው።	**የ**ማን ቤተሰብ ቁጥር ስምንት ነው?	**የ**እኛ
ልጁ አባቱን ይወዳል።	ልጁ ማን**ን** ይወዳል?	አባቱ**ን**

The first question includes the interrogation word formed by combining the prefix “**ለ**” with the word “**ምን**”. This prefix is obtained from the prefix of the answer word “**ለ**መዋጋት”. This is the same for the rest of the above-mentioned examples as indicated by bold font weight.

Even from a single Amharic word, we can create one or more questions as a single Amharic word can be a sentence by itself. From the Amharic word “መጣ” one can construct questions like “ማን መጣ” and “መቼ መጣ”. From the word “በላ”, one can construct questions like “ማን በላ”, “መቼ በላ”, “ምን በላ”.

Chapter Three

3. Research Methodology

After a problem is identified one should have some method, deploy tools and techniques, apply some method as well as pass some steps to conduct research for the identified problem. This over all process is a research methodology. There are two research paradigms [87]. These are behavioral science (the former natural science paradigm by [88]) and design science research paradigms. According to [87] behavioral science "addresses research through the development and justification of theories that explain or predict phenomena related to the identified business need" whereas design science "addresses research through the building and evaluation of artifacts designed to meet the identified business need".

Among the mentioned above paradigms, this study follows a design science research methodology as it is suitable for the objective of this research. From design science research, the result is an artifact that is developed and implemented to address some problem. The design science process includes six basic steps according to [89]. These are identification of problem, objectives definition for a solution, design and development, demonstration, evaluation, and communication.

3.1. Problem identification

The first step in the design science research process is problem identification. The problem regarding automatic question generation from Amharic sentence is identified and understood at this stage. An observation and literature review are performed to have a better understanding of the problem under consideration. Then we move on to the definition of objectives for designing the solution to the identified problem.

3.2. Definition of objectives for a solution

After identifying the problems to be solved, the study objectives should be clearly defined. Then, with our aims in mind, we design and implement the study we are working on. As a result, having recognized the problems, the objectives of this study are well defined, and understanding the objectives, we proceed to the design and development of the study.

Here, the concept of neural machine translation, which is one of the most imperative areas in the field of natural language processing [90], helped us to define our objective. Neural machine translation translates a sentence in one language to its equivalent sentence in another language. Due to the different meaning of words in a sentence, one sentence can be translated into more than one sentence in other language. Likewise, in our task more than one questions can be generated from a single sentence. That means:

Given a sentence S= (s_1, s_2…, s_N) where s_i is a token and N is the length of the sentence, we are going to generate question Q= (q_1, q_2, q_3…, q_M) with length M, which has its answer embedded in S. So, our objective here is to find Q so that the conditional probability p(Q|S) is maximized. This indicates that the probability of each q_t depends on the previously generated words and the input sentence S.

3.3. Design and development

The design and development phase of this study includes developing a design and architecture to automate the generation of factual Amharic questions from a sentence. The architecture we use is an encoder-decoder architecture where there are two recurrent neural networks: the encoder, which reads the input sequence one word at a time and converts it into a vector representation, and the decoder, which generates the output sequence according to the encoder's output and previously generated words.

The designing process starts with data collection and analyzing the data, according to [91]. For this research different types of data are collected from different sources. Most of the data for training the word embedding model are collected from Amharic WIKIPEDIA website articles. And the rest are collected from the internet and through web form.

These days electronic data is available over the internet. The very challenging task is getting relevant data for our specific study. In this study, we need a sentence-question pair and/or a sentence-question-answer triple dataset. However, no source provides a large set of this dataset. So, to collect our sentence we started manually distributing sentences for Amharic familiar individuals to create possible factual questions. Nonetheless, we could collect only 5105 questions from a voluntary two Amharic teachers and 38 university students who are studying Amharic. This is almost nothing for training a deep learning model. As a result, we shift to another method. In this method we tried to collect data through the web form over the internet. We created a web form that has enough description about how to fill it.

First of all, we created a data collection system that uses form to accept sentences/ዓረፍተ ነገር, questions/ጥያቄዎች from the sentence, answer/መልስ to the question and Type of question/ የጥያቄው አይነት as a choice like Who, Where, When, What and how much/many. To distribute the webform over the internet, we first collect very important public information especially new Vacancy notices, and we store those vacancy links in our database. We then share the links of that important information, vacancy announcements and other sources on social medias including Facebook and Telegram. Specially telegram groups with a larger number of members. When an internet user clicks on the link we shared, it will redirect them to our data collection webform with a brief description about what could do and why they are redirected to it.

The web form consists of examples that describe the process of form filling. When the user completes filling out the web form and submitting it, our system then provides the target link with a great acknowledgment. However, we got the problem of repetition of sentence-question-answer-type quadruples. Many of the sentences are almost the same except for the differences in subject like አበበ መኪና ገዛ, አለሙ መኪና ገዛ, ሶስና መኪና ገዛች…

So, we used our third method which ends with an interesting result. In this method, we collect 34,302 sentences from different media's websites including waltainfo.com and fanabc.com. On these websites there are interesting sentences prepared by professional journalists. More interestingly, the sentences cover various domains like social, economic, political, sport and others which helped us to create an open domain dataset.

We then store the collected sentences in our database. Now, instead of letting the user fill the sentences on their own, we provide the sentences automatically through our data collection system. That means, that when the user clicks on the link we shared over the social media, the user will be redirected to our data collection system web form. In this web form, the system displays a sentence from the previously-stored sentences from which no question was created from it by another user. Then the user's task is to create a question from the displayed sentence, answer it based on the sentence and select the type of question from the provided lists. The data collection system also helps the user by filling the answer filled automatically when they select from the sentence in order to save their time and keep them from getting bored while writing. After completion of form filling the system redirects the user to the target link again with acknowledgment.

Finally, we collect 60,023 sentence-question-answer-type quadruples. For this task we need only sentence-question pairs and/or sentence-question-answer triples. We collect the other things like answer type for future use. Once the necessary data is collected, we should make it understandable for the machine. To do this, we performed character normalization, special character removal and tokenization and many more (see detail in section 4.2.1) on the collected data.

3.4. Demonstration

The design science research method verifies the application of the artifact on the problem according to Peffers et al [91]. We used a prototype to demonstrate this artifact and understand how valuable and usable is this artifact for the intended users.

We used an encoder-decoder architecture with an attention mechanism where the encoder is an LSTM that encodes the input and produces the context vector which will be used as a decoder initial step from the decoder after passing through the attention mechanism. In order to implement a design and build a prototype, we used different techniques and tools such as Python programing language, and the TensorFlow library. In this research we implement our model using the Python language which is a programming language that allows us to work quickly and efficiently integrate systems [92]. More specifically, the TensorFlow library is used in developing the model. TensorFlow is an end-to-end open-source machine learning platform. It has a rich, flexible ecosystem of tools, libraries, and community resources that allow academics to push the limits of machine learning and developers to rapidly build and deploy ML-powered applications [93]. Python programming language is also used in this study to train a word embedding model, which consists of the similarities between words in the form of vectors.

3.4.1. Word2vec model development

To understand text inputs, machines need numeric representation of inputs. This research trains a model that can convert the input words into vectors. These vectors then become an input to the encoder when necessary. The vectors in word2vec are not just numbers, rather they contain the similarity information between words.

To develop this model the researcher collects the data from the internet. We scraped the Amharic Wikipedia website using BeautyfulSoap4 which yields 191,834 sentences. We then perform preprocessing tasks including special character removal, short word expansion, normalization and tokenization. The preprocessed data is fed to the Word2vec algorithm, and trained (see section 4.2.2 for detail). This model is used during the training and testing of the Automatic Amharic Question Generation model.

3.4.2. Training and generation

The training's purpose is to reduce the training corpus's negative log-likelihood with respect to all model parameters. For training our model. we use a strategy called teacher forcing, which means using the exact word from the original question when feeding the next target word to the decoder rather than the decoder’s own prediction. This is because of the fact that in the early stages of training, decoder makes many mistakes during generation, and feeding those mistakes as the next input to the decoder can make the training process harder and longer. However, during testing, we feed the model’s own output as the next target word to the decoder. This technique prevents the model from learning from mistakes in the process.

3.4.3. Encoder

An encoder network used in the model development is an RNN that maps an input sequence into a word vector and then converts this word vector into hidden states. It takes a single element from the input sequence, collects information for that element, and forwards it. The input sequence in this question-generation task is a collection of all words from the sentence. Each word is represented as a vector through word embedding, which transforms each word into a fixed-length vector.

Each word is represented as a vector using the word embedding, which converts each word into a vector of fixed length. The internal state then learns what the LSTM has read until time step t. For example, when t=2, it remembers that LSTM has read two words or tokens of the input sequence. Then the final state of the LSTM encoder contains the heart of the entire input. In addition to the final internal state, the LSTM predicts an output when it has read the entire input sequence. However, this output is of no use for initializing the decoder, so we discard it. The final state of the encoder is then used to initialize the decoder.

In sequence-to-sequence learning, the input sequence is encoded understandably so that it can help to predict the output sequence. Our LSTM encoder take the sentence-question pairs. These pairs are separated by tab in the way shown below.

ቋንቋዎችን ማሳደግ የዜጎች ሀላፊነት ነው።	ቋንቋዎችን ማሳደግ የማን ሀላፊነት ነው?
ኢትዮጵያ ብዙ ታሪካዊ ስፍራዎች አሏት።	ኢትዮጵያ ስንት ታሪካዊ ስፍራዎች አሏት?
የአስራ ሁለተኛ ክፍል ተማሪዎች ዉጤት ዛሬ ተለቀቀ።	የአስራ ሁለተኛ ክፍል ተማሪዎች ዉጤት መቼ ተለቀቀ?

The dataset is sorted by the length of a sentence from the shortest to the longest. It is known that the Amharic Full stop(።) and Question mark (?) punctuations are important to show the end of sentence and question respectively. In fact, these operators are appended to the last word of a sentence or question. However, in this work appending these punctuations is unnecessary as it makes the model to understand the appended token as a single word. For instance, from the first example “ነው” and “።” are put together in the sentence side and “ነው” and “?” are put together. That means “ነው።” and “ነው?” are considered two words. To prevent this, we create a space between a word and the punctuation following it as follow:

ቋንቋዎችን ማሳደግ የዜጎች ሀላፊነት ነው ።	ቋንቋዎችን ማሳደግ የማን ሀላፊነት ነው ?

We then inserted a start and end token to the sentence to tell the algorithm when to predict and when to stop. The word tokenization is performed on both the sentences and the question like ['ቋንቋዎችን', 'ማሳደግ', 'የዜጎች', 'ሀላፊነት', 'ነው', '።'] for the above sentence, and we create a Vocabulary with word index (mapping from word → id) and reverse word index (mapping from id → word).

We use a batch size of 64, an embedding size of 300, and a unit of 1024 which is the dimension of the inner cells in LSTM. The encoder then takes the input vocabulary size, embedding dimension, the encoder units, and batch size parameters and start producing sample output, hidden state and cell state.

We also apply the same process for the Amharic Sentence-Question-Answer triple corpus. The difference is that we added the third column by separating it with an <ans> tag as shown below:

ቋንቋዎችን ማሳደግ የዜጎች ሀላፊነት ነው ።	ቋንቋዎችን ማሳደግ የማን ሀላፊነት ነው ?	<ans>የዜጎች

Here, the <ans> token is added to separate the answer span from the question column. This token is also important during post-processing to identify the answer from the predicted text. We used this dataset to develop a model that can take Amharic sentences as an input and generate factual questions from the input sentence with the possible answer. In another work [7] answer span is used to increase the performance of the question generation model.

3.4.4. Decoder

The decoder generates the output sequence according to the encoder's output and previously generated words. Specifically, an attention-based decoder focuses on a particular range of the input sequence during the automatic question generation task [51]. This is similar to the process of forming a question in a human's mind, where one pays attention to a specific part of a sentence and creates a question regarding that part.

The decoder itself is another LSTM whose input is the encoder's output. Then this decoder creates a sequence of words as the question. Only the final output of the encoder, known as the context vector, is utilized to establish the hidden state of the decoder in its most basic version. In this way, the context vector is responsible for encoding the whole sentence. We then apply an attention mechanism to ease the burden on the context vector. This method allows the decoder to concentrate on different areas of the encoder's output.

