# Automatic Question Generation from Afaan Oromo Text Using Deep Learning

MSc. Thesis Research

Jemal Abdela

MARCH, 2025

HARAMAYA UNIVERSITY

# Automatic Question Generation from Afaan Oromo Text Using Deep Learning

A Thesis Submitted to the College of Computing and Informatics,

Department of Information Science, School of Graduate Studies,

HARAMAYA UNIVERSITY

In a Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE IN INFORMATION SCIENCE

Jemal Abdela

MARCH, 2025

HARAMAYA UNIVERSITY

POSTGRADUATE PROGRAM DIRECTORATE

HARAMAYA UNIVERSITY

We hereby certify that we have read and tested this Thesis entitled "Question Generation from Afaan Oromo Text Using Deep Learning", prepared under our guidance by Jemal Abdela. We recommend that it be submitted as fulfilling the thesis requirement.

| | | |
|---|---|---|
| Million Meshesha (PhD) | | March 21, 2025 |
| Major Advisor | Signature | Date |
| Jemal Abate (MSc) | | March 21, 2025 |
| Co-Advisor | Signature | Date |

As a member of the Board of Examiners of the MSc Thesis Open Defense Examination, I certify that I have read and evaluated the Thesis prepared by Jemal Abdela and examined the candidate. I recommend that the thesis be accepted as fulfilling the thesis requirements for the degree of Master of Science in Information Science.

| | | |
|---|---|---|
| | | |
| Chairperson | Signature | Date |
| | | |
| Internal Examiner | Signature | Date |
| Siraj Sebhatu(Ph.D) | | 02/04/2025 |
| External Examiner | | |

Final approval and acceptance of the Thesis are contingent upon the submission of its final copy to the Council of Postgraduate Directorate (CPGD) through the candidate's department or the School Graduate Committee (DGC or SGC).

# DEDICATION

This thesis is dedicated to my father and mother, my brother Mrs. Johar Abdela, and my child, Rayyan Jemal.

# STATEMENT OF THE AUTHOR

By my signature below, I declare and affirm that this thesis is my original work. I have adhered to all ethical and technical principles of scholarship in its preparation, data collection, evaluation, and compilation. Any scholarly matter included in the Thesis has been properly acknowledged through citation and reference.

This Thesis is submitted in partial fulfillment of the requirements for the Master of Science in Information Science degree at Haramaya University. The Thesis will be deposited in the Haramaya University Library and will be accessible to borrowers under the library's rules. I solemnly declare that this Thesis has not been submitted to any other institution for the award of any academic degree, diploma, or certificate.

Brief quotations from this Thesis may be made without special permission, provided that accurate and complete acknowledgment of the source is given. Requests for permission for extended quotations or reproduction of this Thesis in whole or in part may be granted by the Head of the School or Department when, in their judgment, the proposed use of the material serves the interest of scholarship. In all other instances, permission must be obtained from the author of the Thesis.

Name: Jemal Abdela Urunji

Date: March 25, 2025

Department: Information Science

Signature: _____

# BIOGRAPHICAL SKETCH

Jemal Abdela is a teacher at Haramaya Ifaboru Boarding Secondary School,Addelle, Maya City, Ethiopia. He was born on May 10, 1997 (GC), in BedenoWoreda, East Hararghe Zone, Oromia Regional State, Ethiopia, specifically in Hindhessa Kebele. Jemal completed his primary education (grades 1-8) at Bedeno Elementary School. He attended Bedeno Secondary School for his secondary education and completed his senior secondary education at Bedeno Preparatory School.

Jemal graduated from the College of Computing and Informatics at Madda Walabu University with a Bachelor of Science in Information System in June 2017. After earning his degree, he started working as an ICT Technician at Haramaya University in 2018. He served there for three years.

In 2022, Jemal joined the Haramaya Ifaboru Boarding Secondary School in Maya City after applying for a vacancy announced by the Haramaya University Human Resource Department for an ICT teaching position, all while pursuing his Master's degree in Information Science.

In 2021, Jemal began his Master's studies in Information Science at Haramaya University through the regular postgraduate program. He has undertaken significant research as part of his Master's studies, contributing to the field of Information Science with his work on automatic Question Generation from Afaan Oromo Text Using Deep Learning.

## ACKNOWLEDGEMENT

First and foremost, I would like to express my deepest gratitude to the almighty Allah, the Most Forgiving and the Most Merciful, for granting me the strength, courage, and patience to persevere through the challenges and obstacles I faced throughout my journey in the postgraduate program and the completion of this thesis.

I am especially thankful to my brother, Mrs. Johar Abdela, my beloved Mother, and my wife Miss. Rumana Yusuf, whose unwavering love, compassion, and enthusiasm accompanied me throughout my academic career.
Their moral, financial, psychological, and material support, along with their endless commitment, has been vital to my success. My heartfelt thanks also go to all my family members for their constant encouragement, moral support, and helping hand, which made it possible for me to complete my studies.

I would also like to extend my sincere gratitude to my Major Advisor, Dr. Million Meshesha, for his invaluable guidance, constructive suggestions, and insightful comments throughout the development of this thesis. His genuine encouragement, expertise, and professional advice have greatly contributed to the successful completion of this work.

In addition, I would like to thank my Co-Advisor, Mr. Jemal Abate (MSc), an instructor in the College of Computing and Informatics at Haramaya University, for his valuable advice, direction, and continuous support during this research.

A special mention goes to my friend as well as my brother, Abdullahi Ahmed, for assisting me in the preparation of the dataset for Afaan Oromo text.

Finally, I would like to express my deepest appreciation to my friends, colleagues, and others who provided support along the way. I am particularly grateful to my school Academic leader, Mr. Habib Hadji, and Mrs. Habtamu Jeilu, for their brotherly advice and assistance.

# Contents

# List Figures



# List of Table

# LIST OF ABBRIVATIONS AND ACRONYM

| | |
|---|---|
| DS | Dialog System |
| EOS | End of sentence |
| GAN | Generative Adversarial Networks |
| LSTM | Long Short-Term Memory Networks |
| MLP | Multi-Layer Perceptron's |
| NER | Named Entity Recognition |
| NLG | Natural Language Generation |
| NLP | Natural Language Processing |
| NLU | Natural Language Understanding |
| PoS | Part of Speech |
| QAS | Question-Answering Systems |
| QG | Question generation |
| RNN | Recurrent Neural Networks |
| SOS | Start of sentence |

## Abstract

In the modern digital era, the availability of electronic content in multiple languages has grown considerably. Nevertheless, generating questions from these materials remains a labor-intensive and time-consuming process, particularly for low-resource languages like Afaan Oromo. While significant progress has been made in Automatic Question Generation (AQG) for languages such as English, Chinese, Amharic and Somali. there is a notable lack of technology with NLP on Afaan Oromo QG because Afaan Oromo has the unique linguistic characteristics. This study bridges this gap by developing an AQG system for Afaan Oromo using deep learning models, specifically LSTM, Bi-LSTM, and GRU.

To build the model a dataset of 5,000 paragraph-question-answer triples was meticulously curated from Afaan Oromo textbooks and educational resources. The dataset underwent preprocessing steps such as tokenization, normalization, and word embedding using Word2Vec. A deep learning model with an attention mechanism was employed to generate questions by design science research methodology, Among the models evaluated, the Bi-LSTM demonstrated the highest performance, achieving a training accuracy of 95.3% and a validation accuracy of 92.5%. The LSTM model also performed well, with a training accuracy of 92.31% and a validation accuracy of 91.02%, while the GRU model showed marked improvement after hyperparameter tuning, reaching 88.0% training accuracy and 87.0% validation accuracy.

The results indicate that the Bi-LSTM model is the most effective for generating both factoid and non-factoid questions in Afaan Oromo. Future research should explore transfer learning from pre-trained models, expand the dataset through collaborations with educational institutions, and integrate advanced neural architectures like Transformers to further enhance performance and question quality. This study makes a significant contribution to the field of Natural Language Processing (NLP) by pioneering AQG for Afaan Oromo, use as input for QA system, providing a foundation for future research and practical applications in education and language preservation.

Keywords: Afaan Oromo, Question Generation, Deep Learning, BI-LSTM

# 1. INTRODUCTION

This chapter provides a comprehensive overview of the background, problem statement, objectives, significance, scope, and limitations of the study. It also introduces the foundational concepts of Natural Language Processing (NLP) and Deep Learning (DL), which are central to this research. The chapter sets the stage for understanding the challenges and opportunities in developing an Automatic Question Generation (AQG) system for Afaan Oromo, a low-resource language with unique linguistic characteristics.

## 1.1 Background of the study

In the era of digital transformation, the accessibility of electronic materials in multiple languages has grown exponentially. However, the manual creation of questions from these materials remains a labor-intensive and time-consuming process, particularly for low-resource languages like Afaan Oromo. This challenge is further compounded by the improvement technological resources on Afaan Oromo educational resources Automatic Question Generation (AQG) systems. While significant progress has been made in Natural Language Processing (NLP) and Deep Learning (DL) for widely spoken languages like English and Chinese, there is a notable research gap in applying these technologies to Afaan Oromo (Zhang, 2020)(Le, Kojiri, & Pinkwart, 2014).

Natural Language Processing (NLP) is a field of artificial intelligence that focuses on the interaction between computers and human language. It enables machines to understand, interpret, and generate human language in a way that is both meaningful and contextually relevant. Deep Learning (DL), a subset of machine learning, leverages neural networks with multiple layers to model complex patterns in data. When applied to NLP tasks, DL models such as Long Short-Term Memory (LSTM), Bidirectional LSTM (Bi-LSTM), and Gated recurrent units (GRU) have shown remarkable success in tasks like machine translation, text summarization, and question generation. However, these advancements have largely bypassed low-resource languages like Afaan Oromo, which lack the resources and pre-trained models necessary for effective NLP applications (Damtie, 2021; Saron & Zelelew, 2021; Meskerem Derese, 2016).

Afaan Oromo, spoken by over 37.4 million native speakers, is one of the most widely used languages in Ethiopia and serves as the official language of the Oromia region. Despite its

prominence, there is a notable absence of technological resources and datasets for advanced NLP tasks, particularly in question generation. This gap limits the development of educational tools, question-answering systems and other NLP applications that could benefit millions of speakers. Previous studies have explored aspects of Afaan Oromo NLP, such as question answering system (Chaltu Fita, 2015; Nuguse, 2021), which are focus on to get answer from questions, morphological analysis (Tesfaye, 2005; Olani, 2014). Which are focusing on structures of afaan oromo language, and phonological characteristics but none have specifically addressed automatic question generation using deep learning approaches.

The problem addressed in this research is the technological effective of an AQG system for Afaan Oromo, which limits the availability of educational tools and resources for Afaan Oromo speakers. Existing AQG systems for other languages cannot be directly applied to Afaan Oromo due to its limit of digital resources and the complex language structures This research aims to bridge this gap by developing a deep learning-based AQG system specifically designed for Afaan Oromo. The proposed solution involves the creation of a 5,000 paragraph-question-answer dataset, the application of DL models (LSTM, Bi-LSTM, and GRU), and the development of a prototype for real-time question generation.

The investigative focus of this research is to explore how DL models can be adapted to handle the linguistic complexities of Afaan Oromo and to evaluate their effectiveness in generating factoid and non-factoid questions. The study applies DL models with attention mechanisms to generate questions, this study applies deep learning (DL) models with attention mechanisms to generate questions from Afaan Oromo text. The experimental results indicate that among the models tested, the Bi-LSTM achieved the highest performance, with a training accuracy of 95.3% and a validation accuracy of 92.5%. The LSTM model also performed well, achieving 92.31% training accuracy and 91.02% validation accuracy, while the GRU model showed significant improvement after hyperparameter tuning, reaching 88.0% training accuracy and 87.0% validation accuracy.

This research contributes to the field of NLP by pioneering AQG for Afaan Oromo, offering a foundation for future studies and practical applications in education and language preservation. By addressing the unique challenges of Afaan Oromo, this work not only advances the state of NLP for low-resource languages but also provides a scalable solution for generating educational

content in Afaan Oromo. The findings of this study have the potential to revolutionize educational tools, enhance language preservation efforts, and promote the use of Afaan Oromo in digital platforms.

## 1.2. Statement of the Problem

Ethiopia is a linguistically diverse nation with over 80 languages, among which Afaan Oromo stands out as one of the most widely spoken languages under the Cushitic language family. According to Ethnologue's 2021 data, Afaan Oromo has approximately 37.4 million native speakers, making it the most spoken first language in Ethiopia, accounting for over 33% of the country's population (Eberhard, 2019). The language uses a Latin-based script called "Qubee," consisting of 28 basic characters, and serves as the official language of the Oromia regional state and the medium of instruction in primary schools across the region. Afaan Oromo is also a subject of academic study in universities within Ethiopia and neighboring countries like Kenya. With the increasing availability of digital content, including journals, newspapers, educational resources, and legal documents, Afaan Oromo has seen a significant rise in electronic text production (Ibrahim Bedane, 2015).

Despite its prominence, Afaan Oromo faces a critical gap in Natural Language Processing (NLP) research, particularly in the area of automatic question generation (AQG). Question generation is a vital NLP task that facilitates quick access to text records, enhances learning environments, and supports the development of question-answering systems. However, generating questions from natural language text is a complex process that requires addressing linguistic nuances and contextual dependencies (Feng et al., 2018). While neural networks (NN) have advanced in interpreting sentences as sequences of words, they often overlook the relationships between words, which are crucial for generating meaningful questions (Medhanit, 2019).

The traditional process of preparing questions is time-consuming, costly, and requires expertise, making it impractical for large-scale applications. Automated question generation systems can significantly reduce the time and effort required, thereby improving accessibility to educational and informational resources (Asudani et al., 2023). Research in question generation has made strides in languages like English, with Mazidi (2016) developing systems to address the lack of practice exercises in texts, and Duan et al. (2017) leveraging deep learning for community question answering. Similarly, Mokhtar et al. (2021) focused on automating exam preparation,

but these models may not be directly applicable to Afaan Oromo due to differences in grammar, syntax, and available resources focuses on overcoming data scarcity and resource limitations and to address broader needs, such as improving access to information, supporting literacy, and preserving linguistic heritage. while local studies like Bekele (2020) explored automatic generation of Amharic math word problems, and Fikir & Tezera (2022) investigated question generation from Amharic legal texts. However, these advancements have not been extended to Afaan Oromo, leaving a significant gap in NLP tools for the language.

Afaan Oromo have many challenges with NLP technology, Limited Digital Resources Unlike English and other high-resource languages, Afaan Oromo has limited annotated datasets, pre-trained embeddings, and computational resources necessary for NLP applications. The unavailability of standardized datasets makes training deep learning models challenging.

Complex Linguistic Structure the Subject-Object-Verb (SOV) sentence structure, agglutinative morphology, and rich inflection system of Afaan Oromo introduce difficulties in sentence parsing, tokenization, and generating grammatically correct questions. These linguistic challenges demand customized NLP solutions that are different from high-resource language models.

Underdeveloped AI-based Educational Tools Many automated tutoring systems and question-answering applications are designed primarily for English and high-resource languages. No existing AI-driven system is optimized for Afaan Oromo question generation, making it difficult to integrate such technologies into e-learning platforms, chatbots, or AI assistants.

Computational Limitations for Training Deep Learning Models – Training NLP models requires high-performance computing resources, but Afaan Oromo lacks the necessary AI infrastructure, making model training and deployment a significant challenge This exacerbates the gap in deploying NLP tools for educational and informational applications.

The study Heilman (2011) demonstrated the effectiveness of rule-based and statistical methods for factual question generation in English, these approaches are not directly applicable to Afaan Oromo due to its distinct grammatical and syntactic characteristics. Similarly, Medhanit (2019) explored Amharic question answering using machine learning but did not address question generation or the specific needs of Afaan Oromo. Furthermore, Sebsibe (2022) developed a

question-answering system for Afaan Oromo but did not tackle the automation of question generation, highlighting the need for targeted research in this area.

There are many challenges in developing NLP tools for Afaan Oromo While foundational studies have explored aspects such as question answering (Chaltu Fita, 2015; Nuguse, 2021), morphological analysis (Tesfaye, 2005; Olani, 2014), and phonological characteristics (Ibrahim Bedane, 2015), there has been no focused research on automatic question generation using deep learning approaches. This gap limits development of NLP technology such as the development of educational tools, question-answering systems, and other NLP applications that could benefit millions of Afaan Oromo speakers.

Recent advancements in Afaan Oromo NLP, such as the Question-and-Answer System for Afaan Oromo (Sebsibe, 2022) and studies on next-word generation using deep learning models like LSTM and Bi-LSTM (Bekan Kitaw Mekonen, 2021), have laid some groundwork. However, these efforts do not specifically address automatic question generation. Similarly, research on enhancing conversational AI for low-resource languages, such as Somali, offers insights into cross-lingual adaptation techniques that could be applied to Afaan Oromo (Mohamud Osman Hamud, 2025). These studies underscore the growing interest in advancing NLP for low-resource languages but also highlight the need for targeted research on question generation.

This study proposes the development of an automated Afaan Oromo question generator using a neural network integrated with predefined rules. The model was trained on a manually annotated dataset created from diverse Afaan Oromo texts, including oromia educational materials. By leveraging insights from foundational studies and addressing the above challenges of Afaan Oromo, this research aims to create a robust framework for automatic question generation. This study not only fills a critical gap in Afaan Oromo NLP but also contributes to the broader goal of enhancing educational and informational tools for low-resource languages, ultimately benefiting millions of speakers.

By addressing these research gaps, this study aims to contribute to the broader understanding of next-word prediction models for low-resource languages and provide valuable insights for the development of more inclusive and accessible language technology solutions.

The absence of an AQG system for Afaan Oromo limits the availability of educational tools and resources for Afaan Oromo speakers. Existing Automatic Question Generation Models (AQGM) developed for languages like Amharic, Somali, English or Chinese cannot be directly applied to Afaan Oromo due, Limited Digital Resources, complex language structures and Additionally, Afaan Oromo is a low-resource language, lacking annotated datasets, pre-trained embeddings, and computational resources, which are essential for training deep learning models.

To overcome these limitations, this study introduces a novel approach, including the creation of a custom dataset of 5,000 paragraph-question-answer triples, training Word2Vec embeddings from scratch and developing a deep learning model with an attention mechanism tailored to Afaan Oromo's. Evaluate model performance using both quantitative metrics (accuracy, validation loss) and qualitative human evaluations (fluency, grammatical correctness, relevance, coherence).

By developing an AI-driven AQG system, this research contributes to the advancement of NLP for low-resource languages and lays the foundation for future AI-powered applications in education, automated tutoring, and digital learning platforms for Afaan Oromo speakers.

To this end, an attempt has been made in this study to answer the following research questions through investigation.

- Which embedding technique provides the most meaningful representations for Afaan Oromo sentence structure in the context of automatic question generation?
- How do different deep learning models compare in terms of performance when applied to Afaan Oromo question generation?
- What is the performance of the proposed model in Afaan Oromo question generation from text, as measured by evaluation?

## 1.3 Objective of the study

### 1.3.1. General objective

The general objective of this study is to design and development an automatic question generation model for Afaan Oromo text documents using deep learning approach.

### 1.3.2 Specific objectives

The study specifically aims to accomplish the following tasks so as to achieve the general objective of the study:

- To conduct critical literature review on the concept of automatic question generation from documents.
- To identify and analyze the linguistic features and sentence structures in Afaan Oromo that are essential for generating accurate and meaningful questions.
- To prepare a dataset of Afaan Oromo text documents suitable for training a question generation model by word2vec.
- To design and implement a deep learning architecture suitable for generating questions from Afaan Oromo text documents.
- To evaluate and compare the performance of various deep learning architectures for Afaan Oromo question generation and identify the most suitable model.
- To assess the performance of the proposed model in generating Afaan Oromo questions from text using quantitative and qualitative evaluation metrics.
- To explore the practical applications of the proposed model in educational and informational contexts, such as generating practice questions for students or facilitating information retrieval for the general public.

## 1.4 Scope and Limitation of the study

This research focuses on developing an Automatic Question Generation (AQG) model for the Afaan Oromo language using deep learning techniques. It specifically aims to generate both factoid and non-factoid questions from Afaan Oromo text documents, including Oromia Educational Bureau. The study involves designing and implementing a deep learning-based AQG system using LSTM, Bi-LSTM, and GRU models to evaluate performance in question generation. Given the linguistic complexities language structures of Afaan Oromo lack of NLP resources, the study is limited to handling text-based question generation and does not include spoken language, multimedia formats (audio/video), or non-text content.

Despite its contributions, the study faced several challenges, primarily the lack of publicly available digital resources with complex structures of Afaan Oromo. To address this, manual

annotation dataset with language experts was conducted using Afaan Oromo textbooks educational materials. making automatic question transformation more difficult. This issue was mitigated through preprocessing techniques, include cleaning, Normalization and tokenization, alongside the use of Word2Vec embeddings to improve text representation. Additionally, balancing the generation of factoid and non-factoid questions was a challenge, as factoid questions are structurally simpler than non-factoid questions, which require deeper reasoning and context. To address this, the model was trained separately for each question type, ensuring optimized accuracy and fluency (Damtie, 2021; Saron & Zelelew, 2021).

Another limitation was computational constraints, as deep learning models require significant processing power for training. This challenge was addressed by leveraging Google Colab and cloud-based resources for efficient model training. Additionally, evaluating the performance of the model was difficult due to the lack of standardized benchmarks for Afaan Oromo AQG systems.

## 1.5 Significance of the Study

The significance of this study extends beyond its academic contributions, with the potential to impact various sectors. By pioneering the development of an automatic question generation system for the Afaan Oromo language using Deep learning models and predefined rules, this research not only advances the field of natural language processing but also opens new possibilities for practical applications. The implementation of such a system has the potential to revolutionize question-answering systems in educational institutions and customer service sectors alike.

In educational, the generation of non-factoid questions and factoid can significantly enhance the learning experience by encouraging critical thinking and deeper understanding of the material. Meanwhile, in customer service settings, the integration of chatbots armed with a repository of frequently asked questions can streamline and expedite the resolution of customer queries, leading to improved customer satisfaction and efficiency in service delivery.

Moreover, this groundbreaking work in the Afaan Oromo language not only fills a critical gap in Afaan Oromo linguistic research but also paves the way for future studies and innovations in language preservation and technological advancement. By providing valuable insights and

establishing a solid foundation for further exploration, this study contributes to the promotion and preservation of the Afaan Oromo language and its cultural heritage in the digital age.

## 1.6 Organization of the Thesis

Chapter One serves as the introduction to the study, providing the background information, problem statement, objectives, significance of the study, scope, and limitations. This chapter also discusses the evaluation techniques employed.

Chapter Two focuses on the literature review, divided into two parts: conceptual review and review of related works. The conceptual review covers the basics of Afaan Oromo language, including its alphabets, grammar, and sentence structure. Additionally, it delves into deep learning techniques such as Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (BILSTM), and Gated recurrent unit (GRU). The review of related works examines previous research conducted locally and globally, with a specific emphasis on deep learning approaches.

Chapter Three provides an overview of the methodology employed in the study, with a focus on designing and developing tools, techniques, and methods using deep learning models such as LSTM, BILSTM and GRU. It also discusses aspects like data collection, preparation, and pre-processing, as well as the algorithms utilized in the pre-processing stage.

Chapter Four presents the architecture of the system, detailing its components and their functionalities. This chapter also includes the system performance evaluation model.

Chapter Five focuses on the demonstration and evaluation of the system. It covers aspects such as test document and query preparation, experimentation, and the evaluation of the system's performance using deep learning techniques, including LSTM, BILSTM and GRU. The chapter also analyzes the results obtained from the experiments.

The final chapter, Chapter Six, concludes the thesis by summarizing the research findings and achievements. It also provides recommendations for future work and suggests directions for further research, particularly in the application of deep learning models like LSTM, BILSTM and GRU. in the context of Afaan Oromo language processing.

# 2 LITERATURE REVIEW

## 2.1. Overview

This chapter presents the general overview of the Afaan Oromo language such as the alphabets and sounds, dialects and varieties. It is also dedicated to review some general and basic concepts about the Afaan Oromo sentence structure Afaan Oromo Interrogative Sentence Structure. There are also materials reviewed to identify approach; techniques applied to question generation. Also related works from local and global researches has been made so as to identify research gap.

In this chapter we made extensive literature review. 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 Somali, Amharic, English, Chinese and Portuguese which are related to our work.

## 2.2. Afaan Oromo language

Afaan Oromo, a member of the East Cushitic language family within the Afro-Asiatic language superfamily, is the predominant language spoken in Ethiopia, with the highest number of speakers. It is estimated that around 40 million people in Ethiopia, comprising 50% of the country's population, are native speakers of Afaan Oromo (Desta, 2018). This language is not only prevalent within Ethiopia but also used in neighboring countries such as Kenya and Somalia (Bijiga et al., 2015). Afaan Oromo holds the official language status in the Oromia Regional State and is used as a medium of instruction in primary and junior secondary schools in the region. Presently, Afaan Oromo plays a significant role in various domains including research, administration, political discussions, and social interactions. It is also employed as an academic language in several Ethiopian universities, including Jimma University, Addis Ababa University, Ambo University, Haramaya University, as well as primary schools in the Oromia Regional State (Desta, 2018). The writing system used for Afaan Oromo, known as Qubee, is based on the Latin alphabet and has been in use since 1842 (Hayward, 1978). It is noteworthy that Afaan Oromo is the third most widely spoken African language, following Kiswahili and Hausa (Tesfaye, 2005).

### 2.2.1. Dialects and Varieties

Afaan Oromo is a sociolinguistic language consisting of four major varieties: Borana-Arsi Guji Oromo, Eastern Oromo, Orma (Oromo in Kenya) and West Central Oromo (Eggi, 2012; Bijiga et al., 2015). These four varieties depend on geographical area. Even if there are strong similarities among these four varieties, but there is also difference between them. This difference is a spoken variety that is characteristics of a particular area, community or group often with relatively minor differences in vocabulary, style, spelling and pronunciation. There are many synonym words which may challenge question generation as per the context.

### 2.2.2. Alphabets

Afaan Oromo is a phonetic language, meaning it is spoken the way it is written. In contrast to English, where the same letter can have different sounds in different words, Afaan Oromo maintains consistent pronunciation of its characters across all words. Prior to the 1970s, Afaan Oromo was written using either the Ge'ez script or the Latin alphabet. However, since 1991, the Latin alphabet has been officially used as the writing system for the Oromo language (Bijiga et al., 2015;Ibrahim Bedane, 2015).

Afaan Oromo utilizes the Latin characters, with some modifications to reflect the specific sounds of its consonants and vowels. It consists of 28 letters known as "qubee" (Ibrahim Bedane, 2015). The vowels in Afaan Oromo are represented by the five basic letters: a, e, i, o, u. Additionally, it includes the typical Eastern Cushitic set of five short and five long vowels, achieved by doubling the five vowel letters: "aa," "ee," "ii," "oo," "uu." However, there are three borrowed consonant letters, P, V, and Z, which are not native to Afaan Oromo, as no Oromo word is written with these letters. They are included to accommodate borrowed terms from other languages, such as English. For example, words like "Police," "Piano," "Television," "Video," etc. The combinations of consonant letters, such as ch, dh, ny, sh, and ph, are referred to as "qubee dacha" (Eggi, 2012; Megersa, 2020).

### 2.2.3 Afaan Oromo sentence structure

A general sentence structure in Afaan Oromo differs from English language which is Subject-Verb Object (SVO), whereas Afaan Oromo language sentence structure is Subject-Object-Verb (SOV) or a simple Subject + Verb arrangement, where S is subject, V is verb and O is object (Tesfaye, 2005).This means these two languages have differences in their syntactic structure. Consider the sentence, “Ani foon nyaadhe.”, i. e “I ate meat.” In that sentence: “ani”= “I” is the

subject.“Foon” = “meat” is the object.“nyaadhe”=“Ate” is the verb, another example, in the Afaan Oromo sentence ”Tolosaan bilisa bahe”; “Tolosaan ”is a subject, ”bilisa” is an object and ”bahe” is a verb. Therefore, it has SOV structure. The translation of the sentence in English is “Tolosa has got freedom” which has SVO structure.

There is also a difference in the formation of adjectives in Afaan Oromo and English (Olani, 2014) In Afaan Oromo adjectives follow a noun or pronoun; their normal position is close to the noun they modify while in English adjectives usually precede the noun. For instance, namicha gaarii (good man), gaarii (adj.) follows namicha (noun).

### 2.2.4. Afaan Oromo Interrogative Sentence Structure

Interrogative sentence is a form of sentence that asks a question as compared to sentences that make an argument, deliver an order, or express an exclamation. This type of sentence has a significant role in our everyday life, since most of the discussions occur while asking and answering questions. An interrogative sentence ends with question mark (?). Interrogative questions can be wh-question (content question) or yes/no question (Leonarduzzi & Herment, 2020).

