MATTU UNIVERSITY

COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

AFAAN OROMOO TEXTUAL ENTAILMENT CLASSIFICATION USING DEEP LEARNING APPROACH

MSc THESIS

BY: DIRO TOLOSA GODA

MAIN ADVISOR: ARULMURUGAN RAMU (PhD)

CO-ADVISOR: TESHOME DEBUSHE (MSc)

NOVEMBER, 2024

MATTU, ETHIOPIA

MATTU UNIVERSITY

COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

AFAAN OROMOO TEXTUAL ENTAILMENT CLASSIFICATION USING DEEP LEARNING APPROACH

A THESIS SUBMITTED AS A PARTIAL FULFILLMENT TO THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

BY: DIRO TOLOSA GODA

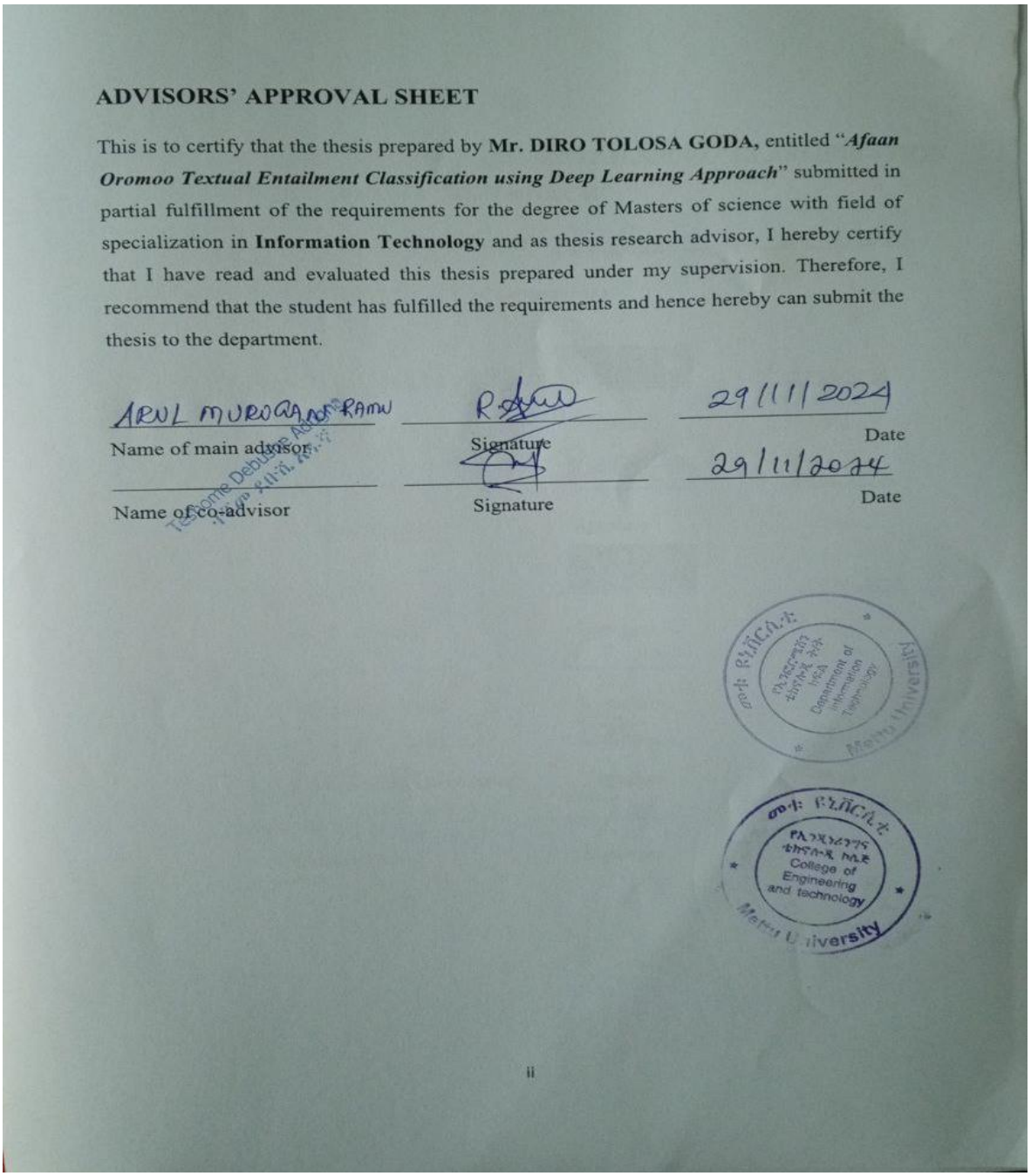
MAIN ADVISOR: ARULMURUGAN RAMU (PhD)

CO-ADVISOR: TESHOME DEBUSHE (MSc)

NOVEMBER, 2024

MATTU, ETHIOPIA

ADVISORS' APPROVAL SHEET



ADVISORS' APPROVAL SHEET

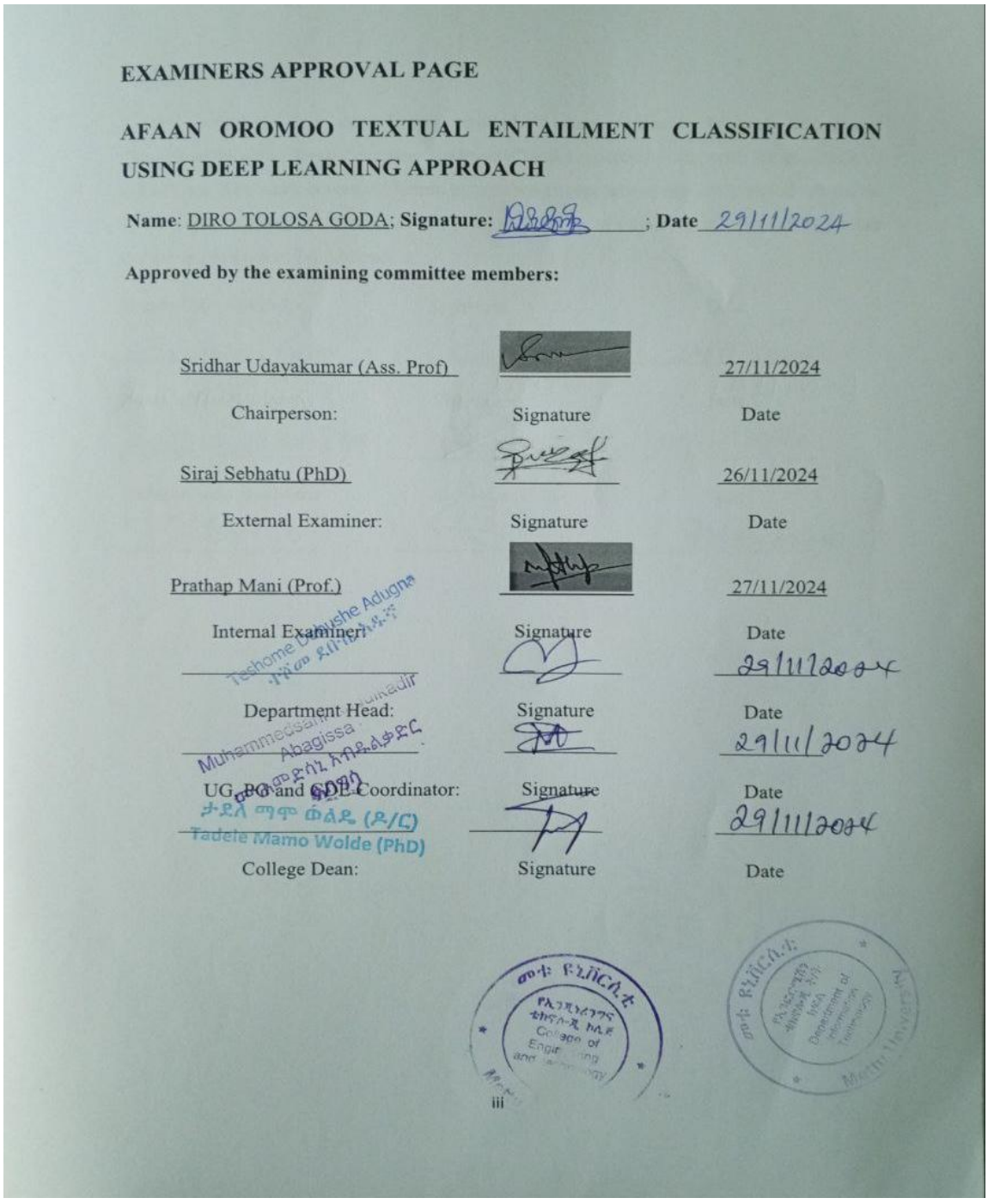
This is to certify that the thesis prepared by **Mr. DIRO TOLOSA GODA,** entitled "***Afaan Oromoo Textual Entailment Classification using Deep Learning Approach***" submitted in partial fulfillment of the requirements for the degree of Masters of science with field of specialization in **Information Technology** and as thesis research advisor, I hereby certify that I have read and evaluated this thesis prepared under my supervision. Therefore, I recommend that the student has fulfilled the requirements and hence hereby can submit the thesis to the department.

ARUL MURUGAN RAMU — Name of main advisor — Signature — 29/11/2024 — Date

Name of co-advisor — Signature — 29/11/2024 — Date

ii

EXAMINERS APPROVAL PAGE



EXAMINERS APPROVAL PAGE

AFAAN OROMOO TEXTUAL ENTAILMENT CLASSIFICATION USING DEEP LEARNING APPROACH

Name: DIRO TOLOSA GODA; Signature: _____; Date 29/11/2024

Approved by the examining committee members:

Sridhar Udayakumar (Ass. Prof)		27/11/2024
Chairperson:	Signature	Date
Siraj Sebhatu (PhD)		26/11/2024
External Examiner:	Signature	Date
Prathap Mani (Prof.)		27/11/2024
Internal Examiner:	Signature	Date
Department Head:	Signature	29/11/2024 Date
UG, PG and SDL Coordinator:	Signature	29/11/2024 Date
College Dean:	Signature	29/11/2024 Date

iii

DECLARATION

I hereby declare that this Thesis entitled **"*Afaan Oromoo Textual Entailment Classification using Deep Learning Approach"*** was prepared by me, with the guidance of my advisor. The work contained herein is my own except where explicitly stated otherwise in the next, and that this work has not been submitted, in whole or in part, for any other degree or professional qualification.