The decoder predicts the next word given the context vector and the previously predicted words. The process stops when the end-of-sentence token is generated. We utilized a softmax layer to get the prediction distribution.

3.5. Evaluation

For evaluating automatic question generation systems, recall, precision, accuracy and F-Measure can be used [50]. According to [65], the development of the question generation system and evaluation methodology should be parallel. The systematic analysis of evaluation methodologies could play a central role in the effort to construct machines capable of achieving linguistic communication standards that are human-like. First of all, two Amharic teachers judged the questions generated from the system. Two criteria are considered in their judgment: syntactic correctness and relevance. Syntactic correctness indicates the grammaticality and fluency of the generated questions and relevance demonstrates whether the generated question is meaningful and related to the sentence it is generated from.

The judges were asked to classify the questions as either correct or incorrect with Syntactic correctness and relevance. They performed the evaluations on 227 questions generated from 105 randomly selected Amharic sentences. Finally, since this research used accuracy, as an evaluation metrics, we define Accuracy [94] as defined below:

$$\boxed{Accuracy = \left(\frac{CGQ}{TGQ}\right) \times 100}$$

Where CGQ=Correctly generated questions and TGQ=Total Generated Questions by the system.

3.6. Communication

Communication is the final stage of the design science research process. In this step the final results from the experiment should be done. In addition, the complete research process was thoroughly documented and communicated as a thesis work.

Chapter Four

4. Design of Amharic Question Generation

This chapter discusses the design and prototype development of the AQG model using recurrent neural network. The major components, that are utilized to accomplish the Amharic QG, in the architecture of AQG system are the training component which in turn consists of Input Preprocessing and Input Processing sub-components, and the Testing component which consists of Question Generation (QG) sub-component and shares the preprocessing subcomponents from the training component. In the next section we discuss the AQG system architecture along with the system interaction in the training and testing phases.

4.1. System Architecture

In this section the above-mentioned Amharic question generation architectures are discussed in detail. Figure 5.1 shows the overall system architecture of the AQG model that involves the mentioned components.

The first step in the AQG model is the Input Preprocessing. This component helps us to get more clean data by including Normalization, Special Character Removal, and Tokenization to the raw data. The raw data in this task is a Sentence-Question pair and/or Sentence-Question-Answer triple. The sentence is where the question is generated from, the question is the possible factual question that can be generated from the given sentence, and the answer is a text span from the sentence that can be a possible answer for the question created from the same sentence. The next step in training the AQG model is to let the system understand the preprocessed input data by representing them in a vectorized form. To make the AQG model understand the preprocessed data we use the previously trained Word2vec (see section 4.2.2 for detail) model. This model provides the vectorized form of each token to the AQG model. In case the new token is obtained, we use Word Modeling in the training component to convert the preprocessed structured data into a vectorized form.

The Word Modeling takes a preprocessed input from the Amharic Sentence-Question pair and/or the Sentence-Question-Answer triple Corpus and extracts a semantic relation of words using a sparse distribution. It also saves the semantically related vectorized words in the Word2vec Model for future use in the Testing component. This word vectorizer gives a richer semantic meaning to make the machine understand the inputs well.

The testing component is the final component of the system architecture. This component shares similar architecture to the training component by using the preprocessing and vectorization of words similarly.

The Question Generator can then automatically generate questions without necessarily extracting or learning the features again.

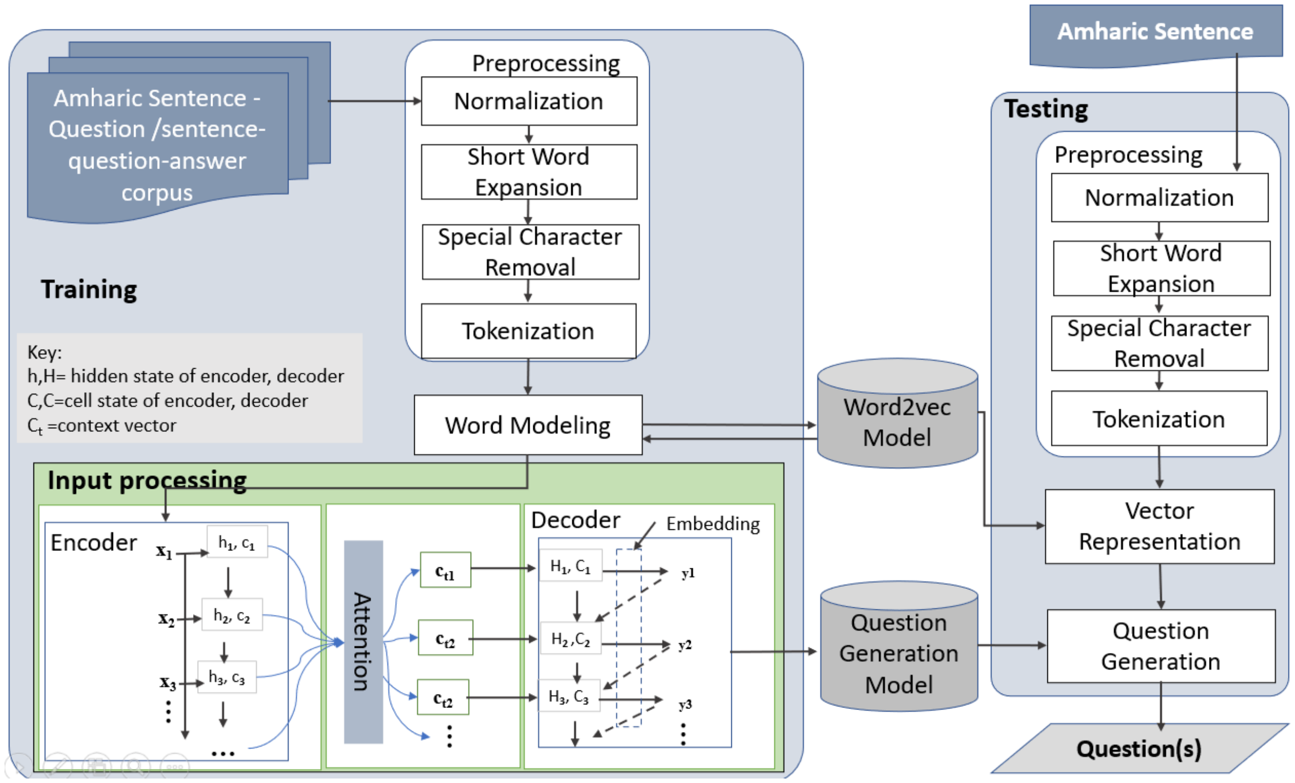


Figure 4.1 The architecture of Amharic Question Generation Model

4.2. Training

The training component is one part of the AQG system architecture that is used to train the AQG model using the Amharic Sentence-Question pair and/or Sentence-Question-Answer dataset. In this component various tasks are performed to train the AQG system: Input Preprocessing, Input Processing, and development of the Question Generation Model. Input preprocessing is where the necessary preprocessing techniques are applied to the collected data and Input Processing, the major component in training the AQG model, is all about representing the vectorized words into a fixed size word matrix, and training the AQG model to generate a learned knowledge base i.e., Question Generation Model as it is shown in Figure 5.1.

4.2.1. Preprocessing

Preprocessing is a method of structuring the raw data using different data formatting and cleaning techniques to make those data suitable for the learning process. The AQG model uses the Amharic Sentence-Question pair Corpus and/or the Sentence-Question-Answer triples to generate questions by primarily preprocessing them. The Preprocessing component is critical in enabling the AQG model to learn from well-structured input.

Character Normalization: the purpose of character normalization in this study is to reduce the vocabulary size. In Amharic, one word can be written in different characters or fidels but the sound and the meaning are the same. For example, the Amharic characters ሀ, ሃ, ሐ, ሓ, ኀ, ኃ and ኻ have sound and ሰ, and ሠ have the same, and ፀ, ጸ sounds the same. That means, words like ፀሐፊ, ፀሓፊ, ፀሀፊ, ፀሃፊ, ፀኀፊ, ፀኃፊ, ፀኻፊ, ጸሐፊ, ጸሓፊ, ጸሀፊ, ጸሃፊ, ጸኀፊ, ጸኃፊ, ጸኻፊ would be considered as different words by the model, which is not desirable because all of these words have the similar meaning in reality. Therefore, it is very important to normalize these characters to avoid any differences in similar words that are written using different characters with the same pronunciation but different structural presence in order to make the machine understand each meaning.

Special Character Removal: Characters that are not necessary for the Amharic Question Generation task are removed from the corpus. These non-alphanumeric characters are often found in punctuation marks like adding quotations to refer to other people's speech, as well as symbolizing currencies. These non-alphanumeric characters add no value to the data other than causing noise to the system.

Short word expansion: While collecting data several short word forms found. Using the short and expanded word forms interchangeably in the dataset make the model to consider those words as different words. To avoid these problem, short word expansion is made based on the predefined short words that we extract during data collection. Some of those short words are written in Appendix 4 with their equivalent expanded form.

Tokenization: The technique of dividing a given text into single words or tokens is known as word tokenization. It helps us to analyze the semantic meaning of each word. The tokens can be either a sentence or a single word, punctuation mark or number. Hence, there are two kinds of tokenization: sentence tokenization and word tokenization. Sentence tokenization is performed on a given paragraph splitting it into a set of sentences. Since this work focuses on sentence-level question generation, sentence tokenization is used to split paragraphs into sentences so that questions can be generated from them. To make sentence tokenization we used an Amharic full stop (።) name as አራት ነጥብ/ārati net'ibi.

Word tokenization is the process of splitting a given sentence into single words. This is done by finding the word boundaries, the ending point of a word and the start of the next one. In this study we perform word tokenization on the whole dataset. The reason behind this process is to get all the unique words in our corpus in order to build the vocabulary or dictionary. Finally, we assign an index to each of these words to make them a distinctive member of the dictionary. For performing this task, we used white space as a splitter to distinguish the boundary between tokens.

4.2.2. Word2vec Model

Machines cannot understand the Amharic or any other language texts directly. Rather they need a numerical form of these text data that we call a vector. FastText word2vec model provides a 300-dimensional word vectors for 157 languages including Amharic. More specifically, for Amharic it contains vectors of 304,651 tokens. However, FastText did not perform Amharic character normalization. That means the same words are written differently due to the different nature of Amharic letters having different vector representations. That makes those act differently. For example, the following words are all in the FastText Amharic word vector.

ጸሐይ -0.0019 -0.0119 0.0696 ...	ፀሓይ 0.0077 0.0018 -0.0687 ...
ጸሀይ -0.0263 0.0111 0.0066 ...	ፀሐይ -0.1033 -0.0438 0.0228 ...
ጸሃይ -0.0283 -0.0048 0.0677 ...	ፀሃይ 0.0185 0.0129 0.0503 ...
ጸሓይ -0.0166 0.0258 -0.0441 ...	ፀሀይ 0.0058 0.0028 -0.0324 ...

Although the all the above words have the same meaning, they appear eight times in FastText each having different vector value. While the change in letter form changes only the structure of the word itself, it has no any impact on the structure of Amharic text. For example, from the Amharic sentence "ፀሐይ በምስራቅ ትወጣለች።" The word "ፀሐይ" is the beginning of the sentence. If we replace this word with "ጸሐይ", the sentence becomes "ጸሐይ በምስራቅ ትወጣለች።" The beginning of the sentence is now "ጸሐይ". That is to mean, that the structure of the sentence does not get changed due to the change in unnormalized character change in a word. Therefor those words should have the same vector value.

In addition, the presence of these repeated words with their different forms increases the vocabulary size which in turn increases the computation time during training. For instance, with the addition of a preposition like የ, በ, ከ, ወደ, ስለ etc., the above verities of ፀሐይ will have another eight words for each preposition. The same case will happen when those words come with other prefixes and suffices. Moreover, the Amharic version of the FastText word vector consists of thousands of English words.

To make the machine understand our input we model our inputs in the form of vectors. First of all, we pre-train a word embedding model that we use to look up during training our generator model. Training the word embedding model begins by scrapping the Wikipedia Amharic website. Wikipedia Amharic consists of lots and lots of Amharic articles that they called "መጣጥፍ" in Amharic. By the time we collect the Amharic articles/መጣጥፎች, we got 17,584 articles. Among these, 2,583 articles have no content inside other than their title. So, we got a total of 15,001 Amharic articles which is pretty enough to train our word embedding model.