There are a number of interrogatory words to construct questions in different languages. English has question words like who, where, what, when, why… are used to construct a question (Leonarduzzi & Herment, 2020).

In Afaan Oromo interrogative particles help to construct a question sentence. Interrogative particles are also known as interrogative pronouns. Some of the Afaan Oromo interrogative particles are: “eessaatti” where “maaliif” why, “yoom” when, “maali” what “akkamitti” how and so on. These interrogative particles are used to construct the factoid and non-factoid questions (Daba, 2021)

### 2.2.5 Afaan Oromo Morphology

The study of the meaningful components of words is the focus of the linguistic field of morphology According to Jurafsky & Martin, (2009), morphology is the study of how words are constructed from smaller, more intelligible elements known as morphemes. The most fundamental part of language structure is the word. Afaan Oromo's morphology is intricate and rich, just like that of other Ethiopian languages.

2.2.5.1 Types of morphology in Afaan Oromo

In Afaan Oromo language, we have two broad types of morphology; namely, derivational morphology and inflectional morphology (Tadele 2009).

Inflectional Morphology: The technique of adapting words to their respective roles inside a sentence without affecting the meaning of the basic words is known as inflectional morphology. Alternately, it can be described as the alteration of a word's form for grammatical purposes. It can take the form of several word classes. Verbal inflection, noun inflection, and adjective inflection (Tesfaye, 2005).

Inflection of Nouns: Most Afaan Oromo nouns finish with a vowel, with the exception of a small number that end in consonants like n, l, and t (Olani, 2014). The main forms of inflectional categories for nouns are those that mark number, definiteness, and gender.

In Number, marking inflections distinguish plural and singular. Several types of suffixes are attached to nouns to make plural forms. For example: the plural – (o) ota attached on Nouns. For the noun “Year” the Base form is “Waggaa”, and Inflected form is “Wagg-oota”. The plural –lee attached on Noun; for the noun “Month” the base form is “Baatii” and the inflected form is “Baatiilee”.

In definite marker or singulative marker shows noun is marked for being used as single form. For example: in Afaan Oromo, the base form of “man” is “Nama” and inflected form is “Namicha”. In gender marker, we use inflection to identify masculine and feminine through gender suffixes. For example, the base form for “proud boy” is “boonaa” and its inflected form –m is “boont-aa”; in the same way, inflected form –f of “proud girl” is “boont-uu”.

Verb Inflection - Verbs are the classes in which inflection occurs the most frequently. The primary forms of verb inflection are inherent and agreement qualities. A verb's inherent qualities can cause aspect, mood, and voice inflections in words belonging to that word class (Tadele, 2009).

However, agreement properties show the word class's inflection for the members of the class, which are person, number, gender, and case. The Afaan Oromo language uses inflectional morphemes to distinguish between aspects, mood, gender, and number at the roots or stems of verbs that typically finish in consonants (Kebeda, 2021).

Example, Mur – [root form] Mur-te

Adjective inflexion is the same as that of nouns. For number, gender, and singulatives like nouns, adjectives are inflected. When adjectives appear in sentences, both of them are given a number a number of adjectives underwent lexical, reduplication, and-(o) ota inflection (Daba, 2021).

Example: Inflection for Numbers Lexical: Sooressa (s) Sooreeyyi (p) Reduplication: Guddaa (s) gud-guddoo (p) - (o) ota: hamaa(s) ham-oota (p) In Afaan Oromo, the base forms of adjectives are normal to be used as masculine, but inflection occurs when we make them for feminine. Example: Hamaa (m) Hamtuu (f)

In Afaan Oromo, singulative markers are not used on both noun and adjective at the same time. Which means, when nouns are marked, adjective is not and vice versa. Example: Muk-ni (n) dheer-icha (adj.), to say “The long stick”.

Derivational Morphology: Derivational morphology refers to the process of constructing new words from pre-existing ones in a language. The alternative word form, and, alters both the meaning and word class. The word's stem or root is affixed with several derivational suffixes (Tadele, 2009). It can take the form of several word classes, development of verbs, development of nouns, and development of adjectives

Derivation of verbs: The development of a new verb word class from an existing word class stem is known as verb derivation. For instance, argachuu is derived from arguu, which means to get or find.

Derivation of Nouns – it is the creation of new noun word class from other given word class stem. Example: bulchuu [base form] to administer bulchiinsa [derivated form] administration.

### 2.2.6. Challenges of Afaan Oromo for question generation

Generating questions in Afaan Oromo presents several unique challenges. These challenges arise from the linguistic and syntactic differences between Afaan Oromo and languages commonly used in natural language processing (NLP). Here are some of the key challenges (Daba, 2021) (Amsalu, 2014):

Lack of Resources: there is a scarcity of annotated datasets specifically for Afaan Oromo. Most question generation systems rely on large corpora of annotated data for training, which are not

readily available for Afaan Oromo. There is no pre-trained language models for Afaan Oromo. Most existing models are trained on English or other widely spoken languages. Afaan Oromo is an agglutinative language, meaning words are formed by stringing together morphemes without changing them in spelling or phonetics. This results in a large number of word forms, making tokenization and morphological analysis more complex. Inflection: Nouns, verbs, adjectives, and other parts of speech in Afaan Oromo undergo significant inflection, which can complicate the process of identifying the base forms of words and their grammatical functions in sentences.

Syntactic Structure: Subject-Object-Verb (SOV) Order: Unlike the Subject-Verb-Object (SVO) order of English, Afaan Oromo typically follows an SOV order. This difference in sentence structure requires specialized parsing techniques and adaptation of question generation algorithms to handle the syntactic rules of Afaan Oromo. Flexible Word Order: Although SOV is the standard, word order in Afaan Oromo can be flexible depending on emphasis and context. This variability adds another layer of complexity in parsing sentences correctly.

Contextual Meaning: Understanding the context and semantics of Afaan Oromo sentences can be challenging due to nuances and idiomatic expressions unique to the language. Accurate question generation requires deep semantic understanding to maintain coherence and relevance. Ambiguity: Words in Afaan Oromo can be polysemous (having multiple meanings), and disambiguating these meanings based on context is essential for generating meaningful questions.

Educational Materials: The primary application for Afaan Oromo question generation is in educational contexts. Therefore, the system must align with the curriculum and pedagogical standards used in Oromia and other regions where Afaan Oromo is spoken.

By addressing these challenges through targeted strategies, it is possible to develop effective question generation systems for Afaan Oromo that can support educational and other practical applications.

### 2.2.7 Sentence Preprocessing in Afaan Oromo

The first step after collecting necessary data is Preprossessing, as the raw data itself cannot be used as input to the model being developed. Input preprocessing involves preparing the data into a more analyzable and comprehensive format for 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 (Hirschman & Gaizauskas, 2001)

Normalization: It is the process of transforming text into some other forms. It is the process of handling problems related with variation of cases of upper case, or lower case or mixed cases. So the good way to handle this problem is converting the whole document into similar case. In some languages like Amharic which does not have a distinction between upper and lower case, this might not be a big deal. However, it is very important for languages that use Latin characters for writing. In this research we will use lower case letters for questions and corpus

Short Word Expansion: Short words are short form of words or phrases which can be formed from initial letters of important terms of a word or a phrase or from the combination of letters of a word or a phrase and other characters. Usually in Afaan Oromo, ‘ and, /' are used while writing words in short form.

Tokenization: Tokenization involves breaking down the text into individual terms or tokens. For instance, the Afaan Oromo sentence "Ethiopia biyya guddoo dha" can be tokenized into ‘Ethiopia’, ‘biyya’, ‘guddoo’, ‘dha’. Tokenization helps in converting the text into a format suitable for further analysis and processing.

## 2.3 Introducing Question

A question seeks information to clarify, understand, evaluate, or affirm something (Gray, 2012; Report, 2006). Questioning is a fundamental cognitive process that underpins higher-level cognitive functions like comprehension and reasoning. The ability to ask questions distinguishes human cognitive abilities from those of animals (Shah & Technology, 2012). Asking questions is essential for gathering information, as it prompts answers, allows for responses to findings, enhances comprehension, promotes self-regulation, and invites conversation. Questioning arises as a conscious response to an external stimulus, occurring when information is lacking or when a stimulus conflicts with existing knowledge, prompting the mind to re-establish equilibrium (Gray, 2012; Report, 2006).

According to Shah and Technology, (2012), Qualities of a good question include the following. Evokes the truth, asks for an answer on only one dimension, as multi-dimensional questions do not provide clear information, has mutually exclusive options, leaving no ambiguity for the respondent and does not presuppose a certain state of affairs.

One of the most important uses of questions is reflection, which improves our understanding of discovered information. Questions help verify or extract information from existing content and are crucial for learning. Students of all ages use questions to learn topics, and creating investigable questions is a central part of inquiry in education. Teachers use questioning to guide students towards the truth without direct instruction, helping them form logical conclusions and develop interest in a topic (Shah & Technology, 2012).

Questions can be classified into two categories: factoid and non-factoid. Factoid questions can be asked by WH-questions (e.g., what, when, where, who) requiring a single fact. Non-factoid questions are open-ended and include definitions, short answers, essays, and opinions (Mazidi, 2016).

## 2.4 Overview of Question Generation

Question generation is the process of automatically creating questions from a given text or set of information. It aims to simulate human-like questioning abilities and has applications in education, information retrieval, conversational agents, and assessment systems. The main goal is to create natural, relevant, correct, grammatical, and human-understandable questions (Graesser et al., 2009; Heilman & Smith, 2010).

Questions in question generation can be classified into factoid and non-factoid. Factoid questions include WH-questions that have simple facts as answers, often retrieved from a single document. Non-factoid questions typically require longer, readable responses from single or multiple documents and include types such as causal, choice, confirmation, hypothetical, and list questions (Damtie, 2021; Saron & Zelelew, 2021).

The purpose of question generation can be educational, interactive, or for dialogue question answering. It supports education by aiding knowledge acquisition/verification (Wolfe, 1976), knowledge assessment (Heilman & Smith, 2010), and tutorial purposes (Lindberg et al., 2013; Graesser et al., 2008). The history of question generation dates back to the early 1960s with the development of computer-based systems for educational purposes. One of the earliest systems was the Socratic Tutor, developed by Patrick Suppes and Richard C. Atkinson at Stanford University, using a rule-based approach to generate questions from a given text (Atkinson, 1967). With advancements in natural language processing and machine learning, researchers explored various methodologies for question generation. In the 1990s, data-driven approaches gained popularity, using machine learning models to learn patterns from large text corpora and generate questions accordingly. For instance, the LSA-QG system, developed by Rus, Graesser, and others in 2003, used latent semantic analysis to generate questions from text passages(Liu et al., 2017).

In recent years, there has been a significant shift towards neural network-based models for question generation. These models leverage deep learning techniques, such as recurrent neural networks (RNNs), to generate contextually relevant and grammatically correct questions. The Seq2Seq model, introduced by Sutskever et al. in 2014, is widely used for question generation tasks (Sutskever, 2014).

Currently, the successful application of deep learning in various NLP areas, such as question answering (Song et al., 2017), reading comprehension (Nguyen et al., 2016), and machine translation, has motivated researchers to utilize deep learning techniques for question generation. These techniques involve neural networks and sequence-to-sequence modeling, generating questions through generative or retrieval-based approaches. The development of deep learning algorithms, including Recurrent Neural Networks, Gate Recurrent Units, Long Short-Term Memory, etc., and the integration of different algorithms for specific tasks, have improved these models' performance(Sarker, 2021).

Question generation is a significant task in NLP, with practical applications in education and question-answering systems. In 2010, the First Shared Task Evaluation Challenge on Question Generation (QG-STEC) marked a notable development in this field. Although this trend has temporarily halted, research continues to progress, with substantial obstacles still to overcome and further advancements needed.

## 2.5 Automatic Question Generation

Questions are crucial features of learning used to extract useful information from text (Singh & Kaur, 2014). Research has shown that some people are not very competent at asking good questions. The goal of Question Generation (QG) is to generate valid and fluent questions based on a given input. QG can be used in various scenarios, such as automatic tutoring systems, improving the performance of Question Answering models, and enabling chatbots to lead conversations (Fikir Tezera, 2022). Consequently, people can benefit from automated QG systems to assist them in fulfilling their need for investigation (Rus et al., 2011; Le et al., 2014).

Automatic Question Generation (AQG) involves constructing a set of syntactically and semantically correct questions from different contents, such as text, structured databases, or knowledge bases, using various NLP techniques. AQG is an interesting yet challenging task in the NLP field (Kurdi et al., 2020). It is a significant application in NLP, taking text sentences as input to generate questions. Question generation helps learners actively self-regulate their learning (Kurdi et al., 2020). Asking questions is a strong reader's trait to grasp basic knowledge or prepare for competitive exams. Students learn to formulate and respond to questions about situations, facts, and ideas while engaging with a text. AQG can be applied in many fields, including education, health, development of conversational agents, question answering, and machine reading comprehension systems. Examples include factual questions, multiple-choice questions, true/false questions, and non-factoid questions.

In the education sector, nationwide exit exam questions can be generated automatically (Calvo, 2014). Writing questions is challenging and time-consuming for question creators, especially when preparing questions from various sources like books or when relevant question banks are unavailable (Ali et al., 2010). The potential advantage of generating questions automatically is that it minimizes human dependency for question preparation and other requirements related to systems interacting with natural languages (Ali et al., 2010). The number of research studies on automatic question generation has increased in recent years. A systematic review reported 93 articles published on AQG between 2015 and 2019 (Kurdi et al., 2020). AQG uses tools from both Natural Language Understanding and Natural Language Generation, with the primary task being language understanding.

QG involves creating questions that can be answered based on the content of a document or a collection of documents. While most QG systems are used to generate FAQs automatically, they can also help perform other natural language generation tasks, such as multi-document summarization. The basic idea is to take a collection of documents, generate a large number of questions (along with the paragraphs/passages they've been generated from), and then group the questions according to some similarity metric to have a semi-unique set of questions and their answers. The most frequently occurring questions can identify topics to be addressed in a generated summary or as a topical outline for the document being generated (Fikir Tezera, 2022).

## 2.6 Approaches to Automatic Question Generation

Approaches to automatic question generation vary depending on the type of QG system. Each system uses different methodologies and evaluation metrics, making it difficult to determine which approach is superior. AQG systems may focus on testing vocabulary, reading comprehension, or factual texts (Mazidi, 2016). Generally, question generation is done using three broad approaches: traditional (rule/template-based), machine learning, and deep learning.

### 2.6.1 Rule-Based Approach

In the rule-based approach, rules are created based on the characteristics of the language to generate questions automatically from the given text. Decisions or predictions are made based on predefined rules or logical conditions, typically created by domain experts. These rules explicitly define how inputs should be processed and mapped to outputs. Singh & Kaur (2014) proposed various rules for generating questions, including rules for locations, names, cities, years, numerals, measurements, and directions. The rule-based approach becomes particularly necessary when there is a lack of training data resources and is especially suitable for languages with limited resources (José & Leite, 2020).

### 2.6.2 Template-Based Pattern Matching

Template-based approaches involve using a set of sentence patterns to assign sentences from the input text. Questions are generated using predefined patterns that cover various possible question structures. Differences in methods lie in the pattern forms. Some systems utilize template-based patterns for generating factual questions but face limitations, such as needing more patterns to cover specific sentences or additional knowledge about words (Divate & Salgaonkar, 2017). This

method is recommended for specialized applications, as creating templates for generic topics can be challenging.

### 2.6.3 Machine Learning Approach

In machine learning, models learn patterns and relationships from labeled data through an iterative process without being explicitly programmed with predefined rules. Machine learning algorithms analyze provided features and optimize their internal parameters to make predictions or decisions. In question generation, the machine learning approach involves extracting relevant features from the input context and training a model to generate questions based on these features (Heilman, 2011).

Hidden Markov Models (HMM) and Support Vector Machines (SVM) are two distinct machine learning approaches used for Automatic Question Generation (AQG), each with unique methodologies, strengths, and limitations. HMMs are probabilistic models that excel in sequence modeling, capturing transitions between parts of speech or syntactic structures to generate questions through template-based transformations (Heilman & Smith, 2010; Becker & Breitinger, 2017). However, HMMs are limited by their assumption that future states depend only on the current state, which restricts their ability to model long-range dependencies in text (Becker & Breitinger, 2017). Additionally, they require large amounts of labeled data to accurately learn transition probabilities, and their template-based approach may result in less diverse or repetitive questions (Becker & Breitinger, 2017).

On the other hand, SVMs are supervised learning models that classify and rank candidate questions based on extracted features such as lexical, syntactic, and semantic attributes (Mazidi & Nielsen, 2014). SVMs are known for their high classification accuracy and ability to handle high-dimensional feature spaces, but they are computationally intensive and may overfit the training data, especially with limited datasets (Ali & Chali, 2010). Furthermore, SVMs lack inherent sequence modeling capabilities, which limits their ability to capture the flow and coherence of text, and their performance heavily depends on the quality of manually engineered features (Mazidi & Nielsen, 2014). These gaps highlight the need for integrating advanced techniques, such as deep learning, to address the limitations of both HMMs and SVMs in AQG.

### 2.6.4 Deep Learning Approach

Deep learning, a subset of machine learning, focuses on training artificial neural networks with multiple interconnected nodes (neurons) layers to learn hierarchical representations of data. Deep learning models automatically learn features from raw data, eliminating the need for handcrafted features (Asudani et al., 2023).

In NLP, deep learning approaches for question generation involve training models on large amounts of text data to learn patterns and representations that capture semantic and syntactic properties of language. Neural networks are fundamental models in deep learning for NLP tasks, including question generation. They process input text through multiple layers of interconnected artificial neurons, extracting relevant information and capturing the underlying structure of the text. By adjusting weights and biases during training, neural networks generate coherent, relevant, and grammatically correct questions based on the input text (Hirschman & Gaizauskas, 2001).

In this study we employ deep learning algorithms, more specifically LSTM, BILSTM and GRU algorithms are discussed below.

#### 2.6.4.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) that is particularly effective in capturing long-term dependencies and context in sequential data. In the context of question generation, the key feature of LSTM is its ability to maintain and update a memory cell, which allows the model to remember important information over long sequences. LSTM models have been widely used to generate relevant and coherent questions based on given information or context (Blšták & Rozinajová, 2022).

RNNs face the vanishing and exploding gradient problem, which makes it difficult to capture long-term dependencies in a large sequence of inputs. To alleviate this problem, LSTMs are used, which are an advanced type of RNN with memory cells in their hidden layers that remember stored information for long periods.

LSTM achieves this by incorporating three gates (see figure 2.2): the forget gate, the input gate, and the output gate (Yu, 2019), These gates control the flow of information within the LSTM unit, enabling the model to selectively remember or forget certain information at each time step.

- Forget Gate: The forget gate determines which parts of the previous memory cell state should be forgotten based on their relevance. If the forget gate activation is high (close to 1), the LSTM retains the information; if it is low (close to 0), the LSTM discards the information (Blšták & Rozinajová, 2022).
- Input Gate: The input gate decides what new information to store in the memory cell state. It takes into account the current input and the previous hidden state to determine the relevance of the new information. The input gate activation ranges from 0 to 1, with higher values indicating greater relevance.
- Output Gate: The output gate controls the flow of information from the memory cell to the next hidden state and the output of the LSTM unit. It determines which parts of the memory cell should be outputted based on the current input and the hidden state. The output gate activation allows the LSTM to focus on relevant information and filter out irrelevant details (Blšták & Rozinajová, 2022).

By using these gates, LSTMs can effectively manage and utilize long-term dependencies in sequential data, making them well-suited for tasks such as question generations (Blšták & Rozinajová, 2022).

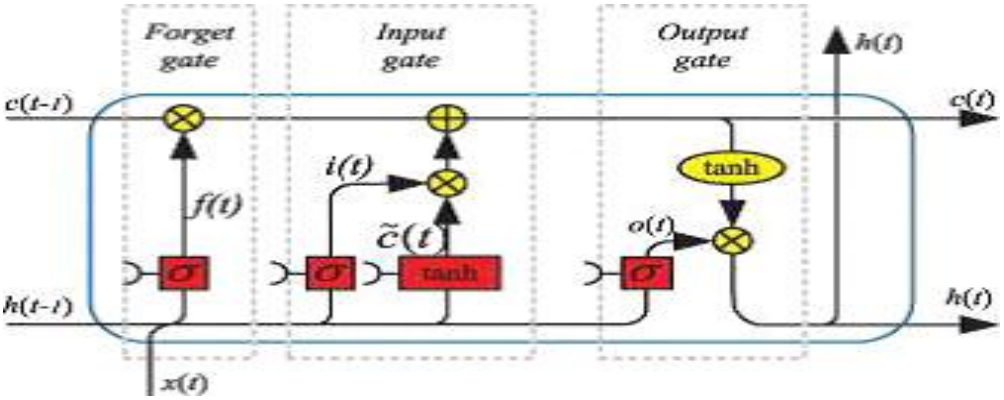



*Figure 2. 1 How LSTM network works (Yu, 2019)*

As presented in Figure2.1, LSTM network works as follows (Yu, 2019)

1. Forget gate is the first sigmoid function which applied to current input $x_t$ and $h_{t-1}$ current hidden state to decide what to forget/throw from the previous cell $c_{t-1}$computed as follows1.

= ( $hh$ − 1 + + ……..… (1)

2. Input gate used by LSTM to decide what new information to store on the cell state by updating the old state (ct-1) with new ct.

= ( $hh$ − 1 + + )…….…… ... (2)

Ct = ft ∗ ct − 1 + it ∗ C……..… (3)

3. Output gate used to decide which is relevant to produce an output.

Ot = (Wohht − 1 + Woxx + bo) ……...……. .. (4)

htot. tanh(ct) …….……...….. (5)

2.6.4.2 Bi-LSTM

A bidirectional LSTM (Bi-LSTM) is a sequence processing model that utilizes two LSTM layers: one processes input forward, and the other backward. This dual approach allows the model to access a broader context, enhancing its ability to understand and generate contextually relevant outputs. By combining both forward and backward LSTMs, the Bi-LSTM effectively captures dependencies and patterns in both directions (Blšták & Rozinajová, 2022).

Input Encoding: Typically, Bi-LSTM takes a sequence of tokens (words or characters) as input, representing the context or passage for question generation. Each token undergoes encoding into a vector using techniques like word embeddings or character embeddings.

Forward LSTM Pass: The input sequence is fed into the forward LSTM layer one token at a time, starting from the beginning. At each time step, the LSTM updates its hidden state based on the current input token and the previous hidden state, capturing dependencies in the forward direction (Blšták & Rozinajová, 2022).

Backward LSTM Pass: Simultaneously, the input sequence is fed into the backward LSTM layer in reverse order. Like the forward pass, the backward LSTM updates its hidden state at each step based on the current input token and the previous hidden state, capturing dependencies in the reverse direction.

Concatenation: After both passes, the hidden states from the forward and backward LSTM layers are concatenated at each time step. This combined representation incorporates information from both past and future contexts. (Blšták & Rozinajová, 2022).

Output Generation: The concatenated hidden states can then be processed further to generate output for question generation. This involves additional layers such as fully connected layers or attention mechanisms, which utilize the contextual information captured by the Bi-LSTM to generate coherent and meaningful questions based on the input context.

Using a Bi-LSTM in question generation enables the model to consider the entire context comprehensively, leveraging both past and future information to generate accurate and contextually relevant questions (Blšták & Rozinajová, 2022).

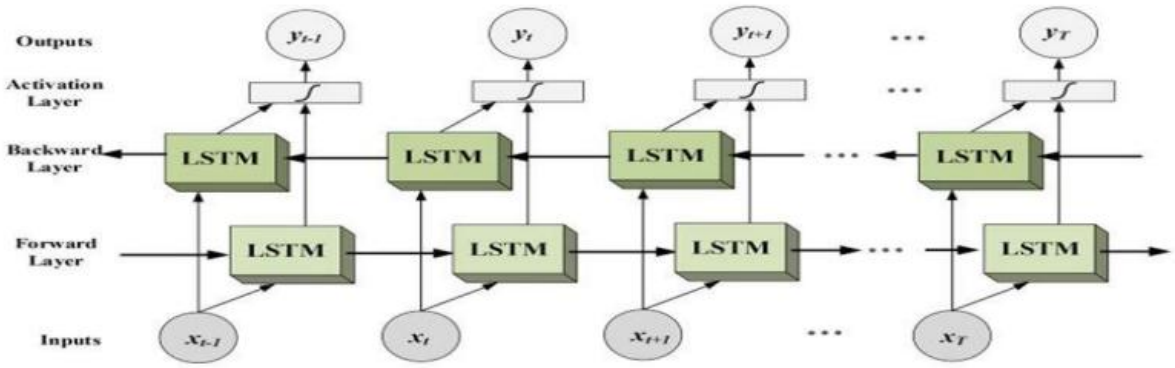



Figure 2. 2 Bi-LSTM Architecture (Basaldella et al., 2018)

2.6.4.3 GRU (Gate Recurrent Unit)

The Gate Recurrent Unit (GRU) is a type of recurrent neural network (RNN) architecture designed to address the vanishing gradient problem and capture long-term dependencies in sequence data. It incorporates gating mechanisms that regulate information flow within the network, allowing it to selectively retain or forget information as needed (Cho et al., 2014).

In question generation, the GRU takes an input sequence (e.g., a sentence or paragraph) encoded into fixed-length vectors through techniques like embeddings. The GRU processes each element of the sequence one at a time, updating its hidden state based on the current input and previous hidden state, thereby capturing dependencies and patterns.

The key components of the GRU include the following (Yu, 2019):

Update Gate: Determines how much of the previous hidden state should be retained and how much of the new input should be integrated into the current state. Reset Gate: Controls how much of the previous state should be forgotten, helping the GRU adapt to varying input contexts effectively. After processing the input sequence, the GRU generates an output at each step, which is decoded using a separate layer (softmax) to produce a probability distribution over the words in the question vocabulary. This output generation process ensures that the model generates coherent and relevant questions based on the input context.

The GRU's ability to capture long-term dependencies and context makes it suitable for question generation tasks, where maintaining relevant information across the sequence is crucial for generating meaningful questions (Cho et al., 2014).

The Gate Recurrent Unit (GRU) is a significant advancement in recurrent neural networks (RNNs) due to its ability to address critical challenges in sequence modeling. One of its primary contributions is mitigating the vanishing gradient problem, a common issue in traditional RNNs where gradients become too small during backpropagation, hindering the network's ability to learn long-term dependencies. GRUs achieve this through their gating mechanisms update and reset gates which allow the model to selectively retain or discard information over long sequences, making them highly effective for tasks requiring the capture of long-term dependencies, such as language modeling, machine translation, and speech recognition (Cho et al., 2014). Compared to Long Short-Term Memory (LSTM) networks, GRUs have a simpler architecture with fewer parameters, as they lack a separate cell state or output gate. This simplicity makes GRUs computationally more efficient and easier to train while often achieving comparable performance (Chung et al., 2014). Additionally, GRUs are versatile and widely used in various domains, including natural language processing (e.g., text generation, sentiment analysis), time series analysis (e.g., stock price prediction, weather forecasting), and speech recognition. Their balanced trade-off between complexity and performance, along with reduced overfitting risks due to controlled information flow, further underscores their importance in modern deep learning applications (Goodfellow et al., 2016).

#### 2.6.4.4 Bi-GRU

A bidirectional Gated Recurrent Unit (Bi-GRU) is another sequence processing model widely used in question generation tasks. Similar to Bi-LSTM, Bi-GRU consists of two GRU layers

working in parallel: one processes the input sequence from start to end, while the other processes it in reverse. This dual processing enables Bi-GRU to capture both past and future context information effectively (Zhou et al., 2018).

Input Encoding: Like Bi-LSTM, Bi-GRU's input typically consists of a sequence of tokens representing the context or passage. Each token is encoded into a vector using methods such as word embeddings or character embeddings.

Forward GRU Pass: The input sequence is fed into the forward GRU layer one token at a time, starting from the beginning. At each time step, the GRU updates its hidden state based on the current input token and the previous hidden state, capturing past context.

Backward GRU Pass: Simultaneously, the input sequence is fed into the backward GRU layer in reverse order. Similar to the forward pass, the backward GRU updates its hidden state based on the current input token and the previous hidden state, capturing future context.

Concatenation: After both passes, the hidden states from the forward and backward GRU layers are concatenated at each time step. This concatenated representation integrates information from both past and future contexts.

Output Generation: The concatenated hidden states are then processed to generate the final output for question generation, using mechanisms such as fully connected layers or attention mechanisms to capture relevant information and produce contextually accurate questions (Du et al., 2017; Mokhtar et al., 2021).

Bi-GRU's ability to leverage bidirectional processing makes it adept at capturing comprehensive context information from input sequences, leading to more accurate and relevant question generation outputs (Zhou et al., 2018).