Name of the Student	Signature	Date
_____	_____	_____
Name of the Advisor	Signature	Date
_____	_____	_____
Name of the Co-advisor	Signature	Date
_____	_____	_____

ABSTRACT

Human communication relies on natural language, such as Afaan Oromoo. However, for computers to interact effectively with humans, they must be able to understand natural language. Natural language processing is the field that enables computers to understand and use human language. Since textual entailment determines the relationship between sentence pairs, it is a crucial task in natural language processing. Recent development in deep learning has offered promising solutions for automating feature engineering and learning semantic representations. This study used a deep learning approach for classifying textual entailment in Afaan Oromoo into three categories: Entailment, Contradiction, and Neutral. Despite its widespread use in the Horn of Africa, Natural Language Processing tools for Afaan Oromoo are limited. To address this gap, we collected a dataset of 13,060 sentence pairs in Afaan Oromoo, preprocessed the data, and developed model architecture for classification. We developed the model using various deep learning approaches, including CNN, LSTM, and BiLSTM, comparing their performance to identify the most effective approach. The BiLSTM model showed highest performance, achieving 91.23% accuracy on the training dataset, 82.15% accuracy on the validation dataset, and 80.47% accuracy on the test dataset. Considering that there are currently little resources available for Afaan Oromoo Natural Language Processing, these results are encouraging. As a starting point for future research, this study offers a basis for additional investigation and advancement in this field. This research is expected to make a substantial contribution to the development of Afaan Oromoo's natural language processing capabilities.

Keywords: Afaan Oromoo, Bidirectional Long Short Term Memory, Convolutional Neural Network, Deep Learning, Natural Language Processing, Textual Entailment Classification.

DEDICATED TO:

My younger sister, **Alami Tolosa Goda** and my little younger brother, **Abdi Tolosa Goda**.

ACKNOWLEDGMENTS

I am deeply grateful to **God** for the countless blessings that have made this accomplishment possible. God, your guiding hand and divine inspiration have been key in my journey.

Next, I would like to express my sincere gratitude to my advisor, **Dr. Arulmurugan Ramu**, for his invaluable guidance, unwavering support, and encouragement throughout my thesis research. His expertise, mentorship, and dedication have been instrumental in my success.

I am also indebted to **Mr. Teshome Debushe** for his exceptional support and contributions to this study. His insights and assistance have been invaluable.

Additionally, I would like to thank **Mr. Ramatha Mosissa (Ass. Prof)** for his support in giving me his laptop to accomplish this work.

At the end, my heartfelt thanks go to **Mrs. Gadise Birhanu Olkeba**, a promising young scholar whose technical expertise and enthusiasm have been a source of inspiration. Her assistance in dataset preparation and her insightful discussions have been invaluable

Table of Contents

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS OR ACRONYMS

AOTEC	Afaan Oromoo Textual Entailment Classification
Bi-LSTM	Bidirectional Long Short-Term Memory
CIAN	Character-level Intra Attention Networks
CLTE	Cross-lingual Textual Entailment
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
MLTE	Multilingual Textual Entailment
MTE	Monolingual Textual Entailment
NLI	Natural Language Inference
NLP	Natural Language Processing
NLU	Natural Language Understanding
PTE	Partial Textual Entailment
RBM	Restricted Boltzmann Machines
RNN	Recurrent Neural Network
RTE	Recognizing Textual Entailment
SMO	Sequential Minimal Optimization
SMT.	Social media Text
TAC	Text Analysis Conference
TEC	Textual Entailment Classification

CHAPTER: ONE

INTRODUCTION

1.1 Background

Humans can communicate with each other using some form of language either by text or speech. They can exchange the message with their natural language without having any trouble. But, to make interaction with Computers, the language that human being use should be understandable by computers. This aims enabling computers to read and comprehend the natural language used by humans[1]. The ability of computer software to understand spoken and written human language is known as natural language processing. The main goal is to teach computers to understand, comprehend, analyze, and manipulate human languages. Nowadays, many of the programs we use on a daily basis are based on machines' capacity to understand human language[2].

Natural language allows for several meanings to be stated by a single text as well as multiple texts that express the same meaning indicating a many-to-many mapping relationship between the meanings and phrases of language. Theoretically, a comprehensive semantic interpretation into a logic-based representation of a text's meanings would be necessary for accurate interpretation [3].

The evaluation of Natural Language Processing systems and the question of whether model performs better at generating text or understanding language is becoming more crucial as NLP technologies become more extensively used[4]. The way computers understand the structure and meaning of human language can be called as Natural language understanding. Here, understanding the meaning of a given piece of text is an open problem in NLP [5].

Text Entailment is a branch of Natural Language Processing that focuses on understanding the meaning of sentences. It involves determining the relationship between pairs of texts (e.g., entailment, contradiction, neutral) [5]. The aim of textual entailment recognition (RTE) is to ascertain if the hypothesis can be reasonably deduced from the premise given a pair of sentences. In 2005, the significance of Textual Entailment Recognition has become a generic task that addresses the main requirements for semantic inference in a wide range of applications using natural language processing.

Many NLP applications[6]., such as Question Answering, Information Extraction, text summarization and Machine translation evaluation, need automation of textual entailment in order to recognize that a particular target meaning can be inferred from different text variants[7]. So, textual entailment recognition (RTE) has been a very popular research area in the last years and current also.

Recognizing Textual Entailment (RTE) has gained popularity in the NLP community since its inception in 2005 because it appears to serve as a common framework for analyzing, contrasting, and assessing various approaches used in NLP applications to address semantic inference, a problem that many NLP applications share. Different approaches are developed for Textual Entailment Recognition. Most of Recognizing Textual Entailment systems done are based on Machine Learning, lexical or semantic approaches. These approaches use features such as lexical, syntactic and semantic features. Lexical approach works directly on the input surface strings, it performs poorly on the task of recognizing textual entailment. Because of textual entailment is a directional relation, where the text contains more information than the hypothesis. Another known approaches of recognizing textual entailment are syntax based, in which syntactic information is usually represented as directed graph[3].

The rapid development of deep learning with natural language processing brings hope for possibly alleviating the problem of avoiding manual feature engineering and learning semantic representations for natural language [8] . Deep learning models have achieved impressive performance on various Natural language inference tasks[9].. They leverage neural networks to learn complex patterns and representations from large amount of text data [9]. In this study, the researcher focused on the deep learning approach developed for Afaan Oromoo language textual entailment classification.

1.2. Statement of the Problem

The process of classifying Afaan Oromoo textual entailment involves determining, given two Afaan Oromoo sentences, whether the meaning of one text is implied (may be deduced) from another. Because of Afaan Oromoo is a low-resourced language, currently there is no work done for Afaan Oromoo textual entailment classification. For different highly resourced language like English, many researches have been done by many scholars. It is

known that, the model designed and developed for other language is not used for Afaan Oromoo's syntax and semantics rules. Afaan Oromoo language is semantically complex, morphologically complex and word orders (Subject-Object-Verb) are different when we compare with other languages[10].

The desire for this research comes from that, Afaan Oromoo textual entailment classification has not yet been done. This gap inspires the researcher to do this Afaan Oromoo Textual Entailment Classification using deep learning. The dataset required for this work is a pair of sentences (premise-hypothesis) of Afaan Oromoo texts. The researcher prepared the dataset with Afaan Oromoo experts. The traditional way of Textual entailment classification mainly based on manually created features. But, this traditional method comes with many challenges in textual entailment recognition due to the language variability, i.e. different text fragments can have the same meaning[11]. So, using deep learning approach can solve the challenges related with traditional methods.

1.3. Research Questions

The following are the research questions that will be addressed in this study.

1) Which deep learning approach performs better for Afaan Oromoo textual entailment classification?
2) How can Afaan Oromoo textual entailment be classified using the chosen approach?
3) To what extent the model is effective?

1.4. Objective

1.4.1. General objective

The general objective of this research is to develop Afaan Oromoo textual entailment classification model using Deep learning approach.

1.4.2. Specific objectives

- To identify the existing gap for Afaan Oromoo textual entailment classification
- To collect and organize Afaan Oromoo sentence pair texts dataset
- To identify which feature extraction is suitable for Afaan Oromoo textual entailment classification

- To develop Afaan Oromoo textual entailment classification model using deep learning approach
- To evaluate the performance of the model using test dataset

1.5. Methodology

1.5.1. Literature review

Literature review is a critical review of previously published and existing literature (books, articles, dissertations, etc.) relevant to a particular research area. It involves summarizing, evaluating, and synthesizing the findings of various sources to offer a thorough comprehension of the area. For this study, we have reviewed different approaches of Textual entailment classification, algorithms, research conference papers, Journal articles, and etc.

1.5.2. Data collection

It is known that; data collection is the most significant task of all in any research activity. Afaan Oromoo is a low resourced language, that is a challenging to work with. Because of there is no publicly available dataset, especially in Natural Language Processing researches. Here, for Afaan Oromoo Textual Entailment Classification, the researcher collected data with Afaan Oromoo Language experts.

1.6. Significance of the Research

The research is done on Afaan Oromoo textual entailment classification by using deep learning approaches. The dataset organized will be useful for other researcher to work more on this language and with translation for other under-resourced language. The developed model for Afaan Oromoo textual entailment recognition would contribute to advancing NLP capabilities for this language. It can help the community, speakers of the language by providing them with access to advanced NLP technologies. It enables learners in understanding the relationships between different texts, aids in language learning, facilitates good communication, it contributes to the development of linguistic diversity, helps researchers in developing techniques that can be used to other under-resourced languages. Generally, Afaan Oromoo textual entailment classification using deep learning shows significant advantages not only for the Afaan Oromoo NLP research communities

but also for the development of Afaan Oromoo language, helps speakers and the communities.

1.7. Scope and Limitation of the Research

1.7.1. Scope of the research

The scope of this research is on Afaan Oromoo Textual Entailment Classification using deep learning approaches. It contains textual entailment classification at the sentence level for Afaan Oromoo language [Monolingual Textual Entailment]. There are three class labels used [Entailment, Contradiction and Neutral]. The model we developed checks whether sentence 2 (hypothesis) is inferred from corresponding sentence 1 (premise). The dataset has been collected from Afaan Oromoo texts and we translated some from SNLI dataset prepared for English language.

1.7.2. Limitation of the research

The source of data was from Afaan Oromoo texts, like: reports, newspapers, religious and health care data. Texts pertaining to proverbs, pottery, and multi-sentence inference are not included in the study, because of they are multi-dimensional sentences.

1.8. Organization of the Thesis

The thesis is organized into five chapters, including Chapter One. Chapter Two provides a review of the literature related to previous studies and the Afaan Oromoo language. Chapter Three details the methods used, including the materials, design, and architecture of the Afaan Oromoo Textual Entailment Classification using a deep learning approach. Chapter Four presents the data analysis and discusses the results to enhance the research's significance. Finally, Chapter Five summarizes the work, presents conclusions, and offers recommendations for future research.

CHAPTER: TWO

LITERATURE REVIEW

2.1. Introduction

Previous related researches, books, papers, and journals are studied in order to support and accomplish the goal of the study. The researcher studied different studies and publications on Natural language processing, Textual Entailment, approaches of textual entailment recognition, textual entailment classification in various languages like Ethiopian and Non-Ethiopian languages, Afaan Oromoo language and its writing system, and other related works are discussed in detail in order to achieve the objective of the research.