As a first step, we extract titles as the URL of each article is made of its title. For example: ወፎች፣ የፍርድ ቀን, የኢትዮጵያ ንግድ ባንክ, ትናንሽ ልጆች ትናንሽ ሀብቶች etc. Titles with more than one word are made concatenate by an underscore (e.g., ትናንሽ_ልጆች_ትናንሽ_ሀብቶች) as white spaces cannot be used in URL, and also the format of the Wikipedia Amharic URL is following this method. We then navigate through each article and extract each text in each article using BeautifulSoup4 [95].

In the collected sentences there were lots of unnecessary special characters, and Wikipedia reference numbers that have no relation to the written text for our purpose. When the same thing is written by two or more bodies, its content will not be all the same. However, writers use different characters to write the same thing. For instance: የአክሱም ሀውልት, የአክሱም ሃውልት, and የአክሱም ሐውልት are titles of three different articles. Storing all these things as it is, increases our dictionary size. So, we apply normalization to the entire sentences so that ሀውልት, ሃውልት, ሐውልት become one word. The articles also consist of English texts which again are not necessary for our task.

Hence, the preprocessing steps: special character removal, and normalization are applied to the collected sentences. In addition, on this website several Amharic articles which contain texts delimited by colon (ሁለት ነጥብ). To make our task easier, we replace the characters ፡, ፤, ፣, and ፥ by whitespace so that we could use only whitespace during tokenization.

There were also lots of short word forms in the collected data. We collect all short words. We then create a dictionary in python where the index stores the short word forms and the value part of the dictionary stores the expanded form of those short words. Then we used this dictionary while applying the short word expansion step of preprocessing.

After preprocessing the collected text, we apply word-level tokenization to make our dictionary. We also preprocess the collected sentences and questions in the above same way, and add the words to our dictionary. From this process we got 210,008 unique words/tokens. We then develop an embedding model with a dimension of 300. Below is a table-wise description of the word embedding training data.

Table 4.1 Description of word embedding training dataset

The data used for modeling our word2vec model	
Total number of articles	15,001
Total number of paragraphs	74,827
Total number of sentences	191,834
Total number of unique words	210,008

As shown in Table 5.1 above the total number of Amharic Wikipedia Articles (መጣጥፎች) which has contents inside is 15,001. From those articles we could extract 74, 827 paragraphs which in turn consists of 191,834 sentences. The unique words/tokens in those sentences are 210,008. Even though Amharic Wikipedia has lots of Amharic articles, there are repeated and empty articles. This is the reason behind the small number of unique words relative to the number of Articles found in Amharic Wikipedia articles.

4.2.3. Input processing

Input Processing represents the vectorized words into a 300-dimensional word matrix, extracts features (keyword) and, provides a learned model that can generate questions from Amharic sentence. This component is generally very valuable to let the generator model learn from the vectorized Amharic Sentence-Question and/or Sentence-Question-Answer Corpus by involving the mentioned operations in Sentence Processing. The question processor outputs trained models after learning features. The saved trained model is then used in the Testing component to the generated Amharic Questions. The general architecture in the Input Processing component of training consists of three main sub-components. The first one is the encoder, which uses Long-Short-Term-Memory (LSTM) layer to make the model understand the input sequence and produce hidden states that are used as an input for the decoding process. The second sub-component is the attention mechanism which lets the model to focus on very important words during learning. The last sub component in the input-processing component of the Amharic Question Generation is the decoder which is another LSTM. The decoder predicts output at each time step using the encoder's output and the current target word vector. To predict the next target word using a probabilistic distribution over the entire target vocabulary, we used the softmax layer.

Encoder

In sequence-to-sequence learning, the input sequence is encoded understandably so that it can help to predict the output sequence. Our LSTM encoder take the sentence-question pairs. These pairs are separated by tab in the way shown below.

ቋንቋዎችን ማሳደግ የዜጎች ሀላፊነት ነው።　　ቋንቋዎችን ማሳደግ የማን ሀላፊነት ነው?

ኢትዮጵያ ብዙ ታሪካዊ ስፍራዎች አሏት።　　ኢትዮጵያ ስንት ታሪካዊ ስፍራዎች አሏት?

የአስራ ሁለተኛ ክፍል ተማሪዎች ዉጤት ዛሬ ተለቀቀ።　የአስራ ሁለተኛ ክፍል ተማሪዎች ዉጤት መቼ ተለቀቀ?

The dataset is sorted by the length of a sentence from the shortest to the longest. It is known that the Amharic Full stop(።) and Question mark (?) punctuations are important to show the end of sentence and question respectively. In fact, these operators are appended to the last word of a sentence or question. However, in this work appending these punctuations is unnecessary as it makes the model to understand the appended token as a single word. For instance, from the first example "ነው" and "።" are put together in the sentence side and "ነው" and "?" are put together. That means "ነው።" and "ነው?" are considered two words. To prevent this, we create a space between a word and the punctuation following it as follow:

ቋንቋዎችን ማሳደግ የዜጎች ሀላፊነት ነው ።　　ቋንቋዎችን ማሳደግ የማን ሀላፊነት ነው ?

We then inserted a start and end token to the sentence to tell the algorithm when to predict and when to stop. The word tokenization is performed on both the sentences and the question like [*'ቋንቋዎችን', 'ማሳደግ', 'የዜጎች', 'ሀላፊነት', 'ነው', '።']* for the above sentence, and we create a Vocabulary with word index (mapping from word → id) and reverse word index (mapping from id → word).

We use a batch size of 64, an embedding size of 300, and a unit of 1024 which is the dimension of the inner cells in LSTM. The encoder then takes the input vocabulary size, embedding dimension, the encoder units, and batch size parameters and start producing sample output, hidden state and cell state. We also apply the same process for the Amharic Sentence-Question-Answer triple corpus. The difference is that we added the third column by separating it with an <ans> tag as shown below:

ቋንቋዎችን ማሳደግ የዜጎች ሀላፊነት ነው ።　　ቋንቋዎችን ማሳደግ የማን ሀላፊነት ነው ?　　<ans> የዜጎች

Here, the <ans> token is added to separate the answer span from the question column. This token is also important during post-processing to identify the answer from the predicted text. We used this dataset to develop a model that can take Amharic sentences as an input and generate factual questions from the input sentence with the possible answer. In another work [7] answer span is used to increase the performance of the question generation model.

Decoder

The decoder is another LSTM that is responsible for predicting the output sequences based on the encoders hidden and cell states which together is a context vector. Therefore, the decoder LSTM is initialized with the encoder state with the Luong Attention [51]. Luong attention get the decoder's hidden state, compute attention scores, and then predict using the context vector concatenated with the decoder's hidden state. To let our model, generate more than one questions from a single sentence, we used a beam search decoder where the default width of the beam is set to 3, and the average number of questions generated from a single question in the whole dataset, which is calculated by dividing the total number of sentences to the total number of questions from the whole dataset. We then train our model using Adam optimizer and defined loss and metrics. Finally, the model is let to train for 50 epochs.

4.3. Testing

This component is somehow similar to the Training component except that it uses a learned Question Generation Model to generate the Amharic Questions from the given sentence. This component generates questions from sentences that were never fed into the system before to evaluate the performance of the AQG model. The Question Generation Model gets saved and updated every time the Training component passes through some training. The Testing component as shown in Figure 5.1, retrieves the saved Question Generation Model to generate questions without necessarily going through the processes carried out in the training component.

4.3.1. Preprocessing

The preprocessing component receives the Amharic Sentences as input. The sentences here share a similar process i.e., Normalization, Special character removal, and Tokenization as what has been done in the training preprocessing sub-component. Each of the preprocessing sub-components is discussed in Section 4.2.1.

4.3.2. Word embedding

After preprocessing the Amharic sentences, we vectorize each sentence using the Word2vec Model. The trained Word2vec Model solves this issue by training on a large amount of data to identify the semantically related words. The Word2vec Model stores the learned vectorized questions in a 300-

dimensional word vector. Its purpose is to mainly store the modeled words with their semantically related vectorized words.

The Vectorization in the Testing component of the AQG system architecture converts the preprocessed raw data into a vectorized form by using the learned Word2vec Model. We then get the word matrix from the stored vectorized trained data using a lookup table functionality. Each word is mapped to a vector with multi-dimensional related vectorized words. This process makes the Testing process easier by getting the tokenized input words alongside the related vectors. Then again, words with a different meaning are represented with a vectorization far from the target word.

The Word2vec Model is trained on words having similar words across the 300 dimensions. There are actually 50, 100, 200, and 300-dimensional word embeddings. In this study we use a 300-dimensional word vector to get a more accurate representation of the words. The represented words in the Word2vec model are loaded into our AQG model to represent the input sentences with a 300-dimensional word2vec word matrix.

4.3.3. Question generation

The Question Generation is like the intersection between the Question Generation Model and the Vectorized Amharic sentences that we get from the Testing component. The Question Generation Model stores the learned features using the tab-separated Amharic Sentence-Question pairs and/or Sentence-Question-Answer triple Corpus. The testing component takes sentences as input unlike the training component which take sentence-question pairs. This sentence (e.g., ኢትዮጲስ የኩሽ ልጅ ነው።) is tokenized into words ([‘ኢትዮጲስ’, ‘የኩሽ’, ‘ልጅ’, ‘ነው’, ‘።’]). The <sos> and <eos> tokens are also added to help the model know when to start and end the prediction. These inputs are then converted to tensors using a convert_to_tensor method of TensorFlow and padded to the maximum length. Then, the model passes the tensor inputs to the decoder. The decoder processes these tensor inputs in a way it has trained before and the encoder returns the hidden state and cell state which the model uses to initialize its decoder hidden state.

The model’s decoder then uses this hidden state and the input vector to predict a word. After predicting a word, the decoder then takes the next input. Using this input and the previous hidden state, the decoder produces the second word. This process continues until all the inputs are completed. The model understands that the prediction process is finished when it gets the <eos> token.

The generation of multiple questions from a single sentence is performed by the use of a beam search whose width is set to 3, which is the average number of questions generated from all the sentences in the corpus.

Sentence	Questions
አበበ በ2014 ዓ.ም አሜሪካ ሄደ።	አበበ መቼ አሜሪካ ሄደ?
	አበበ በ2014 ዓ.ም የት ሄደ።
	ማን በ2014 ዓ.ም አሜሪካ ሄደ።

From the above sentence (አበበ በ2014 ዓ.ም አሜሪካ ሄደ።), three factoid questions of type Who, When and Where are generated. The generation of those questions is based on the beam width and the beam weight. The question with high weight is generated first and with the lower weight will be generated later.

Chapter Five

5. Implementation and Evaluation

5.1. Dataset Preparation

For evaluating automatic Amharic question generation. We utilized two datasets: a sentence-question dataset which contains around 60,023 Sentence-Question pairs and a sentence-question-answer triples with the same number as the first dataset, which was created from 34,302 sentences which are collected from waltainfo.com and fanabc.com as mentioned in section 3.3. We shuffle all the categories through the dataset and since all of the examples within a batch should also come from different parts of the dataset, we shuffle all the samples before training. Batches are also randomly selected during training by using a function that returns a random permutation of integers from 0 to [(size of the input) – 1].

We do not set a fixed-length vectors for our input sentences. Instead, we add a special end-of-sentence token to our dictionary of words which helps with learning variable-length sentences. In addition, the <sos> and <eos> tokens are added to the start and end of each sample respectively so that the model continues decoding until it generates the token. In the generation process, for producing the next word, the decoder can either choose the most likely word according to the model or apply a beam search to generate best sequences according to the given beam width. In our case, the default beam width is set to 3 which is the approximate average value of the total number of questions in the dataset divided by the total number of sentences from which the questions are created. The distribution of questions is described below in Table 5.1.