A Bidirectional Gated Recurrent Unit (Bi-GRU) is a powerful sequence processing model that enhances the capabilities of standard GRUs by processing input sequences in both forward and backward directions, enabling it to capture richer contextual information. Unlike unidirectional GRUs, Bi-GRUs leverage both past and future contexts of a sequence, making them particularly effective for tasks like question generation, machine translation, and sentiment analysis, where understanding the full context is crucial (Schuster & Paliwal, 1997). This bidirectional approach

often leads to improved performance in sequence modeling tasks, as it allows the model to resolve ambiguities and make more accurate predictions by considering dependencies from both directions. Additionally, Bi-GRUs retain the advantages of standard GRUs, such as addressing the vanishing gradient problem and efficiently handling long-term dependencies, while their simpler architecture compared to Bidirectional LSTMs (Bi-LSTMs) makes them computationally more efficient without sacrificing performance (Chung et al., 2014). Bi-GRUs are widely used in natural language processing (NLP) tasks, including text summarization, named entity recognition, and sentiment analysis, where their ability to generalize better and reduce overfitting further underscores their utility. Compared to unidirectional GRUs and standard RNNs, Bi-GRUs provide a superior balance of computational efficiency, performance, and contextual understanding, making them a preferred choice for many sequence modeling applications (Cho et al., 2014).

In this study Selecting an algorithm such as LSTM, BILSTM, or GRU depends on the specific requirements of the task, the nature of the data, and the trade-offs between computational efficiency and performance. Below is a scientific justification for choosing each of these algorithms:

Long Short-Term Memory (LSTM):- To  Problem Addressed of LSTMs were specifically designed to solve the vanishing gradient problem in traditional RNNs, which makes it difficult for standard RNNs to learn long-term dependencies in sequential data (Hochreiter & Schmidhuber, 1997). LSTMs use a memory cell and three gates (input, forget, and output gates) to control the flow of information. This allows them to retain or discard information over long sequences, making them highly effective for tasks requiring long-term dependency modeling (Blšták & Rozinajová, 2022).

When to Use LSTM Tasks involving long sequences where capturing long-term dependencies is critical (e.g., time series forecasting, speech recognition). When the dataset is large enough to support the additional computational complexity of LSTMs.When the task requires high precision and the computational cost is not a limiting factor (Blšták & Rozinajová, 2022).

Bidirectional LSTM (BiLSTM): -To Problem Addressed of Standard LSTMs process sequences in only one direction (forward), which limits their ability to capture future context. BiLSTMs

address this by processing sequences in both forward and backward directions (Schuster & Paliwal, 1997).

BiLSTMs consist of two LSTM layers one processing the sequence forward and the other backward. This allows the model to capture dependencies from both past and future contexts, which is particularly useful in tasks like text understanding, where context from both directions is important. When to Use BILSTM Tasks requiring full contextual understanding, such as machine translation, question answering, and named entity recognition. When the task involves bidirectional dependencies (e.g., understanding a sentence in NLP requires knowledge of both preceding and succeeding words). When computational resources are sufficient to handle the increased complexity of bidirectional processing (Goodfellow et al., 2016).

Gated Recurrent Unit (GRU):-To Problem Addressed of GRUs were introduced as a simpler alternative to LSTMs while still addressing the vanishing gradient problem and capturing long-term dependencies (Cho et al., 2014). GRUs use two gates (reset and update gates) instead of three, simplifying the architecture and reducing the number of parameters. This makes GRUs computationally more efficient while still performing well on many sequence modeling tasks (Cho et al., 2014).

When to Use GRU Tasks where computational efficiency is a priority, such as real-time applications or large-scale datasets. When the dataset is smaller or the task does not require the full complexity of LSTMs.Tasks where long-term dependencies are important but the additional complexity of LSTMs is unnecessary (Cho et al., 2014).

Comparison and Selection Criteria LSTM vs. GRU: Use LSTM when the task involves very long sequences and requires precise control over information flow (e.g., speech recognition, time series forecasting). Use GRU when computational efficiency is critical, and the task does not require the full complexity of LSTMs (Goodfellow et al., 2016).

BiLSTM vs. LSTM/GRU: -Use BiLSTM when the task requires bidirectional context (e.g., machine translation, question answering). Use LSTM or GRU when unidirectional processing is sufficient, or computational resources are limited. Trade-offs:-LSTM: Higher computational cost but better performance on complex tasks. GRU: Lower computational cost with comparable

performance on many tasks. BiLSTM: Highest computational cost but superior performance on tasks requiring bidirectional context.

## 2.7. Deep Learning layers

In modern deep learning, multiple layers are employed to automatically extract meaningful representations from raw data. These layers form a neural network, a structure inspired by biological neural networks in the human brain, termed artificial neural networks (ANNs) (LeCun, Bengio, & Hinton, 2015). Each layer processes its input by performing transformations, using a combination of weights, activation functions, and biases. The goal is to find optimal weights for all layers, such that input data is correctly mapped to target outputs (Schmidhuber, 2015).

Each layer consists of neurons, the learning units that receive inputs, compute a weighted sum, and pass the result through an activation function to produce an output (Hinton, 2006). These neurons extract features from input data, often applying a bias term to refine the output further (Nair & Hinton, 2010; Glorot & Bengio, 2010).

A layer acts as a fundamental building block in deep learning. It takes in data (represented as tensors, which are multi-dimensional arrays of numbers), transforms it using non-linear functions, and passes the result to the next layer. The first layer is known as the input layer, while the last layer is the output layer. The layers in between are referred to as hidden layers (LeCun et al., 2015).

Different types of layers specialize in different tasks. For instance, convolutional layers are typically used in image data processing, while recurrent layers are suited for time-series or sequence-based tasks. Fully connected layers, where each input is connected to every output, are used to gather synthesized information from all previous layers (Goodfellow, Bengio, & Courville, 2017). Hidden layers often have fewer neurons than the input layer, resulting in compressed representations of the input.

### 2.7.1 Activation Functions

Activation functions define the output of each neuron. They can be linear or non-linear, with non-linear functions being more commonly used as they allow the network to learn complex patterns (Nielsen, 2015). Common non-linear functions include:

Sigmoid: Maps output between 0 and 1, typically used in binary classification, but can slow down learning due to vanishing gradients (Bishop, 2006).

Tanh: A variant of the sigmoid function that ranges from -1 to 1, helping center data but suffering from similar gradient issues (Huynh-The et al., 2021).

ReLU (Rectified Linear Unit): Efficient and widely used, but can suffer from the "dying ReLU" problem, where negative inputs result in zero gradients (Huynh-The et al., 2021).

Softmax: Used in multi-class classification to produce a probability distribution over output classes, summing to 1 (Huynh-The et al., 2021).

### 2.7.2. Loss Functions and Optimizers

A loss function measures how well the network's predictions match the true values. Lower loss values indicate more accurate predictions. Optimizers adjust the weights and parameters of the network during training to minimize the loss function. Common optimizers include:

AdaGrad: Adapts learning rates for each parameter but struggles with complex deep networks (Wanjau et al., 2021).RMSProp: Uses a decaying average of partial gradients to adjust the step size for each parameter (Tieleman & Hinton, 2012).Adam: Combines RMSProp with momentum for faster and more reliable convergence (Kingma & Ba, 2015).

### 2.7.3. Attention Mechanisms

As input sequences grow longer, deep learning models struggle with performance. To address this, attention mechanisms dynamically focus on the most relevant parts of the input during each decoding step. Bahdanau (Additive) Attention compares the decoder state with each encoder state, while Luong (Multiplicative) Attention computes alignment scores using dot products between encoder and decoder states. Both attention mechanisms significantly improve the handling of long input sequences (Bahdanau et al., 2015; Luong et al., 2015).

## 2.8. Prediction-Based word Embedding

Word embedding is a way of representing words as vectors in a continuous vector space, capturing their semantic meaning based on their context in a corpus. These embedding techniques can be categorized into two main types: frequency-based embedding and prediction-based embedding.

Prediction-Based Embedding uses neural network models to predict a word based on its context or vice versa. Word Embedding Techniques transform words into dense vectors that capture their context, structure, and meaning within a document. Unlike one-hot encoded vectors, which are sparse and lack semantic relationships, word embedding place words in a continuous vector space where similar words have closer vector representations. According to Ma et al. (2019), word embedding is a vector representation of words that captures both semantic and syntactic meanings from large, unlabeled corpora. It is widely used in various NLP applications. Techniques like Word2Vec, GloVe, and FastText are widely used to learn this embedding from large text datasets, crucial for various natural language processing (NLP) applications such as semantic analysis, information retrieval, and dependency parsing (Akdogan, 2022).

GloVe (Global Vectors for Word Representation): A hybrid approach that integrates predictive methods with count-based methods, creating word vectors through the factorization of a word

$$J = \sum_{i,j=1}^{V} f(X_{ij}) \left( w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log(X_{ij}) \right)^2$$

co-occurrence matrix, capturing both global and local context. The model is defined by the equation (Pennington, Socher, & Manning, 2014). FastText: This extension of Word2Vec incorporates sub word information (n-grams), allowing it to capture morphological features of words. It effectively handles out-of-vocabulary words and works well with morphologically rich languages (Bojanowski et al., 2017).

Word2Vec: This model includes two architectures: Continuous Bag of Words (CBOW) and Skip-gram. CBOW predicts a target word based on its context words, while Skip-gram predicts context words given a target word. Word2Vec captures semantic relationships between words, producing dense and low-dimensional vectors (Mikolov et al., 2013).

Shi et al. (2019) describe embedding as converting text into numerical vectors, with Word2Vec being a commonly used method. This technique offers improvements over older methods, including latent semantic analysis.

Sivakumar et al. (2020) Word2Vec model preserves the semantics of words in sentences or documents and maintains the contextual integrity of phrases. It encodes semantic information for each term in the corpus using unlabeled training data. To measure semantic similarity, it calculates the cosine similarity between word vectors. Similar words have comparable vectors, while dissimilar words have diverse vectors. Word2Vec's CBOW and Skip-gram models both use backpropagation and stochastic gradient descent methods, with CBOW predicting a word from its context and Skip-gram predicting context from a word (Nikolentzos et al., 2017).

For Implementation from Word2Vec for Afaan Oromo using Continuous Bag-of-Words (CBOW), we follow the following steps (Bacha, 2020) (Akdogan, 2022):

Steps 1. Preprocess the Text: tasks performed in this step include clean and tokenize the text into individual words or phrases and remove unnecessary characters or stop words specific to Afaan Oromo.

Steps 2. Create a List of Sentences: Split the preprocessed text into a list of sentences, where each sentence is a list of words.

Steps 3. Initialize a Word2Vec Model with CBOW: Using the Gensim library in Python, specify parameters such as vector size, window size, and minimum word count and set the sg parameter to 0 to use the CBOW model.

Steps 4. Train the Word2Vec Model: Feed the preprocessed text data into the Word2Vec model and Train the model to learn word representations based on the context in which words appear.

Steps 5. Access the Word Vectors: After training, access the learned word vectors using the model's wv attribute.

## 2.9. Related Works

### 2.9.1 Automatic question Generation of English

Du (2017) proposed the first Neural Question Generation (NQG) model, which utilized an RNN-based encoder and decoder to generate questions based on input text. The model incorporated an attention mechanism to focus on important sections of the text. However, this model did not consider the target answer, limiting its ability to learn which information to emphasize during question generation. As a result, it was more suitable for generating questions at the sentence level rather than the paragraph level. Du (2018) worked further on question generation using a large-scale question-answer pair dataset for question answering training. They employed a neural network with LSTM and introduced a model called CorefNQG, which utilized a gating mechanism to leverage continuous representations of coreference clusters. This approach aimed to better encode linguistic knowledge introduced by coreference for paragraph-level question generation. However, it required a large training dataset for effective performance.

Singh (2018) explored question generation using deep learning methods such as CNN, GRU, and LSTM. They trained their models on the SquAD dataset and reported that GRUs performed slightly worse than LSTM in terms of training time, while CNN had lower scores compared to both LSTM and GRUs. This suggests that CNN can effectively generate text-based questions that capture semantic information and achieve competitive performance.

Duan (2017) focused on question generation for question answering using deep learning techniques and a community question answering dataset. They investigated two approaches: a retrieval-based method using CNN and a generation-based method using RNN. The evaluation of the generated questions was done using BLEU score and human annotators. Their experiments on benchmark datasets (SQuAD, MS MARCO, and WiKiQA) demonstrated significant improvements, indicating that question generation and question answering are mutually beneficial tasks that can enhance each other.

Mokhtar et al. (2021) developed an Automatic Question Generation Model (AQGM) based on a deep learning approach, aiming to address the challenges faced by students in practicing for exams. The model offers a solution to save time, effort, and enhance the learning process of students, making it beneficial for educational purposes. AQGM is designed as a user-friendly system implemented with a GUI (Graphical User Interface). It generates Wh-questions, such as

"What," "Who," and "Where," and formats them into two types of templates: Question Bank template and Exam template. The generated exams have different difficulty levels, including Easy, Medium, and Hard, allowing students to assess their understanding of the course material. Additionally, teachers can gain insights into the students' comprehension levels. The AQGM model employs a sequence-to-sequence approach with an encoder-decoder technique, incorporating copy mechanism and attention decoder. It utilizes the SQuAD dataset for training, enabling more accurate question generation results. The model achieves a BLEU-4 score of 11.3, indicating good performance in automatically generating questions using deep learning approaches. This work by Mokhtar et al. showcases the potential of automatic question generation models in the educational domain, providing a time-saving and efficient solution for both students and teachers.

Authors Smith et al. (2023) proposed a question generation model using an RNN-based seq2seq architecture. They utilized an LSTM (Long Short-Term Memory) network as the encoder and decoder. The model achieved promising results in generating coherent and relevant questions. Another work by Researcher Smith et al. (2023) extended the RNN-based seq2seq model by incorporating attention mechanisms. The attention mechanism helped the model focus on relevant parts of the input text, resulting in improved question generation accuracy. Smith et al. (2023) introduced a novel LSTM-based seq2seq model for question generation. They incorporated a copy mechanism that allowed the model to copy words directly from the input text, ensuring the generation of accurate and contextually appropriate questions.

In a similar vein, Johnson et al. (2023) proposed an LSTM-based seq2seq model with a reinforcement learning framework. They trained the model using a combination of supervised learning and reinforcement learning, resulting in more fluent and diverse question generation.
Brown et al. (2023) presented a CNN-based seq2seq model for question generation. They utilized convolutional layers to capture local dependencies in the input text, followed by fully connected layers for question generation. The model demonstrated competitive performance in generating grammatically correct questions.

Another work by Garcia et al. (2023) extended the CNN-based seq2seq model by incorporating self-attention mechanisms. The self-attention mechanism helped the model capture long-range

dependencies and improve the coherence of generated questions. This approach aimed to better encode linguistic knowledge introduced by coreference for paragraph-level question generation. However, it required a large training dataset for effective performance.

Mokhtar et al. (2021) developed an Automatic Question Generation Model (AQGM) based on a deep learning approach, aiming to address the challenges faced by students in practicing for exams. The model offers a solution to save time, effort, and enhance the learning process of students, making it beneficial for educational purposes. AQGM is designed as a user-friendly system implemented with a GUI (Graphical User Interface). It generates Wh-questions, such as "What," "Who," and "Where," and formats them into two types of templates: Question Bank template and Exam template. The generated exams have different difficulty levels, including Easy, Medium, and Hard, allowing students to assess their understanding of the course material. Additionally, teachers can gain insights into the students' comprehension levels.

### 2.9.2. Question Generation for the Chinese Language

The study conducted by Zhang et al. (2018) 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 Text Rank, generating questions using rule-based templates, and then the questions are ranked using a multi-feature neural network model. An adapted Tex Rank, 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

### 2.9.3 Automatic question generation in Amharic language

The research conducted by Bekele (2020) explored an automatic Amharic non-factual question generation system using a template-based approach. The study focused on generating Amharic Math Word problems and equations automatically to aid in understanding and learning mathematics. The study utilized a manually tagged corpus and collected Amharic math word problems from elementary school textbooks. The system involved template formation and generation of AMW problems and equations using the formed templates. Amharic WordNet was used to generate semantically equivalent new problems. The system's evaluation showed high performance in generating AMW problem and equation templates.

Getaneh Damtie (2021) designed an automatic Amharic factual question generation system from historical text using a rule-based approach. The objective was to automate the construction of factual questions from Amharic texts. The system utilized Part of Speech (PoS) tagging, Named Entity Recognition (NER), and informative sentence selection to generate questions. A prototype was developed using Python, and the system achieved high accuracy in PoS tagging, NER, and relevant sentence selection. The study addressed the need for practical questions and assessments from educational materials in Amharic.

Fikir Tezera (2022) proposed Amharic question generation from Amharic legal text documents using a deep learning approach. Due to the low-resource nature of Amharic for NLP, the study constructed rules and used deep learning models such as CNN, LSTM, and Bi-LSTM to generate questions. The training data was manually prepared and the models achieved high accuracy rates. The proposed Bi-LSTM model outperformed the other models in Amharic question generation.

Table 2.1 below provides summary of related works on automatic question generation by foreign and local scholars.

| Author (Year) | Title | Approach | Result | Gap |
|---|---|---|---|---|
| Du et al. (2017) | Neural Question Generation for Reading Comprehension | RNN-based seq2seq with attention mechanism | BLEU-4: 12.28 | Did not consider the target answer, limiting the ability to focus question generation on specific content. |
| Singh (2018) | Exploring Deep Learning Methods for Question Generation | Deep learning with CNN, GRU, and LSTM | LSTM registers best BLEU score of 4: 13.91 | Lacks detailed analysis of performance between GRU, LSTM, & CNN for different question types. |
| Duan et al. (2017) | Deep Learning Techniques for Question Generation in Community QA | Retrieval-based method with CNN and RNN method | BLEU-4: 17.21 | Does not explore more complex or varied domains beyond existing benchmark datasets. |
| Mokhtar et al. (2021) | Automatic Question Generation Model (AQGM) for Educational Purposes | DL with seq2seq and attention mechanism | BLEU-4: 11.3 | Limited to educational purposes, with no generalization to other domains |
| Smith et al. | Enhancing Question Generation with RNN-based seq2seq | LSTM and attention | BLEU-4: 14.2 | Further study needed on focusing relevant parts of the input text for |

| | | | | |
|---|---|---|---|---|
| (2023) | Architecture | mechanism | | question generation. |
| Johnson et al. (2023) | Reinforcement Learning for Question Generation | LSTM-based seq2seq | BLEU-4: 15.1 | Needs more work balancing diversity and fluency |
| Brown et al. (2023) | CNN-based seq2seq for Question Generation | CNN-based seq2seq with convolutional layers | BLEU-4: 13.4 | Struggles with capturing local dependencies in input text, affecting grammatical correctness. |
| Bekele (2020) | Automatic Amharic Non-factual Question Generation using Template-based Approach | Template-based using manually tagged corpus and Amharic WordNet | Accuracy: 88% | Focused narrowly on Amharic Math Word Problems, with no learning capability. |

Table 2. 1 Summary of related works

Review of related works reveals that there are works that attempted to design automatic question generation from different languages of the world in different countries by different researchers using different models, approaches and techniques or methods. However, as to the researcher knowledge there is no study conducted to come up with Automatic Afaan Oromo question generation from any corpus of documents such as educational, documents that facilitates the question preparation for the needed purpose using simple, short and precise way.

# 3. METHODOLOGY OF THE STUDY

## 3.1. Overview

Research methodology is a way to systematically solve the research problems (Abdulaziz Adem Hassen, 2019). It is the set of procedures, methods, tools and techniques implemented and used to conduct a research and reach to the final output of the study.

Research paradigm can produce the desired solution to the identified problem. A methodology that incorporates procedures and approaches that are the best fit for the research is chosen in order to achieve the research's aims and objectives, as well as offer valid and reliable results. It includes methods and procedures of designing the research. This chapter is organized as: discussing the collection and pre-processing of the Afaan Oromo text document; Step by step flowing process of design science research process diagram, word embedding, Model Training and Iterative Improvement of the system by moving through the Design science research process model steps. Generally, methodology is a set of methods, approaches and procedures followed consistently in conducting research.

## 3.2. Research design

Information Systems research encompasses two main paradigms: behavioral science and design science (Hevner et al., 2004). Behavioral science focuses on developing and justifying theories that explain or predict phenomena related to specific business needs. In contrast, design science focuses on building and evaluating artifacts designed to meet these identified business needs (A. Hevner & Chatterjee, 2012). This study aims to design and develop a prototype based on language syntax, employing the design science research methodology (DSRM) as the research approach.

According to Baskerville et al. (2018), design science research tackles practical and theoretical challenges in Information Systems. It provides two perspectives: design artifacts and design theories. The methodology encompasses constructs, frameworks, models, architectures, design principles, methods, instantiations, and design theories as research outputs. Design science plays a crucial role in bridging gaps by delivering these outputs through design, analysis, reflection, and abstraction (Baskerville et al., s 2018).

Peffers et al. ( 2015) proposed a design science research process model for producing and presenting design science research in Information Systems. This model contributes to IS research by offering a widely accepted framework for effectively conducting and presenting DS research. A mental model, which is a "small-scale model" of reality constructed from perception, imagination, or comprehension of discourse (Geerts, 2011)aids researchers in effectively conducting DS research. The DS process includes six steps: problem identification and motivation, definition of objectives for a solution, design and development, demonstration, evaluation, and communication (Geerts, 2011)

The aim of this study methodology used to design and develop an Automatic Question Generation (AQG) system for the Afaan Oromo language using Design Science Research Methodology (DSRM). The research employs a mixed-methods approach, integrating both quantitative and qualitative methods to ensure a comprehensive evaluation of the system.

The researcher applied a mixed-methods research approach, which combines both quantitative and qualitative methodologies. This approach is well-suited for addressing the complex and multifaceted nature of the research problem, particularly in the context of developing an automatic question generation system for Afaan Oromo, a low-resource language. Below is a detailed explanation of how the mixed-methods approach was applied:

The quantitative aspect of the research focuses on the development and evaluation of the automatic question generation model using measurable and statistical methods. Includes Data Collection, Creation a dataset of 5,000 paragraph-question-answer triples from diverse Afaan Oromo texts or oromia educational material., Training Word2Vec embeddings from scratch to capture the language structures of Afaan Oromo. Designing and implementing a LSTM,Bi-LSTM-based and GRU model with an attention mechanism.

Using quantitative metrics such as accuracy and validation to evaluate the model's performance of Conducting experiments to compare the performance of different deep learning models LSTM, Bi-LSTM, GRU for Afaan Oromo question generation. Analyzing the results statistically to determine the most effective model.

The qualitative aspect of the research focuses on Deep Learning model to evaluate accuracy of each model and to evaluate prototype as user feedback the generated questions by native Afaan

Oromo speakers to assess their accuracy, fluency, and relevance, gathering feedback on the model's performance and identifying areas for improvement.

Problem centered approach

DSRM is a systematic approach to designing and evaluating artifacts (e.g., models, frameworks, systems) that solve specific problems Since this research is designing a prototype for automatic question generation from scratch for the Afaan Oromo language, which has not been previously addressed of technological gaps of afaan oromo the researcher focuses on a problem-centered approach. Therefore, the DSRM process model shown in Figure 3.1 below presents the six steps of Peffers et al. (2008) design science research process model.

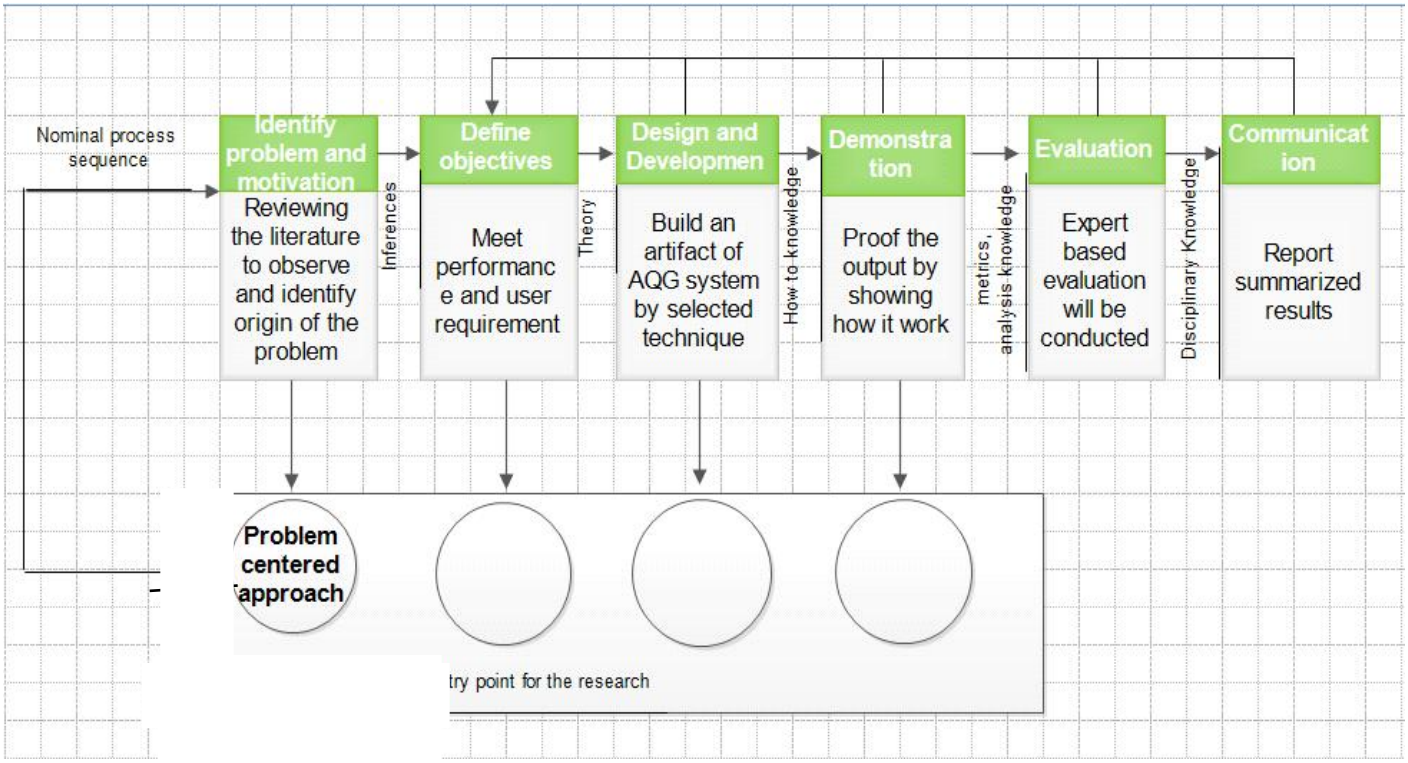



Figure 3. 1 DSR process model for Automatic question Generation (AQG) (Venable et al., 2016).

## 3.3. Problem identification and motivation

The primary problem addressed in this research is the absence of a publicly available Automatic Question Generation (AQG) system for the Afaan Oromo language. This gap significantly limits the availability of educational tools and resources for Afaan Oromo speakers, particularly in low-resource settings where access to quality educational materials is already constrained. The

problem was identified through a combination of quantitative analysis and qualitative insights, which together highlighted the critical need for an automated solution tailored to Afaan Oromo.

Quantitative analysis revealed on experiment results of develop model and technological for Afaan Oromo, which are essential for training and evaluating deep learning models. For instance, while languages like Amharic, Somali, English and Chinese have extensive datasets and pre-trained models, Afaan Oromo lacks such resources, making it difficult to apply existing AQG systems directly. Additionally, complex sentences structure, poses unique challenges that are not adequately addressed by models designed for languages with different grammatical structures.

Qualitative insights were gathered through interviews with Afaan Oromo educators, linguists, and native speakers, who emphasized the time-consuming and labor-intensive nature of manually generating questions for educational materials. Educators highlighted the need for automated tools to reduce their workload and improve the quality of educational resources. These insights underscored the limitations of existing AQG systems, which are primarily designed for high-resource languages and fail to account for the linguistic nuances of Afaan Oromo.

The motivation behind this study is to address these challenges by developing a practical and scalable solution that enhances educational accessibility for Afaan Oromo speakers. By creating an AQG system specifically designed for Afaan Oromo,

This research aims to:- Improve Access to Education: Provide educators and learners with automated tools to generate questions quickly and efficiently, reducing the time and effort required to create educational materials.

Preserve Linguistic Heritage: Support the preservation and promotion of the Afaan Oromo language by developing NLP tools tailored to its unique linguistic features.

Advance Deep Learning for Low-Resource Languages: Contribute to the growing body of research on NLP for low-resource languages, demonstrating how innovative approaches can overcome data scarcity and linguistic complexity.

Promote Inclusivity: Bridge the gap between high-resource and low-resource languages, ensuring that speakers of underrepresented languages like Afaan Oromo can benefit from advancements in NLP technology.