2.2. Natural Language Processing

NLP is a branch of computational linguistics and artificial intelligence that aims to understand human language by machines. Its goal is to make it possible for computers to produce, comprehend, and interpret human language in a meaningful and suitable manner for the given context. NLP's primary objective is to give computers human-like language understanding and manipulation capabilities. NLP combines several techniques, including computational linguistics, machine learning, and deep learning models. It enables computers to process texts and speeches and extract their content[12]. In Natural language processing, enabling computers to understand the meaning of the sentences is the bold challenges researchers are facing. These semantic inference challenges can get solution through textual entailment recognition.

2.3. Textual Entailment

For several decades, linguistics and artificial intelligence have been interested in the idea of textual entailment, commonly referred to as natural language inference (NLI). Textual entailment is a specific task within the field of natural language understanding[13].. It focuses on determining the logical relationship between two pieces of text, typically referred to as the premise (sentence 1) and the hypothesis (sentence 2)[13].

The task of textual entailment involves determining whether the hypothesis can be inferred or logically derived from the premise, and it requires understanding the semantic relationship between the two texts. In natural language understanding, recognizing and

coping with inferences is a key point. The majority of NLP systems need to comprehend and draw conclusions from text, even though they may be trained to carry out various tasks including translating, responding to queries, or extracting information from text. RTE was therefore developed as a framework for assessing NLP systems. The goal of RTE, which has its roots in linguistics, is to determine whether the meaning of one sentence can likely be inferred from another [4].

2.3.1. Types of Textual Entailment

A. Mono-lingual Textual Entailment

The process of deciding whether a text, known as the "hypothesis," can be logically deduced from another text, known as the "premise," when both texts are written in the same language is known as monolingual textual entailment (MTE). It entails determining if the premise logically supports the hypothesis or if the meaning of the hypothesis can be inferred from it.

Here's an example to illustrate MTE:

a. **Premise:** All dogs are mammals (Saroonni hundi hoosiftoota)

b. **Hypothesis:** My dog is a mammal (Sareen koo hoosiftuudha).

In this case, the hypothesis can be inferred from the premise, so the entailment holds true.

MTE is a fundamental problem in natural language processing (NLP) with applications in various tasks such as question answering, summarization, information retrieval, and machine translation. It helps in understanding and reasoning about textual information, ensuring that systems can make logical connections between different pieces of text.

B. Cross Lingual Textual Entailment

An extension of monolingual textual entailment to many languages is called cross-lingual textual entailment, or CLTE. The purpose of CLTE is to determine if a text (the premise) in one language can be inferred from a text (the hypothesis) in another language. This task requires understanding the semantic content of texts across different languages and involves translation as an intermediate step or leveraging multilingual representations [14].

Here's an example to illustrate CLTE:

a. **Premise**: "All dogs are mammals"

b. **Hypothesis**: "Sareen sun ni hoosifti" (Translation: That dog is mammal)

In this example, the hypothesis in Afaan Oromoo can be inferred from the premise in English, so the entailment holds true.

CLTE is a challenging task because it involves not only understanding and generating logical inferences but also dealing with the complexities of language translation, such as handling nuances, idiomatic expressions, and language-specific structures. Successful CLTE systems require robust multilingual models or effective translation mechanisms to accurately capture the semantic equivalence between texts in different languages. CLTE enables the processing and understanding of information across language barriers, enhancing the accessibility and interoperability of global information systems.

C. Multilingual Textual Entailment

Multilingual textual entailment (MLTE) is a textual entailment that involves determining whether a piece of text (the hypothesis) in one language can be logically inferred from another piece of text (the premise) in the same or a different language. Unlike cross-lingual textual entailment, which specifically deals with different languages for the premise and hypothesis, multilingual textual entailment encompasses scenarios where both texts could be in the same or different language.

Here's an example to illustrate MLTE:

a. **Premise:** "All dogs are mammals"
b. **Hypothesis:** "Saroonni hundinuu hoosiftoota" (Translation: "All dogs are mammals")

2.3.2. Classifications of Textual Entailment

Textual entailment can be classified into different types based on the nature of the relationship between the premise and hypothesis[13]. Here are some common classifications.

A. Entailment: In entailment, the premise and the hypothesis follow logically from one another. The hypothesis must also be true if the premise is. For example:

a. **Premise**: "The man is sleeping on the bed" (Namichi siree irra rafaa jira)
b. **Hypothesis**: "There is a man on the bed" (Namichi siree irra jira)
c. **Relationship**: **Entailment** (the hypothesis logically follows from the premise)

B. Contradiction: When two statements are directly at odds with one another, they are said to be in contradiction. This means that if the premise is true, the hypothesis must also be incorrect, and vice versa. For Example:

a. **Premise**: “The man is good” (Namichi gaariidha)
b. **Hypothesis**: “The man is not good” (Namichi gaarii miti)
c. **Relationship**: **Contradiction** (the hypothesis contradicts the premise)

C. Neutral: There is no logical relationship between the hypothesis and the premise in a neutral relationship. One's truth does not automatically indicate the other's truth or untruth. For Example:

a. **Premise**: "The man is playing the ball." (Namichi kubbaa taphachaa jira).
b. **Hypothesis**: "The sun is shining." (Aduun ba’aa jirti).
c. **Relationship**: **Neutral** (no logical connection between the premise and hypothesis)

2.4. Textual Entailment approaches

2.4.1. Lexical Approaches

Direct manipulation of the input surface strings is the focus of the lexical approach. All it does is use a string comparison to compare the text and hypothesis. Lexical entailment seeks to ascertain the entailment between two pairs of sentences solely through lexical concepts. Determining entailment relationships mostly requires examining the text's lexicon and linguistic elements. Common approaches include word overlap, sub-sequence matching, longest sub string using sliding window approach[3][15]. The following are some of the common approaches.

Lexical Overlap: This approach measures the word overlap or phrases between texts.

WordNet: it is a lexical database that organizes a set of synonymous words.

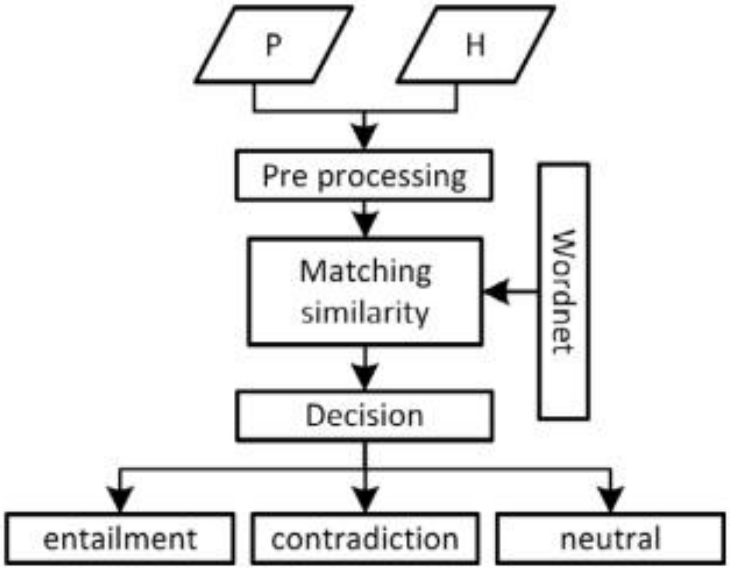


Figure 1: A Lexical based approach for RTE

2.4.2. Syntax Based approaches

Syntax based approach is one of the popular approaches used for recognizing textual entailment[3]. It mainly focuses on analyzing the grammatical structure and syntactic relationships between sentences to determine entailment. In syntax based approach, syntactic information is usually represented as directed graph [3]. The text and hypothesis are represented by a directed graph in this approach. In that graph, each word or phrase is represented by a node and the edges in the graph represent the relation between those nodes. Here, entailment depends upon the amount of semantic content of the given hypothesis present in the text[3]. Some of a syntax-based approaches are:

A. **Syntactic parsing**: it is the step of parsing the sentences in the two texts to identify the grammatical structure of sentences. The constituency parsing or the dependency parsing are the techniques used for syntactic parsing. Dependency parsing identifies the relationship between words in sentences, while Constituency parsing breaks down sentences into hierarchical structures (parse trees) based on their syntactic constituents.
B. **Syntactic structures alignment**: the syntactic structure of the two texts should be aligned next to the sentence parsing. Here, the process involves matching corresponding syntactic constituents or dependency relations between the sentences.

C. **Entailment patterns identification**: After aligning the syntactic structures, the system looks for patterns that indicate entailment relationships.

D. **Decision**: Finally, inference rules and logical reasoning techniques are applied to draw conclusions about entailment based on the syntactic analysis and matching. This may involve employing deductive reasoning principles to infer entailment based on the syntactic and semantic relationships between the texts.

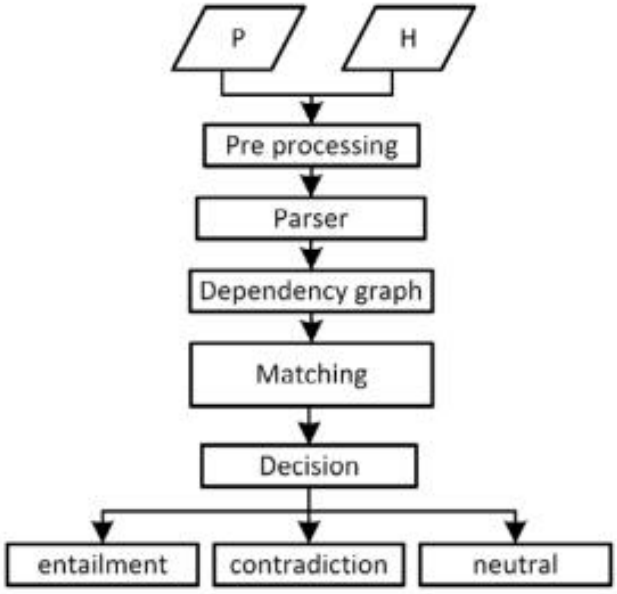


Figure 2: Syntax based RTE model with a parser

2.4.3. Semantics-based approaches

The meaning of the texts directly influences the semantics-based approach. The method connects verbal expressions with semantic representations. Similarities that are not visible at the surface or syntactic level can be revealed by semantic representations. However, there is a chance that the semantic analysis process will add errors into the representation. This could be offset by the ability to identify semantic similarities that are difficult to identify in syntactic or surface level representations. Semantic based methods, those that use some sort of semantic representation of the text and the hypothesis, are usually based on a predicate argument structure [3][16]. Some of the overviews of semantic based approach are:

A. Semantic representation: in the first step, the meaning of sentences should be represented in a structured format. This may involve converting the sentences into

semantic representations such as semantic graphs, semantic role structures, or logical forms.