Table 5.1 Detail description of the dataset

Sentence-question pairs and/or sentence-question-answer triples	**Quantity**
Number of sentence-question pairs and/or sentence-question-answer triples where the question kind is Who|ማን	11,067
Number of sentence-question pairs and/or sentence-question-answer triples where the question kind is What|ምን	15,502
Number of sentence-question pairs and/or sentence-question-answer triples where the question kind is Where|የት	8,473

Number of sentence-question pairs and/or sentence-question-answer triples where the question kind is When|መቼ	10,831
Number of sentence-question pairs and/or sentence-question-answer triples where the question kind is How much|ስንት	14,150
Total number of sentence-question pairs and/or sentence-question-answer triples collected	**60,023**

As shown in the above table, the collected data for the who, what, where, when and how much/many factual question types is 11067, 15502, 8473, 10831, 14150 respectively which gave us a total of 60,023 sentence-question pairs and/or sentence-question-answer triples. The difference between the sentence-question pair and the sentence-question-answer datasets is that, the former was used to train a model that can generate questions from an Amharic sentence and the latter was used to train the model to generate question from an Amharic sentence with their answers without the need to train an independent question answering model. That means in the latter dataset we added the answer text to know whether it has impact on our model's performance. Among the collected data 80% is used for training the model and 20% for validating it. These data split is chosen after testing with different splitting trials like 70% for training and 30% for validation. Finally, testing is done manually using human evaluator there is no standard automatic evaluation metrics for question generation task as it is discussed in detail at the end of section 5.2

5.2. Tools and experimental setup

After collecting and preprocessing the necessary data, one needs to setup the environment in a suitable way for the preprocessed data. To implement our model, we used the TensorFlow library, which is a promote public machine learning environment. It includes a robust, flexible set of tools, libraries, and community resources that allow academics to push the limits of machine learning and developers to swiftly create and use ML-powered systems [93].

To train our model, we used Google Colab, which is a Jupyter notebook environment created by Google to help spread machine learning education and research, because of its effortlessness in use and its fast processing [96]. First of all, we build our source and target vocabulary. To do this we got all the unique words on the source and target side, and assigned an index to each unique word. We then set the size of those source and target vocabularies to length of the unique inputs. When there are words out of the limit, those words were replaced with the UNK token. In addition, we added the <sos> token at the beginning of the sentence to indicate the start of the sentence, and <eos> token at the end of the sentence. The <eos> token helped us for defining variable-length sequences in our model. It also indicates where the generation of question from that sentence should stop.

We set the dimension of the word embedding vector to 300, which is obtained from the weight of our pre-trained word embedding model. If there is a word in our vocabulary that does not exist in our word embedding, the weight of that word is initialized from a uniform distribution. We trained an LSTM based and GRU based learning model with a hidden unit size of 1024 for our model.

Adam is an adaptive learning rate optimizer created particularly for deep neural network training [33]. It adapts the learning rate for each weight of the neural network by estimating the first and second moments of the gradient. Adam has a quicker computation speed and requires less modification parameters. As a result, we used Adam as our optimizer for both the LSTM based and GRU based models.

The training of our model took 50 epochs. The number of samples that go through one forward, which is the batch size, is set to 64. Moreover, to prevent our model from overfitting, which occurs when the model fits the current data very well but fails to generalize for new examples, we utilized dropout with a probability of 0.5 because dropout in a hidden layer need to be between 0.5 and 0.8 to allow the model to train successfully [97]. For training and validating the model we made a percentage split of 80% by 20% which is selected after different percentage split testing like 70% by 30% for training and validation respectively.

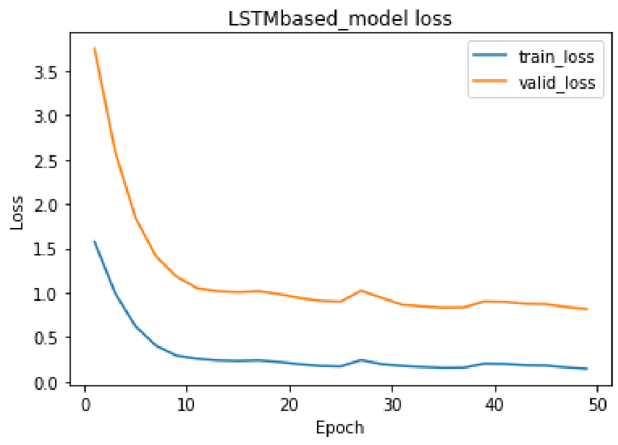


Figure 5.1 LSTM based AAQG model loss

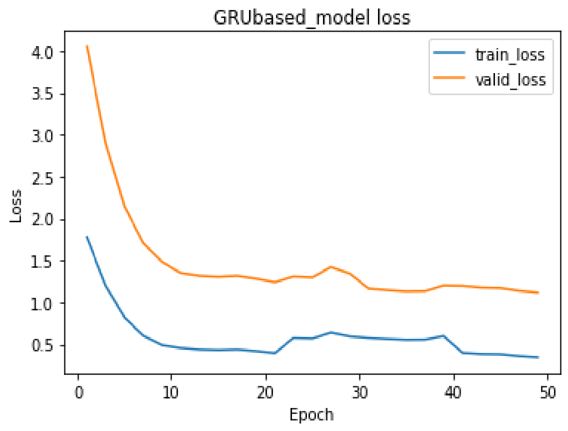


Figure 5.2 GRU based AAQG model loss

The figures 5.1 and 5.2 above shows the training and validation loss of the LSTM based and GRU based Automatic Amharic Question Generation model with the same training and validation datasets. The LSTM based model could learn better than the GRU based model. The GRU based model shown an increased loss both in the training and validation especially around the middle its iteration although it showed a decreased loss at the last quarter of its iteration. In the last ten epochs of the GRU based model, the validation loss shown a little improvement compared to its training loss. Unlike the GRU based model, the LSTM based model shown lower loss value both in training and validation, comparatively. In this model also, a little increase in loss has shown nearest to epoch thirty also it lowered after some iterations. Even in the last ten iterations the validation loss is lower that the GRU based model.

The testing is made based on another 105 sentences which enabled the model to generate 227 questions and those questions were given for human evaluators as it is discussed in section 5.3. The reason behind human evaluation is that there is no standard automatic evaluation metrics for question generation task. Based on the study made on 37 question generation papers [71], most of the papers used human evaluation and some used automatic evaluation metrics which was mainly designed for machine translation work like BLEU [73] and METEOR [74]. However, none of these metrics or the automatic accuracy could evaluate the question generation model.

For example, for the Amharic sentence "አበበ በ1985 ዓ.ም አዲስ አበባ ተወለደ።", the reference questions could be "አበበ በስንት ዓ.ም ተወለደ?", "ማን በ1985 ዓ.ም አዲስ አበባ ተወለደ?", and "አበበ የት ተወለደ?". Even though many questions can be created from this sentence we could not collect all possible questions that can be created from all the sentences in our dataset due to the flexibility of the langage nature. Some sentences have only one question, some have two etc. And the system could generate questions like "አበበ መቼ ተወለደ?", "አበበ በ1985 ዓ.ም የት ተወለደ". This time, based on the automatic accuracy evaluation, the system performs nothing because number of correctly generated questions according to the system is zero though they are correct in reality. This is because matching sequence is very complex unlike classification tasks which matches labels.

BLEU too could not evaluate our task well as it is dependent on precison and do not understand the semantics. That means, BLEU could not understand that "መቼ" is equivalent to "በስንት ዓ.ም" and also to calculate the precision, it should count the predicted words that match the reference and divide it to the total number of predicted words which makes the evaluation unfair. The same is true for METEOR as it also counts the predicted words for the purpose of recall calculation.

5.3. Evaluation of Questions

To evaluate the questions, we collect 105 different Amharic sentences and give them to the system. The system generates 227 questions. The generated questions include 58, 31,33,61 and 44 questions of type ምን, የት, መቼ, ስንት, and ማን respectively by considering other questions types like ከየት, ማንን,

በስንት, and ለማን as የት, ማን, and ስንት respectively etc. We then gave these questions to a human evaluator to classify the questions as correct or incorrect based on grammaticality and relevance. Using the result of the human evaluator, we used the accuracy, which is the most widely used evaluation measure, to evaluate our system. It is calculated as:

$$\boxed{Accuracy = (\frac{CGQ}{TGQ}) \times 100}$$

Where CGQ is the number of Correctly Generated Questions from the system according to the human evaluator, and TGQ is a Total number of Generated Questions. Table 5.2 shows the correctly evaluated and total generated questions of each question types with the accuracy independently as well as the overall accuracy.

Table 5.2 Evaluation result of the GRU based question generation system

Type of Question	**ማን**	**መቼ**	**ስንት**	**የት**	**ምን**
Total questions	**44**	**33**	**61**	**31**	58
Correct Questions	**39**	**27**	**56**	**27**	54
Incorrect Questions	**5**	**6**	**5**	**4**	4
Accuracy (%)	**86.64**	**81.82**	**91.80**	**87.1**	**91.38**
Over all accuracy (%)	**88.10**				

As shown in table 5.2 above our question generation system generates 39 out of 44, 27 out of 33, 56 out of 61, 27 out of 31 and 54 out of 58 correct questions, and 3, 5, 4, 6 incorrect questions for who, when, how much/many type questions respectively. Based on these, the system shown an overall accuracy of 88.10%. The When question type shown lower accuracy relative to other question types.

Table 5.3 Evaluation result of the LSTM based question generation system

Type of Question	**ማን**	**መቼ**	**ስንት**	**የት**	**ምን**
Total questions	44	33	61	31	58
Correct Questions	41	28	57	25	52

Incorrect Questions	3	5	4	6	6
Accuracy (%)	**93.18**	**84.85**	**93.44**	**80.65**	**89.66**
Over all accuracy (%)	**88.36**				

As shown in table 5.2 above our question generation system generates 41 out of 44, 28 out of 33, 57 out of 61, 25 out of 31 and 52 out of 58 correct questions, and 3, 5, 4, 6 incorrect questions for who, when, how much/many type questions respectively. Based on these, the system shown an overall accuracy of 88.36%.

Table 5.4 Sample comparison of actual and system generated questions

Sentence	**Actual question**	**System generated question**
▪ ተዝካረ ቃል በሊማሊሞ ገደል ተወርውሮ ሞተ።	▪ ተዝካረ ቃል በየት ተወርውሮ ሞተ? ▪ ማን በሊማሊሞ ገደል ተወርውሮ ሞተ? ▪ በሊማሊሞ ገደል ተወርውሮ የሞተው ማን ነበር?	▪ ተዝካረ ቃል የት ሞተ? ▪ ማን በሊማሊሞ ገደል ተወርውሮ ሞተ?
▪ በጦርነት ምክንያት አንድ ሚሊየን በላይ ሰዎች ተፈናቀሉ።	▪ በጦርነቱ ምክንያት ስንት ሰዎች ተፈናቀሉ? ▪ አንድ ሚሊየን ሰዎች በምን ምክንያት ተፈናቀሉ?	▪ በጦርነቱ ምክንያት ስንት ሰዎች ተፈናቀሉ? ▪ አንድ ሚሊየን ሰዎች በምን ምክንያት ተፈናቀሉ? ▪ ስንት ሰዎች በምን ምክንያት ተፈናቀሉ?
▪ ሲዳማ ክልል ምክር ቤት ለ2014 14 ቢሊዮን ብር በጀት አፀደቀ።	▪ የየት ክልል ምክር ቤት ለ2014 14 ቢሊዮን ብር በጀት አፀደቀ? ▪ የሲዳማ ክልል ምክር ቤት ለመቼ 14 ቢሊዮን ብር በጀት አፀደቀ? ▪ የሲዳማ ክልል ምክር ቤት ለ2014 ስንት ቢሊዮን ብር በጀት አፀደቀ?	▪ የሲዳማ ክልል ምክር ቤት ምን አፀደቀ? ▪ ማን ለ2014 14 ቢሊዮን ብር በጀት አፀደቀ? ▪ ሲዳማ ክልል ምክር ቤት ለስንት ቢሊዮን ብር በጀት አፀደቀ።

As shown in the above table, most of the system generated questions are similar to the actual questions, some are not similar but they are still correct questions based on the sentence, and some of the system generated questions are not correct.

For example, from the first row of the above table the actual factual question is ተዝካረ ቃል በየት ተወርውሮ ሞተ?" and the system generated question is "ተዝካረ ቃል የት ሞተ?". Even though these questions are not perfectly equal, both of them are grammatically correct questions. This is the reason that automatic evaluation of question generation task is very difficult. That means, in reality many questions can be created from a single sentence, and the number of questions that can be created from different sentence is not known. When the dataset is prepared all possible questions that can be created from the given sentence could not be included due to the complex nature of natural languages. However, based on the training from different sentences-question pairs, the generation system generates correct questions which are not included in the test data. At that time the automatic evaluation metrics understands that the system made wrong prediction.