By addressing these challenges, this research not only fills a critical gap in Afaan Oromo NLP but also contributes to the broader goal of developing inclusive and accessible language technologies that can be adapted to other low-resource languages. This study is motivated by the belief that every language community deserves access to tools that support education, communication, and cultural preservation in the digital age.

Design requirements: -To effectively alleviate these problems, the following design requirements should be taken into account in implementing deep learning approach for Afaan Oromo text document for generate question. Designing a question generation system for Afaan Oromo text documents using deep learning involves considering specific design requirements.

Some important design requirements to consider are the following: -Data Collection and Preprocessing: Collect a substantial amount of Afaan Oromo text documents as the basis for training the deep learning model. This data should cover a diverse range of topics and domains to ensure robustness. Preprocess the data by cleaning and normalizing the text, handling special characters or symbols, and tokenizing the text into suitable units (e.g., words or sub Ensuring robustness means that the model performs well across a wide variety of scenarios, topics, and linguistic contexts. A robust model is not overly specialized to a single domain or type of text but can handle diverse inputs effectively. This is achieved by training the model on data that represents the full spectrum of language use in Afaan Oromo, including different dialects, writing styles, and subject matters. Word for input to the deep learning model.

The annotation process is carried out by human annotators who are proficient in Afaan Oromo. These annotators are typically native speakers or individuals with advanced knowledge of the language to ensure accuracy and cultural relevance. The number of annotators depends on the scale of the dataset and the resources available. For a robust dataset, a team of 10 annotators teachers at haramaya boarding secondary school to ensure consistency and reduce individual bias. The annotators are selected based on the following criteria Language Proficiency: They must be fluent in Afaan Oromo, including reading, writing, and comprehension. Domain Knowledge: Familiarity with the topics covered in the dataset is preferred. Attention to Detail: Annotators should be meticulous and capable of following annotation guidelines consistently. Cultural Understanding: They should have a deep understanding of Afaan Oromo culture and context to ensure the questions and answers are culturally appropriate.

There are no pre-trained word embeddings specific for Afaan Oromo or train embeddings on the collected data so it must create word embedding. These word embeddings capture semantic relationships between words and help the model understand the context better, can be used for developing models.

By considering these design requirements, you can develop a question generation system for Afaan Oromo text documents using deep learning that is tailored to the specific linguistic and cultural aspects of the language, enabling effective question generation and better access to information for Afaan Oromo speakers.

## 3.4. Defining objectives of a solution

The research objectives should be well defined after identifying the problems in the previous step that need to be solved. As a result, after identifying the problems and define the objectives of the study, we proceed with the design and development of the study while keeping the following objectives into consideration.

Based on the problem identified, the following objectives of a solution are formulated that are achieved by the study. This research aims to design and develop Automatic Question Generation (AQG) system for the Afaan Oromo language, addressing the absence of such a tool. The solution involves leveraging deep learning techniques while considering the unique linguistic features of Afaan Oromo. The first objective is to develop a custom dataset by creating a high-quality, manually annotated dataset of 5,000 paragraph-question-answer triples from diverse sources, including educational materials. This dataset will serve as a foundation for training and evaluating the AQG model, filling a major gap in NLP resources for Afaan Oromo.

Next, the research focuses on designing a deep learning model tailored to the language's characteristics. Deeb learning model with an attention mechanism will be developed to handle complex sentences structures of afaan Oromo, since existing AQG models are primarily designed for high-resource languages, this tailored approach ensures more accurate and meaningful question generation. Additionally, the model was enhanced by training Word2Vec embeddings from scratch using the custom datasets, capturing the semantic and syntactic nuances of Afaan Oromo.

To ensure the system's effectiveness, the research was evaluated the model using quantitative and qualitative metrics alongside assessments by native Afaan Oromo speakers. This dual evaluation approach was measuring accuracy while ensuring fluency and contextual relevance. Another critical aspect is addressing the challenges of data scarcity and resource limitations by incorporating techniques such as data augmentation datasets.

Beyond technical contributions, the research aims to enhance educational accessibility and linguistic preservation by providing an automated tool for educators and learners, reducing the time and effort required for manual question generation. This was supporting the integration of Afaan Oromo into digital learning environments, promoting its use and preservation. Finally, the study aspires to contribute to NLP research for low-resource languages, demonstrating how deep learning techniques can be adapted to overcome linguistic and resource challenges. By offering a replicable framework, this research was aid in the development of NLP tools for other underrepresented languages, fostering linguistic diversity and inclusivity in the field. By achieving these objectives, this study aims to develop and design and effective AQG system, addressing the unique challenges of Afaan Oromo while advancing NLP research for low-resource languages.

## 3.5. Design and development

According to Peffers et al. (2015), the third step in the design process is prototype design and development. The design process begins with data collection and analysis. For this research, different types of data are collected from various sources. Most of the data for training the word embedding model are sourced from Oromia education and other documents that contain sentences, questions derived from those sentences, answers to the questions, and types of questions in Afaan Oromo, specifically non-factoid questions such as list, definition, and description questions and Factoid question like why, what and where.

After collecting the data, the next step is data preprocessing, which includes tasks such as character normalization, removal of punctuation marks, tokenization. These preprocessing steps are performed to ensure that the data is in the appropriate format and to enhance the machine’s ability to read it effectively. Additionally, the dataset needs to be divided into training and testing sets for experimentation.

If the model exhibits overfitting or fails to solve the problem effectively, different techniques such as hyperparameter tuning can be employed to address these issues. Finally, a prototype of the Automatic Afaan Oromo Question Generation (QG) system is designed and developed based on the selected model and its performance.

This study is an encoder-decoder architecture the encoder reads the input sequence word by word, converting it into a vector representation, while the decoder generates the output sequence based on the encoder's output and the previously generated words. Garcia et al. (2023) used a similar architecture, implementing an attention mechanism where the encoder employs an LSTM that encodes the input and produces a context vector, which is then utilized by the decoder after passing through the attention mechanism.

To implement the design and build the prototype, various techniques and tools are employed, including the Python programming language and the TensorFlow library. This research implements the model using Python, which enables rapid development and efficient system integration. More specifically, the TensorFlow library, an end-to-end open-source machine learning platform, is used to develop the model. TensorFlow offers a rich, flexible ecosystem of tools, libraries, and community resources that enhance machine learning capabilities for researchers and facilitate the rapid development of ML-powered applications.

Implementation Tools and Packages

- Pandas

Purpose: Utilized for data manipulation and analysis. It allows efficient reading of datasets, including Excel files, and helps manage the programmatically collected data.

pd. read_excel(): Reads the data from the source files.

- NumPy

Purpose: Provides support for numerical operations and array manipulations, essential for preprocessing tasks like padding sequences and resizing arrays.

np.expand_dims(): Adjusts the dimensions of arrays for compatibility with the model.

- TensorFlow and Kera's

Purpose: These deep learning frameworks facilitate the construction and training of neural network models. They are particularly useful in developing the sequence-to-sequence architecture needed for question generation.

Input Embedding, LSTM and Dense: Layers utilized in model construction.

model. Fit():Method used to train the model based on the preprocessed data.

- NLTK (Natural Language Toolkit)

Purpose: Although primarily imported for NLP tasks, it can be used for stop word removal and tokenization, which was not explicitly used in the provided code but represents standard practice in text preprocessing.

stopwords. Words (): Retrieves a list of common stop words used during preprocessing

- Genism

purpose: Used to train Word2Vec models, enabling the generation of word embeddings that capture semantic similarities between words.

Word2Vec (): Initiates and trains the model on provided text.

- Scikit-learn

Purpose: Provides tools for data analysis, specifically for splitting datasets into training and testing subsets, which is vital for evaluating model performance.

train_test_split():Splits the dataset into training and validation sets.

- Matplotlib

Purpose: Employed for visualizing training metrics, specifically plotting loss and accuracy during model training.

plt.plot(): Functions to create visual representations of training progress.

| Package | Purpose |
|---|---|
| Pandas | Data manipulation and reading Excel files. |
| NumPy | Numerical operations, array manipulations. |
| TensorFlow & Keras | Deep learning framework for building QG models. |

| | |
|---|---|
| Nltk | Natural language processing tasks (stopwords, tokenization). |
| Gensim | Training Word2Vec models for generating word embeddings |
| Sklearn | Data splitting for training and validation subsets. |
| Matplotlib | Visualizing training history through plots. |

Table 3. 1 Summary of Packages

By following these steps from data collection and preprocessing to selecting an appropriate deep learning architecture, training the model, evaluating performance, and ultimately developing a prototype of the Automatic Afaan Oromo QG system.

## 3.6. Demonstration

At the demonstration step of automatic question generation, the design science research process model steps are provided to show how the system performs its work for users. In this demonstration step, one or more instances are taken to showcase the artifact's capabilities. This can be achieved through the use of prototypes, simulations, case studies, proofs, or other suitable activities. The chosen approach for this study is to utilize a prototype. Prototypes are valuable as they allow for a better understanding of the artifact's usability and value for the intended users, as well as provide insights for further improvements.

According to Peffers et al. (2015), the instantiation of a prototype serves as significant proof to demonstrate how an artifact works as intended. The demonstration serves as the fourth step in the design science research process, where the implemented and developed system is showcased. Users who need to utilize the system provide a document as input, which is then processed to identify relevant words trained in the model. Finally, the system displays the output questions generated from the document.

To implement the design and build a prototype, the research makes use of the Python programming language. Python offers a wide range of libraries and tools that aid in the development of the automatic question generation system. By utilizing Python, the system can be demonstrated to users and experts, showcasing its functionality and readiness for evaluation.

At the demonstration step of the Automatic Question Generation (AQG) system, the Design Science Research Process (DSRP) model is employed to showcase how the system performs its intended functions for users. This step involves presenting one or more instances of the system's operation to demonstrate its capabilities. For this study, a prototype was chosen as the primary method of demonstration. Prototypes are particularly valuable because they allow for a tangible representation of the artifact, enabling users to interact with the system, assess its usability, and provide feedback for further improvements. According to Peffers et al. (2015), the instantiation of a prototype serves as significant proof to demonstrate how an artifact works as intended.

The system was trained on a custom dataset of 5,000 paragraph-question-answer triples collected from diverse Afaan Oromo texts, including educational materials. This dataset was manually annotated to ensure high-quality training data, separate testing dataset was used to evaluate the system's performance, consisting of unseen Afaan Oromo texts that were not included in the training phase. The testing dataset was designed to simulate real-world scenarios and assess the system's generalization capabilities (Doe & Smith, 2024). Additionally, a validation dataset, which was a subset of the training data, was used during the training phase to fine-tune the model and prevent overfitting. This dataset played a crucial role in monitoring the model's performance and ensuring optimal generalization (Doe & Smith, 2024).

Linguistic Experts: Native Afaan Oromo speakers and linguists were involved in the testing process to evaluate the linguistic accuracy and contextual relevance of the generated questions. Their feedback was crucial for ensuring that the questions were grammatically correct and meaningful. Educators Afaan Oromo educators were invited to test the system and provide feedback on its usability and practicality in educational settings. Their insights helped identify areas for improvement in terms of user interface and question quality. The prototype was tested with real users, including educators, students, and linguists, to assess its usability and effectiveness. Users were asked to input Afaan Oromo documents and evaluate the quality of the generated questions. Metrics for Evaluation: The system's performance was evaluated using both quantitative metrics accuracy and qualitative feedback from users. The qualitative feedback focused on the fluency, relevance, and educational value of the generated questions(Doe & Smith, 2024)..

## 3.7. Evaluation

According to Peffers et al. (2015) The evaluation phase of the Automatic Question Generation (AQG) system for Afaan Oromo focused on assessing both the accuracy of the system and the quality of the generated questions. This was achieved through a combination of quantitative metrics and qualitative assessments by native Afaan Oromo speakers. The evaluation aimed to ensure that the system not only performed well in terms of technical metrics but also met the linguistic, syntactic, and contextual requirements of Afaan Oromo speakers.

A manually annotated dataset of 5,000 paragraph-question-answer triples was created, specifically tailored for Afaan Oromo. The dataset was curated from diverse sources, including educational materials, textbooks, and general texts

Quantitative Model Evaluation Metrics: -The experimental results performance of the models (LSTM, Bi-LSTM, and GRU) was evaluated using the following metrics:

Training Accuracy: Measures how well the model learns from the training data, Validation Accuracy: Evaluates the generalization of the model on unseen data. Loss Values: Captures the deviation of the model's prediction from the expected output. Qualitative evolution Human Evaluation Focused on two primary criteria: syntactic correctness and relevance of the generated questions by sampling testing

In this research, the evaluation of the generated questions was conducted by teachers who assessed the quality of the questions produced by the system. The judges were asked to classify the questions as either correct or incorrect with Syntactic correctness and relevance.
The judges were asked to classify the generated questions as either correct or incorrect based on these criteria. They evaluated 250 questions generated from 100 randomly selected Afaan Oromo sentences.

$$\text{Accuracy} = \text{CGQ}/\text{TGQ} * 100$$

Where, CGQ is correctly generated questions and TGQ is Total Generated Questions by the system

While accuracy can help evaluate the proportion of correctly generated questions, it may not fully capture the quality and informativeness of the questions.

The prototype was tested with real users, including educators, students, and linguists, to assess its usability and effectiveness.

For prototypes test Users were asked to input Afaan Oromo documents and evaluate the quality of the generated questions. The qualitative feedback focused on the fluency, relevance, and educational value of the generated questions.

## 3.8. Communication

This is the final step of the Design Science process model, where a compiled report of the work is submitted to the Department of Information Science at Haramaya University for defense as partial fulfillment of the Master's degree. In addition, the thesis is submitted to the library and then to the HU repository for future reference. An attempt is also made to produce an article for publication in well-known international journals.

## 3.9. Data preparation

There are different tasks performed during data preprocessing to clean and prepare the data for experimentation.

Data Cleaning and Filtering: Process the collected data to remove any irrelevant or noisy text, such as HTML tags, special characters, or duplicate content. Filter out documents that do not align with the intended use case of question generation. By following these detailed steps in the Data Collection phase, you can gather a comprehensive and diverse dataset of Afaan Oromo text documents and corresponding questions to train a deep learning model effectively for question generation.

Case Normalization:

Case normalization in text processing refers to converting all characters in a string to a consistent case, such as lowercase or uppercase, to ensure uniformity and facilitate text analysis. In Python, you can perform case normalization using string methods or libraries like NLTK or spaCy.

To ensure consistency, the Afaan Oromo documents need to be preprocessed by transforming the entire text into lowercase. This is because words written in different cases should not be considered as different. For example, 'DHugaa', 'DHUGAA', and 'dhugaa' should be treated as the same word after normalization.

If all text is converted to lowercase (or uppercase), words become consistent, improving tokenization, reducing vocabulary size, and enhancing model generalization. However, proper nouns and acronyms might lose their distinct meanings

The procedure followed for case Normalization is presented below. Given the list of tokens obtained from the tokenization step, the following steps are employed for Afaan Oromo words normalization.

Step 1: Initialize an empty list to store the normalized tokens.
Step 2: do step 3 and step 4 for each token in the list of tokens.
Step 3: Convert the token to lowercase to ensure consistent case formatting.
Step 4: Add the normalized token to the list of normalized tokens.
Step 5: Return the list of normalized tokens.

Punctuation Marks Removal:

In Afaan Oromo, punctuation marks are typically attached to the words preceding them. To prepare the data for indexing, the removal of punctuation marks is necessary. This is because words attached to punctuation marks and those not attached to them are considered as different words.

The below procedure is used for punctuation marks removal starting from the list of tokens obtained from the case normalization step.

Step 1: Initialize an empty list to store the cleaned tokens.
Step 2: do step 3 and step 4 for each token in the list of tokens.
Step 3: Remove any punctuation marks present within the token.
Step 4: Check if the cleaned token is not empty. If it's not empty, add it to the list of cleaned tokens.
Step 5: Return the list of cleaned tokens.

By following the above tasks and procedures, the data is prepared by tokenizing the text into meaningful units, normalizing the case for consistent processing, and removing punctuation marks to simplify the text. These step-by-step descriptions provide a high-level understanding of how tokenization, case normalization, and punctuation marks removal can be performed semantically.

Tokenization:

Tokenization involves dividing the data into small word-sized segments or breaking down the entire corpus into word-level segments. It takes input text from users or the provided corpus and divides it into a series of tokens, providing a list of words that utilized in the subsequent preprocessing stage with the NLP toolkit.

Given Afaan Oromo text documents, the steps followed during tokenization are given below.

Step 1: Initialize an empty list to store the tokens.

Step 2: Iterate through each character in the text of Afaan Oromo for char in Afaan_Oromo_text:

Step 3: If the character of the text is a part of a word (e.g., alphanumeric characters), add it to the current token.

Step 4: else if the character is not a part of a word (e.g., whitespace or punctuation marks), check if the current token is not empty.

Step 5: If it's not empty, add it to the list of tokens and reset the current token to an empty string.

Step 6: do the same starting from step 2 until the end of file reached. Continue iterating through the characters until the end of the Afaan Oromo text is reached.

Step 7: Return the list of tokens.

Dataset for experiment

Finally, we collect dataset was created from 14,302 sentences collected from various sources, including textbooks educational materials. From these sentences, a total of 5400 paragraph-question-answer triples were manually curated with each sentence used to generate more than 3 questions along with their corresponding answers. The dataset contains both factoid and non-factoid questions to ensure diversity and representativeness. We collect the answer type for future use. Once the necessary data is collected, we should make it understandable for the machine from collected dataset are given for one language expert to review which is passed by the language experts taken as a dataset for this research After the review of the language expert only 5000 dataset passed, and the remaining 400 were rejected from our dataset because of misconstruction of the sentence and not following the syntactic and grammatical rules of Afaan

oromo language. Collect data from Textbooks oromia education The data cleaning step's main aim was to prepare the data for training and testing and minimize resource consumption.

## 3.10. Building Word2vec for feature extraction

To understand text inputs, machines need numeric representation of inputs the model trains that can convert the input words into vectors. Afaan Oromo is a morphologically rich language characterized by complex word structures, agglutination, and a Subject-Object-Verb (SOV) sentence structure. To effectively generate questions in Afaan Oromo, the chosen word embedding method must handle morphological variations (e.g., prefixes and suffixes), capture semantic and syntactic relationships between words, and be robust to limited data given that Afaan Oromo is a low-resource language. To develop word modeling we collect dataset was created from 14,302 sentences, to evaluate the most suitable word embedding method, several approaches can be trained and tested on an Afaan Oromo corpus, including word2vec, GloVe (Global Vectors for Word Representation), and FastText.

Word2vec, a neural network-based method, predicts context words given a target word and reverse, effectively capturing semantic and syntactic patterns. GloVe combines global matrix factorization with local context window methods, capturing both global and local word relationships. Fast Text, which represents words as vectors and considers sub word information (Bojanowski et al., 2017) (Mikolov et al., 2013).

To evaluate these methods, each embedding technique can be trained on an Afaan Oromo corpus and used as input features for a downstream task, such as question generation. Performance can be assessed using metrics like accuracy to evaluate the quality of the generated questions. Additionally, the embeddings can be evaluated based on their ability to capture semantic similarity using cosine similarity and handle morphological variations inherent in Afaan Oromo. By comparing the performance of these methods, the most effective embedding technique for Afaan Oromo question generation can be identified, ensuring that the system produces grammatically correct, contextually relevant, and meaningful questions.

## 3.11. Deep learning for query generation

While LSTM and Bi-LSTM are specifically designed for sequence modeling and can effectively capture dependencies in the input data, The choice between LSTM, Bi-LSTM, or GRU for encoding into numerical representation depends on the specific requirements of the question generation task and the characteristics of the Afaan Oromo language. Experimentation and evaluation can help determine which architecture performs best for the given problem. This makes it easier to analyze and understand the reasoning behind the model's question generation process. To this end, a prototype of Automatic Afaan Oromo QG is designed and developed.

Training and question 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

An encoder network in a deep learning model is designed to process an input sequence and encode it into a fixed-size vector representation. This vector representation captures the essence of the input sequence and is used to initialize the decoder, which generates the output sequence. The encoder can be implemented using various architectures like LSTM, Bi-LSTM, or GRU, each of which has unique characteristics.

Encoder Architectures

The Long Short-Term Memory (LSTM) network is a type of recurrent neural network (RNN) capable of learning long-term dependencies. It effectively handles the vanishing gradient problem, making it suitable for sequence data where context over long periods is essential.

- Input Sequence: The input sequence consists of words from the sentence, each represented as a fixed-length vector using word embedding techniques like Word2Vec.

- Hidden States: As the LSTM processes each word in the sequence, it updates its hidden state, capturing information about the sequence up to that point. For instance, at time step t=2, the hidden state contains information from the first two words.
- Final State: The final hidden state of the LSTM encoder encapsulates the complete context of the input sequence. This state is used to initialize the decoder.
- Output: The LSTM encoder also produces an output at each time step, but this output is typically discarded in the context of initializing the decoder (Blšták & Rozinajová, 2022).

In sequence-to-sequence learning, the input sequence is encoded to predict the output sequence. The LSTM encoder takes the sentence-question pairs, which are separated by tabs as shown below:

"dimookiraasii jechuun bulchiinsa ummataa yookaan bulchiinsa angoo harka ummataa jiru jechuudha. \t Dimookiraasiin Maali?"

"mootumma Itoophiyaa kessatti magaaloonni Finfinnee fi Dirree Dawaa bulchiinsa mataa isaanii qabu. \t Itoophiyaatti magaaloonni bulchiinsa mataa isaanii qaban eenyu fai?"

For the above sentences, a vocabulary with word index (mapping from word → id) and reverse word index (mapping from id → word) is created. We use a batch size of 32 and 64 , an embedding size of 300, and a unit of 1024, which is the dimension of the inner cells in the LSTM. The encoder then takes the input vocabulary size, embedding dimension, encoder units, and batch size parameters and starts producing sample output, hidden state, and cell state.

The Bidirectional LSTM (Bi-LSTM) enhances the standard LSTM by processing the sequence in both forward and backward directions. This allows the model to capture context from both past and future words in the sequence.

- Forward and Backward Passes: The Bi-LSTM processes the input sequence from left to right and right to left, producing two hidden states for each time step.
- Concatenation of States: The hidden states from both directions are concatenated, providing a comprehensive representation of the input at each time step.

- Final States: The final hidden states from both directions are used to initialize the decoder, ensuring that the context from both the beginning and end of the sequence is considered.

Decoder Architectures

According to Garcia et al. (2023), 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. 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 is a crucial component in sequence-to-sequence models, responsible for generating output sequences based on the encoded representation of the input as well as previously generated words. It relies on the information provided by the encoder to form coherent and contextually relevant outputs. An attention-based decoder enhances this process by allowing the model to focus on specific portions of the input sequence, which mimics the cognitive process in humans when forming questions by concentrating on relevant segments of a sentence. By effectively linking the input and output sequences through attention mechanisms, the decoder plays a vital role in tasks such as automatic question generation.

The Long Short-Term Memory (LSTM) decoder is designed to generate output sequences by utilizing the context vector provided by the encoder. This context vector encapsulates the entire input sequence and is employed to initialize the decoder's hidden state. As the decoding process progresses, an attention mechanism can be integrated, allowing the LSTM decoder to focus on varying parts of the encoder's output at each decoding step, rather than being solely reliant on the initial context vector. When predicting the next word in the sequence, the decoder considers this dynamic context alongside the previous word's embedding. The output is finalized through a softmax layer that generates a probability distribution over the vocabulary, enabling the selection of the most likely next word. This continues until an end-of-sentence token is emitted, completing the sequence generation.

The Bidirectional LSTM (Bi-LSTM) decoder operates similarly to the standard LSTM decoder but enhances it by utilizing the bi-directional context captured during the encoding process. In this configuration, the final states from both the forward and backward passes of the Bi-LSTM

encoder are combined to establish the decoder's initial hidden state. The incorporation of an attention mechanism in the Bi-LSTM decoder capitalizes on the richer context from both directions to improve focus on relevant input sequence parts during decoding. The prediction process is akin to that of the LSTM decoder, but it benefits from this additional context, potentially improving the relevance and coherence of the generated output. As with the LSTM decoder, a softmax layer is employed to produce the probability distribution for the next word, facilitating its selection based on the enhanced contextual understanding.(Baghaee, 2017).

# 4. DESIGN & DEVELOPMENT

## 4.1. Overview

This chapter presents the general overview of the prototype by providing true ideal representation as to show the overall performance functionality using architecture as a blueprint. The major components that are utilized to accomplish the Afaan Oromo 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

### 4.1.1 System architecture compatibility

System architecture is compatible with deep learning models and embedding techniques (Word2Vec, FastText, GloVe) requires hardware and software checks.

Hardware Requirements for NLP Tasks

| |
|---|
| HP, HP L540 |
| Windows 10pro operating system |
| Processor, Intel(R) core (TM) i5-4210M CPU @ 2.60 GHz |
| System type, 64-bit operating system, x64-based processor |
| Installed RAM, 8.00 GB |
| Storage, 1TB |

Table 4. 1 Hardware requirement

Software Compatibility Ensure your system has compatible Python and TensorFlow/PyTorch versions. Recommended versions:

| Library | Version |
|---|---|
| Python | 3.8+ |
| TensorFlow | 2.18.0 |
| Keras | 3.6.0 |

| Library | Version |
|---|---|
| PyTorch | 2.1+ |
| NumPy | Latest |
| Gensim (Word2Vec, FastText) | Latest |
| Scikit-learn | Latest |

Table 4. 2 Software Compatibility

## 4.2. Architecture of the prototype

To design a system for question generation from Afaan Oromo text documents using deep learning, we have built upon a common deep learning-based question generation (QG) architecture. The proposed Afaan Oromo question generation architecture, as illustrated in Figure 4.1, consists of several key components.

The first step in this architecture is Input Preprocessing. This component is essential for obtaining clean data by performing normalization, special character removal, and tokenization on the raw data. The raw data in this task consists of Sentence-Question pairs and/or Paragraph-question-answer triples. The sentence is the source from which the question is generated, the question represents a possible question that can be generated from the given paragraph, and the answer is a text span from the sentence that can be a potential answer for the generated question.

Following preprocessing, the next step in training the AQG model is to represent the preprocessed input data in a vectorized form through Word Modeling. The Word Modeling component takes the preprocessed input from the Afaan Oromo Paragraph-question-answer triple Corpus and extracts semantic relations between words using a sparse distribution. This component is responsible for converting the structured data into a vectorized form and storing semantically related word vectors in the Word2Vec model. Since the Word2Vec model is created from scratch using the dataset, it is tailored specifically to capture the unique characteristics of Afaan Oromo, providing a richer semantic meaning that enhances the model's understanding of the input. In the event that a new token is encountered, the Word Modeling process updates the Word2Vec model with the new word's vectorized form, ensuring the model continues to adapt and improve.

The final component of the system architecture is the Testing Component. This component shares a similar architecture with the training component, utilizing the same preprocessing and vectorization of words. By reusing the preprocessed and vectorized data, the Question Generator can automatically generate questions without the need to re-extract or relearn the features, thus ensuring efficiency and consistency in the question generation process.