B. **Semantic similarity measures**: semantic similarity measures should be applied to quantify the degree of relatedness between the meanings of the sentences next to representing sentences semantically. These measures assess the semantic overlap between words, phrases, or concepts in the sentences. Techniques such as cosine similarity, Jaccard similarity, or semantic vector representations may be used for this purpose.

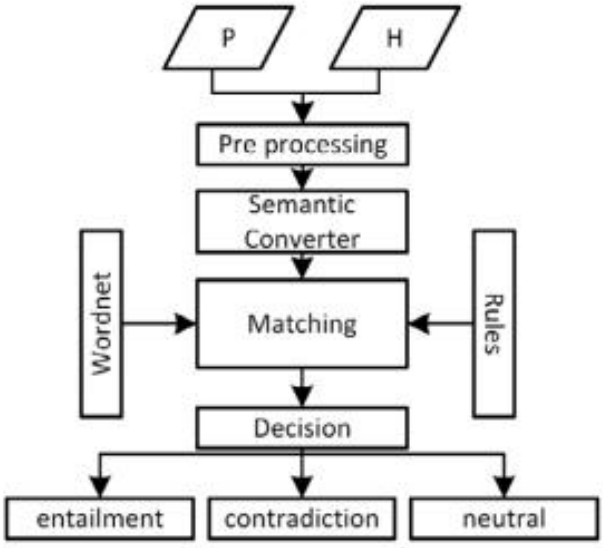


Figure 3: A semantic based RTE model

2.4.4. Logical Form Based approach

Logical form-based methods are one of the most knowledge intensive set of approaches and that relies on a logical meaning representation[3]. A logical meaning representation is able to expose similarities that are not found in other representation levels. Such a representation is a meaning representation that is backed with a sound and understandable formal semantics and can take advantage of formal reasoning algorithms to derive information[3].

2.4.5. Hybrid approach

Hybrid methods cover approaches that use a combination of methods to recognize textual Entailment[3]. Hybrid approaches are usually based on only two methods with one act as primary strategy and the other as a backup[3].

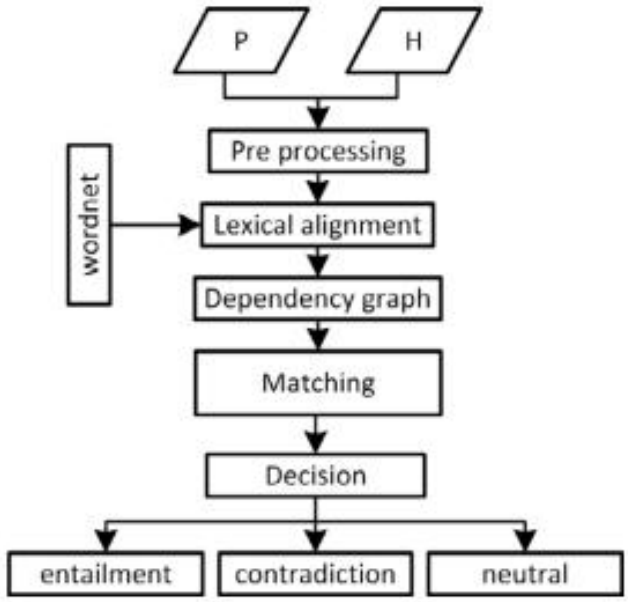


Figure 4: A hybrid Approach for RTE

2.4.6. Machine Learning Approach

Machine learning based textual entailment recognition approach involves training models to automatically learn patterns and features from labeled data, which consist of pairs of text and their corresponding entailment relationships[17]. Machine learning approach uses a supervised learning algorithm that identifies the class to which an instance belongs. Support Vector Machine, Random Forest, Naïve Bayes classifier, decision tree and others classification algorithms are used [17].

2.4.7. Deep learning Approach

2.4.7.1. Basics of Deep Learning

Deep learning is a branch of machine learning that offers a new method for extracting meaningful representations from data by focusing on learning several layers of progressively more detailed representations. It incorporates techniques designed to model high-level abstractions in data using structures composed of several nonlinear transformations. Neural networks, another name for these architectures, are based on the structure and functions of the human brain. The amazing capacity of deep learning to learn from vast amounts of data, automatically identify complex patterns, and make predictions or judgments without explicit programming has garnered it a great deal of attention and appeal. Deep learning algorithms can learn data representations by utilizing layers of neural

network neurons—interconnected nodes—in neural networks. Several hidden layers can exist in these networks, which is why "deep" learning got its name[18].

Strong machine learning models, Deep Neural Networks (DNNs) excel at solving complex tasks like test recognition. DNN aims to develop a deep and intelligent encoder capable of converting a text into encoded vectors. The encoder is trained to preserve as much information as possible when an input is run through it, and also to make the new representation have various nice properties [19]. By applying neural networks, especially deep architectures, we can achieve automatic text representation and textual entailment recognition.

2.4.7.2. How Deep learning works for textual entailment classification?

It is known that deep learning works with huge amount of data[19]. So, data gathering is the first necessary step in the deep learning approach. Here is some of the steps should be followed.

A. **Data gathering:** Collect a dataset of text pairings and their labels indicating entailments. Make that the dataset is distributed evenly among the various entailment classes (e.g., neutral, entailment, and contradiction).
B. **Data preprocessing:** Perform preprocessing on the text data, like lowercasing, tokenization, punctuation removal, and potentially stemming or lemmatization. Transform the text into numerical formats that neural networks can use as input.
C. **Word embedding**: Use word embedding to represent words in the text; these embedding collect semantic information about words depending on their usage in context. Possible to use pretrained word embedding like Word2Vec, GloVe, or FastText or train them from scratch using the dataset.
D. **Selection of model**: Select a suitable deep learning architecture for the recognition of textual entailments. Commonly used architectures are:
    a. **Recurrent Neural Networks (RNNs):** Captures sequential information in text.
    b. **Long Short-Term Memory Networks (LSTMs)**: has ability in capturing long-range dependencies.
    c. **Gated Recurrent Units (GRUs)**: Simplified versions of LSTMs with fewer parameters.

d. **Convolutional Neural Networks (CNNs):** Effective at capturing local patterns in text.

e. **Transformer-based architectures** (e.g., BERT): State-of-the-art models for contextualized embedding and understanding textual relationships.

Many deep learning models have power to work for textual entailment. As the cases are described in Chapter 3, section 3.4.2, we selected CNN, LSTM and BiLSTM for our study.

E. Model design: it involves designing the neural network architecture for textual entailment recognition. This consists of combining layers such as embedding layers, recurrent layers, convolutional layers, attention mechanisms, and fully connected layers.

F. Model training: the model should be trained using the training dataset. During training, the model learns to predict the entailment relationship (e.g., entailment, contradiction, or neutral) based on the input text pairs.

G. Model evaluation: the trained model should be evaluated on a separate validation set to assess its performance. Common evaluation metrics include accuracy, precision, recall, F1-score, and confusion matrices are used. For our model, we selected Accuracy, Precision, recall, F1-score and Confusion matrix.

H. Model testing: Evaluate the final trained model on a test set to obtain an unbiased estimate of its performance.

2.5. Afaan Oromoo language

One of the main languages used and spoken in Horn of Africa and especially with many speakers in Ethiopia is Afaan Oromoo. It is currently the official language of the Oromia regional state[20]. The Oromo people, who make up 35.8% of Ethiopia's total population and are the country's largest ethnic group, use it[21]. This information was obtained from the 2007 census. Regarding the writing system, "Qubee" has been accepted and used as the official Afaan Oromoo script since 1991[20][21]. In Ethiopia, Afaan Oromoo is currently a commonly spoken and written language. Afaan Oromo is the medium of instruction in primary and junior secondary schools throughout the region and its administrative zones, in addition to being the official working language of Oromia regional State[21]. It is also listed as the department in many Ethiopian universities. As a result, the language has a well-established, standardized structure for both writing and speaking [22].

2.5.1. Afaan Oromoo Alphabets

Afaan Oromoo is a language that it is spoken in the way it is written[23]. Thus, we can call it as phonetic language. It uses a Latin scrip like English language and other Latin-based languages. Like as English, Afaan Oromoo has the same vowels and consonants. A, E, I, O, U are the vowels found in Afaan Oromoo language. There are some few additional combinations of Qubee's like "**ch**", "**dh**", "**ny**", "**sh", "ph", "ts"** called "Qubee Dachaa" in Afaan Oromoo is added to the consonants available in English language. We use them commonly, for example, to say **hubachiisa '**Notice', **dhadhaa** 'Butter', **nyaata '**Food'**, shamarree '**Girl'**, buuphaa '**Egg'. In general, Afan Oromo has 32 letters, 27 consonants including special combinations and 5 vowels[23].

All Alphabets found in Afaan Oromoo are:

A a	B b	C c	CH ch
D d	DH dh	E e	F f
G g	H h	I i	J j
K k	L l	M m	N n
NY ny	O o	P p	PH ph
Q q	R r	S s	SH sh
T t	TS ts	U u	V v
W w	X x	Y y	Z z

Table 1: Afaan Oromoo Alphabets

2.5.2. Afaan Oromoo punctuation marks

A. **Period** (.): Used in abbreviations and shows the sentences are end.
B. **Question mark** (?): indicates a question or interrogatives.
C. **Exclamation mark** (!): Expresses strong emotion or used at the end of exclamatory sentences.

D. **Comma** (,): separates items in a list or clauses within a sentence, used to divide parts.
E. **Colon** (:): introduces a list, explanation or quotation.
F. **Semicolon** (;): connects related independent clauses
G. **Quotation marks** (“” or ‘’): enclose direct speech or quoted material
H. **Hyphen** (-): join words or parts of words
I. **Ellipsis** (…): indicates omission of words or a pause
J. **Apostrophe (‘)**: used when two different vowels happened together.

2.5.3. Afaan Oromoo Morphology

Morphology is the study and description of word formation in a language within the field of linguistics. In Afaan Oromoo language, we have two types of morphology. They are:

A. **Derivational morphology**: It is a process where new words are created. Words in Afaan Oromoo are formed through derivational methods. These procedures entail endowing root words with prefixes, suffixes, or infixes.
B. **Inflectional morphology**: Rather than coining new words, it is the process of giving already-existing terms additional meaning. Words undergo inflectional modifications according to grammatical characteristics like as gender, number, and tense. Nouns and verbs have complex inflections.

2.6. Related works

2.6.1. Ethiopian Language Textual Entailment

Afaan Oromoo text classification is one of the Natural Language Processing applications used to classify a group of text documents into specified categories [24]. According to Sena et.al [24], 2125 news text were used for multi-label Afaan Oromoo text classification using deep learning approach. They used LSTM model. They obtained 96.07% training accuracy and testing accuracy 90.19%.