From the sentence "ሲዳማ ክልል ምክር ቤት ለ2014 14 ቢሊዮን ብር በጀት አፀደቀ።", the system generalizes the string "ለ2014 14" into one and produces an incorrect prediction "ሲዳማ ክልል ምክር ቤት ለስንት ቢሊዮን ብር በጀት አፀደቀ።". That means, the system understands the two tokens 2014 and 14 as one and classify them to a how much question. However, when we say "በ1985 ተወለደ" in our day-to-day activities, it is known that 1985 to indicate the year instead of saying 1985 ዓ.ም.

Table 5.5 Evaluation result of the LSTM based question generation system that is trained by the sentence-question-answer triple dataset

Type of Question	**ማን**	**መቼ**	**ስንት**	**የት**	**ምን**
Total questions	44	33	61	31	58
Correct Questions	39	27	56	23	48
Incorrect Questions	5	6	5	8	10
Accuracy (%)	**88.63**	**81.82**	**88.52**	**70.97**	**82.75**
Over all accuracy (%)	**82.54**				

Our system outperforms the previous results. The rule-based question generation system by [9] got 86.4% overall accuracy for the factoid question types Who, Where, When and How much/many while excluding the What question types. Table 5.6 shows the comparison of our system with the baseline automatic Amharic question generation system.

Table 5.6 Comparison of our system with the previous study *[9]*

Type of question	**Previous generation system accuracy**	**Our GRU based generation system**	**Our LSTM based generation system accuracy**	**Our LSTM based generation system with answer text**
ማን	91.6%	86.64	93.18%	88.63
መቼ	83.3%	81.82	84.85%	81.82
ስንት	94.1%	91.80	93.44%	88.52
የት	73.0%	87.1	80.65%	70.97
ምን	–	91.38	89.66%	82.75
Overall accuracy	**86.4 %**	**88.10**	**88.36%**	**82.54%**
Generate more than one question from a single sentence	No	No	No	Yes

5.4. Training and experimental results

To get a high-performance model and choose the best learning algorithm for the Amharic Question Generation, we train two different models using the same dataset. In the first step, we train an encoder-decoder model with GRU, and LSTM learning algorithms. We use the same dataset for all of them for both training and testing. The dataset, as mentioned above in section 3.3, consists of 60,023 *sentence-question-answer-answer type* quadruples. However, for our experiment we use the *sentence*-question pairs which are separated by a tab from one another and the sentence-question-answer triples where the sentence and the questions are separated by tab and the answer part is separated by our special token (<ans>). Among the whole dataset 80% is used for training and 20% is used for validating the model. We also used our pre-trained word vector data. The parameters for all the model are set equally. The batch size is set to 64, the embedding dimension is set to 300 and the hidden unit is set to 1024.

In the first phase of our training, we use an LSTM encoder which takes the input sequences in the form of vectors. The encoder then produces hidden states. Then the Luong Attention [51] is applied so that the hidden states encoded by the encoder are concatenated with the previous predicted output. This concatenated result is then passed through dense layers to form the attention weights. These weights are then used to compute the context vector. Eventually, the LSTM decoder take this context vector from the softmax layer as an input. We then train our model using the Adam optimizer. Finally, our model achieves an accuracy of 88.36% after 50 epochs based on human evaluation.

To train the GRU model, we adopt the above LSTM model as GRU has one state unlike LSTM which has two states. We do not make any change whether on the sentence-question dataset or on the word vector dataset, and the same attention mechanism is applied here. However, the accuracy is seen to be lower than the LSTM model by 0.26 % as shown in Table 5.6. This indicates that the LSTM-based model outperforms the GRU-based model for our task of generating Amharic questions from Amharic sentence automatically.

5.5. The effects of adding answer text on the AQG models

So far researchers conducted research on the area of question generation including [22], [64], [68], and [98]. They tried to increase the performance of their question generation system using the answer and answer position features. In our experiment however, we used the answer span to train the model to learn how to provide an answer for the generated question instead of just only learning the modet to generate questions given the sentence. In real-world, when we want to test ourselves after reading some information, we want to get question concerning what we read. For example, assume a journalist collects 50 historical economic, political, and general current issues sentences for radio question and answer program. This time the journalist should prepare questions from those collected sentences. That means the journalist is wasting time and labor. Our study tried to solve this problem by training the model how to get the answer text from the given sentence without further independent trained question answering model.

To solve the above-mentioned problem, we train our model by adding answer span in the previously used dataset in a format like this →Sentence[tab]question[space]<ans>[space]answer. The other preprocessing task is similar to the previous sentence-question pair dataset. We added the <ans> tag to separate the answer text from the question text as well as to split the answer text from the question text during post-processing as the model understands the question and the answer as a single sequence.

After we trained the model with the same parameters as the previous model, we evaluated it with the same number of questions that we used previously. However, in this case the model generates questions and answers for the given sentences. Based on the human evaluation, the system achieves 82.54 % accuracy.

Relatively, the performance of this model decreases. This is because of the longer sequence in comparison to the previous sentence-question pair dataset. For example, from the sentence "አበበ ቀይ መኪና ገዛ", we have the following format of the previous dataset (sentence-question pair) and the dataset used in this experiment (sentence-question-answer triple) respectively.

አበበ ቀይ መኪና ገዛ	አበበ ምን ገዛ ?
አበበ ቀይ መኪና ገዛ	አበበ ምን ገዛ ? <ans> ቀይ መኪና

Our model understands that "አበበ ቀይ መኪና ገዛ" as one sequence and "አበበ ምን ገዛ? <ans> ቀይ መኪና" as another sequence. This time when the sentence, question, answer text becomes long, the length of the sequence increases. This leads the system to make poor predictions.

Chapter Six

6. Conclusion and Recommendation

6.1. Conclusion

In this study, an attempt was made for the implementation of automatic factual question generation from Amharic sentences. The system uses an encoder-decoder architecture with an attention mechanism where the encoder and decoder are recurrent neural networks specifically LSTM for encoding the input sequence and decoding it respectively. Furthermore, the attention mechanism helps the system to focus on necessary parts only to make the prediction. The dataset used in this study is collected from the internet using our data collection system that was developed for this specific task. The sentences that help the data annotators during question construction are crawled from the websites which use the Amharic language more specifically from waltainfo.com and fanabc.com.

We implemented our question generation system using the python programming language. We basically used the TensorFlow library to build the model. The model is trained by the collected dataset in the Google Colab environment. To generate questions, the encoder part of the system takes the input sequence in the form of tensors and produces the context vector that is used to initialize the decoder. Later on, the decoder predicts the output words based on the current input and the previously predicted word with the help of an attention mechanism.

The system also used a pretrained word embedding that we trained with sentences collected from the Amharic Wikipedia website. For crawling content from Wikipedia, we used BeautifulSoup4 [95].

The model was trained by using two different datasets: one is the sentence-question pair and the other is the sentence-question-answer triple. The model trained with the first dataset generates questions from the given sentence while the model trained with the second dataset generates questions with their answer. Human evaluation was made for both models and the model trained with the sentence-question pair has 88.36% accuracy while the model trained with the sentence-question-answer triple achieves 82.54% accuracy. This is because in the second model, the sequence length is longer due to the addition of answer text for to learn the model how to answer the generated question from the given sentence.

Having these performance, this study contributes to the world by (1): preparing relatively large dataset from the scratch which can be used not only for the question generation task but also for question answering and question classification tasks using deep learning approaches, (2): providing a normalized and cleaned embeddings of the Amharic words unlike FastText which provides embeddings of unnormalized Amharic words as well as thousands of English words in it, (3): building a model that provides answers for the generated Amharic questions instead of adding question-answering model distinctly.

6.2. Recommendation

In this research we developed a system that can generate question(s) from the given Amharic sentence. From this work we understood that question generation task is very challenging very specifically such task is difficult for low resource languages like Amharic. Based on the results from the experiments in this study, the researcher recommends the following future works to improve the performance of the question generation system:

- Language is complex and flexible that it can be used in different forms for describing even a single thing. So, to make the machine learn more about generating questions from sentence, using deep learning, a very huge and clean dataset is necessary.
- For generating better questions from the longer sentences, using the transformer model [49] can also be a future work as transformers do not use recurrence and are based only on the attention mechanism.
- Generating factual questions is not the only case in reality. Therefore, developing a system that can generate deep questions which require complex inferences will make the question generation full. For example, from the Amharic sentence “ነገ ለወጣቶች ብቻ የተፈቀደ ስልጠና ይሰጣል”, one can ask “አበበ ከነገው ስልጠና ላይ ይሳተፋል። ለምን?” and the answer is “ወጣት ሰለሆነ”. However, creating such a question need understanding the semantics of the sentences deeply. Therefore, working on such questions is crucial for developing a full-fledged question generation system.
- This study considers question generation from sentences as paragraph-level question requires other tasks like text simplification and anaphora resolution which requires much time by themselves. Not only, these tasks but also a data set of paragraph-list of questions from the paragraph is necessary for training the model. So, in future work, one can incorporate those tasks to develop a paragraph-level question generation system.

References

[1] A. Reshamwala, D. Mishra and P. Pawar, "Review on natural language processing," *Engineering Science and Technology: An International Journal (ESTIJ)*, vol. 3, no. 1, pp. 113-116, February 2013.

[2] IBM Cloud Education, "Natural Language Processing (NLP)," IBM, 2 July 2021. [Online]. Available: https://www.ibm.com/cloud/learn/natural-language-processing#:~:text=Natural%20language%20processing%20(NLP)%20refers,same%20way%20human%20beings%20can.. [Accessed 10 December 2021].

[3] R. Weischedel, J. G. Carbonell, B. J. Grosz, W. Lehnert, M. Marcus, C. R. Perrault and R. Wilensky, "White paper on natural language processing," in *Speech and Natural Language: Proceedings of a Workshop Held at Cape Cod, Massachusetts, October 15-18, 1989*, Massachusetts, 1989.

[4] M. Heilman, "Automatic Factual Question Generation from Text," Pittsburgh, 2011.

[5] Y. Zhao, X. Ni, Y. Ding and Q. Ke, "Paragraph-level Neural Question Generation with Maxout Pointer and Gated Self-attention Networks," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, October 31 - November 4 ,2018.

[6] X. Du, J. Shao and C. Cardie, "Learning to Ask: Neural Question Generation for Reading Comprehension," pp. 1342-1352, 2017.

[7] Q. Zhou, N. Yang, F. Wei, C. Tan, H. Bao and M. Zhou, "Neural Question Generation from Text: A Preliminary Study," in *National CCF Conference on Natural Language Processing and Chinese Computing*, Beijing, 2017.

[8] A. Assefa, "Automatic generation of amharic math word problem and equation," *Journal of Computer and Communications*, vol. 8, no. Scientific Research Publishing, pp. 59-77, 2020.

[9] G. Damtie, "Automatic Amharic Factual Question Generation From Historic Text Using Rule Based Approach," Addis Ababa, June.

[10] Q. Zhou, N. Yang, F. Wei, C. Tan, H. Bao and M. Zhou, "Neural Question Generation from Text: A Preliminary Study," in *National CCF Conference on Natural Language Processing and Chinese Computing*, Beijing, 2017.

[11] B. Liu, M. Zhao, D. Niu, K. Lai, Y. He, H. Wei and Y. Xu, "Learning to Generate Questions by Learning What not to Generate," in *World Wide Web Conference*, NewYork, 2019.

[12] H.-T. Zheng, J. Han, J. Chen and A. K. Sangaiah, "A novel framework for automatic Chinese question generation based on multi-feature neural network model," *Computer Science and Information Systems*, vol. 15, no. 3, pp. 487-499, 2018.

[13] B. Leite, H. L. Cardoso, L. P. Reis and C. Soares, "Factual Question Generation for the Portuguese Language," in *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, 2020.

[14] Y. l. Chali and S. A. Hasan, "Classifying the Question by the Answer Complexity," in *Proceedings of the Workshop on Question Answering for Complex Domains*, Mumbai, 2012.