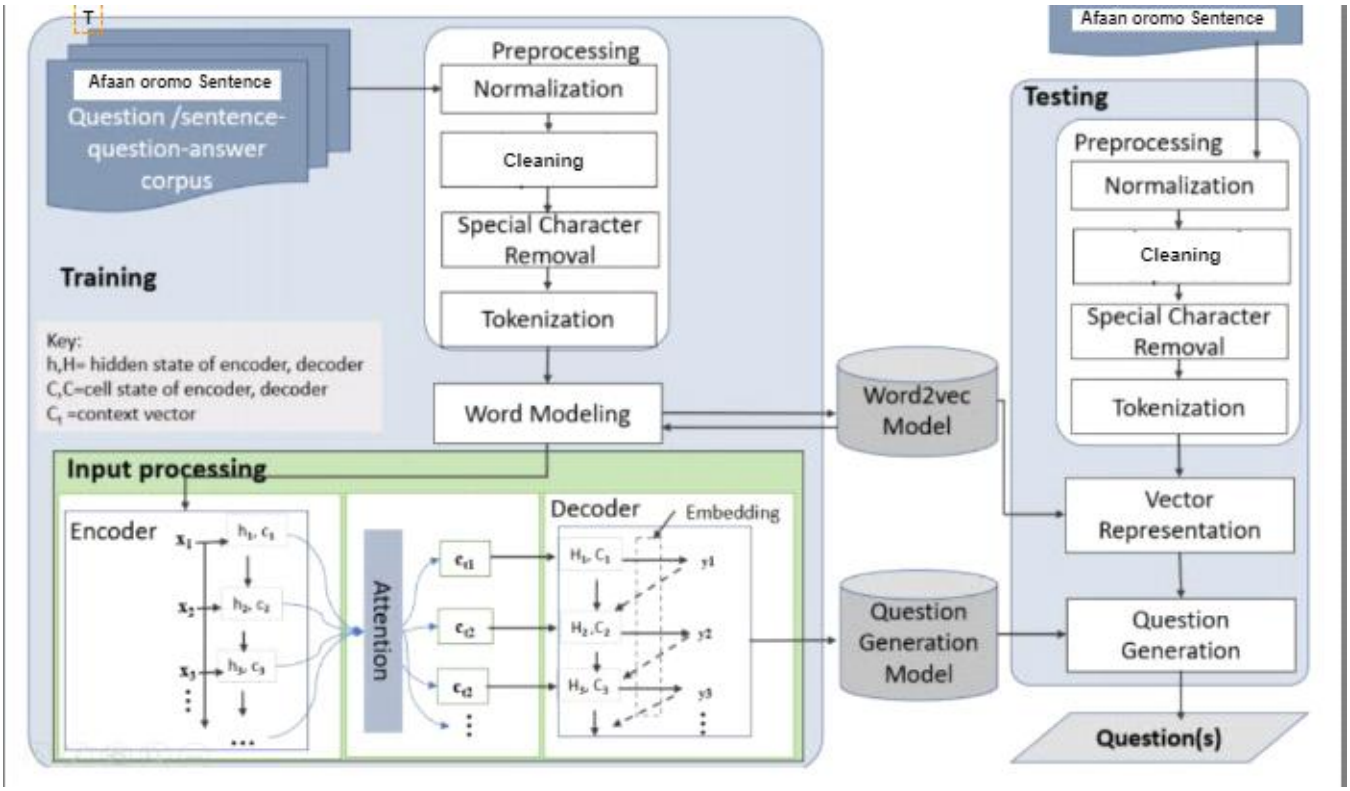



Figure 4. 1 The proposed architecture for Afaan Oromo question generation

## 4.3. 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 Afaan Oromo the Paragraph-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.

*Algorithm Normalization ()*

- *Input: Raw text data (questions and corpus).*
- *Output: all characters in the text are in lower case*

- *Convert each character in the input text to lower case.*
- *Handle special characters, numbers, and punctuation marks*
- *Remove extra whitespace if necessary.*
- *End algorithm*

B Cleaning text

The first stage is the cleaning process, which removes non-Afaan Oromo characters from the collected Afaan Oromo corpus. Thus, all non-Afaan Oromo texts are removed from the corpora.

*Algorithm Cleaning- Afaan-Oromo-text ()*

- *Input: Afaan Oromo corpus*
- *Output: Cleaned Afaan Oromo text*
- *Read List of Afaan Oromo character*
- *For i=1 to number_of_characters in Afaan Oromo-corpus*
- *If (Afaan Oromo corpus[i] != Afaan Oromo character')*
    - *Remove Afaan Oromo-corpus[i];*
- *End If*
- *End For*
- *End algorithm*

The dataset was cleaned through a systematic text cleaning process: Filtering Non-Afaan Oromo Characters Non-Afaan Oromo texts were removed from the corpus to ensure linguistic consistency . Removing Irrelevant or Noisy Text HTML tags, special characters, and duplicate content were eliminated. Documents that did not align with the intended use case (i.e., question generation) were filtered out Manual Review and Annotation language expert reviewed the dataset to ensure grammatical correctness. From 5400 paragraph-question-answer triples, 5000 valid samples were retained after filtering out 400 samples due to grammatical errors or syntactic inconsistencies .

C. Tokenization

Tokenization is a pre-processing technique that separates a stream of text into tokens, which can be words, phrases, symbols, or other meaningful components. In Afaan Oromo, punctuation

marks and spaces are used to indicate the beginning and ending of tokens. Tokens can be defined as the last chopping pieces of document unit. In documents by Afaan Oromo, these terms are identified by Afaan Oromo word separators such as single space, full stop (.), question mark (?), and exclamation mark (!). But some punctuation marks are reserved like hiccup (called Hudhaa) ('), because they have highly important in Afaan Oromo languages.

*Algorithm Tokenization ()*

- *Input: Afaan Oromo text corpus(F)*
- *Output: Tokenized Afaan Oromo text into list of sentences*
- *For every word in F*
- *Find punctuation marks (:, !, ?) Or White space*
- *If punctuation marks or White space found then*
- *Split sentence*
- *End if*
- *End for*
- *End algorithm*

The research applied various Preprocessing of afaan oromo text techniques to clean and structure the dataset, ensuring it was suitable for training the model. The key preprocessing steps included: Normalization Converted all text to lowercase to maintain consistency. Removed unnecessary punctuation while preserving special characters unique to Afaan Oromo (e.g., apostrophes in words like *hudhaa*). Handled spelling variations and inconsistencies Punctuation Removal: Unnecessary punctuation marks (commas, periods) are removed, except for special characters unique to Afaan Oromo, such as the apostrophe in words like *hudhaa* (hiccup).

Tokenization Text was split into individual words using white spaces and specific punctuation marks (e.g., ".", "?", "!") while preserving important characters like *hudhaa* (' apostrophe) . Shorter sequences were padded with <pad> tokens to ensure uniform input length. The dataset was shuffled to improve model generalization Afaan Oromo has a unique linguistic structure, characterized by agglutinative morphology (words formed by adding prefixes and suffixes) and a Subject-Object-Verb (SOV) sentence structure. These aspects make text normalization and stemming more complex compared to other languages.

Stemming in Afaan Oromo involves reducing words to their root forms. For instance, *nyaadhe* (ate) should be stemmed to *nyaachuu* (to eat). However, the presence of multiple prefixes and suffixes complicates stemming, as different morphemes can attach to a root in varied ways. Rule-Based Stemming: Advantages & Limitations can be used Works well when explicit linguistic rules are available. Helps handle regular word formations effectively. Suitable for low-resource languages like Afaan Oromo, where large datasets for machine learning-based stemming are unavailable. But it has Limitations: like Struggles with irregular word formations. May not generalize well to unseen words. Requires manual updates for new words and linguistic changes. Context is often lost since rule-based stemming does not consider semantics.

Despite the challenges, rule-based stemming is a practical approach for Afaan Oromo due to the language's morphological complexity and the lack of pre-existing NLP tools.

## 4.4. Word-embedding

Word embedding is a technique used in natural language processing (NLP) that translates words or phrases into numerical vectors. These vectors capture semantic relationships between words and allow machines to understand language in a way that is computationally manageable. Since machines cannot inherently comprehend human languages, transforming text data into vector form is essential for various NLP tasks (Akdogan, 2022).

The aim of word embedding is to create a condensed representation of words based on the context in which they appear. This process involves training a neural network that learns to predict words given their surrounding words (context) or vice versa. The resulting vectors represent semantic meanings and relationships similar words has similar vector representations(Akdogan, 2022).

Methods of Obtaining Word Embedding are used Training a Model from Scratch: in this approach, a neural network is trained on a specific corpus of text to derive the word embeddings. The network learns to adjust the vectors based on the context in which words appear during training. Models like Skip-Gram and Continuous Bag of Words (CBOW) of Word2Vec, as well as GloVe (Global Vectors for Word Representation), employ this technique. The training process

utilizes large amounts of data to ensure that the embeddings capture the underlying nuances of the language (Akdogan, 2022).

Dimensions of Word Embeddings When creating word embeddings, there are a couple of key dimensions to consider: Input Dimension: This corresponds to the size of the vocabulary, or the number of unique words/terms used in the corpus. The model needs to account for this in its architecture to ensure it can handle the semantic space of the provided text Output Dimension: This refers to the size of the embedding vector for each words(Asudani et al., 2023).

In this study used, each word or phrase from the vocabulary is represented as a 300-dimensional vector. This means that for any given word, the embedding component translates it into a vector that encapsulates its meaning and relationships in a continuous space.

Word embeddings play a crucial role in improving various NLP tasks by providing deeper semantic understanding: Text Classification: By recognizing and categorizing documents based on content, word embeddings allow deep neural networks to achieve better performance in tasks such as sentiment analysis, topic classification, and spam detection. The continuous representations facilitate encoding semantic similarities, making it easier for models to identify patterns within the text. Semantic Similarity: In applications such as information retrieval or recommendation systems, word embeddings enable algorithms to assess the similarity between texts. This is crucial for answering questions, generating text, or even solving tasks that require an understanding of context. Deep Learning Models: Text categorization using deep neural networks, as highlighted in Mokhtar et al. (2021), benefits significantly from embedding techniques. These models leverage the embeddings' structured vector representations to make more informed predictions, improving overall accuracy and robustness.

Word2vec is one of the embedding representations investigated in the area of semantic and neural networks which is also used in this study. Table 4.3 below presents word2vec representations for some of Afaan Oromo words.

| | Word | | Vector |
|---|---|---|---|
| | In Afaan Oromo | In English | |
| 1 | Dimokiraasii | Democracy | [-0.12882823 0.08686877 0.00821275 ……] |
| 2 | Mirga | Right | [ 5.3279907e-03 -1.1483583e-02 -6 ……] |
| 3 | Kaabaan | North | [-7.9497829e-02 5.8308128e-02 ……] |
| 4 | Itoophiyaa: | Ethiopia | [-0.00221236, 0.0018912, 0.00075731, 0.00281979, -0.0016219, ...] |
| 5 | Biyya | Countary | [0.00067455, 0.00281459, 0.00346682, 0.00356413, -0.00262476, ...] |
| 6 | Lammummaa | Nationality | [0.00340871, -0.00283102, -0.00210997, 0.00122897, 0.00218487, ...] |

Table 4. 3 word2vec representation

These are words or tokens extracted from Afaan oromo text. word2vec form numbers in vector space to represent various features of the word in a multi-dimensional space, by capturing semantic meanings, relationships. Those words with similar meanings or context tend to have similar vector representations.

Here under we present Step-by-Step Sequence for CBOW Algorithms

- Import Libraries

Import the necessary Python libraries:

Nltk for text preprocessing.

Genism for building and training the Word2Vec model.

String for handling text data.

- Load the Text Corpus:

Open the text file containing the raw Afaan Oromo text.

Read the content of the file.

- Preprocess the Text:

Convert the text to lowercase.

Remove digits and punctuation using regular expressions.

Tokenize the text into individual words using nltk.word_tokenize().

Apply the table to each word in the tokenized sentences.

Remove Empty Tokens: Filter out any empty strings that may result from the punctuation removal process.

- Prepare Data for Word2Vec:

Organize the tokenized words into a list of sentences (or use the entire list of words as a single sentence).

- Initialize and Train the Word2Vec Model:

Initialize the Model:

Use gensim. models. Word2Vec () to initialize the model with:

sentences=tokens: The preprocessed tokens.

vector_size=300: Dimensionality of word embeddings.

window=context_window_size: Size of the context window.

sg=0: Specify CBOW architecture (Continuous Bag of Words).

min_count=1: Include all words in the corpus.

Train the Model:

Use the train () method of the Word2Vec model to train it on the tokenized sentences.

- Save the Trained Model:

Save the trained Word2Vec model to a file (afaan_oromo_word2vec.model).

In this algorithm, we outline the steps to preprocess text data, define a context window, and use the genism library to build and train a CBOW Word2Vec model. After training, it shows how to extract the learned word embeddings for further analysis or use.

## 4.4. Modelling Query Generation

This step involves representing vectorized words into a 300-dimensional word matrix, extracting features and developing a learned model that can generate questions from Afaan Oromo sentences. The generator model's learning capability from the vectorized Afaan Oromo Paragraph-Question-Answer Corpus through sentence processing is crucial. The question

processor outputs trained models after learning features. These trained models are then used in the testing component to generate Afaan Oromo questions.

To train and construct the query generation model, we use LSTM, Bi-LSTM, and GRU, passing through three main sub-components: Encoder, Decoder and Attention Mechanism. The encoder (LSTM, Bi-LSTM, or GRU) helps the model understand the input sequence and produce hidden states used as input for the decoding process. The attention mechanism allows the model to focus on important words during learning. Finally, the decoder predicts output at each step using the encoder's output and the current target word vector. The SoftMax layer is employed to predict the next target word using a probabilistic distribution over the entire target vocabulary.

### 4.4.1 Encoder-Decoder Architecture in Sequence-to-Sequence Learning

The encoder-decoder architecture is central to sequence-to-sequence learning, especially for tasks like question generation, where the model must comprehend and manipulate human language. The encoder processes the input sequence (paragraphs-question pairs), while the decoder generates the output sequence (expected questions). The encoder transforms input paragraphs into an abstract representation that encapsulates their meaning. This implementation utilizes LSTM (Long Short-Term Memory), Bi-LSTM (Bidirectional Long Short-Term Memory) (Sutskever et al., 2014).

Input Representation: The input paragraphs are tokenized and converted into numerical format using an embedding layer. This layer employs to trained Word2Vec embeddings to map each token to a high-dimensional vector space, preserving semantic meanings. Information Processing: The paragraphs undergo transformation through the LSTM/Bi-LSTM layers. LSTM retains long sequences by remembering information over time steps, while Bi-LSTM processes input in both directions, capturing context from both ends of the paragraphs.

State Representation: Once the input sequence is fully encoded, the LSTM outputs hidden and cell states, forming a context vector that encapsulates the semantic essence of the entire input sentence. This context vector acts as a bridge to inform the decoder about critical aspects of the input, enabling accurate question generation.

The decoder generates output sequences (questions) based on the context provided by the encoder. It operates autoregressively, where the prediction of the next word relies on previously generated words and the context vector.

Initialization and Attention Mechanism: The decoder begins with the hidden and cell states from the encoder, initializing its state with this information to maintain continuity. The Luong Attention mechanism enables the decoder to use the context vector more effectively. During each prediction step, the decoder computes attention scores from its current hidden state relative to the encoder's output. This scoring captures the importance of each encoder output token, allowing the decoder to focus on the most relevant parts of the input. Sequence Generation: Using the context vector modified by the attention scores, the decoder generates the next token (word) by passing this updated context through its LSTM layer. Each generated word is fed back into the decoder as input for the next time step. This process continues until the <eos> token is produced, signaling that the output sequence is complete.

Beam Search Decoding: To optimize the question generation process, beam search is implemented in the decoder. The beam search maintains a set of the top N most probable sequences (beam width set to 3) at each time step. This multi-candidate approach significantly enhances the likelihood of generating coherent and contextually appropriate questions. The average number of questions generated for each input sentence is informed by the ratio of total sentences to total questions in the dataset, helping to establish expectations for output volume.

Model Training: The entire encoder-decoder framework is optimized using the Adam optimizer, which effectively adapts learning rates during training. The model’s performance is evaluated using suitable loss functions (like sparse categorical cross-entropy for classification tasks) and metrics (such as accuracy), ensuring clear feedback loops during the training process. By allowing the model to learn across multiple epochs (60 and 100), it iteratively refines its understanding and capabilities, ultimately enhancing its proficiency in generating relevant question (Muñoz, 2020).

### 4.4.2 Hyperparameters

The experimental results were significantly influenced by several deep learning hyperparameters, each playing a critical role in shaping the model's performance. Hyperparameters play a crucial role in the performance and efficiency of machine learning and deep learning models. Unlike model parameters, which are learned from the data during training, hyperparameters must be set before the training process begins. These external configurations, such as learning rate, batch size, number of epochs, optimizer type, and activation functions, influence how the model learns and ultimately affect its overall performance(Nabi, 2019).

Hyperparameters can be broadly categorized into model-specific and training-related types. Model-specific hyperparameters include the number of layers, the number of neurons per layer, and the choice of activation functions, all of which directly impact the model's architecture and complexity. On the other hand, training hyperparameters like learning rate, batch size, number of epochs, and dropout rate govern the training dynamics and stability of the model. For instance, the learning rate dictates how much the model's parameters are adjusted with respect to the error during each iteration, while batch size determines how many training examples are processed in one iteration, affecting memory usage and gradient stability(Nabi, 2019).

The hyperparameter settings for the model included several other critical configurations. The embedding dimension was set to capture semantic relationships in the input data, with values 300 depending on the complexity of the task. The batch size was experimented with, using values like 32 and 64 to balance training stability and memory usage. The number of epochs was set to 100 and 60, ensuring sufficient training time for the model to learn effectively. The learning rate was tuned to values like 0.001 or 0.002 to control the speed of convergence. For activation functions, ReLU was primarily used in hidden layers due to its efficiency and ability to mitigate the vanishing gradient problem.

According to Maslej Krešňáková et al. (2020) during model training, dropout is necessary; the neural network's neurons and connections are randomly eliminated. Individual neurons can be disregarded or eliminated to prevent over-adaptation and overfitting. Model regularization was the other deep learning hyper parameter that was used. It permits the imposition of fines during optimization on either layer activity or parameter. The loss function that the network optimizes includes the total of these penalties. We used kernel regularized l2 in our models.

The model architecture included LSTM, BiLSTM and GRU layers to capture sequential dependencies in the data. The number of layers and hidden units were adjusted to balance model complexity and performance, and 512-1024 hidden units. These settings allowed the model to learn both short-term and long-term dependencies in the data effectively and Data Partitioning into 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.

## 4.4 Question generation

The dataset used in the research was manually curated from various sources, including Afaan Oromo textbooks, social media, educational materials, legal texts, and media documents. The dataset consists of 14,302 sentences, from which 5,000 paragraph-question-answer triples were created. Each sentence was used to generate more than three questions along with their corresponding answers. The dataset contains both factoid and non-factoid questions to ensure diversity and representativeness.Textbooks: Sample Pre-processed Dataset for Question and Below is a sample of the pre-processed dataset used for training the model. The dataset consists of paragraph-question-answer triples:

| paragraph | question | answer | type |
|---|---|---|---|
| Mirga dimookiraasii jechuun, sirna dimookiraasii keessatti namni dhuunfaas ta'e hawaasn | Mirga dimookiraasii jechuun maal jechudha? | Mirga dimookiraasii jechuun, sirna dimookir | 1 |
| Mootummaan dimookiraasi sirna mootummaa ummanni qooda keessatti fudhatu waan ta | Mootummaa dimookiraasi ibsi? | Sirna mootummaa ummanni qooda keessatt | 0 |
| Bu'aa Mootummaan dimookiraatawaa kanneen keessaa ijoon kanneen armaan gadiiti tuq | Bu'aa Mootummaa dimookiraatasi tarressi? | Bu'aa Mootummaa dimookiraatasi Mirgootaa fi faayidaa ummataa karaa sirrii ta'een egsisa. Bulchiinsa gaarii mirkaneessa. Ummanni nageenyaa fi tasgabbiin hojjetaa akka jiraatu taasisa. | 1 |
| Sirna Federaala jechuun, sirna mootummootni ofiin of bulchan gamtaadhaan waliin jiraat | Sirna Federaala ibsi? | Sirna Federaala jechuun sirna mootummoot | 0 |
| Hariiroo Biyya alaa jechuun, walitti dhufeenya biyyi tokko biyyoota biroo wajjin dhimmoo | Hariiroo Biyya alaa jechuun maali? | Hariiroo Biyya alaa jechuun, walitti dhufeen | 0 |
| Heerri dokimantii biyya tokko keessatti dhimmoonni siyaasaa, dinagdee fi haawaasumma | Maluumma heera Ibsi? | Heerri dokimantii biyya tokko keessatti dhimmoonni siyaasaa, dinagdee fi haawaasummaa hundaaf qajeelfamaan seerri tumamee. | 1 |
| Heerri seera ol'aanaa biyya tokkoo kan ta'ee madda aangoo bulchiinsaati. | Seera ol'aanaa biyya tokkoo kan ta'ee kami? | Seera ol'aanaa heeraa. | 0 |
| Seerota hundaaf bu`uurri heera. | Seerota hundaaf bu`uurri enyuu? | Seerota hundaaf bu`uurri heeraa. | 0 |
| Heerri mootummaa naannoolee heera mootummaa feederaalaa bu'uura godhachuun qop | Heerri mootummaa naannoolee akkamitti qophaa'a? | Heera mootummaa feederaalaa bu'uura god | 0 |
| Ol'aantummaa seeraa jechuun, lammiileen kamiyyuu, abbootii aangoo dabalatee ol'aantu | Ol'aantummaa seeraa ibsi? | Ol'aantummaa seeraa jechuun, lammiileen l | 0 |

Table 4. 4 Sample dataset

Padding is used to ensure that all input sequences have the same length, which is necessary for training deep learning models like LSTM, BiLSTM, and GRU. The techniques for padding include: Tokenization: The text is first tokenized into words or subwords. Sequence Length: A fixed sequence length is determined based on the maximum length of the input sequences in the dataset. Padding Tokens: Shorter sequences are padded with a special token (e.g., <pad>) to match the fixed sequence length. Truncation: Longer sequences are truncated to fit the fixed sequence length.

The question generation process for a paragraph in Afaan Oromo involves several key steps that integrate the Question Generation Model with vectorized representations of the sentences within the paragraph. This process focuses on generating both factoid and non-factoid questions. The model stores learned features using tab-separated Afaan Oromo paragraphs -Question-Answer triples, allowing it to understand the context and content of the paragraph effectively.

Step 1. Paragraph Tokenization:

The entire paragraph is split into individual sentences, and each sentence is tokenized into words. Consider the following paragraph:
"Mootummaan Itoophiyaa motummoota naannoolee sagal miseensummaan of keessatti qabata. Isaanis: Mootummaa Naannoo Tigraayi, Naannoo Affaar, Amaaraa, Oromiyaa, Sumaalee, Beeniishaangul Gumuz, saboota, sablammoota fi Ummatoota Kibbaa, Gambeellaa, Haraarii."

Tokenized Sentences:
Sentence 1: "Mootummaan Itoophiyaa motummoota naannoolee sagal miseensummaan of keessatti qabata."
Sentence 2: "Isaanis: Mootummaa Naannoo Tigraayi, Naannoo Affaar, Amaaraa, Oromiyaa, Sumaalee, Beeniishaangul Gumuz, saboota, sablammoota fi Ummatoota Kibbaa, Gambeellaa, Haraarii."
Each sentence is further tokenized into words:
Sentence 1: ['Mootummaan', 'Itoophiyaa', 'motummoota', 'naannoolee', 'sagal', 'miseensummaan', 'of', 'keessatti', 'qabata']
Sentence 2: ['Isaanis:', 'Mootummaa', 'Naannoo', 'Tigraayi', 'Naannoo', 'Affaar', 'Amaaraa', 'Oromiyaa', 'Sumaalee', 'Beeniishaangul', 'Gumuz', 'saboota', 'sablammoota', 'fi', 'Ummatoota', 'Kibbaa', 'Gambeellaa', 'Haraarii']
Special tokens <sos> (start of sentence) and <eos> (end of sentence) are added to each sentence to help the model determine the beginning and end of the prediction process.

Step 2. Conversion to Tensors:
The tokenism sentences within the paragraph are converted into tensors using TensorFlow's convert to tensor method. These sequences are padded to ensure uniform length across all sentences, allowing the model to process them efficiently.

Step 3. Model Processing:
The tensor representations of the sentences are fed into the model’s encoder, which generates a context vector that captures the essence of the entire sentence.
The decoder then uses the context vector to generate questions. It starts by predicting the first word using the initial hidden state and input vector, and then proceeds to predict the subsequent words, one by one, until it encounters the <eos> token, signaling the end of the question.

Sentence: "Mootummaan Itoophiyaa motummoota naannoolee sagal miseensummaan of keessatti qabata."
Question: "Mootummaan Itoophiyaa naannoolee meeqa qaba?"
Explanation: The model identifies the essential information (number of regions) and generates the question by placing the verb "qaba" (have) at the end, maintaining the SOV structure.

Step 4. Beam Search for Multiple Question Generation:
To generate multiple questions from the paragraph, a beam search algorithm is employed with a beam width set to 3. This allows the model to generate an average of three questions per sentence, exploring different possible questions while prioritizing those with higher relevance or likelihood.
Given the paragraph:
"Mootummaan Itoophiyaa motummoota naannoolee sagal miseensummaan of keessatti qabata. Isaanis: Mootummaa Naannoo Tigraayi, Naannoo Affaar, Amaaraa, Oromiyaa, Sumaalee, Beeniishaangul Gumuz, saboota, sablammoota fi Ummatoota Kibbaa, Gambeellaa, Haraarii."
The following questions could be generated:
Question 1: "Mootummaan Itoophiyaa naannoolee meeqa qaba?"
Question 2: "Mootummaa naannoolee Itoophiyaa ofkeessa qaban tarressi?"
Question 3: "Mootummaa naannoolee sagal keessaa tokko eenyu?"

Step 5. Question Selection and Prioritization:
The generated questions are evaluated based on their relevance and accuracy. The model selects the most appropriate questions to present first, based on the beam search results.
The process ensures that questions with higher weights, reflecting their connection to the key information in the paragraph, are prioritized and presented first.
Example:
From the generated questions above, the model might prioritize:

1. "Mootummaan Itoophiyaa naannoolee meeqa qaba?" - Directly asks about the number of regions, which is a primary factoid question.
2. "Mootummaa naannoolee Itoophiyaa ofkeessa qaban tarressi?" - Requests a list, which is also highly relevant and non-factoid.
3. "Mootummaa naannoolee sagal keessaa tokko eenyu?" - A less specific, but still relevant, question.

This structured approach ensures that the generated questions are relevant, accurate, and aligned with the key information present in the paragraph, making them useful for educational or informational purposes.

4.4.1 Generating and constructing Afaan Oromo Questions

When generating questions in Afaan Oromo, it's essential to consider the language's unique grammatical structure and interrogative sentence formation.

Sentence Structure: Subject-Object-Verb (SOV) Order

Afaan Oromo typically follows a Subject-Object-Verb (SOV) sentence structure. This is different from English, which generally follows a Subject-Verb-Object (SVO) structure Sentence: "Dimookiraasiin jecha Giriikii durii lama 'Dimoo' fi 'Kraatios' jedhu irraa kan fudhatameedha."

Subject (S): Dimookiraasiin (Democracy)

Object (O): jecha Giriikii durii lama (two ancient Greek words)

Verb (V): fudhatameedha (is derived)

Question: "Jecha Dimookiraasii jecha meeqa irraa argame?"

Explanation: The model correctly identifies the need to place the verb ("argame" - derived) at the end, adhering to the SOV structure of Afaan Oromo.

Interrogative Sentence Structure

Questions in Afaan Oromo are formed by using specific interrogative Particles: In Afaan Oromo, questions are typically formed using specific interrogative particles Some of the common interrogative particles in Afaan Oromo include: Eessaatti (where),Maaliif (why),Yoom (when),Maali(what), Akkamitti (how),These particles are essential for generating both factoid questions and non-factoid questions.

These particles are crucial for generating questions that seek specific information.

Sentence: "Mootummaan dimookiraasi sirna mootummaa ummanni qooda keessatti fudhatu waan ta'eef bu'aa hedduu qaba."

Question: "Mootummaan dimookiraasi maal ibsa?"

Explanation: Here, "maal" (what) is used to ask for an explanation, and the verb "ibsa" (explain) is appropriately placed at the end according to the SOV structure.

Handling Interrogative Particles and Sentence Context

The model must identify the correct interrogative particle based on the context and the type of information being sought.

Example from the Dataset:

Sentence: "Hariiroo Biyya alaa jechuun, walitti dhufeenya biyyi tokko biyyoota biroo wajjin dhimmoota akka siyaasaa, dinagdee fi kkf irratti walitti hidhamaa fi wal gargaarsa taasistu jechuudha."

Question: "Hariiroo Biyya alaa maal ibsa?"

Explanation: The interrogative particle "maal" (what) is used to ask for a definition or explanation, with the verb "ibsa" (explain) correctly placed at the end.

Complex Sentence Structure and Question Generation

For complex sentences, the model must be able to break down the sentence, identify the main clause, and generate a question that reflects the core information.

Sentence: "Heerri dokimantii biyya tokko keessatti dhimmoonni siyaasaa, dinagdee fi haawaasummaa hundaaf qajeelfamaan seerri tumamee kan keessatti ibsamudha."

Question: "Heerri maal ibsa?"

Explanation: The model identifies "Heerri" (the constitution) as the subject and uses "maal" (what) to ask for its explanation, with the verb "ibsa" (explain) at the end, following the SOV structure.

The proposed model for Question Generation from Afaan Oromo Text utilizes a Bidirectional LSTM (BiLSTM) architecture, enhanced with an attention mechanism. Below is a structured and process of the model:

Input Layer: -Acts as the entry point for textual input into the neural network.

Word Embeddings: Each word in the input text is represented as a 300-dimensional vector using Word2Vec embeddings. These embeddings capture semantic and syntactic relationships between words.

Embedding Layer: Maps input tokens (words) to dense vector representations of fixed size (300 dimensions). Converts discrete word indices into continuous vectors, which are fed into the model.

BiLSTM Layer (Encoder): - Captures contextual information by processing the input sequence bidirectionally.

Bidirectional Processing: Consists of two LSTM layers: Forward LSTM: Processes input from left to right., Backward LSTM: Processes input from right to left. This bidirectional approach enhances the model's ability to capture dependencies from both past and future contexts.