Helina Mesfin[25], developed “**Amharic textual entailment classification by using deep learning approach”**. They used reverse side sentence matching model with five main layers. Word embedding, sentence embedding, sentence matching, aggregation and prediction layers are the layers used. In sentence embedding layer they used Bi-LSTM to remember long sentence sequences of the dataset. They used pair of 8700 sentences for Amharic textual entailment and their model produced scoring 77% accuracy for training dataset and 72% accuracy for test dataset accuracy.

2.6.2. Non-Ethiopian Language Textual Entailment

A. English language Textual Entailment

Han et.al [26], in their paper entitled as "**Recognizing Textual Entailment using Deep Learning Techniques"** they used the character-level convolutional network to replace the standard word-level embedding layer, and they used the intra attention layer to capture the intra-sentence semantics. They demonstrated the CIAN model, a sequence encoder with extensive semantic and orthographic properties that can encode lengthy sentences at the character level. Furthermore, the attention mechanism makes the model highly interpretable, making it possible for users to comprehend how the model operates. In the matched test set, the model's accuracy was 90%, and in the mismatched test set, it was 60%.

In the paper of Lyu et.al[11], they developed a novel two-step procedure to recognize textual entailment. In order to learn the joint representation of the text-hypothesis pairings, they first constructed a joint layer of Restricted Boltzmann Machines (RBM). Next, in order to identify textual entailment for each pair, the reconstruction error is computed by comparing the original representation with the rebuilt representation obtained from the joint layer. A sizable news corpus is automatically used to create the joint RBM training data. The findings of the experiment demonstrate how the concept affected textual entailment performance, with accuracy of 67.5% for training pairs and 66.5% for test pairings.

According to Haggag et.al[19], they proposed a "**hybrid approach"** which is based on lexical, syntactic, semantic analysis and Deep Learning classifier entailment module. The labeled texts and hypothesis pairs provided by the Text Analysis Conference (TAC) are used. By training the model with 3567 sample and test the model with 1000 sample it showed 89.8% accuracy.

Another method applied on the textual entailment recognition is a lexical and syntactic features hybrid method. Partha.et.al[27], in their paper entitled as "**A Hybrid Textual Entailment System using Lexical and Syntactic Features"** have described a two-way textual entailment recognition system that uses lexical and syntactic features. They used a hybrid textual entailment system that is Support Vector Machine (SVM) based. They developed the lexical and syntactic hybrid system utilizing 2400 text-hypothesis pairs and a

collection of test gold, development, and annotated sets. The test set evaluation scores revealed that for YES decisions, there was 55.30% precision and 58.40% memory, and for NO decisions, there was 55.93% precision and 52.80% recall. The semantic rules have not been taken into account.

B. Indian language Textual Entailment

Dwijen et.al[28], In their paper they have investigated the idea of "**Partial Textual Entailment for Indian social media text (SMT)"**. They created an annotated corpus for Bengali tweets using Partial Textual Entailment and suggested a Sequential Minimal Optimization (SMO) based method. They employed the recognition technique known as Partial Textual Entailment, or PTE. Their approach was evaluated using experiment findings, and the accuracy of the F measure was 98.3%.

C. Japanese Language Textual Entailment

Quang Nhat et.al[29], they presented a "**machine learning based textual entailment recognition system for Japanese language".** To learn the entailment classifier, they are tagged to a binary class and various entailment features taken from original Japanese pairs and their English translation are combined. Among the participant groups, they achieved a 58% accuracy rate in the Japanese Binary class subtask on the test set. Their research suggested that the RTE system's performance could be enhanced via machine translation.

D. Arabic Language Textual Entailment

Mariam et.al[30], in their paper entitled as "**Textual Entailment for Arabic Language based on Lexical and Semantic Matching**", they used a lexical analysis technique of Textual Entailment to study the suitability of the technique for Arabic language. Furthermore, a semantic matching technique was incorporated to improve the suggested entailment system's accuracy. Word overlap calculations, bigram extraction, and matching provide the foundation of the lexical analysis. To improve word matching accuracy, semantic matching has been combined with word overlap. Using a binary classification, the system was able to classify the relationship between pairs of sentences with a recall of 61% and a precision of 68%, 58%, and 58% for Entails and NotENtails, respectively.

Nada et.al [31], In their paper, they addressed **the problem of textual entailment in Arabic**. They made use of distributional representations in addition to traditional features. When compared to other fundamental surface level matching features, they have demonstrated that the employment of word representation-based features yields a satisfactory outcome. The size of the set they assessed on was constrained. Lastly, there are still issues with the current system, such as the different methods word embedding could be applied. Their method produces a 76.2% accuracy.

Mabrouka et.al[32], they developed "**semantic method for recognizing textual entailment in Arabic".** The directional semantic entailment relationship between text/hypothesis pairings is detected by the model. To be more precise, they concentrated on work at the sentence level, using word sense disambiguation and semantic similarity measures to identify entailment relationships in the context of the Arabic question/answer system. They achieved a 70% accuracy rate.

Summary of some related works in table format

Author	**Title**	**Method**	**Gap**	**Result**
Partha .et.al	A Hybrid Textual Entailment System using Lexical and Syntactic Features [27]	SVM	They have not considered the meaning of words. To enhance the performance of the classifier, more training data is required. It is necessary to upgrade the two-way task to the multi-labelled way task.	The test set's evaluation results revealed that, for YES decisions, there was 55.30% precision and 58.40% recall, while for NO decisions, there was 55.93% precision and 52.80 recall.
Dwijen et.al	Recognition of Partial Textual Entailment for Indian Social Media Text[28]	Sequential Minimal Optimization (SMO) classifier-based PTE recognition approach, Random Forest, Logistic Regression, and Decision	Enhancing the system's resilience and suitability for code-mixed social media text.	Achieved an accuracy of F measure 98.3%.

		Tree.		
Mabrouka et.al	Recognizing Textual Entailment for Arabic using semantic similarity and Word Sense Disambiguation[32]	Semantic similarity and Word Sense Disambiguation	Used less dataset.	The results obtained has accuracy of 70%.
Mariam et.al	Textual Entailment for Arabic Language based on Lexical and Semantic Matching[30]	Based on lexical and semantic matching.	PoS has not been considered in the sentence Word overlap is not calculated.	With an overall recall of 61%, the system has achieved 68% and 58% precision for Entails and NotEntails, respectively.
Helina Mesfin	Amharic Textual Entailment Classification Model Using Deep Neural Network[25]	BiLSTM	They did not use a really large dataset. It is well known that any DL-based NLP work yields more fruitful outcomes the larger the dataset.	77% training and 72% of test accuracy,

Han Yang	Recognizing Textual Entailment using Deep Learning Techniques[26]	CIAN (Character-level Intra Attention Networks)	The model relies only on character-level inputs.	90 % in matched test set, and 60% in mismatched test set.

Table 2: Summary of some related works

2.7. Summary

The overview of Natural Language Processing, textual entailment and different approaches of textual entailment classification, Afaan Oromoo language and some related works with Afaan Oromoo textual entailment classification has been described in this literature review section. The related works done on textual entailment classification are also discussed. From the analysis of the available studies, most of the research has been conducted on highly resourced languages like English, Arabic and also local language Amharic. No one researcher has touched the area of textual entailment recognition for Afaan Oromoo Language.

CHAPTER: THREE

MATERIALS AND METHODOLOGY

3.1. Introduction

The chapter goes over the proposed Deep Learning Approach for Afaan Oromoo Textual Entailment Classification. In order to construct a proposed solution of Afan Oromoo textual entailment classification, we developed a new Afaan Oromoo Textual Entailment Classification (AOTEC) dataset from various Afaan Oromoo texts in this chapter.

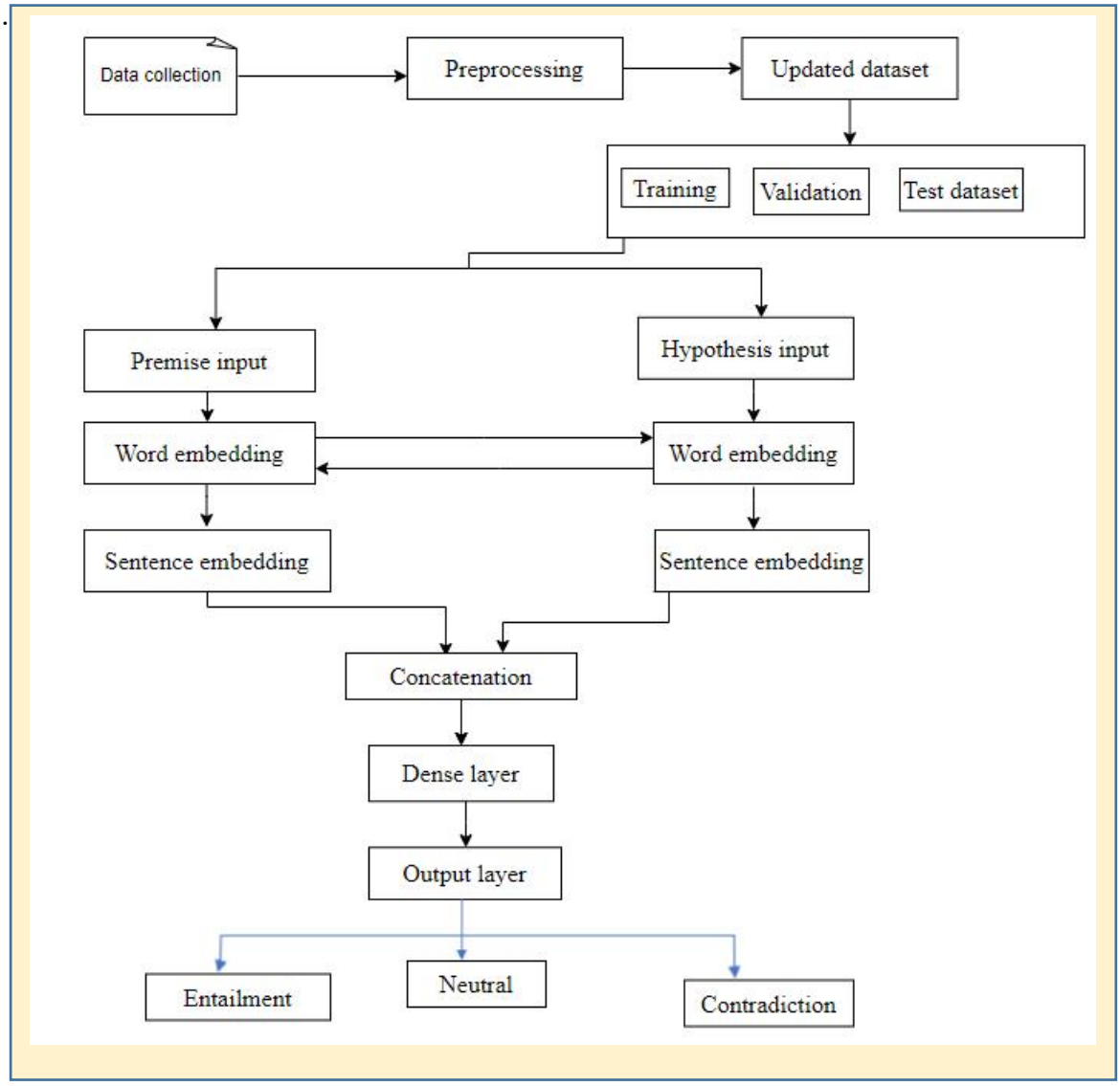


Figure 5: Afaan Oromoo Textual Entailment Classification Architecture

3.2. Data Collection and Preparation

Data collection is a significant task of all for this research. It is about collecting of data that achieves the objectives. In deep learning process, the huge number of data is required to perform the required activity. Because deep learning performs best with larger corpora, the quantity of datasets collected influences the model's performance. For this work the dataset that includes pairs of premises (Sentence 1) and corresponding hypothesis (Sentence 2) labelled with their entailment relationship (e.g., entailment, contradiction, or neutral) should be gathered from different Afaan Oromoo texts, like: Newspaper, Websites and others. With the help of Afaan Oromoo experts, the collected data is annotated to its appropriate labels. After annotation data succeed, the 80% of the dataset is used for training, 10% is used for testing and validating the model.