[15] teachingenglish.org.uk, "Gap-fill," [Online]. Available: https://www.teachingenglish.org.uk/article/gap-fill#:~:text=A%20gap%2Dfill%20is%20a,intervals%2C%20e.g.%20every%20five%20words.. [Accessed 21 March 2020].

[16] A. Narendra, M. Agarwal and R. Shah, "Automatic cloze-questions generation," in *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, 2013.

[17] T. Alsubait, B. Parsia and U. Sattler, "Ontology-Based Multiple Choice Question Generation," *KI - Künstliche Intelligenz,* vol. 30, pp. 183-188, 2016.

[18] A. Hoshino and H. Nakagawa, "A real-time multiple-choice question generation for language testing: a preliminary study," in *Proceedings of the second workshop on Building Educational Applications Using NLP*, 2005.

[19] L. Stanescu, C. S. Spahiu, A. Ion and A. Spahiu, "Question generation for learning evaluation," in *2008 International Multiconference on Computer Science and Information Technology*, 2008.

[20] R. Das, A. Ray, S. Mondal and D. Das, "A Rule based Question Generation Framework to deal with Simple and Complex Sentences," in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2016.

[21] M. Agarwal, R. Shah and P. Mannem, "Automatic Question Generation using Discourse Cues," in *Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications*, 2011.

[22] Xinya Du et al, "Learning to Ask: Neural Question Generation for Reading Comprehension," 2017.

[23] H. D. A. D. Ali, "Automatic question generation: a syntactical approach to the sentence-to-question generation case," 2012.

[24] Aravindpai, "An Essential Guide to Pretrained Word Embeddings for NLP Practitioners," 16 March 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/03/pretrained-word-embeddings-nlp/. [Accessed 22 September 2020].

[25] A. C. Graesser and N. K. Person, "Question asking during tutoring," *American educational research journal,* vol. 31, no. 1, pp. 104-137, 1994.

[26] J. Pennington, R. Socher and C. D. Manning, "GloVe: Global Vectors for Word Representation," nlp.stanford.edu, August 2014. [Online]. Available: https://nlp.stanford.edu/projects/glove/. [Accessed 05 April 2021].

[27] https://fasttext.cc, "Word vectors for 157 languages," [Online]. Available: https://fasttext.cc/docs/en/crawl-vectors.html.

[28] V. Rus, Z. Cai and A. C. Graesser, "Experiments on generating questions about facts," in *International Conference on Intelligent Text Processing and Computational Linguistics*, 2007.

[29] N. BM, "What is an encoder decoder model?," 07 October 2020. [Online]. Available: https://towardsdatascience.com/what-is-an-encoder-decoder-model-86b3d57c5e1a. [Accessed 05 February 2021].

[30] M. Heilman and N. Smith, "Good Question! Statistical Ranking for Question Generation.," 2010.

[31] N. Duan and Y. Liang, "Question Generation (QG)," microsoft.com, [Online]. Available: https://www.microsoft.com/en-us/research/project/question-generation-qg/. [Accessed 25 September 2021].

[32] M. Z. Alom, T. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. Nasrin, M. Hasan, B. Essen, A. Awwal and V. Asari, "A State-of-the-Art Survey on Deep Learning Theory and Architectures," *Electronics,* vol. 8, p. 292, 2019.

[33] F. Chollet, Deep learning with Python, Simon and Schuster, 2021.

[34] E. Amor, "Understanding Non-Linear Activation Functions in Neural Networks," 29 May 2020. [Online]. Available: https://medium.com/ml-cheat-sheet/understanding-non-linear-activation-functions-in-neural-networks-152f5e101eeb. [Accessed 06 June 2021].

[35] A. Khan, B. Baharudin, L. H. Lee and K. Khan, "A review of machine learning algorithms for text-documents classification," *Journal of advances in information technology,* vol. 1, no. 1, pp. 4-20, 2010.

[36] A. Rao, "Convolutional Neural Network Tutorial (CNN) – Developing An Image Classifier In Python Using TensorFlow," Jul 2020. [Online]. Available: https://www.edureka.co/blog/convolutional-neural-network/. [Accessed 09 Augest 2021].

[37] DeepAI, "Logit," [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/logit. [Accessed 23 March 2021].

[38] N. Buduma and N. Locascio, "Fundamentals of deep learning: Designing next-generation machine intelligence algorithms," O'Reilly Media, Inc, 2017.

[39] I. Goodfellow, Y. engio and A. Courville, Deep learning, MIT press, 2016.

[40] DEEPLIZARD, "Deep Learning Fundamentals - Classic Edition," [Online]. Available: https://deeplizard.com/learn/video/FK77zZxaBoI. [Accessed 03 October 2021].

[41] P. Vieito and P. José, "Convolutional neural networks for efficient object detection on ultra low-power platforms," Universitat Politècnica de Catalunya, 2017.

[42] S. SHARMA, "Activation Functions in Neural Networks," 06 September 2017. [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. [Accessed 17 December 2021].

[43] IBM Cloud Education, "Convolutional Neural Networks," IBM, 20 October 2020. [Online]. Available: https://www.ibm.com/cloud/learn/convolutional-neural-networks. [Accessed 13 June 2021].

[44] J. Brownlee, "Loss and Loss Functions for Training Deep Learning Neural Networks," machinelearningmastery.com, 23 October 2019. [Online]. Available: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/. [Accessed 05 February 2022].

[45] S. Doshi, "Various Optimization Algorithms For Training Neural Network," Towards Data Science, January 2019. [Online]. Available: https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6. [Accessed 08 April 2020].

[46] F. Chollet, "A ten-minute introduction to sequence-to-sequence learning in Keras," The Keras Blog, 29 September 2017. [Online]. Available: https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html. [Accessed 26 October 2021].

[47] G. Colab, "Sequence to Sequence Learning with Neural Networks," Google Colab, [Online]. Available: https://colab.research.google.com/github/bentrevett/pytorch-seq2seq/blob/master/1%20-%20Sequence%20to%20Sequence%20Learning%20with%20Neural%20Networks.ipynb. [Accessed 11 December 2021].

[48] K. Cho, B. v. Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar.

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention is all you need," in *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, Red Hook, NY, United States, 2017.

[50] D. Bahdanau, K. Cho and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," 2014.

[51] M.-T. Luong and C. D. M. Hieu Pham, "Effective Approaches to Attention-based Neural Machine Translation," 2015.

[52] V. Rus, B. Wyse, P. Piwek, M. Lintean, S. Stoyanchev and C. Moldovan, "The first question generation shared task evaluation challenge," 2010.

[53] H. Ali, Y. Chali and S. A. Hasan, "Automatic question generation from sentences," in *Actes de la 17e conférence sur le Traitement Automatique des Langues Naturelles. Articles courts*, 2010, pp. 213-218.

[54] R. Mitkov, "Computer-aided generation of multiple-choice tests," in *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing*, 2003.

[55] J. Brown, G. Frishkoff and M. Eskenazi, "Automatic question generation for vocabulary assessment," in *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, 2005.

[56] S. Kalady, A. Elikkottil and R. Das, "Natural language question generation using syntax and keywords," in *Proceedings of QG2010: The Third Workshop on Question Generation*, 2010.

[57] M. Heilman and A. S. Noah, "Extracting simplified statements for factual question generation," in *Proceedings of QG2010: The Third Workshop on Ques-tion Generation*, 2010.

[58] Z. Zheng, X. Si, E. Chang and X. Zhu, "K2q: Generating natural language questions from keywords with user refinements," 2011.

[59] L. Becker, S. Basu and L. Vanderwende, "Mind the gap: learning to choose gaps for question generation," *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* pp. 742-751, 2012.

[60] O. Rokhlenko and I. Szpektor, "Generating synthetic comparable questions for news articles," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.

[61] K. Mazidi and R. Nielsen, "Linguistic considerations in automatic question generation," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2014.

[62] C. Zong and M. Strube, "Deep questions without deep," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 2015.

[63] Y. Chali and S. Golestanirad, "Ranking automatically generated questions using common human queries," in *Proceedings of The 9th International Natural Language Generation conference*, 2016.

[64] M. Blšták and V. Rozinajová, "Machine learning approach to the process of question generation," in *International Conference on Text, Speech, and Dialogue*, 2017.

[65] mathworks.com, "deep-learning," www.mathworks.com, 26 October 2019. [Online]. Available: https://www.mathworks.com/discovery/deep-learning.html. [Accessed 3 December 2020].

[66] I. V. Serban, A. García-Durán, C. Gulcehre, S. Ahn, S. Chandar, A. Courville and Y. Bengio, "Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus," *arXiv preprint arXiv:1603.06807,* 2016.

[67] N. Mostafazadeh, I. Misra, J. Devlin, M. Mitchell, X. He and L. Vanderwende, "Generating natural questions about an image," *arXiv preprint arXiv:1603.06059,* 2016.

[68] L. Song, Z. Wang and W. Hamza, "A unified query-based generative model for question generation and question answering," *arXiv preprint arXiv:1709.01058,* 2017.

[69] D. Tang, N. Duan, T. Qin, Z. Yan and M. Zhou, "Question answering and question generation as dual tasks," *arXiv preprint arXiv:1706.02027,* 2017.

[70] A. Patel, "Introduction to CNN, LSTM, GRU, OBJECT RECOGNITION & DATASETS," medium.com, 20 December 2018. [Online]. Available: https://medium.com/@apatel67/introduction-to-cnn-lstm-gru-object-recognition-datasets-7dfa4f8ad8f6. [Accessed 3 December 2020].

[71] J. Amidei, P. Piwek and A. Willis, "Evaluation methodologies in automatic question generation 2013-2018," *The 11th International Natural Language Generation Conference,* 2018.

[72] Albert Gatt and Anja Belz et al, "Introducing shared tasks to NLG: The TUNA shared task evaluation challenge," Berlin Heidelberg, 2010.

[73] Kishore Papineni et al, "BLEU: a method for automatic evaluation of machine translation," in *the 40th Annual meeting of the Association for Computational Linguistics*, 2002.

[74] Banerjee and Laviel, "Meteor: An automatic metric for MT evaluation with improved correlation with human judgments," in *the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 2005.

[75] Lin and Och, "Automatic evaluation of machine translation quality using using longest common subsequence and skip-bigram statistics," 2004.

[76] P. Khullar, K. Rachna, M. Hase and M. Shrivastava., "Automatic Question Generation using Relative Pronouns and Adverbs," in *Proceedings of ACL 2018, Student Research Workshop*, Melbourne, 2018.

[77] Miroslav Blstak and Viera Rozinajov, "Machine Learning Approach to the Process of Question Generation," Springer, 2017.

[78] N. Duan, D. Tang, P. Chen and M. Zhou., "Question Generation for Question Answering," in *Proceedings of the 2017 conference on empirical methods in natural language processing*, Copenhagen, 2017.

[79] B. Leite, H. L. Cardoso, L. P. Reis and C. Soares, "Factual Question Generation for the Portuguese Language," in *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, 2020.

[80] lonweb.org, "AMHARIC," [Online]. Available: http://www.lonweb.org/link-amharic.htm. [Accessed 13 Jun 2020].

[81] amharic.com, "Welcome to Amharic.com," [Online]. Available: http://www.amharic.com/. [Accessed 13 Jun 2020].

[82] Wondwossen Teshome, "Designing Amharic Definitive Question Answering," Addis Ababa University, Addis Ababa, 2013.

[83] S. Muhie, "AMHARIC QUESTION ANSWERING SYSTEM FOR FACTOID QUESTIONS," Addis Ababa University, Addis Ababa, 2009.

[84] C. W. Isenburg, "Grammar of The Amharic Language.," London, 2002.

[85] A. Atelach, "Automatic Sentence Parsing for Amharic Text: An Experiment Using Probabilistic Context Free Grammars," MSc. Thesis, Addis Ababa University, Addis Ababa, 2002.

[86] Brook Eshetu, "Amharic Question Answering for list questions," Addis Ababa University, Addis Ababa, 2013.

[87] S. T. J. P. a. S. R. Alan R, "Design science in information systems research," 2004.

[88] G. F. S. Salvatore T. March *, "Design and natural science research on information technology," *Information and Decision Sciences Department,* 1995.

[89] K. Peffers, "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems,* vol. 24, no. 3, pp. 45-77, 2007.