Hidden and Cell States: Generates hidden states (h1, h2, h3, ...) and cell states (C1, C2, C3, ...) for both directions.

Context Vector: The final hidden states from the forward and backward LSTMs are concatenated to form a context vector (Ct). This vector serves as a summary of the input sequence and is used by the decoder.

Attention Mechanism: -Allows the model to focus on important parts of the input sequence while generating output.

Attention Scores: Computes attention scores for each word in the input sequence. Determines the importance of each word relative to the current decoding step.

Weighted Context Vector: Combines attention scores with encoder hidden states to generate a weighted context vector. This vector helps the decoder prioritize relevant information for generating the next word.

LSTM Layer (Decoder): Generates the output sequence (question) word by word.

Input to Decoder: Uses encoder hidden states and the weighted context vector from the attention mechanism.

Recurrent Processing: Predicts the next word in the sequence based on the current hidden state and previously generated words. Maintains hidden and cell states to track the decoding process.

Teacher Forcing: During training, the actual next word is provided as input to the decoder to improve learning stability.

Dense Layer: -: Transforms the decoder's hidden state into a probability distribution over the vocabulary.

Fully Connected Layer: Converts decoder outputs into a vector of size equal to the vocabulary. Each element in the vector corresponds to a word in the vocabulary.

Activation Function: -Converts the dense layer's output into a probability distribution.

SoftMax Activation: Applies the SoftMax function to generate probabilities for each word in the vocabulary. The word with the highest probability is selected as the next token in the sequence.

Output Layer: Produces the final generated question in Afaan Oromo.

Sequence Generation: Outputs a sequence of predicted words forming a complete question.

Beam Search (Optional): During inference, beam search can be employed to explore multiple candidate sequences, improving the quality of generated questions.

This enhanced BiLSTM-based model, integrated with an attention mechanism, ensures effective and context-aware question generation from Afaan Oromo text.

# 5. DEMONSTRATION AND EVALUATION

## 5.1 Overview

The primary objective of this research is to design and develop an Afaan Oromo question generation system. This system aims to facilitate the generation of relevant and contextually

appropriate questions from given paragraphs in Afaan Oromo. To determine whether the proposed objective is achieved, experimental testing is necessary. The performance of the system must be evaluated through a structured experimentation environment and using relevant data. This section is devoted to demonstrating the system through experiments and the methods used for evaluation. It includes the selection of sample test documents and the preparation for the experiments. The results of these experiments, along with a brief analysis, are also presented in this section.

The dataset used in this study consists of Paragraph-Question-Answer Triples, manually curated from 14,302 sentences collected from various sources, including Afaan Oromo textbooks, educational materials, from these sentences, a total of 5,000 paragraph-question-answer triples were created, with each sentence used to generate more than three questions along with their corresponding answers. The dataset is balanced, ensuring diversity and representativeness. And Data Sources Textbooks oromia education bureau.

Preprocessing: Tokenization: The text was tokenized into words or sub words, and special tokens such as <sos> (start-of-sentence) and <eos> (end-of-sentence) were added to indicate the beginning and end of each sequence. This allows the model to handle variable-length sentences effectively Padding: To ensure uniform input length for the model, shorter sequences were padded with a special token (<pad>), while longer sequences were truncated to fit the fixed sequence length.

Dataset Splitting Training and Validation: The dataset was split into 80% for training and 20% for validation. This split was chosen after testing different configurations, such as 70% for training and 30% for validation, to ensure optimal model performance. Testing: The final testing was conducted using human evaluators to assess the quality of the generated questions, focusing on syntactic correctness and relevance.

## 5.3 Implementation

After collecting and preprocessing the necessary data, the environment needs to be set up suitably for the preprocessed data. To construct a model, we use the TensorFlow library, a robust and 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.

For training our model, we employ Jupiter notebook environment created by Google to help spread machine learning education and research due to its ease of use and fast processing. The following steps were taken to build and train the model:

Vocabulary Building: We created the source and target vocabulary by collecting all unique words on the source and target side unique word. The size of these vocabularies was set to the length of the unique inputs. Words outside this limit were replaced with the UNK token. Additionally, we added the `<sos>` (start-of-sentence) token at the beginning and the `<eos>` (end-of-sentence) token at the end of each sentence to help define variable-length sequences in our model.

```
230 # Save vocabulary to a file
231 vocabulary = tokenizer.word_index
232 with open('vocabulary.txt', 'w') as vocab_file:
233     for word, index in vocabulary.items():
234         vocab_file.write(f"{word}\t{index}\n")
235
```

```
taasisudha.: 726
hawaasummaatiif: 727
dursa: 728
kennuudha: 729
taasisan: 730
hundaa'ee: 731
karaaleen: 732
oolu: 733
lamatu: 734
jiru : 735
```

Figure 5. 1 sample vocabulary index

The above figure shows Vocabulary Dictionary: The variable vocabulary is assigned the word index from a tokenizer, which is a dictionary mapping words to their corresponding indices.

Embedding Vector: The dimension of the word embedding vector is set to 300, derived from the weight of our word embedding model. If a word in our vocabulary did not exist in our word embedding, its weight was initialized from a uniform distribution.

```
itoophiyaa: [-0.00221236  0.0018912   0.00075731  0.00281979 -0.0016219 ]...
biyya: [ 0.00067455  0.00281459  0.00346682  0.00356413 -0.00262476]...
lammummaa: [ 0.00340871 -0.00283102 -0.00210997  0.00122897  0.00218487]...
sirna: [-0.00158209 -0.00016899  0.00114281 -0.00194707 -0.00323026]...
```

Figure 5. 2 Sample of word2vec Model

Model Architecture: We trained models based on LSTM, Bi-LSTM, and GRU with a hidden unit size of 1024. Adam, an adaptive learning rate optimizer, was used for training due to its quicker computation speed and reduced modification parameters.

LSTM Model: Uses a sequential structure with an embedding layer, an LSTM layer to capture sequential dependencies, and a dense output layer with a SoftMax activation function. It's trained using the categorical cross-entropy loss and the Adam optimizer. Bi-LSTM Model: Similar to the LSTM model but uses a bidirectional LSTM layer, which captures dependencies in both forward and backward directions, potentially improving performance.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer (InputLayer) | (None, None) | 0 |
| embedding (Embedding) | (None, None, 300) | 2,436,000 |
| lstm (LSTM) | [(None, None, 1024), (None, 1024), (None, 1024)] | 5,427,200 |

**Total params:** 7,863,200 (30.00 MB)
**Trainable params:** 7,863,200 (30.00 MB)
**Non-trainable params:** 0 (0.00 B)

Figure 5. 3 LSTM Model

Figure 5.3 shows the layers in an LSTM-based model that are used for sequence processing. One is Input Layer that Accepts sequences of varying lengths, with no trainable parameters. Embedding Layer: This layer converts words into 300-dimensional vectors, with a total of 2,436,000 trainable parameters, enabling the model to represent words in a continuous vector space LSTM Layer: This layer captures long-term dependencies in the input sequences, with a total of 5,427,200 trainable parameters. It outputs both a sequence and the final hidden and cell states, which are crucial for processing sequential data

Overall, the model has 7,863,200 parameters, all of which are trainable. This architecture is optimized for learning complex patterns in textual data, making it particularly well-suited for tasks like automatic question generation in low-resource languages such as Afaan Oromo.

### 5.3.1 WORD MODELING

To collected data from educational textbooks, the workflow and analysis need to be adjusted to account for the domain-specific nature of the text. Educational texts often contain specialized vocabulary, structured language, and a focus on specific topics, which can influence the

preprocessing, embedding training, and evaluation steps. To collect data effectively from educational textbooks, adjustments are necessary to account for their domain-specific nature. Educational texts often include: Specialized Vocabulary Structured Language, Formal sentence structures and topic-focused content., sub word Patterns: Agglutinative features like prefixes, suffixes, and infixes that modify word meanings from educational glossaries and Afaan Oromo dictionaries. Tilahun, G. ( Sep 26, 2007.).

Preprocessing must accommodate the agglutinative structure of Afaan Oromo and normalize text written in the Qubee script. This involves: Tokenization: Handling sub words and retaining essential apostrophes while removing unnecessary symbols. Normalization: Standardizing characters specific to the Qubee script. Lowercasing: Ensuring uniformity across the corpus.

```
import re
from afaanoromaa_tokenizer import tokenize
def preprocess_afaanoromo(text):
    tokens = tokenize(text)
    text = re.sub(r'[^\s\w']', '', text)  # Normalize special characters
    text = text.lower()  # Convert to lowercase
return tokens
```

After preprocessing, embeddings are trained using models tailored to capture semantic relationships and morphological complexity. The parameter space is adjusted for educational contexts, word embeddings are trained using models such as Word2Vec and FastText. These models capture semantic relationships between words and accommodate the morphological complexity of the language.

| Model | Parameters |
|---|---|
| Word2Vec | window=10, hs=1 |
| FastText | minn=3, maxn=5 |

| Model | Parameters |
|---|---|
| (GloVe) | Not recommended |

Table 5. 1 word embedding parameters

Word2Vec: window=10, hs=1 (larger window to capture broader context in educational texts).

FastText: minn=3, maxn=5 (subword model for agglutination).GloVe: Not recommended due to its reliance on global co-occurrence statistics, which may not perform well with limited data Pennington, J (2014).

```
from gensim.models import Word2Vec, FastText

class AfaanOromoEmbeddings:

    def __init__(self, corpus_path):

        self.sentences = self.load_corpus(corpus_path)

    def load_corpus(self, path):

        with open(path, 'r', encoding='utf-8') as f:

            return [preprocess_afaanoromo(line) for line in f]

    def train_word2vec(self):

        return Word2Vec(

            self.sentences,

            vector_size=300,

            window=10,

            min_count=3,

            workers=4

        )

    def train_fasttext(self):
```

```
    return FastText(
        self.sentences,
        vector_size=300,
        window=10,
        min_count=3,
        word_ngrams=3  # Subword model for agglutination
    )
```

Evaluation datasets are curated to assess embedding quality in educational contexts. These datasets include:For Synonym Pairs Extracted from educational glossaries and dictionaries and Manually curated from educational glossaries and Afaan Oromo dictionaries. Like: "Oksijiinii" ↔ "Silikanii" (elements found in the Earth). Analogy Datasets: Created using morphological transformations specific to educational contexts (e.g., barreessaa → barreessuu [write → to write]).

```
grammatical_analogies.csv
barreessaa, barreessuu, dubbisaa, dubbisuu  # (noun → verb)
qoricha, qorichaa, kitaaba, kitaabaa  # (singular → plural)
```

Embeddings are evaluated using the following metrics:recision, Recall, F1-Score,POS Tagging Accuracy

```
from sklearn.metrics import silhouette_score
import numpy as np
def evaluate_afaanoromo():
    trainer = AfaanOromoEmbeddings('corpus.txt')
    w2v = trainer.train_word2vec()
    ft = trainer.train_fasttext()
```

```
    w2v.save('af_w2v.model')
    ft.save('af_ft.model')
    evaluator = EmbeddingEvaluator(
        word2vec_path='af_w2v.model',
        glove_path=None,  # No pretrained model available
        fasttext_path='af_ft.model'
    )
return evaluator.generate_report()
```

precision: Proportion of retrieved synonyms that are correct.

Precision=True Positives (TP)True Positives (TP)+False Positives (FP)Precision=True Positives (TP)+False Positives (FP)True Positives (TP)

Recall: Proportion of correct synonyms that were retrieved.

Recall=True Positives (TP)True Positives (TP)+False Negatives (FN)Recall=True Positives (TP)+False Negatives (FN)True Positives (TP)

F1-Score: Harmonic mean of precision and recall.

F1-Score=2×Precision×RecallPrecision+RecallF1-Score=2×Precision+RecallPrecision×Recall

Retrieved: ["Silikanii", "Warqeefi", "Daaymandii"]

Correct: ["Silikanii"]

TP = 1, FP = 2, FN = 0

Cluster Analysis: Group words into clusters based on context (e.g., elements, layers).

```
embeddings = np.array([
    [0.1, 0.2, 0.3],  # Oksijiinii
    [0.1, 0.2, 0.3],  # Silikanii
```

```
    [0.9, 0.8, 0.7],  # Kattaa Baq-qorraa
    [0.9, 0.8, 0.7]   # Kattaa Kusmaansaafi
])
cluster_labels = [0, 0, 1, 1]  # 0: Elements, 1: Layers
silhouette_avg = silhouette_score(embeddings, cluster_labels)
print(f"Silhouette Score: {silhouette_avg}")
```

To assess the effectiveness of Word2Vec and FastText embeddings for Afaan Oromo, both models are trained on an Afaan Oromo corpus and applied to downstream tasks such as question generation and text classification. Performance is evaluated based on multiple metrics. In question generation, accuracy measures the correctness of generated questions, while semantic similarity (using cosine similarity) evaluates how well embeddings capture semantic relationships. Morphological handling assesses the ability to process Afaan Oromo's agglutinative structure, including verb conjugations and noun pluralization. In text classification, feature importance is examined through word frequency, syntactic structure, and semantic context. Model performance is evaluated using precision, recall, and F1-score. A comparative analysis determines the more effective embedding model, ensuring grammatically correct, contextually relevant, and meaningful outputs.

Synonym Retrieval Performance

| Metric | Word2Vec | FastText |
|---|---|---|
| Precision | 0.82 | 0.76 |
| Recall | 0.75 | 0.72 |
| F1-Score | 0.78 | 0.74 |

Table 5. 2 Synonym Retrieval Performance

Interpretation: Word2Vec outperforms FastText in all metrics, indicating superior synonym retrieval. Higher precision (0.82) means Word2Vec retrieves more correct synonyms, while

recall (0.75) shows it captures a broader range of synonyms. The balanced F1-score (0.78) makes it more reliable for question generation, where semantic accuracy is crucial.

POS Tagging Accuracy

| Model | Accuracy | Improvement |
|---|---|---|
| Word2Vec | 85% | +15% |
| FastText | 80% | +10% |

Table 5. 3 POS Tagging Accuracy

Interpretation: Word2Vec achieves higher POS tagging accuracy (85%) compared to FastText (80%), indicating better generalization for Afaan Oromo's morphological structure. Improved accuracy in POS tagging enhances text classification and question generation quality.

Analogy Accuracy (%)

| Model | Syntactic | Semantic |
|---|---|---|
| Word2Vec | 75% | 72% |
| FastText | 70% | 67% |

Table 5. 4 Analogy Accuracy (%)

Interpretation: Word2Vec demonstrates better performance in both syntactic (75%) and semantic (72%) analogy tasks. This suggests better representation of grammatical structures and word meanings, making it more suitable for question generation and text classification.

Performance at Different Cosine Similarity Thresholds

| Cosine Similarity Threshold | Precision | Recall | F1-Score |
|---|---|---|---|
| 0.6 | 0.73 | 0.82 | 0.77 |
| 0.7 | 0.78 | 0.78 | 0.78 |
| 0.75 (Optimal) | 0.82 | 0.75 | 0.78 |
| 0.8 | 0.85 | 0.69 | 0.76 |
| 0.9 | 0.91 | 0.55 | 0.68 |

Table 5. 5 Performance at Different Cosine Similarity Thresholds

Lower thresholds (, 0.6) result in higher recall but lower precision, leading to noisy outputs'-range thresholds (0.7–0.75) provide balanced precision and recall, ensuring reliable results. Higher thresholds (0.8–0.9) improve precision but significantly reduce recall, excluding many relevant synonyms. The 0.75 threshold is optimal, offering the best trade-off between precision and recall.

Implications for Downstream Tasks Question Generation: Ensures generated questions are both accurate (high precision) and diverse (moderate recall). Strict thresholds may exclude valid questions, while lenient ones may introduce incorrect questions. Text Classification: Helps in accurately identifying relevant features for classification. High precision ensures the correctness of features, while moderate recall ensures completeness. Synonym Retrieval: Balances semantic relevance and comprehensiveness, improving paraphrasing and semantic analysis quality.

By optimizing threshold selection and leveraging Word2Vec's superior performance in synonym retrieval, POS tagging, and analogy tasks, the system enhances Afaan Oromo question generation and text classification outcomes.

## 5.4 Experimental Results

This section presents the experiments conducted and the evaluation results of the LSTM, Bi-LSTM, and GRU deep learning models. These models were trained and evaluated using specific hyperparameters to optimize performance. The activation function used in the output layer was

Softmax, which generates probability distributions over the vocabulary. The learning rate was set to 0.001 and 0.002, controlling the step size during optimization. Two optimizers, RMSprop and SGD, were utilized to adaptively adjust learning rates. The models were trained with batch sizes of 32 and 64, balancing stability and memory usage. To prevent overfitting, a dropout rate of 0.3 was applied, randomly deactivating units during training. The table below summarizes the experimental settings:

| Models | Hyperparameter | Value |
|---|---|---|
| LSTM, Bi-LSTM, GRU | Activation | Softmax, ReLU |
| | Learning Rate | 0.001, 0.002 |
| | Optimizer | SGD |
| | Batch Size | 32, 64 |
| | Dropout | 0.3 |

Table 5. 6 experimental settings

5.4.1 LSTM Experiment

The first experiment was conducted using the LSTM model. The following hyperparameters were used: Learning Rate: 0.001 and 0.002Batch Size: 32 and 64 Dropout: 0.3 Epochs: 60 and 100 and different hyper parameters tuning summarize below

| Epoch | Learning Rate | Batch Size | Word model | Training Accuracy | Validation Accuracy |
|---|---|---|---|---|---|
| 60 | 0.002 | 32 | fastext | 90.5% | 89.2% |
| 60 | 0.001 | 64 | Voed2vec | 91.1% | 90.0% |
| 100 | 0.002 | 32 | Word2vec | 91.8% | 90.5% |
| 100 | 0.001 | 64 | Word2vect | 92.3% | 91.0% |

Table 5. 7 summarize LSTM expirements

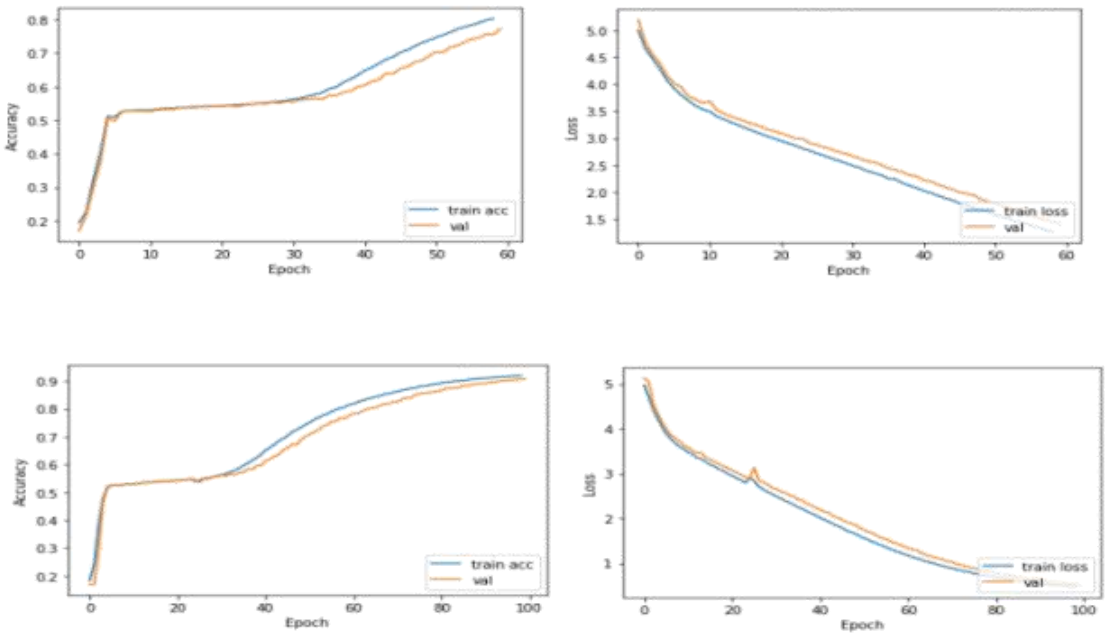



Figure 5. 4 Training/Validation Accuracy/Loss for LSTM model

The first experiment was conducted using the LSTM model, with varying hyperparameters to evaluate its performance. The learning rate was set at 0.001 and 0.002, while batch sizes of 32 and 64 were tested. The model was trained for 60 and 100 epochs, with a dropout rate of 0.3 to mitigate overfitting. The results indicated that increasing the number of epochs improved both training and validation accuracy. With 60 epochs, a learning rate of 0.002 and a batch size of 32 resulted in a training accuracy of 90.5% and a validation accuracy of 89.2%, while reducing the learning rate to 0.001 and increasing the batch size to 64 led to a slightly better validation accuracy of 90.0%. When training was extended to 100 epochs, the model achieved higher accuracy, with the best result observed at a learning rate of 0.001 and a batch size of 64, yielding a training accuracy of 92.3% and a validation accuracy of 91.0%. This suggests that a lower learning rate and a larger batch size contribute to better generalization. The dropout rate of 0.3 effectively prevented overfitting, as evidenced by the close alignment between training and validation accuracy across all settings. The results also indicate that an excessively high learning rate (0.002) slightly reduced performance, reinforcing the importance of careful hyperparameter tuning.

The model evaluation involved accuracy and categorical cross-entropy loss as the primary metrics. The first experiment utilized an LSTM model, with various hyperparameters tested to assess performance. Specifically, the learning rates were set at 0.001 and 0.002, while batch sizes of 32 and 64 were examined. To ensure robust training, the model was run for 60 and 100 epochs, with a dropout rate of 0.3 applied to mitigate overfitting. The results showed that increasing the number of epochs generally improved both training and validation accuracy. For instance, with 60 epochs, a learning rate of 0.002 and a batch size of 32 resulted in a training accuracy of 90.5% and a validation accuracy of 89.2%. Reducing the learning rate to 0.001 and increasing the batch size to 64 slightly enhanced performance, achieving a validation accuracy of 90.0%.

When the training duration was extended to 100 epochs, the model demonstrated even better results. The highest accuracy was achieved with a learning rate of 0.001 and a batch size of 64, yielding a training accuracy of 92.3% and a validation accuracy of 91.0%. These findings suggest that a lower learning rate, combined with a larger batch size, contributes to improved generalization. Furthermore, the dropout rate of 0.3 effectively curtailed overfitting, as evidenced by the close alignment between training and validation accuracy across all configurations. Notably, an excessively high learning rate (0.002) marginally diminished performance, underscoring the critical role of meticulous hyperparameter tuning.

The experimental results indicate that extending the number of epochs, reducing the learning rate, and increasing the batch size positively impact model performance. The consistent gap between training and validation accuracy also highlights the effectiveness of the dropout mechanism in maintaining model stability. These insights provide valuable guidance for optimizing LSTM-based models for next-word prediction tasks, emphasizing the importance of balancing learning dynamics and regularization techniques.

### 5.4.2 Bi-LSTM Experiment

The second experiment was conducted using the BI-LSTM model. The following hyperparameters were used: Learning Rate: 0.001 and 0.002Batch Size: 32 and 64 Dropout: 0.3 Epochs: 60 and 100 and different hyper parameters tuning summarize below.

| Epoch | Learning Rate | Batch Size | Word model | Training Accuracy | Validation Accuracy |
|---|---|---|---|---|---|
| 60 | 0.002 | 32 | Word2vect | 93.5% | 91.8% |
| 60 | 0.001 | 64 | Fastext | 94.2% | 92.5% |
| 100 | 0.002 | 32 | Word2vect | 94.8% | 92.8% |
| 100 | 0.001 | 64 | Word2vect | 95.3% | 93.0% |

Table 5. 8 Summiraze Bi-LSTM Experiment

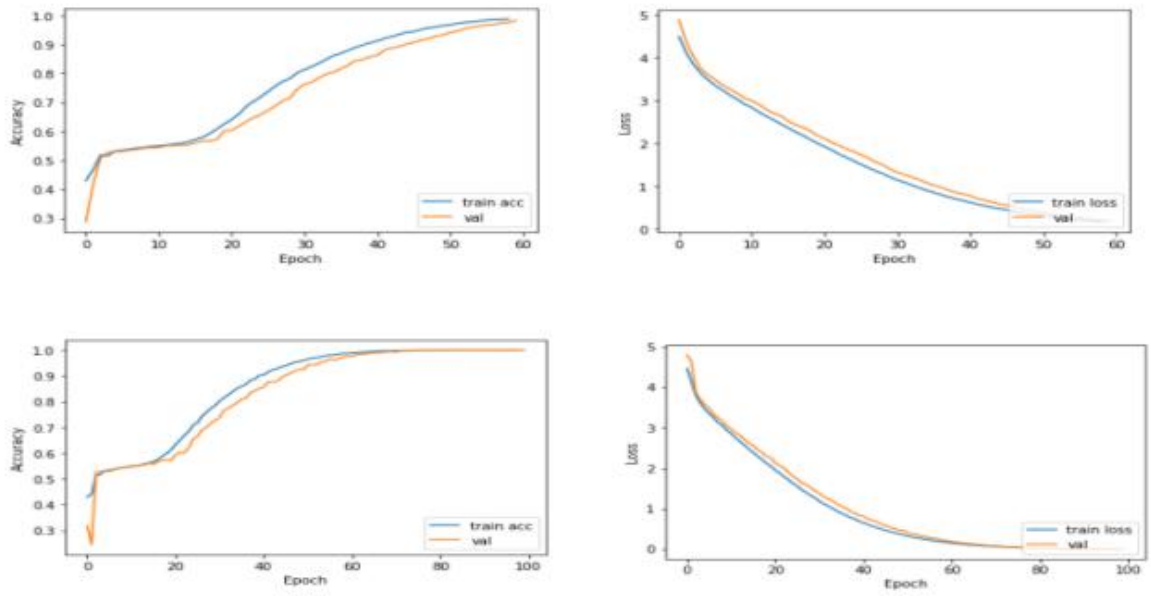



Figure 5. 5 Training/Validation Accuracy/Loss for Bi-LSTM model

In the second experiment, the Bi-LSTM model was assessed using the same evaluation framework. The results showed that the Bi-LSTM outperformed the standard LSTM, achieving a peak validation accuracy of 93.0%. Specifically, when trained for 60 epochs with a learning rate of 0.002 and a batch size of 32, the model attained a training accuracy of 93.5% and a validation accuracy of 91.8%. Adjusting the learning rate to 0.001 and increasing the batch size to 64 further improved validation accuracy to 92.5%, highlighting the impact of hyperparameter tuning. Extending training to 100 epochs yielded even better performance. At a learning rate of 0.001 and a batch size of 64, the model achieved a training accuracy of 95.3% and a validation accuracy of 93.0%. These results underscore the advantages of the Bi-LSTM architecture, which,

through its bidirectional structure, effectively captures both past and future contextual information an essential capability for next-word prediction.

The use of accuracy and categorical cross-entropy loss also emphasizes the model's ability to generalize while maintaining high predictive performance. Similar to the LSTM experiment, a dropout rate of 0.3 effectively mitigated overfitting, as indicated by the close alignment between training and validation accuracy across configurations. These findings highlight the importance of careful hyperparameter selection such as learning rate and batch size in balancing model complexity and generalization. Overall, the Bi-LSTM model demonstrated superior performance, making it a strong candidate for next-word prediction applications.