3.2.1. Data Annotation

The process of labeling data with pertinent information to increase its value for analysis and decision-making is known as data annotation. Data annotation should be followed by the guideline of the language. Here, Afaan Oromoo textual entailment guideline is described in the following subtopic.

3.2.1.1. Afaan Oromoo Textual Entailment Dataset annotation guidelines

It is known that Afaan Oromoo is a highly inflectional language that requires a careful consideration of linguistics experts during the annotation. A set of guidelines should be developed for annotation. The following are some of them.

a) **Clearly defining the terms**

The three classification of entailment relationships are defined clearly. So, providing a clear explanation of entailment, contradiction, and neutral is the most important. The label between two sentences is Entailment when sentence 1 (Premise) supports the sentence 2 (Hypothesis). The label between two sentences is Contradiction when sentence 1 (Premise) contradicts the sentence 2 (Hypothesis). The label between two sentences is Neutral when there is no correlation between two pair of sentences

Example:

Premise (Sentence 1): *Caalaan laaqana nyaateera (Caalaa ate the lunch).*

Hypothesis (Sentence 2): *Caalaan laaqana hin nyaanne (Caalaa didn't eat the lunch).*

Label: *Contradiction*

From the above example, we can understand that, the two sentences contradict each other.

b) **The annotators**

The annotators are an expert of Afaan Oromoo language. They considered linguistic features specific to Afaan Oromoo, such as word order, morphology, and syntactic structures, when annotating entailment relationships. Our Annotators were Afaan Oromoo language lecturers at Mattu University.

c) **The dataset**

The scope and complexity of the dataset is defined, including the types of texts to be included like news articles, social media posts, scientific texts and the level of difficulty of the entailment relationships (e.g., simple vs. complex).

d) **Annotation tool**

The annotators used manual annotation technique. They annotated manually without using any other annotation.

e) **Reliability**

Measuring the consistency of annotations, having feedback session with annotators and etc has been assessed for annotation reliability.

f) **Considerations**

Ethical considerations like privacy, sensitivity, and bias, when selecting and annotating texts for the dataset has been considered.

The guideline is provided to annotators for labelling data into their classification. For our annotation work, we considered the above criteria's and also the synonyms, the antonyms, negation, long sentences, paraphrae and semantic features of the language. Whether the Premise infers or contradicts the hypothesis or no relationship between them is considered.

.

	Premise	Hypothesis	Label
0	Namni tokko farda bo'oo irra utaalchisa.	Namichi dorgommiidhaaf farda isaa leenjisa.	Entailment
1	Namni tokko farda bo'oo irra utaalchisa.	Namichi sun farda isaa bo'oo irra hin utaalchisu.	Contradiction
2	Namni tokko farda bo'oo irra utaalchisa.	Namni tokko ala jira.	Neutral
3	Ijoolleen kolfaa jiru.	Ijoollonni warra isaaniitti kolfaa jiru	Entailment
4	Ijoolleen kolfaa jiru.	Ijoolleen ni argamu	Neutral
...	...	...	...
13055	Namni tokko teessoo mana keessaa irra taa'a.	Namichi mana keessa taa'ee barreeffama dubbisa.	Entailment
13056	Namni tokko teessoo mana keessaa irra taa'a.	Namichi mana keessa taa'e.	Neutral
13057	Namni soofaa irra taa'ee kolfa.	Namichi tokko ala taa'ee boo'a.	Contradiction
13058	Namni soofaa irra taa'ee kolfa.	Namni tokko mana keessa jira.	Neutral
13059	Namni soofaa irra taa'ee kolfa.	Namni soofaa irra taa'ee jiru sun jaarsa.	Entailment

13060 rows × 3 columns

Figure 6: Sample of dataset collected

3.3. Text data preprocessing

One of the most crucial tasks that researchers should take attention of is data preprocessing. After the dataset has been gathered and annotated, this is the first phase. The correctly preprocessed data, results the most accurate than that of not preprocessed incorrectly. Preprocessing may include cleaning and tokenizing text, handling stop words, and other necessary steps. It is about checking anything wrong with data, wrongly labelled data, biased data and etc [33].. let us see in relation with Afaan Oromoo language.

3.3.1. Data cleaning

It involves locating and removing non-textual material from the dataset that contains text. It is also about detecting, removing and correcting incomplete, in accurate, irrelevant and bad data available in the dataset like empty cells, data in the wrong format, wrong data, duplicated data and etc. [34] .

Algorithms used for data cleaning

Input Sentence: Afaan Oromoo Sentences

Output Sentence: cleaned Afaan Oromoo Sentences

I. Read each word in sentences.

II. Divide them up into tokens.

III. If the token is punctuation, stop word, numbers or irrelevant Unicode character conduct replacement.
IV. Join the words to create the original sentences.
V. Return well-formed sentences

3.3.2. Tokenization

Tokenization is a preprocessing technique that involves breaking down a text into individual tokens, often words or sub words [35]. For instance, you might use whitespace tokenization or more advanced methods like character-based tokenization or sub word tokenization for Afaan Oromoo language.

Tokenization Algorithm

Input: Afaan Oromoo sentences

Output: Tokenized sentences

I. Read each sentence from list
II. Convert each character to lower case
III. Then divide them up into tokens
IV. Return list of words

3.3.2. Stop word removal

It is about removing common words that do not contribute much to the meaning of the text. There might be language-specific stop words for Afaan Oromoo that you'd want to remove. Many stop words are available in Afaan Oromoo language, this is why stop word removal is required.

Input: tokenized text and a list which contains stop words

Output: stop words free text

I. Read each sentence from list
II. Check for stop words
III. Then remove them
IV. Return stop word free texts

3.4. Feature Extraction

It is the process that uses preprocessed text as an input. It represents the premise (Sentence 1) and hypothesis (Sentence 2) using suitable feature vectors. This stage is where the

dataset is transformed into a vector of numbers. Common methods include like word embedding.

3.5. Word Embedding

Words are represented as dense vectors of real numbers in a high-dimensional space using a technique called word embedding, which is utilized in machine learning and natural language processing (NLP). These vectors capture semantic relationships between words, enabling algorithms to understand the contextual meaning of words and phrases [36].

Each word in a word embedding is represented by a fixed-length vector, and adjacent words are mapped to adjacent places in the embedding space by similar words. This allows algorithms to leverage the semantic similarity between words, even if they do not appear in the same context in the training data [36].

For example, the system that transforms ‘Man’ into ‘Woman’ will also transforms ‘Queen’ if we apply it to ‘King’.

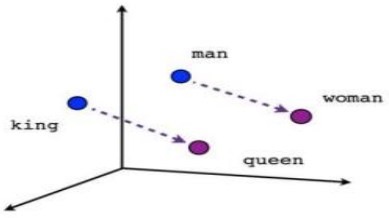


Figure 7: Word embedding example

A. Word2vec

It is the word embedding model that is most widely used. To learn word embedding, it makes use of the CBOW and Skip-gram models. In natural language processing (NLP), this is a fascinating method that gives us vector representations, or embedding, for words. Rich contextual and semantic information about words is captured by these embedding. Let's examine word2vec's two architectures[37].

a. **CBOW (Continuous bag of words):** it looks at the previous words as a context for predictions.

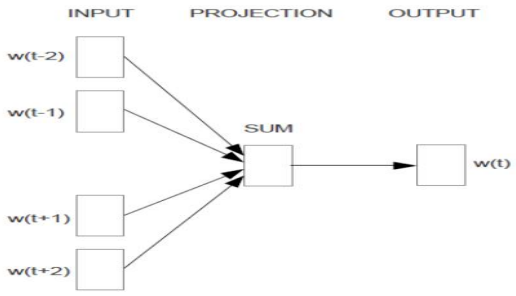


Figure 8: Continuous bag-of-words architecture

b. Skip gram: It classifies a word based on another word in the same sentence rather than guessing the current word based on context.

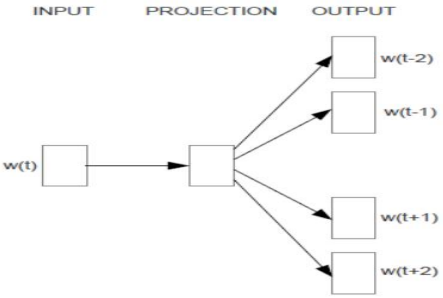


Figure 9: Skip gram architecture

B. Glove (Global bilinear regression method)

Another popular word embedding technique in natural language processing (NLP) is called GloVe, short for Global Vectors for Word Representation. GloVe, created by Stanford University academics, is intended to gather a corpus's global co-occurrence statistics and generate dense vector representations for each word based on these facts [38].

The key idea behind GloVe is to factorize a matrix of word co-occurrence statistics in order to learn word embedding. The frequency with which words occur together inside the corpus within a specific window size is represented by this matrix. GloVe learns embedding that store syntactic and semantic information about words by optimizing a loss function that compares word vectors and their co-occurrence statistics[38].

Unlike methods like Word2Vec, which focus on predicting individual words based on their context, GloVe directly leverages the co-occurrence statistics of words across the entire

corpus. This approach allows GloVe embedding to capture global word relationships more effectively [38].

C. Sub word embedding

Facebook's AI Research (FAIR) lab developed a framework called FastText for effective word representation and text classification learning. PyTorch, Facebook's open-source deep learning technology, serves as the foundation for FastText [39].