[90] G. Neubig, "Neural machine translation and sequence-to-sequence models: A tutorial," *arXiv preprint arXiv:1703.01619,* 2017.

[91] K. Peffers, M. Rothenberger, T. Tuunanen and R. Vaezi, "A design science research methodology for information systems research," *International Conference on Design Science Research in Information Systems,* pp. 398-410, 2012.

[92] "Python," Python.org, [Online]. Available: https://www.python.org/about/. [Accessed 12 December 2021].

[93] "Why TensorFlow," TensorFlow, [Online]. Available: https://www.tensorflow.org/. [Accessed 10 June 2021].

[94] R. Kohavi and Ijcai, "A study of cross-validation and bootstrap for accuracy estimation and model selection," Montreal, Canada, 1195.

[95] "beautifulsoup4 4.10.0," pypi.org, [Online]. Available: https://pypi.org/project/beautifulsoup4/4.10.0/. [Accessed 08 September 2021].

[96] G. Colab, "Colaboratory," [Online]. Available: https://research.google.com/colaboratory/faq.html. [Accessed 18 December 2021].

[97] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research,* pp. 1929-1958, 2014.

[98] M. W. Prabowo, I. Budi and H. B. Santoso, "Developing Question Generation System for Bahasa Indonesia Using Indonesian Standard Language Regulation," in *ICSCA 2021: 2021 10th International Conference on Software and Computer Applications,* 2021.

[99] Nan Duan et al, "Question Generation (QG)," microsoft, 2018. [Online]. Available: https://www.microsoft.com/en-us/research/careers/. [Accessed 19 1 2020].

[100] Michael Heilman Noah A.Smith, "GoodQuestion! StatisticalRankingforQuestionGeneration," Carnegie Mellon University, Pittsburgh, 2010.

[101] Klein and C. D. Manning, "Fast exact inference with a factored model for natural language parsing," 2003.

[102] M. Collins, "Head-Driven Statistical Models for Natural Language Parsing," University of Pennsylvania, 1999.

[103] N. P. Er and I. Cicekli, "A Factoid Question Answering System," 2013.

[104] D. Melo, I. Pimenta Rodrigues and V. Beires Nogueira, "A review on cooperative question-answering systems," pp. 23-30, 2013.

[105] E. Riloff and M. Thelen, "A rule-based question answering system for reading comprehension tests," in *ANLP-NAACL 2000 Workshop: Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems*, 2000.

[106] C. Fellbaum, "WordNet," in *Theory and applications of ontology: computer applications*, Springer, 2010, pp. 231-243.

[107] T. Gruber, "Ontology," Springer, 2018.

[108] H. Hussein, M. Elmogy and S. Guirguis, "Automatic english question generation system based on template driven scheme," in *International Journal of Computer Science Issues (IJCSI)*, 2014.

Appendix

Appendix I: Amharic Alphabets

ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ			ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ	ዃ
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ	ሏ		ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ	
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ	ሗ		ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ	
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ	ሟ		ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ	ዟ
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ	ሧ		ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ	ዧ
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ	ሯ		የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ	
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ	ሷ		ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ	ዷ
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ	ሿ		ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ	ጇ
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ	ቋ		ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ	ጓ
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ	ቧ		ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ	ጧ
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ	ቷ		ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ	ጯ
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ	ቿ		ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ	ጷ
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ	ኋ		ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ	ጿ
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ	ኗ		ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ	
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ	ኟ		ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ	ፏ
አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ			ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ	ፗ
ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ	ኳ		ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ	ቯ

Appendix 2: Amharic Numerals

፩	፪	፫	፬	፭	፮	፯
ኣሓዱ	ክልኤቱ	ሠለስቱ	ኣርባዕቱ	ሓምስቱ	ስድስቱ	ሰብዓቱ
1	2	3	4	5	6	7
፰	፱	፲	፳	፴	፵	፶
ሰመንቱ	ተስዓቱ	ዓሠርቱ	ዕሥራ	ሠላሳ	ኣርባዓ	ሓምሳ
8	9	10	20	30	40	50
፷	፸	፹	፺	፻	፼	
ስሳ	ሰብዓ	ሰማንያ	ተስዓ	ምእት	እልፍ	
60	70	80	90	100	10000	

Appendix 3: Amharic punctuation marks

Punctuation	Name in Amharic	English equivalent
፣	ነጥላ ሰረዝ	Comma
።	አራት ነጥብ	Full stop / period
፤	ድርብ ሰረዝ	Colon
፡	ሁለት ነጥብ	Whitespace

Appendix 4: Short words and their expanded form

Short word	Expanded form		Short word	Expanded form
ም/ር/መ	ምክትል ርዕሰ መስተዳድር		ም/ቤት	ምክር ቤት
ገ/መድህን	ገብረ መድህን		ም/ጠ/ሚ	ምክትል ጠቅላይ ሚኒስትር
ም/ከ	ምክትል ከንቲባ		ም/ር/መስተዳድር	ምክትል ርዕሰ መስተዳድር
ም/ፕሬዚዳንት	ምክትል ፕሬዚዳንት		ደ/ብ/ብ/ህ/ክልል	ደቡብ ብሄር ብሄረሰቦች እና ህዝቦች ክልል
ቤ/ክርስቲያን	ቤተ ክርስቲያን		የኢ/ኦ/ተ/ቤተክርስቲያን	የኢትዮጵያ ኦርቶዶክስ ተዋህዶ ቤተክርስትያን
ፍ/ቤት	ፍርድ ቤት		ዶ/ር	ዶክተር
ጽ/ቤታቸው	ጽህፈት ቤታቸው		ሌ/ጀኔራል	ሌተናል ጀኔራል
ት/ሚ	ትምህርት ሚኒስትር		ጠ/ሚ	ጠቅላይ ሚኒስትር
ጠ/ሚር	ጠቅላይ ሚኒስትር		ም/ጠ/ሚ/ር	ምክትል ጠቅላይ ሚኒስትር
ም/ቤቱ	ምክር ቤቱ		ጠ/ሚ/ር	ጠቅላይ ሚኒስትር
ም/ኮሚሽነር	ምክትል ኮሚሽነር		ብ/ጄኔራል	ብርጋዴል ጄኔራል
ወ/ሮ	ወይዘሮ		ም/ጠ/ሚኒስትር	ምክትል ጠቅላይ ሚኒስትር
ወ/ት	ወይዘሪት		ም/ጠ/ሚኒስትሩ	ምክትል ጠቅላይ ሚኒስትሩ
ት/ቤቱ	ትምህርት ቤቱ		ኮ/ር	ኮማንደር
ጠ/ፍ/ቤቱ	ጠቅላይ ፍርድ ቤቱ		ፐ/ር	ፐሮፌሰር

ጠ/ፍርድ	ጠቅላይ ፍርድ	ም/ሊቀመንበር	ምክትል ሊቀመንበር
ጠ/ፍ/ቤት	ጠቅላይ ፍርድ ቤት	ው/ጉ/ሚ	ውጭ ጉዳይ ሚኒስትር
ም/ቤቶቹ	ምክር ቤቶቹ	ጽ/ቤት	ጽህፈት ቤት
መ/ቤቶች	መስሪያ ቤቶቹ	ወ/ሮ	ወይዘሮ
ገ/ሥላሴ	ገብረ ሥላሴ	ም/ቤቱ	ምክር ቤቱ
ክ/ሃገር	ክፍለ ሃገር	ም/ጠ/ኢ.ታማዦር	ምክትል ጠቅላይ ኢ.ታማዦር
ኮ/ል	ኮለኔል	ገ/ትንሣይ	ገብረ ትንሳይ
ም/አፈጉባኤ	ምክትል አፈጉባየ	ብ/ጄ	ብርጋዴል ጄኔራል
ጠ/ሚሩ	ጠቅላይ ሚኒስትሩ	ሜ/ጀ	ሜጀር ጄነራል
ብ/ጀኔራል	ብርጋዴል ጀኔራል	ገ/ሚካኤል	ገብረ ሚካኤል
ሌ/ኮሎኔል	ሌተናል ኮሎኔል	ም/ከንቲባ	ምክትል ከንቲባ
ቤ/ክ	ቤተ ክርስትያን	ሌ/ጄ	ሌተናል ጀኔራል
ባ/ዳር	ባህር ዳር	ክ/ከተማ	ክፍለ ከተማ
ዲ/ን	ዲያቆን	ጽ/ት	ጽህፈት ቤት
ገ/መስቀል	ገብረ መስቀል	ረ/ፕሮፌሰር	ረዳት ፕሮፌሰር
ደ/ወሎ	ደቡብ ወሎ	ወ/ማሪያም	ወልደ ማሪያም
ጄ/ል	ጄነራል	በጽ/ቤታቸው	በጽህፈት ቤታቸው
ጠ/ኤታማዦር	ጠቅላይ ኤታማዦር	ጠ/ር	ጠቅላይ ሚኒስትር
ወ/ሪት	ወይዘሪት	ጠ/ሚር	ጠቅላይ ሚኒስትር
መ/ር	መምህር	ጠ/ሚኒስትር	ጠቅላይ ሚኒስትር
ገ/መድህን	ገብረ መድህን	ሚ/ር	ሚኒስትር
ገ/ሚካዔል	ገብረ ሚካዔል	መ/ቤቱ	መስሪያ ቤቱ
ሰ/ሸዋ	ሰሜን ሸዋ	ፍ/ቤቱ	ፍርድ ቤቱ
ገ/ተንሳይ	ገብረ ተንሳይ	ኢ/ር	ኢንጂነር
ጠ/ም	ጠቅላይ ምክር ቤት	ር/መስተዳድር	ርዕሰ መስተዳድር
ፅ/ቤቶች	ፅህፈት ቤቶች	ፅ/ቤት	ፅህፈት ቤት
ወ/ገብርኤል	ወልደ ገብሬል	ጠ/ሚኒስትሩ	ጠቅላይ ሚኒስትሩ
ሌ/ጀ	ሌተናል ጀኔራል	መሥ/ቤቶች	መስሪያ ቤቶች
ጠ/አቃቤ	ጠቅላይ አቃቢ	ፕ/ት	ፕሬዝደንት
ሀ/ማርያም	ሀይለ ማሪያም	ወ/ስላሴ	ወልደ ስላሴ
ወ/ጊወርጊስ	ወልደ ጊወርጊስ	ም/ጠቅላይ	ምክትል ጠቅላይ
ቅ/ማርያም	ቅድስተ ማሪያም	ኃ/ማርያም	ኃይለ ማሪያም
ም/ዳይሬክተር	ምክትል ዳይሬክተር	ጠ/ም/ቤት	ጠቅላይ ምክር ቤት
ም/ኃላፊ	ምክትል ኃላፊ	ዓ/ም	ዓመተ ምህረት
ረ/ኮሚሽነር	ረዳት ኮሚሽነር	ወ/ሚካኤል	ወልደ ሚካኤል
ጠ/ሚሩ	ጠቅላይ ሚኒስትሩ	ት/ቤቶች	ት/ቤቶች
አ/አ	አዲስ አበባ	ገ/ስላሴ	ገብረ ስላሴ
ሃ/የ/የግል	ሃላፊነቱ የተወሰነ የግል	አ/አ	አዲስ አበባ
ኃ/ማርያም	ኃይለ ማሪያም	ት/ት	ትምህርት
ቤ/ን	ቤተ ክርስትያን	ት/ቤት	ትምህርት ቤት
ም/መቶ	ምክትል መቶ	መ/ቤት	መስሪያ ቤት
ገ/እግዚያብሔር	ገብረ እግዚአብሔር	ገ/እግዚአብሔር	ገብረ እግዚአብሔር
ም/ሚኒስትር	ምክትል ሚኒስትር	ት/ቤቶችን	ትምህርት ቤቶችን