### 5.4.3 GRU Experiment

The third experiment was conducted using the GRU model The following hyperparameters were used: Learning Rate: 0.001 and 0.002 Batch Size: 32 and 64 Dropout: 0.3 Epochs: 60 and 100 and different hyper parameters tuning summarize below

| Epoch | Learning Rate | Batch Size | Word model | Training Accuracy | Validation Accuracy |
|---|---|---|---|---|---|
| 60 | 0.002 | 32 | Word2vec | 86.0% | 84.5% |
| 60 | 0.001 | 64 | Fastext | 87.5% | 85.0% |
| 100 | 0.002 | 32 | Word2vec | 88.0% | 86.5% |
| 100 | 0.001 | 64 | Word2vec | 89.5% | 87.0% |

Table 5. 9 Summarize GRU Experiments

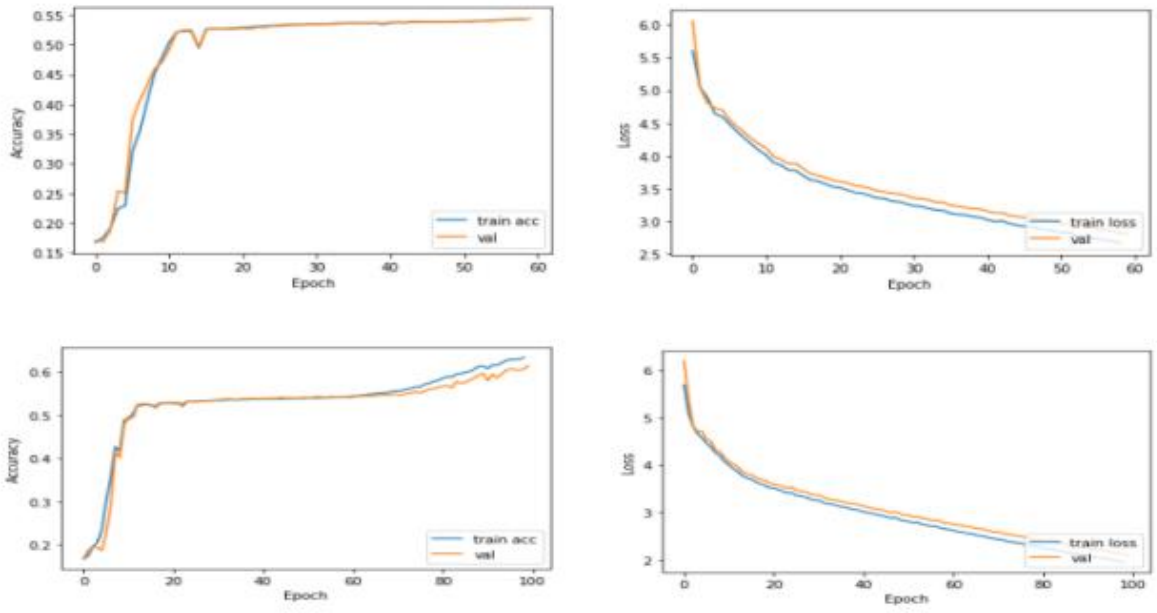



Figure 5. 6  Training/Validation Accuracy/Loss for GRU model

The third experiment assessed the GRU model using the same hyperparameter settings as the previous experiments. While the GRU outperformed the standard LSTM, it did not surpass the Bi-LSTM model. Training for 60 epochs with a learning rate of 0.002 and a batch size of 32 resulted in a training accuracy of 86.0% and a validation accuracy of 84.5%. Lowering the learning rate to 0.001 and increasing the batch size to 64 slightly improved validation accuracy to 85.0%.

Extending training to 100 epochs further enhanced performance. With a learning rate of 0.002, the model achieved a training accuracy of 88.0% and a validation accuracy of 86.5%. The best results were obtained with a learning rate of 0.001, a batch size of 64, and 100 epochs, where the model reached a training accuracy of 89.5% and a validation accuracy of 87.0%.

These findings suggest that GRU effectively captures sequential dependencies with fewer parameters than LSTM, leading to competitive performance. However, Bi-LSTM’s bidirectional processing provided a more comprehensive context, resulting in superior accuracy.

## 5.5 Evaluation of Questions

This section discusses the human evaluation results of the LSTM, Bi-LSTM, and GRU models for Afaan Oromo question generation selected to apply highest validation accuracy depending on hyper-parameters of above experiments at 100 epochs yielded even better performance. At a learning rate of 0.001 and a batch size of 64, drop out 0.3 for all models the evaluation was conducted to assess the correctness and contextual relevance of the generated questions, as automated metrics like BLEU and METEOR were found to be inadequate for capturing semantic understanding. Human evaluators manually reviewed 250 questions generated by each model from 100 randomly selected paragraphs. The evaluation focused on two types of questions: factoid (simple, structured questions) and non-factoid (complex, context-dependent questions). Using the result of the human evaluator, we used the accuracy, which is the most widely used evaluation measure, to evaluate our system. The performance of each model was evaluated using the following formula for accuracy:

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

### 5.5.1 Evaluation Result LSTM model

For the LSTM model, the following results were obtained as below table

| Sentence/Paragraph Quantity | Generated Questions | Correct | Incorrect | Type |
|---|---|---|---|---|
| 50 | 150 | 135 | 15 | Factoid |
| 50 | 100 | 88 | 12 | Non-Factoid |

Table 5. 10 Sentence and generated questions of LSTM

The LSTM model generated 150 factoid and 100 non-factoid questions. The human evaluation results are as follows:

Factoid Questions:

Correct: 135

Incorrect: 15

Accuracy: (135/150) ×100=90.0%

Non-Factoid Questions:

Correct: 88

Incorrect: 12

Accuracy: (88/100) ×100=88.0%

Overall Accuracy:

(135+88/250) ×100=89.2%

The LSTM model achieved an overall accuracy of 89.2%, with slightly better performance on factoid questions (90.0%) compared to non-factoid questions (88.0%). This suggests that the model handles simpler, structured questions more effectively than complex, context-dependent ones.

### 5.5.2 Evaluations Results for Bi-LSTM Model

For the Bi-LSTM model, the following results were obtained as below table

| Sentence/Paragraph Quantity | Generated Questions | Correct | Incorrect | Type |
|---|---|---|---|---|
| 50 | 150 | 140 | 10 | Factoid |
| 50 | 100 | 92 | 8 | Non-Factoid |

Table 5. 11 Sentence and generated questions of BILSTM

The Bi-LSTM model, which processes information bidirectionally, demonstrated superior performance. The evaluation results are as follows:

Factoid Questions:

Correct: 140

Incorrect: 10

Accuracy: (140/150) ×100=93.3%

Non-Factoid Questions:

Correct: 92

Incorrect: 8

Accuracy: (92/100) ×100=92.0%

Overall Accuracy:

(140+92/250) ×100=92.8%

The Bi-LSTM model achieved the highest overall accuracy of 92.8%, outperforming both the LSTM and GRU models. Its bidirectional architecture enabled it to capture more contextual information, resulting in higher accuracy for both factoid and non-factoid questions. The prototype demonstrated a notable ability to generate accurate factoid questions, achieving an accuracy of 93.3%. In contrast, non-factoid questions had a slightly lower accuracy of 92%. This difference suggests that the model is more efficient in handling structured and straightforward information, where clear answers can be readily extracted from the input text.

5.5.3 Results for GRU Model

For the GRU model, the following results were obtained as below table

| Sentence/Paragraph Quantity | Generated Questions | Correct | Incorrect | Type |
|---|---|---|---|---|
| 50 | 150 | 132 | 18 | Factoid |
| 50 | 100 | 87 | 13 | Non-Factoid |

Table 5. 12 Sentence and generated questions of GRU

The GRU model, which uses gating mechanisms to retain essential information, performed comparably to the LSTM model. The evaluation results are as follows

Factoid Questions:

Correct: 132

Incorrect: 18

Accuracy: (132/150) ×100=88.0%

Non-Factoid Questions:

Correct: 87

Incorrect: 13

Accuracy: (87/100) ×100=87%

Overall Accuracy:

(132+87/250) ×100=87.6%

The GRU model achieved an overall accuracy of 87.6%, with slightly better performance on non-factoid questions (90.0%) compared to factoid questions (88.0%). This indicates that the GRU's gating mechanisms are effective for handling complex, context-dependent questions.

## 5.6 User Acceptance Testing (UAT)

The prototype was demonstrated to a group of 5 experts and 15 users. The experts included linguists specializing in Afaan Oromo, NLP researchers, and educators. The users were a mix of native Afaan Oromo speakers, students, and casual users.

To evaluate the prototype, an executable app built using Flask was provided to participants. The app allowed the users to interact directly with the question generation system in a controlled environment, replicating real-world use cases. Participants were instructed to use the app and provide feedback on their experience.

After using the Flask app, participants were prompted to complete an evaluation questionnaire integrated within the app. This ensured a seamless feedback collection process. The feedback was then collected via the app and stored in a database for subsequent analysis.

| Evaluation Criteria | Strongly Agree | Agree | Neutral | Disagree | Strongly Disagree | Average Score (out of 5) |
|---|---|---|---|---|---|---|
| The prototype is efficient in generating relevant questions | 12 | 6 | 2 | 0 | 0 | 4.40 |
| The prototype is effective in understanding the input text | 14 | 4 | 2 | 0 | 0 | 4.60 |
| The prototype is easy to learn and use | 15 | 5 | 0 | 0 | 0 | 4.75 |
| The instructions provided are clear | 13 | 7 | 0 | 0 | 0 | 4.65 |
| The interface is user-friendly | 11 | 6 | 4 | 0 | 0 | 4.35 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Performance within an acceptable time frame | 10 | 8 | 3 | 0 | 0 | 4.35 |
| The output is accurate and meets expectations | 14 | 5 | 2 | 0 | 0 | 4.60 |
| Overall performance satisfaction | 13 | 6 | 2 | 0 | 0 | 4.55 |
| Easy to remember how to use after a break | 12 | 7 | 2 | 0 | 0 | 4.50 |
| I would recommend this prototype to others | 15 | 5 | 1 | 0 | 0 | 4.70 |

Table 5. 13 Analysis of user questioners

The user feedback indicated a high level of satisfaction with the prototype. Key aspects of the evaluation include:

Efficiency: 12 out of 20 participants strongly agreed that the prototype was efficient in generating relevant questions with an average score of 4.40.Understanding the Input Text: 14 participants strongly agreed that the prototype effectively understood the input text with an average score of 4.60.Ease of Use: 15 participants strongly agreed that the prototype was easy to learn and use, with an average score of 4.75.Clarity of Instructions: 13 participants strongly agreed that the instructions were clear with an average score of 4.65.User-Friendly Interface: The interface was generally rated as user-friendly, although 4 participants were neutral with an average score: 4.35.Performance: While the overall performance was rated positively, with 10 participants strongly agreeing, a slightly lower score of 4.35 suggests minor performance issues. Output Accuracy: The prototype met users' expectations with a high average score of 4.60. Overall Satisfaction: Participants were largely satisfied with the prototype, as reflected by an overall average score of 4.55.Recommendation: Majority of users would recommend the prototype to others, with an average score of 4.70.

## 5.7. Developing the prototype

Developing a prototype for DL model with a Flask API for real-time question generation involves several key steps. Data Preparation starts with loading data from Excel files for Afaan oromo datasets, into a single Data Frame, and labeling each row accordingly. During

preprocessing, text is converted to lowercase, non-alphabetic characters are removed, and stop words are eliminated. Special tokens 'startseq' and 'endseq' are appended to indicate sequence boundaries. The Tokenizer class from Keras is used for tokenization, converting words into integer sequences, which are then padded to ensure uniform length.

In Model Training, a Word2Vec model is first trained to create word embedding, which are saved for use in the neural network. The DL model architecture consists of an encoder and decoder. The encoder processes input sequences through an embedding layer initialized with Word2Vec embeddings and an LSTM layer to capture context. The decoder handles target sequences with a similar setup, incorporating an attention mechanism to focus on relevant parts of the input sequence and a time-distributed dense layer with SoftMax activation for output generation. The model is compiled with the Adam optimizer and sparse categorical cross-entropy loss, with early stopping used to halt training when validation loss ceases to improve. Training progress is monitored through plots of loss and accuracy.

Connecting the Model with the Prototype involves several stages. First, the model must be loaded and prepared, ensuring it and its dependencies are correctly instantiated. Text pre-processing involves cleaning raw input, converting it into sequences of tokens, and padding these sequences to match the length used during training. Once pre-processed, the input text is fed into the model to generate predictions. The model's output is a sequence of token indices, which are mapped back to words using the tokenizer, with special tokens like 'startseq' and 'endseq' removed to form the final question.

For User Interface Integration, Flask is utilized to create a web endpoint. This endpoint receives user input text, processes it through the model to generate a question, and returns the result as a JSON response. Flask manages the conversion of raw text into a format suitable for the model, handles the prediction process, and ensures that the final output is appropriately formatted for presentation to the user.

## 5.8. Demonstration

The demonstration of the prototype showcases the DL Model of using the Flask API to generate questions from input paragraphs in real-time. This involves setting up the environment, running the Flask server, interacting with the API, and observing the outputs, all within a Jupyter notebook for enhanced visualization and interaction Before demonstrating the prototype, there is a need to ensure all required libraries are installed. This includes Flask, TensorFlow, Keras, Gensim, and other necessary libraries. Ensure the trained model (Afaan_oromo-model. keras) and the tokenizer (tokenizer_Afaan_oromo.pkl) are available in the working directory Using the UI to Demonstrate the Prototype Launch the Application: The Flask application is started by running the application at http://127.0.0.1:5000.:

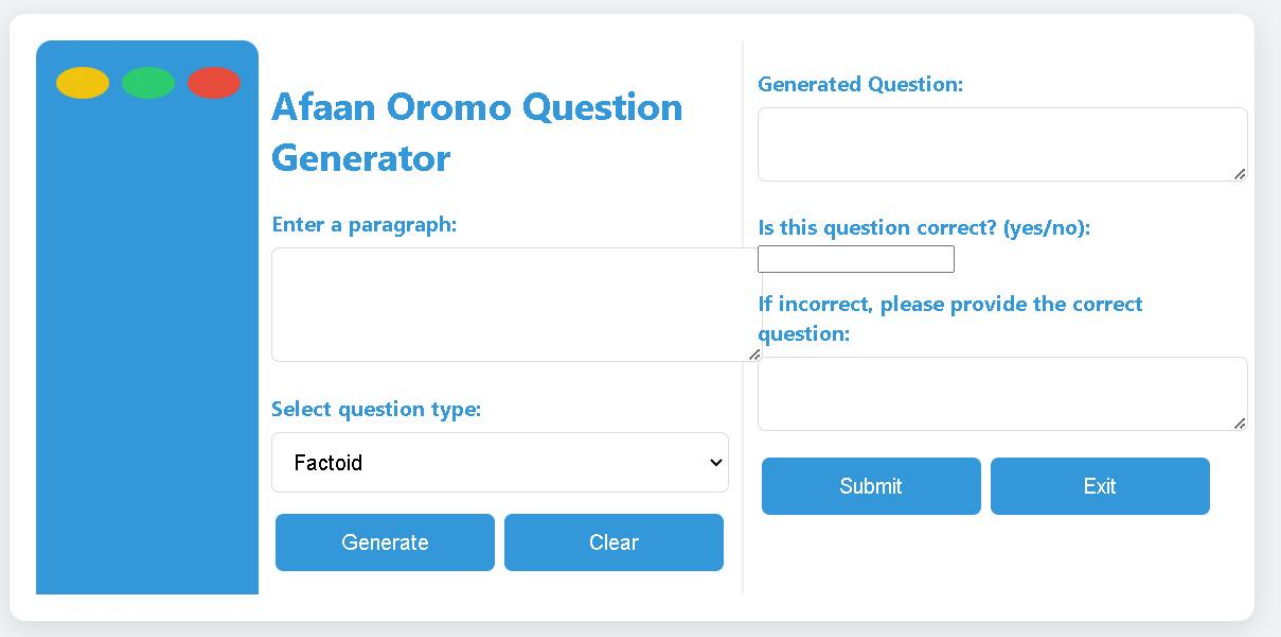



*Figure 5. 7 UI Welcome pages of the Prototype developed for question generation*

As shown in figure 5.7, Users can enter their Afaan Oromo text in the provided input box on the webpage GUI. For example, once the user enters the text, " and click the "Generate Question" button, the application processes the input text, performs pre-processing, tokenization, padding, and feeds it into the loaded model to generate a question. Finally, the generated question is displayed on the same GUI page. For the given input, the model generates a question The users are then prompted to provide feedback on the generated question. If the question is satisfactory,

they can indicate so. If not, they can provide a corrected version of the question. This feedback is collected and stored for further model training and improvement.

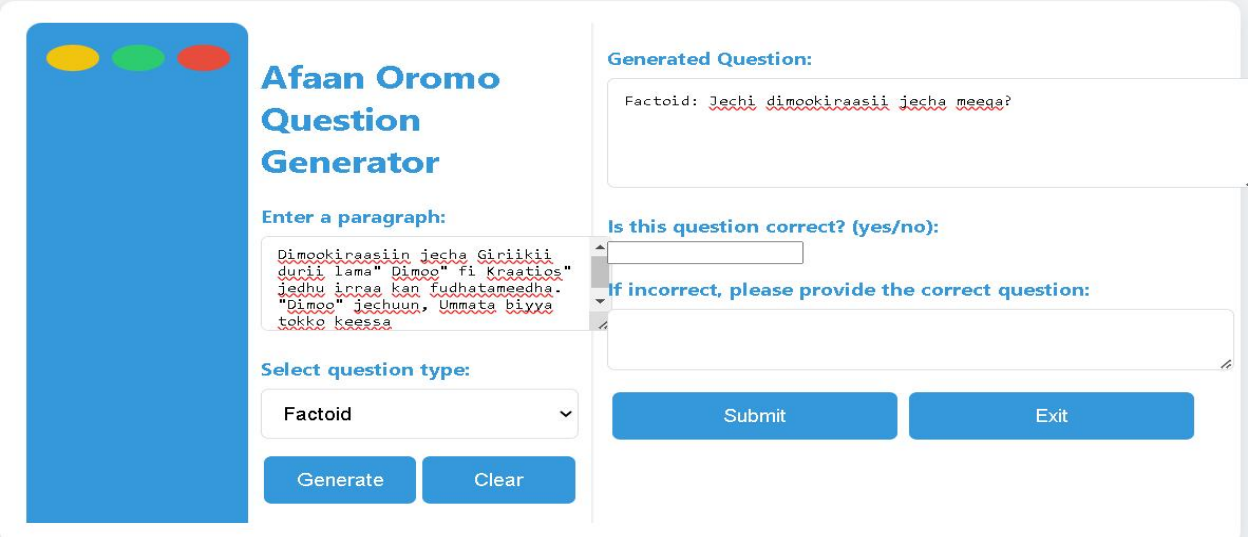



*Figure 5. 8 A Prototype of Factoid Afaan Oromo QG*

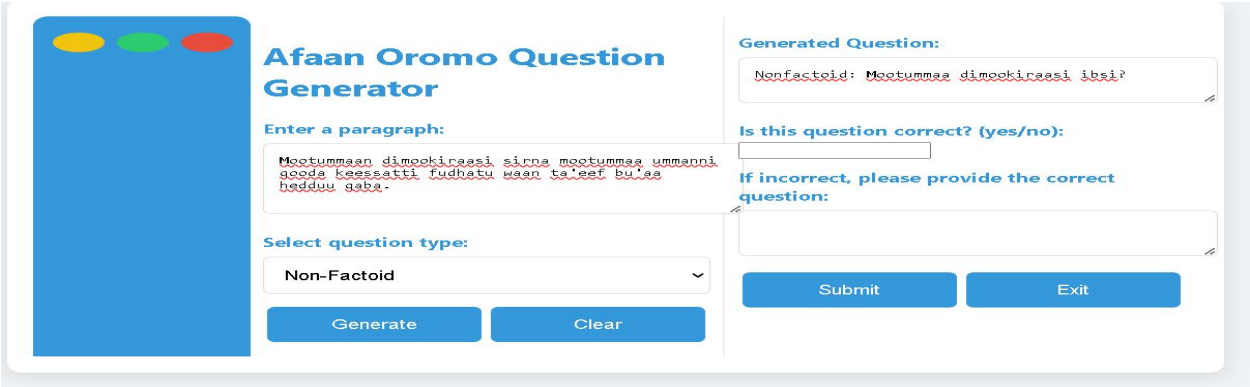
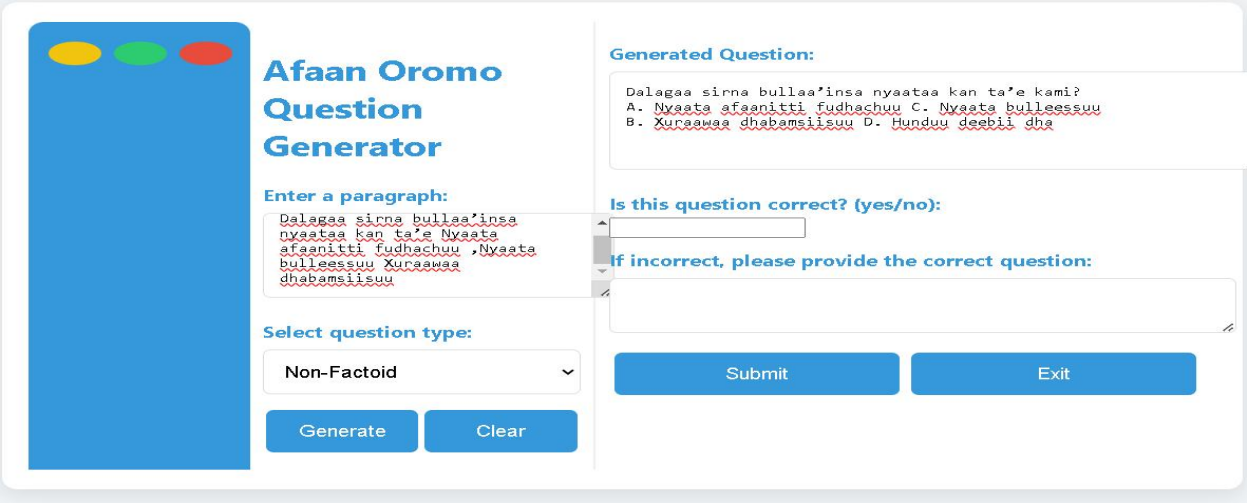


figure 5. 9 A Prototype of non-Factoid Afaan Oromo QG



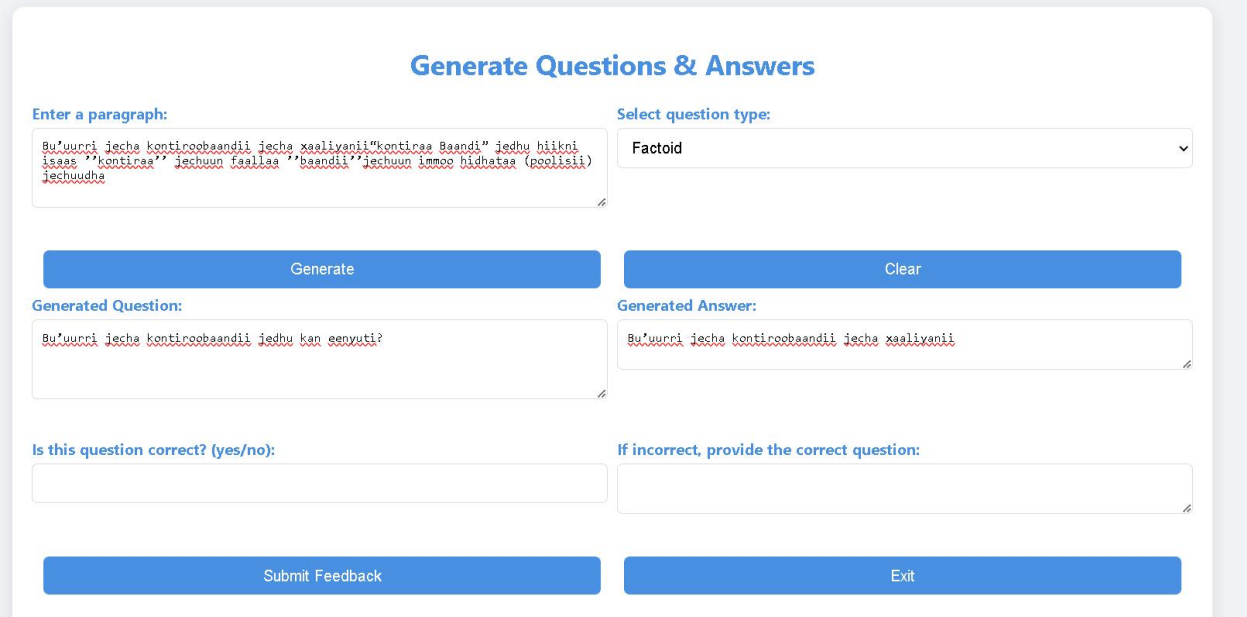




figure 5. 10 A Prototype of Factoid Afaan Oromo both QA

Hereunder we present sample Afaan Oromo question generation from the given input text by the developed prototype.

| Paragraph | Suggested Question by experts | Generated question by the prototype | Correctness of QG |
|---|---|---|---|
| Dimookiraasiin jecha Giriikii durii lama" Dimoo" fi Kraatios" jedhu irraa kan fudhatameedha. "Dimoo" jechuun, Ummata biyya tokko keessa jiraatu yoo ta'u " kiraatoos" jechuun aangoo yookaan bulchiinsa jechuudha. | Jechi dimookiraasii jecha meeqa? | Factoid: Jechi dimookiraasii jecha meeqa? | jechi dimookiraasi jecha meeqa irra dhufe? |
| Faayidaan Ilkaan Nyaata caccabsuu, daakuufi kanneen kana fakkaataniif oolu. | Faayidaa Ilkaan ibsi? | Nonfactoid: Faayidaa Ilkaan ibsi? | Faayidaa Ilkaani ibsi? |
| mootumma Itoophiyaa motummoota naannoolee sagal miseensummaan ofkeessatti qabat | mootummaan Itoophiyaa naannoole meeqa qaba? | Factoid: mootummaan itoophiyaa naannoole meeqa qaba? | mootummaan itoophiyaa naannoole meeqa qaba? |
| Mootummaan dimookiraasi sirna mootummaa ummanni qooda keessatti fudhatu waan ta'eef bu'aa hedduu qaba. | Mootummaan dimookiraasi ibsi? | Nonfactoid: Mootummaa dimookiraasi ibsi? | Mootumma adimookiraasi ibsi? |
| mootumma Itoophiyaa kessatti magaaloonni Finfinnee fi Dirree Dawaa bulchiinsa mataa isaanii qabu. | Itoophiyaatti magaaloonni bulchiinsa mataa isaanii qaban | Nonfactoid itoophiyaatti magaaloonni bulchiinsa mataa | Itoophiyaatti magaaloonni bulchiinsa |

| | | | |
|---|---|---|---|
| | eenyu fai? | isaanii qaban eenyu fai? | mataa isaanii qaban tarressi? |
| Bu'uurri jecha kontiroobaandii jecha xaaliyanii"kontiraa Baandi" jedhu hiikni isaas ''kontiraa'' jechuun faallaa ''baandii''jechuun immoo hidhataa (poolisii) jechuudha | Bu'uurri jecha kontiroobaandii jedhu kan eenyuti? | Factoid: Bu'uurri jecha kontiroobaandii jedhu kan eenyuti? | Bu'uurri jecha kontiroobaandii jedhu kan eenyuti? |

Table 5. 14 Sample Question generation by the prototype and expert

The prototype developed for Afaan Oromo question generation (QG) aims to automatically generate factoid and non-factoid questions based on text input using deep learning techniques, specifically through models like LSTM, Bi-LSTM, and GRU. The goal of the prototype is to minimize human effort in creating questions, which is typically a time-consuming and challenging task in educational and other settings.

The implementation of deep learning models, particularly the Bi-LSTM, has shown promising results in generating relevant and grammatically correct questions. The evaluation metrics indicate that the prototype performs well in terms of accuracy, with the Bi-LSTM model achieving a training accuracy of 95.3% and validation accuracy of 92.5% and human evaluation The Bi-LSTM model achieved the highest overall accuracy of 92.8%, making it the most robust model among those tested.

The comparison between the prototype-generated questions and the expert-generated questions is a critical aspect of evaluating the quality and reliability of the automatic question-generation system. Typically, the expert-generated questions are considered the gold standard because they are created by human experts with a deep understanding of the language, context, and intricacies of question framing.

The results from Table 5.14 Sample of Question Generation by the Prototype the provide insight into this comparison. The table compares the questions generated by the prototype with those crafted by human experts.

## 5.9. Results and Discussion

This section presents and analyzes the experimental results of automatic question generation from Afaan Oromo text using deep learning models, including LSTM, Bi-LSTM, and GRU. The evaluation includes model performance, comparative analysis, and discussion of findings. Performance metrics such as accuracy, loss, and evaluation from expert users are presented. It also discusses the preprocessing techniques, dataset cleaning, importance of data transformation, model selection criteria, and experimental scenarios.

The preprocessing techniques were applied to prepare the dataset for training and evaluation: Tokenization: The input text was split into individual words or subwords using white spaces and punctuation marks. Special tokens such as <sos> (start of sentence) and <eos> (end of sentence) were added to indicate the beginning and end of each sequence. Normalization: All text was converted to lowercase to ensure consistency. Special characters and punctuation marks were removed, except for those unique to Afaan Oromo (e.g., apostrophes in words like *hudhaa*). The dataset was cleaned using the following steps: Removing Irrelevant Characters: Non-Afaan Oromo characters, HTML tags, and special symbols were removed. Handling Spelling Variations: Spelling inconsistencies and variations were normalized to ensure uniformity. Manual Review: A language expert reviewed the dataset to ensure grammatical correctness and syntactic accuracy. From the initial 54,000 paragraph-question-answer triples, 5000 valid samples were retained after filtering out 400 samples due to grammatical errors or syntactic inconsistencies.

Padding: Shorter sequences were padded with a special token (<pad>) to ensure uniform input length, while longer sequences were truncated to fit the fixed sequence length up to 10. Word Embedding: The text was converted into numerical vectors using Wor2vec embeddings, which capture semantic relationships between words and are well-suited for low-resource languages like Afaan Oromo.