FastText is a technique for word embedding that learns distributed representations of words based on the distributional characteristics of words in a text corpus. The FastText algorithm extends the traditional word2vec approach by considering sub word information, which makes it particularly effective for handling rare words and morphologically rich languages. It is very crucial for capturing the meaning and morphology of words, especially for out-of-vocabulary entries[39].. For our work, we selected to compare sub word embedding with word2vec for word embedding.

How we use sub word embedding?

FastText represents each word in the vocabulary as an embedding, or fixed-length vector, of real numbers. These embedding's, which are based on the distributional features of words in the corpus, encapsulate syntactic and semantic information about words [39].

FastText considers sub word information, particularly character n-grams. Instead of representing words as atomic units, FastText decomposes words into a set of character n-grams (substrings of characters) to capture morphological and structural information. For example, the word "Nama" can be decomposed into character n-grams {"<Na", "am", "ma>"} for n=2.

FastText learns word embeddings by utilizing a shallow neural network model that has been trained on a sizable corpus of text. During training, the model predicts the context (neighboring words) of each word based on its embedding and the embeddings of its neighboring character n-grams.

FastText can handle words that are not in its vocabulary. It represents them as the total of their character n-gram embeddings. Because of this, FastText is resilient to uncommon or

unseen words and can produce embeddings even for words that were not seen during training.

Parameters	Description	Values
Window	Size of context window	3
Vector-size	Size of word vectors	300
Sg	Skip-gram	1
Epoch	Number of epochs	10
Min-count	Minimal number of occurrences	1

Table 3: Parameters for training word embedding using fasttext library

The parameters used are described as follows.

a. Vector-size

The word vectors' dimensionality is specified by the vector-size parameter. In this case, a 300-dimensional vector is used to represent each word or sub word. Higher dimensionality typically increases computational complexity but also enables the model to capture more detailed information about the words. 300 is a choice that strikes a balance between efficiency and intricacy.

b. Window

The maximum distance in a sentence between the current and predicted words is defined by this parameter. Here, the model predicts context by examining the three words that come before and after a particular word. While a bigger window can capture a broader context, a smaller window concentrates on closer word associations (local context). In this instance, window=3 suggests a focus on close contextual relationships.

c. Min-count

It specifies the minimum number of occurrences a word must have in the corpus to be considered for training. Words that appear fewer times than min-count are ignored. Making min-count=1, ensures that even rare words are included in the model, which can be important for languages with less training data or for specific domains where rare terms might be significant.

d. Skip-gram

The training algorithm to be used is determined by the skip-gram parameter. For sg = 1, the model employs the Skip-gram method; for sg = 0, the Continuous Bag of Words (CBOW) method is employed. Since skip-gram (sg=1) attempts to anticipate adjacent words given the center word, it is generally better at learning representations for uncommon words. Because of this, it works well with languages like Afaan Oromoo, which have complex morphology and several word forms.

e. Epochs

The parameter indicates how many times the full corpus will be iterated (or passed through) during training. The model can learn more about the connections between words with more epochs, but an excessive number can cause overfitting. The model will iterate over the corpus ten times in this case, as indicated by the epoch = 10, which is typically sufficient for many jobs without overfitting.

3.6. Sentence Embedding

Sentence embedding is a technique similar to word embedding but applied to entire sentences or phrases instead of individual words. It involves representing entire sentences or phrases as fixed-length vectors in a high-dimensional space, capturing the semantic meaning and contextual information of the entire text segment[40]. Sentence embedding techniques, which convert vector input words into sentence representations, is discussed in this section. Textual entailment requires a sentence embedding approach that can able to capture the text semantics and their relationships. Based up on their effectiveness for local patterns, sequential dependencies and contextual information extraction, these three approaches (CNN, LSTMs and Bi-LSTM) are used. Another BERT model is available for sentence embedding feature extraction. It needs a pre-trained huge amount of dataset and requires high processor CPU [41] [42] [43][44]. So, due to BERT model needs pre-trained massive amount of dataset and requires high processor (the researcher uses Intel CORE i3), the researcher has not used it for the purpose. It is known that, Afaan Oromoo is a low-resourced language, which the researcher has collected dataset manually to achieve the objective.

3.6.1. CNN (Convolutional Neural Network)

Convolutional neural network (or CNN) is a one of the type of a multilayer neural network or deep learning architecture that originally invented for computer vision, is showing strong performance on text classification tasks [40]. In textual entailment classification, Convolutional neural networks (CNNs) are used to automatically learn features from textual input and predict the entailment relationship between two text fragments (often a premise and a hypothesis) [41].

In Afaan Oromoo textual entailment classification, CNN is used to capture local patterns and features within sentences. But, CNN needs further model optimization to be applied with higher level representations [41]**.**.

For our Afaan Oromoo textual entailment classification, we used CNN as follows.

Convolutional layers were first applied to the input embedding. Next, in order to identify the most relevant iinformation and lower the dimensionality of the feature maps, we used pooling layers (max pooling). We additionally used the dropout on the convolutional outputs to avoid overfitting. Here, the premise and hypothesis's outputs ought to be combined. Thus, the CNN-encoded representations were concatenated. Additionally, the concatenated vector passes through dropout-filled dense layers. For classification, we added a last dense layer with softmax activation at the end.

The architecture is as follows.

a. **Input Layers**: Make two layers of input: one for the hypothesis and one for the premise.
b. **Embedding Layer**: The embedding layer, initialized with the FastText embedding matrix, transforms the sequences of integers into dense vectors of fixed size (e.g., 300 dimensions). This embedding layer is shared between the premise and hypothesis.
c. **Convolutional Layers**: Apply one-dimensional convolutional layers to the embedded sequences. Convolutional layers slide filters over the input sentences to detect local patterns, such as n-grams or specific phrases.

d. **Max Pooling**: Use max pooling to lower the dimensionality and extract the salient features after convolution. Pooling layers facilitate data downsampling while preserving the most important details.
e. **Flatten**: All of the identified features are combined into a single vector by flattening the pooled outputs.
f. **Concatenation**: Concatenate the flattened outputs from the premise and hypothesis CNNs. This step combines the feature representations of both sentences.
g. **Dense and Output Layers**: To avoid overfitting, the concatenated vector is passed through two layers: a dense layer and a dropout layer. Finally, a probability distribution encompassing the classes (Entailment, Neutral, and Contradiction) is generated by an output layer utilizing a softmax activation function.

3.6.2. LSTM (Long Short-Term Memory)

LSTM stands for Long Short-Term Memory.It is a specific kind of recurrent neural network (RNN) architecture meant to solve the vanishing gradient issue with conventional RNNs, which makes them challenging to train on lengthy data sequences [42].

For handling sequential data, LSTM is designed. Within sentences, it captures context and long-range dependencies. Possess input, output, and forget gates, three in all.

For our Afaan Oromoo textual entailment classification, we used LSTM as follows.

a. **Input Layers**: For the premise and the hypothesis sentences, two distinct input layers are made.
b. **Embedding Layer**: The embedding layer, initialized with the FastText embedding matrix, converts the sequences of integers into dense vectors of fixed size (e.g., 300 dimensions). This layer is shared by both the premise and hypothesis inputs.
c. **LSTM Layers**: Instead of using a bidirectional LSTM, a unidirectional LSTM layer processes the sequences from the premise and hypothesis separately. The LSTM processes the sequence in a single direction, capturing dependencies based on the order of words.

d. **Concatenation**: The outputs from the LSTM layers corresponding to the premise and hypothesis are concatenated. This combined vector captures the relationship between the two sentences.
e. **Dense and Output Layers**: To avoid overfitting, the concatenated vector is first run through a dense layer and then a dropout layer is added. Lastly, the probability distribution for each of the three classes—Entailment, Neutral, and Contradiction—is predicted by the output layer using a softmax activation function.

3.6.3. Bi-LSTM (Bidirectional Long Short-Term Memory)

Bi-LSTM stands for Bidirectional Long Short-Term Memory. It is an extension of the conventional LSTM architecture that makes use of data from past and future time steps to generate predictions. The network's ability to comprehend and represent sequential input is enhanced by this bidirectional capability, which enables it to record dependencies in both directions [43].

BiLSTM model improves LSTM by taking into account the context of the past and future. Sequences entered are processed both forward and backward. especially helpful for comprehending the context of Afaan Oromoo sentences in order to classify entailments.

For our Afaan Oromoo textual entailment classification, we used BiLSTM model as follows.

a. **Input Layers**: Two input layers are created, one for the premise and one for the hypothesis. Each input corresponds to a sequence of tokenized words.
b. **Embedding Layer**: The embedding layer is initialized with the embedding matrix created from FastText. It converts the integer sequences (tokenized text) into dense vectors of fixed size (e.g., 300 dimensions).
c. **Bidirectional LSTM**: The BiLSTM layer, which processes the sequence both forward and backward, is the core of the model. This gives the model greater capability than a unidirectional LSTM by enabling it to gather data from both past and future contexts.
d. **Concatenation**: The outputs from the premise and hypothesis BiLSTM layers are concatenated. This combines the encoded representations of both sentences, which is crucial for capturing the relationship between them.

e. **Dense and output layers**: To avoid overfitting, the concatenated vector is passed through two layers: a dense layer and a dropout layer. Finally, probabilities for each class—Entailment, Neutral, and Contradiction—are generated by an output layer using a softmax activation function.

3.6.4. Drop out

In order to avoid overfitting, dropout is a regularization technique used in neural networks, such as CNN, LSTM, and BiLSTM models. When a model works well on training data but not well on unseen data due to overfitting, it performs poorly on test or validation sets. By introducing noise during training, dropout helps to reduce this problem by pushing the model to pick up more resilient and broadly applicable features.

It functions by releasing neurons randomly. A percentage of the network's neurons are randomly dropped (i.e., set to zero) by dropout throughout each training cycle. This indicates that since any neuron can be removed at any point during training, the model is not dependent on any particular neurons.

In CNN, it is used for:

a. **Feature Detectors**: CNNs are designed to detect features in the input data through convolutional filters. Without dropout, the network might become overly reliant on certain filters or features, which could lead to overfitting.
b. **Regularization**: By preventing the network from being overly dependent on any one set of filters, dropout regularizes the CNN and encourages it to acquire a wider range of characteristics. This variability aids in the model's improved generalization to fresh, untested data.

In LSTM and BiLSTM it is Used for:

a. **Sequential Data**: When dealing with sequential data, like text, LSTMs and BiLSTMs are frequently employed since they require the model to retain past inputs in order to anticipate future ones. Over time, this capacity to retain information might result in complex models that may overfit the training set.
b. **Overcoming Overfitting**: Dropout ensures that the model does not grow overly dependent on particular memory routes or connections, which helps prevent

overfitting in LSTMs and BiLSTMs. This is especially crucial for long-sequence models, as overfitting might result in poor generalization.

c. **Variational Dropout**: In LSTMs and BiLSTMs, dropout can be applied not just to the outputs of the LSTM layers but also to the recurrent connections. This is often done using a technique called "Variational dropout," where the dropout mask is shared across time steps, ensuring that the same neurons are dropped out at every time step.