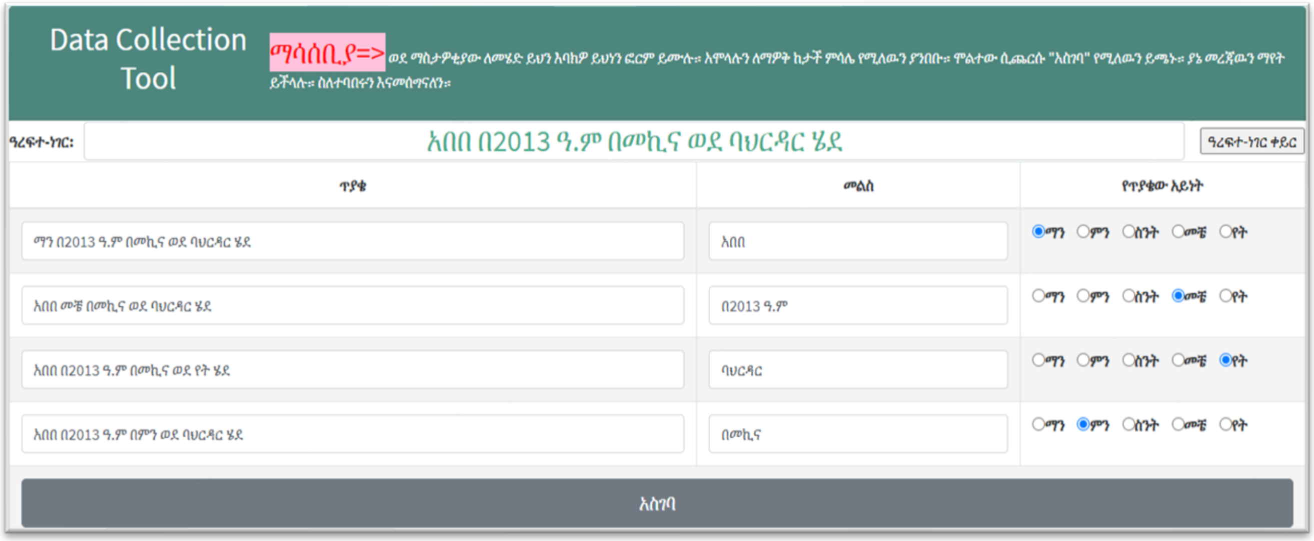
ም/አዛዥ	ምክትል አዛዥ	ቅ/ጽ/ቤት	ቅርንጫፍ ፅህፈት ቤት
ሌ/ጄኔራሎች	ሌተናል ጄኔራሎች	ም/ጠ	ምክትል ጠቅላይ
ቅ/ሲኖዶስ	ቅዱስ ሲኖዶስ	ም/ር	ምክትል ርዕሰ መስተዳድር
ቅ/ሥላሴ	ቅድስተ ሥላሴ	ጽ/ቤቶች	ጽህፈት ቤቶች
ፕ/ት	ፕሬዝደንት	ገ/ማሪያም	ገብረ ማሪያም
ወ/ማርያም	ወልደ ማሪያም	ወ/ጊዮርጊስ	ወልደ ጊዮርጊስ
አ/አበባ	አዲስ አበባ	ክ/ከ	ክፍለ ከተማ

Appendix 5: Sample testing sentences and generated questions with answer

Sentence	Question	Answer
አቶ ንጉሱ ጥላሁን ከአፍሪካ ልማት ባንክ ምክትል ፕሬዝዳንት ጋር ተወያዩ።	ማን ከአፍሪካ ልማት ባንክ ምክትል ፕሬዝዳንት ጋር ተወያዩ?	አቶ ንጉሱ ጥላሁን
	አቶ ንጉሱ ጥላሁን ከማን ምክትል ፕሬዝዳንት ጋር ተወያዩ?	ከአፍሪካ ልማት
	አቶ ንጉሱ ጥላሁን ከማን ጋር ተወያዩ?	ከአፍሪካ ልማት ባንክ ምክትል ፕሬዝዳንት
አፄ ተክለሃይማኖት በአባታቸው ሎሌዎች እጅ በጩቤ ተወግተው ኣረፉ።	አፄ ተክለሃይማኖት በአባታቸው ሎሌዎች እጅ በምን ተወግተው ኣረፉ?	በጩቤ
	አፄ ተክለሃይማኖት በማን እጅ በጩቤ ተወግተው ኣረፉ?	በአባታቸው ሎሌዎች
	በአባታቸው ሎሌዎች እጅ በጩቤ ተወግተው ያረፉት ማን ነበሩ?	አፄ ተክለሃይማኖት
ካናዳ ለኢትዮጵያ የ13 ሚሊዮን ዶላር ድጋፍ አደረገች።	ማን ለኢትዮጵያ የ13 ሚሊዮን ዶላር ድጋፍ አደረገች?	ካናዳ
	ካናዳ ለማን የ13 ሚሊዮን ዶላር ድጋፍ አደረገች?	ለኢትዮጵያ
	ካናዳ ለኢትዮጵያ የስንት ሚሊዮን ዶላር ድጋፍ አደረገች?	የ13
ጀጉል በሀገራችን በርካታ መስጂዶች የሚገኙባት ከተማ ናት።	በሀገራችን በርካታ መስጂዶች የሚገኙባት ከተማ ማን ናት?	ጀጉል
	ጀጉል በየት በርካታ መስጂዶች የሚገኙባት ከተማ ናት?	በሀገራችን
	ጀጉል በሀገራችን በርካታ መስጂዶች የሚገኙባት ምንድን ናት?	ከተማ
ክሪስታቮ ደጋማ በኢትዮጵያ የፖርቹጋሎች ጦር መሪ ነበር።	ክሪስታቮ ደጋማ በኢትዮጵያ የማን ጦር መሪ ነበር?	የፖርቹጋሎች
	ክሪስታቮ ደጋማ በኢትዮጵያ የፖርቹጋሎች ምን ነበር?	ጦር መሪ
	በኢትዮጵያ የፖርቹጋሎች ጦር መሪ ማን ነበር ?	ክሪስታቮ ደጋማ
ከበካፋ በፊት የነበረው ንጉስ በመርዝ ሞተ።	ከበካፋ በፊት የነበረው ንጉስ በምን ሞተ?	በመርዝ
	በመርዝ የሞተው ማን ነው?	ከበካፋ በፊት የነበረው ንጉስ
ቅዱስ ጊዮርጊስ ከሲዳማ ቡና በአቻ ተለያየ።	ማን ከሲዳማ ቡና በአቻ ተለያየ?	ቅዱስ ጊዮርጊስ
	ማንና ሀዋሳ ከነማ አቻ ተለያዩ?	ቅዱስ ጊዮርጊስ

ነጋዴዎች በራሳቸው ፈቃድ በአገሪቱ ውስጥ ይንቀሳቀሳሉ።	በራሳቸው ፈቃድ በአገሪቱ ውስጥ እነማን ይንቀሳቀሳሉ?	ነጋዴዎች
	ነጋዴዎች በራሳቸው ፈቃድ የት ውስጥ ይንቀሳቀሳሉ?	በአገሪቱ ውስጥ
	ነጋዴዎች በማን ፈቃድ በአገሪቱ ውስጥ ይንቀሳቀሳሉ?	በራሳቸው
ሜሪ ወደ ሃዋሳ እንደምትሄድ ለታላቅ ወንድሜ ነገረችው።	ሜሪ ወደ የት እንደምትሄድ ለታላቅ ወንድሜ ነገረችው?	ሃዋሳ
	ሜሪ ወደ ሃዋሳ እንደምትሄድ ለማን ነገረችው?	ለታላቅ ወንድሜ
	ማን ወደ ሃዋሳ እንደምትሄድ ለታላቅ ወንድሜ ነገረችው?	ሜሪ
በድርቁ ምክንያት ጉዳት የደረሰባቸው ኢትዮጵያዊያን ቁጥር ሀምሳ ሽህ ደረሰ።	በድርቁ ምክንያት ጉዳት የደረሰባቸው ኢትዮጲያዊያን ቁጥር ስንት ደረሰ ?	ሀምሳ ሽህ
	ሀምሳ ሽህ ኢትዮጲያዊያን በምን ምክንያት ጉዳት ደረሰባቸው ?	በድርቁ ምክንያት
በአዳማ ከተማ በ100 ሚሊዮን ብር ወጪ የተገነባ ስካይ ሲቲ ሆቴል ተመረቀ።	በየት ከተማ በ100 ሚሊዮን ብር ወጪ የተገነባ ስካይ ሲቲ ሆቴል ተመረቀ?	በአዳማ
	በአዳማ ከተማ በስንት ሚሊዮን ብር ወጪ የተገነባ ስካይ ሲቲ ሆቴል ተመረቀ?	በ100

Appendix 6: Data collection system interface

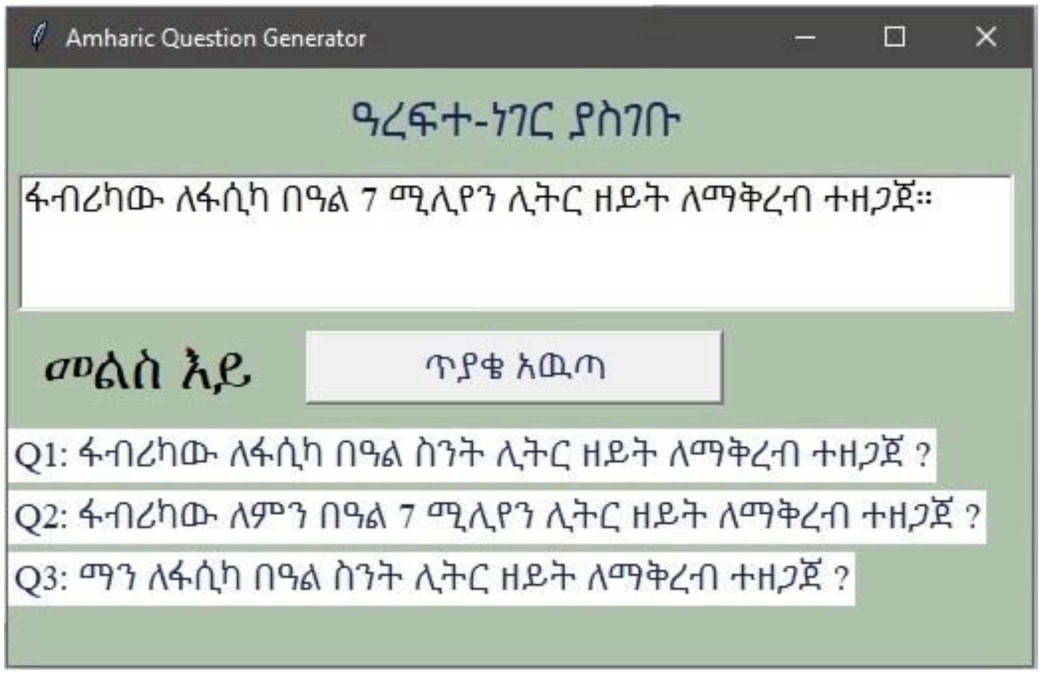


ማሳሰቢያ=> ወደ ማስታዎቂያው ለመሄድ ይህን እባክዎ ይህንን ፎርም ይሙሉ። እሞላሉን ለማዎቅ ከታች ምሳሌ የሚለዉን ያንበቡ። ሞልተው ሲጨርሱ "አስገባ" የሚለዉን ይጫኑ። ያኔ መረጃዉን ማየት ይችላሉ። ስለተባበሩን እናመሰግናለን።

ዓረፍተ-ነገር: አበበ በ2013 ዓ.ም በመኪና ወደ ባህርዳር ሄደ

ጥያቄ	መልስ	የጥያቄው አይነት
ማን በ2013 ዓ.ም በመኪና ወደ ባህርዳር ሄደ	አበበ	[✓ ማን] ምን ስንት መቼ የት
አበበ መቼ በመኪና ወደ ባህርዳር ሄደ	በ2013 ዓ.ም	ማን ምን ስንት [✓ መቼ] የት
አበበ በ2013 ዓ.ም በመኪና ወደ የት ሄደ	ባህርዳር	ማን ምን ስንት መቼ [✓ የት]
አበበ በ2013 ዓ.ም በምን ወደ ባህርዳር ሄደ	በመኪና	ማን [✓ ምን] ስንት መቼ የት

Appendix 7: Amharic Question Generation System Interface



ፋብሪካው ለፋሲካ በዓል 7 ሚሊየን ሊትር ዘይት ለማቅረብ ተዘጋጀ።

Q1: ፋብሪካው ለፋሲካ በዓል ስንት ሊትር ዘይት ለማቅረብ ተዘጋጀ ?

Q2: ፋብሪካው ለምን በዓል 7 ሚሊየን ሊትር ዘይት ለማቅረብ ተዘጋጀ ?

Q3: ማን ለፋሲካ በዓል ስንት ሊትር ዘይት ለማቅረብ ተዘጋጀ ?