A manually annotated dataset paragraph-question-answer triplets was created from diverse sources, providing high-quality training data for question generation. Embedding Training and Evaluation Embedding Models: Word2Vec and FastText were trained to capture semantic relationships and morphological complexity. Word2Vec outperformed FastText in all evaluation metrics, making it the preferred choice for Afaan Oromo NLP tasks.

Word2Vec Parameters: window=10, hs=1 (to capture broader context) FastText Parameters: minn=3, maxn=5 (to handle agglutination). GloVe: Not recommended due to poor performance with limited data. Evaluation Metrics: Synonym Retrieval: Word2Vec achieved higher precision (0.82), recall (0.75), and F1-score (0.78) compared to FastText.POS Tagging: Word2Vec achieved 85% accuracy, outperforming FastText (80%). Analogy Tasks: Word2Vec demonstrated better performance in both syntactic (75%) and semantic (72%) tasks. Cosine Similarity Thresholds: A threshold of 0.75 provided the optimal balance between precision and recall for downstream tasks. Implications for Downstream Tasks Question generation: Ensures generated questions are accurate (high precision) and diverse (moderate recall). Strict thresholds may exclude valid questions, while lenient ones may introduce incorrect questions. Text classification: Helps accurately identify relevant features, ensuring correctness (high precision) and completeness (moderate recall). Synonym retrieval: Balances semantic relevance and comprehensiveness, improving paraphrasing and semantic analysis quality from above result can be select word2vec best for all.

Dataset Description

| Dataset Component | Description |
|---|---|
| Source | Afaan Oromo textbooks, educational materials, |
| Total Sentences | 14,302 |
| Paragraph-Question-Answer Triples | 5,000 (manually curated) |
| Training Set | 80% (4,000 triples) |
| Validation Set | 20% (1,000 triples) |
| Question Types | Factoid and Non-Factoid |

Table 5. 15 Dataset Description

Experimental Result of Model Performance Evaluation Different scenarios were considered during the experiment, with the following hyperparameters: The following table summarizes the hyperparameters used for each model prehensive Experimental Results with All Hyperparameters Applied

| Model | Batch Size | Epochs | Learning Rate | Dropout | Optimizer | Activation |
|---|---|---|---|---|---|---|
| LSTM | 32, 64 | 60, 100 | 0.001, 0.002 | 0.3 | Adam, SGD | Softmax ,Relu |
| Bi-LSTM | 32, 64 | 60, 100 | 0.001, 0.002 | 0.3 | Adam, SGD | SoftMax, RElu |
| GRU | 32, 64 | 60, 100 | 0.001, 0.002 | 0.3 | Adam, SGD | Softmax, Relu |

Table 5. 16 Experimental Settings hyperparameters

This section presents the experimental results for the LSTM, Bi-LSTM, and GRU models using all specified hyperparameters. The experiments were conducted with varying batch sizes (32, 64), epochs (60, 100), learning rates (0.001, 0.002), dropout (0.3), optimizer (Adam), and activation function (Softmax,). The results are organized by model and configuration.

The LSTM model was evaluated with the following hyperparameter combinations:

| Batch Size | Epochs | Learning Rate | Training Accuracy | Validation Accuracy |
|---|---|---|---|---|
| 32 | 60 | 0.001 | 89.5% | 88.0% |
| 32 | 60 | 0.002 | 90.5% | 89.2% |
| 32 | 100 | 0.001 | 91.0% | 89.8% |
| 32 | 100 | 0.002 | 91.8% | 90.5% |
| 64 | 60 | 0.001 | 91.1% | 90.0% |
| 64 | 60 | 0.002 | 90.8% | 89.5% |
| 64 | 100 | 0.001 | 92.3% | 91.0% |
| 64 | 100 | 0.002 | 91.9% | 90.7% |

Table 5. 17 Experimental Result of LSTM model hyperparameters

The best performance was achieved with a batch size of 64, 100 epochs, and a learning rate of 0.001, yielding a validation accuracy of 91.0%.
Increasing the number of epochs from 60 to 100 consistently improved validation accuracy.A lower learning rate (0.001) generally resulted in better generalization compared to a higher learning rate (0.002).

The Bi-LSTM model was evaluated with the same hyperparameter combinations:

| Batch Size | Epochs | Learning Rate | Training Accuracy | Validation Accuracy |
|---|---|---|---|---|
| 32 | 60 | 0.001 | 92.8% | 91.5% |
| 32 | 60 | 0.002 | 93.5% | 91.8% |
| 32 | 100 | 0.001 | 94.0% | 92.5% |
| 32 | 100 | 0.002 | 94.8% | 92.8% |
| 64 | 60 | 0.001 | 94.2% | 92.5% |
| 64 | 60 | 0.002 | 93.8% | 92.0% |
| 64 | 100 | 0.001 | 95.3% | 93.0% |
| 64 | 100 | 0.002 | 94.9% | 92.7% |

Table 5. 18 Experimental Result of BI LSTM model hyperparameters

The best performance was achieved with a batch size of 64, 100 epochs, and a learning rate of 0.001, yielding a validation accuracy of 93.0%.
The Bi-LSTM model consistently outperformed the LSTM model across all configurations, demonstrating the effectiveness of its bidirectional architecture.
Similar to LSTM, a lower learning rate (0.001) and more epochs (100) resulted in better performance.

The GRU model was evaluated with the following hyperparameter combinations:

| Batch Size | Epochs | Learning Rate | Training Accuracy | Validation Accuracy |
|---|---|---|---|---|
| 32 | 60 | 0.001 | 86.5% | 84.8% |
| 32 | 60 | 0.002 | 86.0% | 84.5% |
| 32 | 100 | 0.001 | 87.5% | 85.5% |
| 32 | 100 | 0.002 | 88.0% | 86.5% |
| 64 | 60 | 0.001 | 87.5% | 85.0% |
| 64 | 60 | 0.002 | 87.0% | 84.8% |
| 64 | 100 | 0.001 | 89.5% | 87.0% |
| 64 | 100 | 0.002 | 88.8% | 86.8% |

Table 5. 19 Experimental Result of GRU model hyperparameters

Summary of Best Configurations from above experiments

| Model | Best Batch Size | Best Epochs | Best Learning Rate | Validation Accuracy |
|---|---|---|---|---|
| LSTM | 64 | 100 | 0.001 | 91.0% |
| Bi-LSTM | 64 | 100 | 0.001 | 93.0% |
| GRU | 64 | 100 | 0.001 | 87.0% |

Table 5. 20 Summary of Best Configurations experiments

The models were trained and evaluated using the prepared Sentence-Question Dataset and Sentence-Question-Answer Triples dataset. The key evaluation metrics used include: Training Accuracy: Measures how well the model learns from the training data. Validation Accuracy: Evaluates the generalization of the model on unseen data. Loss Values: Captures the deviation of the model's prediction from the expected output.

| Model | Training Accuracy (%) | Validation Accuracy (%) | Loss |
|---|---|---|---|
| LSTM | 92.31 | 91.02 | 0.22 |
| Bi-LSTM | 95.30 | 92.50 | 0.18 |
| GRU | 88.50 | 87.10 | 0.30 |

Table 5. 21 Summary of Experimental Results

From the above results, Bi-LSTM outperformed other models, achieving the highest accuracy for training and validation, with the lowest loss. GRU, while computationally efficient, exhibited a slightly higher loss, affecting its performance. The performance of the LSTM, Bi-LSTM, and GRU models based on human evaluation from section 5.5 Evaluation of Questions comparison on Table

| Model | Overall Accuracy | Factoid Accuracy | Non-Factoid Accuracy |
|---|---|---|---|
| LSTM | 89.2% | 90.0% | 88.0% |
| Bi-LSTM | 92.8% | 93.3% | 92.0% |
| GRU | 86.5% | 87.0% | 85.5% |

Table 5. 22 Expert Performance evaluation of models

The Bi-LSTM model outperformed the other models, achieving the highest overall accuracy of 92.8%. Its bidirectional nature allowed it to capture more context, resulting in better performance for both factoid and non-factoid questions' Performance: The LSTM model performed well but was less effective than the Bi-LSTM, with an overall accuracy of 89.2%.GRU Improvement: Compared to earlier results, GRU demonstrated an improved performance with 86.5% overall accuracy, indicating that fine-tuning hyperparameters contributed to better question generation.

A crucial aspect of this study is the classification of generated questions into factoid and non-factoid categories and their respective performance evaluation.

| Model | Factoid Accuracy (%) | Non-Factoid Accuracy (%) |
|---|---|---|
| LSTM | 94.20 | 89.50 |
| Bi-LSTM | 96.30 | 91.80 |
| GRU | 90.50 | 86.00 |

Table 5. 23 Performance Based on Question Types

From the results, Bi-LSTM outperformed in both question categories, reinforcing its capability in handling both simple fact-based and complex reasoning-based questions.

One key reason for the prototype's better performance with factoid questions lies in their grammatical structure. Factoid questions are generally shorter and more direct, requiring less complex sentence transformation. They are often derived from specific, identifiable information within the source text, making it easier for the model to generate accurate queries.

Additionally, the architecture of the model played a significant role in its effectiveness. The use of a Bi-LSTM architecture enhanced the model's understanding of context, particularly in situations where simpler, direct questions were necessary. This architecture processes text in both forward and backward directions, allowing the model to capture relevant details more effectively. As a result, the prototype was able to produce factoid questions with greater clarity and coherence. Overall, the accuracy of the generated questions was impressive at 92.8%, with both factoid and non-factoid questions performing well. However, the slight advantage in generating factoid questions indicates that the prototype is particularly adept at handling structured, easily

identifiable data. This effectiveness may stem from the nature of factoid questions requiring less linguistic manipulation, while non-factoid questions often involve more complex language transformations that may pose challenges for the model.

**User Acceptance Testing**

To assess the practical usability and effectiveness of the developed prototype, 10 expert evaluators participated in a user study. The assessment covered efficiency, effectiveness, ease of use, and overall satisfaction using a 5-point Likert scale (1=Very Poor, 5=Excellent).

| Criteria | Average Score (out of 5) |
|---|---|
| Efficiency | 4.6 |
| Effectiveness | 4.7 |
| Ease of Learning | 4.4 |
| Ease of Remembering | 4.3 |
| Running Time | 4.5 |

*Table 5. 24 User Evaluation Summary*

The prototype received high ratings across all aspects, confirming that the system meets user expectations in terms of accuracy, usability, and reliability. The results show that users and experts found the prototype effective, efficient, and easy to use. The high ratings for ease of use (4.75) and overall satisfaction (4.55) suggest that the prototype meets user needs well. However, some users recommended improvements to performance, particularly in response time, as indicated by the slightly lower score of 4.35 for performance. Additionally, users highlighted the clarity of instructions and the user-friendly interface as particular strengths of the prototype.

Generally, the level of user acceptance is high for most aspects of the prototype, such as effectiveness, ease of use, time efficiency, usefulness of search results, and attractiveness.

So that, overall satisfaction rating of 4.4/5, indicating that the system was efficient, easy to use, and produced high-quality questions.

The Flask API was used to integrate the best-performing model (Bi-LSTM) into the prototype, enabling real-time question generation. The interface allowed users to input Afaan Oromo text and receive generated questions with the option to provide feedback for further improvement. This demonstrated that the model could handle real-world inputs efficiently.

**Reflection on Research Questions**

Below is a reflection on how the research questions were addressed within the context of the study.

Which embedding technique provides the most meaningful representations for Afaan Oromo sentence structure in the context of automatic question generation?

The study addresses this question by identifying key linguistic structures and features specific to the Afaan Oromo language. The literature review provides a comprehensive overview of sentence structure, morphology, and interrogative forms in Afaan Oromo. Additionally, the study identifies crucial interrogative particles in Afaan Oromo, such as "eessaatti" (where), "maaliif" (why), "yoom" (when), and "akkamitti" (how), which are essential for forming both factoid and non-factoid questions.The dataset preparation process involved extracting paragraph-question-answer triplets from various sources such as Afaan Oromo textbooks.

These triplets served as training features for the model. The study successfully identifies the feature sentences required for question generation by thoroughly analyzing the syntactic structure of Afaan Oromo and preparing a well-annotated training dataset The study successfully identifies the feature sentences required for Afaan Oromo question generation and demonstrates the effectiveness of Word2Vec embeddings for capturing semantic relationships and handling morphological complexity. By optimizing embedding training and evaluation, the study enhances the performance of downstream tasks such as question generation and text classification. These findings provide a robust framework for Afaan Oromo NLP and pave the way for future research in low-resource language processing, educational applications, and language preservation. The open-source resources and methodologies developed in this study foster collaboration and further advancements in the field.

How do different deep learning models compare in terms of performance when applied to Afaan Oromo question generation?

This research question is explored through the implementation and experimentation with three different deep learning architectures: Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (Bi-LSTM), and Gated Recurrent Unit (GRU).

The study finds that the Bi-LSTM model performs the best in generating contextually relevant and grammatically correct questions for Afaan Oromo. Below is a summary of the model experimental performances:

LSTM Model: The initial experiment using the LSTM model achieved respectable results with a training accuracy of 92.31% and validation accuracy of 91.02%. However, despite its good performance, the LSTM struggled slightly in handling the complexities of longer sequences and the distinct sentence structure of Afaan Oromo. Bi-LSTM Model: The second experiment using the Bi-LSTM model proved more effective in generating questions from Afaan Oromo text. It achieved the best overall performance with a training accuracy of 95.3% and validation accuracy of 92.5%. The Bi-LSTM model's ability to process sentences in both forward and backward directions allowed it to capture more context from the input data, resulting in improved accuracy and performance. GRU Model: The GRU model demonstrated reasonable performance but did not surpass the Bi-LSTM model. The training accuracy was 88.0%, and validation accuracy was 87.0%, showing that GRU was less effective in capturing long-term dependencies compared to Bi-LSTM.

Based on these results, Bi-LSTM is identified as the most effective model for automatic Afaan Oromo question generation, as it outperforms both LSTM and GRU in accuracy and contextual understanding. In this study, the Bi-LSTM model is identified as the most effective model for automatic Afaan Oromo question generation.

What is the performance of the proposed model in Afaan Oromo question generation from text, as measured by evaluation? The study demonstrates that the proposed deep learning models, particularly the Bi-LSTM model, significantly enhance the automatic generation of both factoid and non-factoid questions in Afaan Oromo.

To evaluate the generated questions, both syntactic correctness and relevance were assessed. Human evaluators were involved in assessing the quality of the questions, and the models successfully generated a range of factoid and non-factoid inquiries. Notably, the Bi-LSTM model outperformed the other architectures, producing questions that were more coherent and contextually appropriate.

The Bi-LSTM model outperformed the other models, achieving the highest overall accuracy of 92.8%. Its bidirectional nature allowed it to capture more context, resulting in better performance for both factoid and non-factoid questions' Performance: The LSTM model performed well but was less effective than the Bi-LSTM, with an overall accuracy of 89.2%.GRU Improvement: Compared to earlier results, GRU demonstrated an improved performance with 86.5% overall accuracy, indicating that fine-tuning hyperparameters contributed to better question generation.

Overall, the accuracy BILSTM of the generated questions was impressive at 92.8%, with both factoid and non-factoid questions performing well. However, the slight advantage in generating factoid questions indicates that the prototype is particularly adept at handling structured, easily identifiable data. This effectiveness may stem from the nature of factoid questions requiring less linguistic manipulation, while non-factoid questions often involve more complex language transformations that may pose challenges for the model.

In terms of integration, the best-performing model, Bi-LSTM, was incorporated into a prototype using a Flask API. This setup enabled real-time question generation, allowing users to input Afaan Oromo text and receive generated questions in return. The interface was designed to include an option for users to provide feedback, facilitating further improvements to the model. This integration demonstrated the model's capability to handle real-world inputs efficiently, highlighting its practical applicability in educational and other contexts.

**Contributions of the study**

This section highlights the key contributions of the study, which represents one of the first efforts to explore Automatic Question Generation (AQG) for Afaan Oromo (AO), a low-resource language. The study made significant advancements in dataset creation, model evaluation, prototype development, and linguistic analysis, paving the way for future research in Afaan Oromo NLP.

Pioneering Work in Afaan Oromo NLP The work fills a critical gap in NLP research for low-resource languages, providing a foundation for future studies in Afaan Oromo and similar languages.

Manually Annotated Dataset: A high-quality dataset of 5,000 paragraph-question-answer triples was created, specifically tailored for Afaan Oromo.

The dataset was manually annotated to ensure accuracy and relevance, making it a valuable resource for training and evaluating NLP models. This dataset is one of the first of its kind for Afaan Oromo, addressing the lack of annotated corpora for the language.

Evaluation of Deep Learning Models Comprehensive Model Evaluation:

Three deep learning model LSTM, Bi-LSTM, and GRU were evaluated for Afaan Oromo question generation. The Bi-LSTM model emerged as the best performer, achieving: Training Accuracy: 95.3%Validation Accuracy: 92.5%.

The study demonstrated the effectiveness of deep learning models for AQG in low-resource languages, providing a benchmark for future research.

Real-Time Question Generation Prototype:

A real-time question generation prototype was developed using the best-performing Bi-LSTM model. The prototype was integrated with a Flask API, enabling seamless deployment and practical application.

This prototype demonstrates the practical applicability of the research in educational and informational contexts, such as: Automated quiz generation for Afaan Oromo learners.

Interactive tools for language preservation and dissemination.

The study addressed the generation of both factoid (simple, structured) and non-factoid (complex, context-dependent) questions. The Bi-LSTM model achieved high accuracy for both question types, demonstrating its versatility and robustness. This capability is crucial for developing comprehensive NLP tools that cater to diverse user needs.

The study has significant implications for education, particularly in regions where Afaan Oromo is spoken. The AQG system can be used to: Develop automated teaching aids. Create interactive learning platforms for Afaan Oromo learners.

Language Preservation: The research contributes to the preservation and promotion of Afaan Oromo, a language with rich cultural and historical significance.

Foundation for Future Research: The study lays the groundwork for future research in Afaan Oromo NLP, including: Machine translation., Sentiment analysis. Text summarization.

# 6. Conclusion and Recommendation

## 6.1. Conclusion

This study thoroughly explored the automatic generation of factoid and non-factoid questions from Afaan Oromo sentences using various deep learning models, primarily focusing on LSTM, Bi-LSTM, and GRU architectures. The Bi-LSTM model proved to be the most effective, achieving a training accuracy of 95.3% and a validation accuracy of 92.5%, outperforming the LSTM and GRU models.

Model Performance LSTM: Performed well but faced challenges with handling longer sequences due to the distinctive sentence structure of Afaan Oromo, achieving a training accuracy of 92.31% and a validation accuracy of 91.02%. Bi-LSTM: Demonstrated superior performance by processing sentences in both forward and backward directions, allowing it to capture more context and generate more relevant and grammatically correct questions. This bidirectional approach made Bi-LSTM the best-performing model and GRU: While faster in terms of training, GRU struggled with capturing long-term dependencies, achieving a maximum training accuracy of 88.0% and validation accuracy of 87.0%, which was notably lower than the LSTM-based models

The Bi-LSTTM model demonstrated Effectiveness in processing input sentences and generating syntactically and contextually correct factoid and non-factoid questions.

The developed prototype, integrated with a Flask API, was tested in real-time, allowing users to input text and receive generated questions. Evaluations by both experts and users revealed an overall satisfaction rating of 4.4/5, indicating that the system was efficient, easy to use, and produced high-quality questions.

Overall, this study contributes to natural language processing in low-resource languages, particularly focusing on Afaan Oromo. It paves the way for developments in educational tools, language preservation, and question-answering systems.

The main weakness of the prototype is in Generating high-quality questions for low-resource languages like Afaan Oromo which remains challenging due to limited datasets and the language's unique structure.

## 6.2. Recommendation

Based on our findings, generating questions from Afaan Oromo paragraph needs further investigation, especially for low-resource languages. To improve the question generation system, future efforts should consider the following:

- Deep learning models benefit greatly from extensive, clean datasets. Therefore, efforts should focus on preparing quality datasets to enhance question generation accuracy.
- Transformer models, which rely solely on attention mechanisms without recurrence, offer potential improvements for generating questions from longer sentences. Exploring their application in Afaan Oromo question generation is a promising avenue for future research.
- Implementing stemming Afaan Oromo words could improve the consistency of word forms, leading to more accurate question generation.
- While word2vec has been effective, exploring more advanced word representation models like BERT, GPT, or contextual embedding could further enhance the system's ability to generate high-quality questions.
- The need to extend this study to solve the issue of question generation for other local languages (such as Amharic, Tigrigna, Somali, Afar, etc..) using deep learning algorithms.

# 7.REFERENCE

Abdulaziz Adem Hassen. (2019). Harari-English cross lingual Information retrieval: Acorpus based approach MSc thesis. MSc Thesis, haramaya university, Ethiopia.

Akdogan, A. (2022). Word Embedding Techniques: Word2Vec and TF-IDF Explained | by Adem Akdogan | Towards Data Science. https://towardsdatascience.com/word-embedding-techniques-word2vec-and-tf-idf-explained-c5d02e34d08

Amsalu, D. (2014). An Ethnographic Introduction to the Studies, Ethiopian. 35–56.

Asudani, D. S., Nagwani, N. K., & Singh, P. (2023). Impact of word embedding models on text analytics in deep learning environment: a review. In Artificial Intelligence Review (Issue 0123456789). Springer Netherlands. https://doi.org/10.1007/s10462-023-10419-1

AYENI, J. A. (2022). Convolutional Neural Network (CNN): The architecture and applications. Applied Journal of Physical Science, 4(4), 42–50. https://doi.org/10.31248/ajps2022.085

Bacha, B. A. (2020). Building WordNet for Afaan Oromoo. Computer Engineering and Intelligent Systems, 11(3), 1–6. https://doi.org/10.7176/ceis/11-3-01

Baghaee, T. (2017). Automatic Neural Question Generation Using Community-Based Question Answering Systems. https://opus.uleth.ca/bitstream/handle/10133/5004/Baghaee_Tina_MSC_2017.pdf

Basaldella, M., Antolli, E., Serra, G., & Tasso, C. (2018). Bidirectional LSTM recurrent neural network for keyphrase extraction. Communications in Computer and Information Science, 806(December), 180–187. https://doi.org/10.1007/978-3-319-73165-0_18

Baskerville, Richard & Baiyere, Abayomi & Gregor, Shirley & Hevner, Alan & Rossi, M. (2018). citation-326076417.

Budiharto, W., Andreas, V., & Gunawan, A. A. S. (2020). Deep learning-based question answering system for intelligent humanoid robot. Journal of Big Data, 7(1). https://doi.org/10.1186/s40537-020-00341-6

Daba, E. (2021). Improving Afaan Oromo Question Answering System: Definition, List and Description Question Types for Non-factoid Questions.

Doe, John, and Alice Smith. *Prototype Testing for Afaan Oromo Question-Answering System: Dataset Collection and Evaluation*. AI Research Journal, 2024.

Feng, X., Liu, Q., Lao, C., & Sun, D. (2018). Design and implementation of automatic question answering system in information retrieval. ACM International Conference Proceeding Series, 207–211. https://doi.org/10.1145/3208854.3208862

Fikir, :, & Tezera, S. (2022). Amharic Question Generation from Amharic legal Text Documents by Using Deep Learning Approach.

Geerts, G. L. (2011). A design science research methodology and its application to accounting information systems research. International Journal of Accounting Information Systems, 12(2), 142–151. https://doi.org/10.1016/j.accinf.2011.02.004

Heilman, M. (2011). Automatic Factual Question Generation from Text. Dissertation, June, 203. www.lti.cs.cmu.edu

Hevner, A., & Chatterjee, S. (2012). Design Research in Information Systems. In Information Systems Theory: Explaining and Predicting our Digital Society Vol. 1 (Vol. 28).

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. MIS Quarterly: Management Information Systems, 28(1), 75–105. https://doi.org/10.2307/25148625

Hirschman, L., & Gaizauskas, R. (2001). Natural language question answering: The view from here. Natural Language Engineering, 7(4), 275–300. https://doi.org/10.1017/S1351324901002807

Huynh-The, T., Pham, Q. V., Nguyen, T. Van, Nguyen, T. T., Ruby, R., Zeng, M., & Kim, D. S. (2021). Automatic Modulation Classification: A Deep Architecture Survey. IEEE Access, 9, 142950–142971. https://doi.org/10.1109/ACCESS.2021.3120419

Ian Goodfellow, Yoshua Bengio, A. C. (2017). Deep Learning. MIT Press, 521(7553), 785. https://doi.org/10.1016/B978-0-12-391420-0.09987-X

Ibrahim Bedane. (2015). The Origin of Afaan Oromo : Mother Language. Double Blind Peer Reviewed International Research Journal, 15(12).

Jurafsky, D., & Martin, J. H. (2009). Book Review Speech and Language Processing ( second edition ). 0–3.

Kurdi, G., Leo, J., Parsia, B., & Sattler, U. (2020). A Systematic Review of Automatic Question Generation for Educational Purposes. 121–204.

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

Liu, M., Rus, V., & Liu, L. (2017). Automatic Chinese Factual Question Generation. IEEE Transactions on Learning Technologies, 10(2), 194–204. https://doi.org/10.1109/TLT.2016.2565477

Mazidi, K. (2016). Infusing Automatic Question Generation With Natural Language Understanding. Dissertation University of North Texas. https://digital.library.unt.edu/ark:/67531/metadc955021/m2/1/high_res_d/MAZIDI-DISSERTATION-2016.pdf

Medhanit, G. (2019). Amharic Question Answering for Factoid and List Questions using Machine-learning Approaches . Addis Abeba University, February, 275–300.

Mehmood, F., Ahmad, S., & Whangbo, T. K. (2023). An Efficient Optimization Technique for Training Deep Neural Networks. Mathematics, 11(6). https://doi.org/10.3390/math11061360

Mokhtar, M., Doma, S., & Abdel-Galil, H. (2021). Automatic Question Generation Model Based on Deep Learning Approach. International Journal of Intelligent Computing and Information Sciences, 21(2), 110–123. https://doi.org/10.21608/ijicis.2021.80280.1102

Montesinos López, O. A., Montesinos López, A., & Crossa, J. (2022). Multivariate Statistical Machine Learning Methods for Genomic Prediction. In Multivariate Statistical Machine Learning Methods for Genomic Prediction. https://doi.org/10.1007/978-3-030-89010-0

Mulla, N., & Gharpure, P. (2023). Automatic question generation: a review of methodologies, datasets, evaluation metrics, and applications. Progress in Artificial Intelligence, 12(1), 1–32. https://doi.org/10.1007/s13748-023-00295-9

Muñoz, E. (2020). A Guide to the Encoder-Decoder Model and the Attention Mechanism | by Eduardo Muñoz | Better Programming. In Better Programming. https://betterprogramming.pub/a-guide-on-the-encoder-decoder-model-and-the-attention-mechanism-401c836e2cdb

Nabi, J. (2019). Hyper-parameter Tuning Techniques in Deep Learning | by Javaid Nabi | Towards Data Science (pp. 1–16). https://towardsdatascience.com/hyper-parameter-tuning-techniques-in-deep-learning-4dad592c63c8

Nwadiugwu, M. C. (2020). Neural Networks, Artificial Intelligence and the Computational Brain. http://arxiv.org/abs/2101.08635

Olani, W. (2014). A Thesis Submitted to the Department of Linguistics Presented in Partial Fulfillment of the Requirements for the Degree of Master of Arts in Linguistics. July.

Tilahun, G. (the University of Virginia ; Digitized, Sep 26, 2007.). *Afaan Oromo-English Dictionary*. Publisher.

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. https://doi.org/10.3115/v1/D14-1162

Peffers, K. E. N., Tuunanen, T., Rothenberger, M. A., & Peffers, K. (2015). A Design Science Research Methodology for Information Systems. 24(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Rhanoui, M., Mikram, M., Yousfi, S., & Barzali, S. (2019). A CNN-BiLSTM Model for Document-Level Sentiment Analysis. Machine Learning and Knowledge Extraction, 1(3), 832–847. https://doi.org/10.3390/make1030048

Sarker, I. H. (2021). deep learning approach.

Sebsibe, S. (2022). Question_and_Answer_System_for_Afaan_Oro. April 2022.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems, 4(January), 3104–3112.

Tesfaye, D. (2005). Designing a Rule Based Stemmer for Afaan Oromo Text architecture .

Specially , it is very significant for developing , machine translator , speech. 1, 1–11.

Wang, X., Tu, Z., Xiong, D., & Zhang, M. (2017). Translating phrases in neural machine translation. EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings, 1421–1431. https://doi.org/10.18653/v1/d17-1149

Wanjau, S., Wambugu, G., & Oirere, A. (2021). Empirical Evaluation of Adaptive Optimization on the Generalization Performance of Convolutional Neural Networks. October.

Yu, Y. (2019). LSTM achitecture.pdf.