3.7. Concatenation Layer

Concatenation is a powerful technique used in CNNs, LSTMs, and BiLSTMs to combine different types of information into a single, comprehensive representation. In CNNs, it allows the model to merge features from different filters or layers. In LSTMs and BiLSTMs, it helps integrate contextual information from different directions or sequences. This combined representation is essential in tasks for textual entailment, where understanding the relationship between two sequences is key.

In CNNs, it is combining the output from layers. To combine the low-level and high-level features, outputs are concatenated. This can assist the model in producing more accurate predictions by taking into account data from various angles.

In LSTMs, it is combining forward and backward passes. In standard LSTM models, concatenation is often used to combine the hidden states from different layers or time steps. For instance, when processing a sequence, the hidden state from the last time step might be concatenated with additional features or other hidden states to form a richer representation.

Also, it integrates the contextual information. In textual entailment, where two sequences (premise and hypothesis) are processed separately, their LSTM-encoded representations can be concatenated. This combines the information from both sequences into a single vector that represents the relationship between them.

In BiLSTMs, it is combining forward and backward hidden states. BiLSTMs produce two sets of hidden states by processing sequences both forward and backward. In order to create a single vector that captures context from both directions, these forward and backward hidden states are frequently concatenated at each time step.

Also, it is creating richer sentence representations. Concatenated forward and backward hidden states, which capture information from both past and future contexts, offer a more comprehensive comprehension of the sentence when utilizing BiLSTM for sentence embedding. This is particularly important in tasks like textual entailment, where understanding the entire context of both sentences is crucial.

For our Afaan Oromoo textual entailment, after we encoded the premise and hypothesis separately with BiLSTMs, their final hidden states are concatenated to combine the information from both sentences. This combined representation is then used to determine the relationship between the two sentences.

3.8. Dense layer and output layer

3.8.1. Dense layer

The output layer and the dense layer are critical in Afaan Oromoo textual entailment classification because they convert the learned features from CNN, LSTM, or BiLSTM models into a final determination of the relationship between the premise and hypothesis (Entailment, Neutral, Contradiction). The dense layer, which is a fully connected layer, is an important step prior to classification because it enables the model to combine and refine features in a non-linear manner. The input features are transformed into a new feature space by the dense layer using a non-linear activation function (ReLU), a weight matrix, and a bias vector. This aids the model in identifying intricate patterns in the data.

3.8.2. Output Layer

The ultimate classification choice is made by the output layer. It predicts the probability distribution over the possible classes (Entailment, Neutral, and Contradiction) using the transformed features from the dense layer.

Softmax Activation: A softmax activation function is commonly used in the output layer to transform the outputs into a probability distribution. Each element in this distribution represents the likelihood that the input sequence pair (premise and hypothesis) belongs to a specific class. the softmax function is commonly used in textual entailment classification tasks to compute the probability distribution over the possible classes (e.g., entailment, contradiction, neutral) for a given pair of sentences [45].

Here is the summary of Dense and Output layers of CNN, LSTM and BiLSTM models.

1. CNN (Convolutional Neural Network)

a) **Dense Layer**: The feature maps are usually flattened into a 1D vector after the convolutional and pooling layers. After that, this vector is run through one or more dense layers in order to acquire an abstract, high-level representation of the input text.

b) **Output Layer**: An output layer with a softmax activation function comes after the dense layer and generates a probability distribution across the classes. For the final categorization to be made, the output layer is essential.

2. LSTM (Long Short-Term Memory

a. **Dense Layer**: The output from the last time step or the final hidden states are sent to a dense layer in LSTM models following the processing of the sequences (premise and hypothesis). The learnt sequence representations are improved and made ready for categorization with the aid of this layer.

b. Output Layer: The LSTM model's output layer classifies the relationship between the two sentences using a softmax activation function, just like the CNN does.

3. BiLSTM (Bidirectional LSTM)

a. **Dense Layer**: In BiLSTM models, the hidden states from the forward and backward processing of the sequences are combined after processing. Then, one or more dense layers are traversed using this concatenated vector. For improved classification, the dense layer aids in combining and honing the forward and backward context information.

b. **Output Layer**: The probability distribution over the possible classes is obtained by feeding the output of the final dense layer into an output layer that is activated by softmax. The ultimate forecast of the connection between the premise and the hypothesis is made by this layer.

3.9. Development tools

The researcher used Intel(R) Core (TM) i3-7020U CPU @ 2.30GHz 2.30 GHz, 4GB of RAM, and 500GB of hard disk space and run on 64-bit Microsoft Windows 10. To conduct the analysis, Python, which is an open-source programming language is used along with various libraries such as TensorFlow, keras, and genism.

3.10. Performance Evaluation

Deep learning approach employ various algorithms, each suited to specific tasks. While no algorithm is perfect, some are particularly effective for certain applications. Developing a comprehensive understanding of the key algorithms is essential for making informed decisions.

Different methods are used to evaluate the performance of a deep learning model. For example, **BLUE score** is used to measure the quality of machine-generated text (e.g., in translation, summarization, or text generation) by comparing it to one or more reference texts.

A performance metric for classification models that shows how well the model predicts different classes is evaluated using **Confusion matrix. Confusion matrix** is one of the tools for assessing a model's performance. By comparing the model's predictions to the actual target values, it provides an overview of the model's performance. It is commonly used to evaluate the performance of classification models [46].

The confusion matrix for the three classes (Entailment, Neutral, and Contradiction) in the Afaan Oromoo textual entailment classification will be a 3x3 matrix. The predicted class is represented by each column, and the actual class is represented by each row. Here's how the confusion matrix is structured:

Actual/Predicted	**Predicted Entaiment**	**Predicted Neutral**	**Predicted Contradiction**
Actual Entailment	True Entailment (TE)	False Neutral (FN)	False Contradiction (FC)
Actual Neutral	False Entailment (FE)	True Neutral (TN)	False Contradiction (FC)
Actual Contradiction	False Entailment (FE)	False Neutral (FN)	True Contradiction (TC)

Table 4: Confusion matrix table

Descriptions of the terms are as follows.

- **True Entailment** (TE): The number of instances correctly predicted as Entailment.
- **True Neutral** (TN): The number of instances correctly predicted as Neutral.
- **True Contradiction** (TC): The number of instances correctly predicted as Contradiction.
- **False Neutral** (FN): The number of Entailment or Contradiction instances incorrectly predicted as Neutral.
- **False Entailment** (FE): The number of Neutral or Contradiction instances incorrectly predicted as Entailment.
- **False Contradiction** (FC): The number of Entailment or Neutral instances incorrectly predicted as Contradiction.

CHAPTER: FOUR

RESULT AND DISCUSSION

4.1. Introduction

For the Afaan Oromoo textual entailment classification, we have performed a comparative analysis using various deep learning models. These models are:

a. BiLSTM: Bidirectional Long Short-Term Memory

b. LSTM: Long Short-Term Memory

c. CNN: Convolutional Neural Network

4.1.1. Bidirectional Long Short-Term Memory

The BiLSTM model for AOTEC has been summed up as follows.

Model: "functional_1"

Layer (type)	Output Shape	Param #	Connected to
premise_input (InputLayer)	(None, 23)	0	-
hypothesis_input (InputLayer)	(None, 23)	0	-
embedding_1 (Embedding)	(None, 23, 300)	1,607,700	premise_input[0][0], hypothesis_input[0][0]
bidirectional_3 (Bidirectional)	(None, 23, 256)	439,296	embedding_1[0][0], embedding_1[1][0]
dropout_2 (Dropout)	(None, 23, 256)	0	bidirectional_3[0][0], bidirectional_3[1][0]
bidirectional_4 (Bidirectional)	(None, 256)	394,240	dropout_2[0][0]
bidirectional_5 (Bidirectional)	(None, 256)	394,240	dropout_2[1][0]
concatenate_1 (Concatenate)	(None, 512)	0	bidirectional_4[0][0], bidirectional_5[0][0]
dense_2 (Dense)	(None, 64)	32,832	concatenate_1[0][0]
dropout_3 (Dropout)	(None, 64)	0	dense_2[0][0]
dense_3 (Dense)	(None, 3)	195	dropout_3[0][0]

Total params: 2,868,503 (10.94 MB)
Trainable params: 1,260,803 (4.81 MB)
Non-trainable params: 1,607,700 (6.13 MB)

Figure 10: Summary of BiLSTM model for AOTEC

4.1.2. Convolutional Neural Network

The CNN model for AOTEC can be summed up as follows.

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
premise_input (InputLayer)	(None, 23)	0	-
hypothesis_input (InputLayer)	(None, 23)	0	-
embedding (Embedding)	(None, 23, 300)	1,607,700	premise_input[0][0], hypothesis_input[0][0]
conv1d (Conv1D)	(None, 19, 128)	192,128	embedding[0][0], embedding[1][0]
max_pooling1d (MaxPooling1D)	(None, 9, 128)	0	conv1d[0][0], conv1d[1][0]
dropout (Dropout)	(None, 9, 128)	0	max_pooling1d[0][0], max_pooling1d[1][0]
global_max_pooling1d (GlobalMaxPooling1D)	(None, 128)	0	dropout[0][0]
global_max_pooling1d_1 (GlobalMaxPooling1D)	(None, 128)	0	dropout[1][0]
concatenate (Concatenate)	(None, 256)	0	global_max_pooling1d[_ global_max_pooling1d_…
dense (Dense)	(None, 64)	16,448	concatenate[0][0]
dropout_1 (Dropout)	(None, 64)	0	dense[0][0]
dense_1 (Dense)	(None, 3)	195	dropout_1[0][0]

Figure 11: Summary of CNN model for AOTEC

4.1.3. Long Short-Term Memory

The LSTM model for AOTEC can be summed up as follows.

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
premise_input (InputLayer)	(None, 23)	0	-
hypothesis_input (InputLayer)	(None, 23)	0	-
embedding (Embedding)	(None, 23, 300)	1,607,700	premise_input[0][0], hypothesis_input[0][0]
lstm (LSTM)	(None, 23, 128)	219,648	embedding[0][0], embedding[1][0]
dropout (Dropout)	(None, 23, 128)	0	lstm[0][0], lstm[1][0]
lstm_1 (LSTM)	(None, 128)	131,584	dropout[0][0]
lstm_2 (LSTM)	(None, 128)	131,584	dropout[1][0]
concatenate (Concatenate)	(None, 256)	0	lstm_1[0][0], lstm_2[0][0]
dense (Dense)	(None, 64)	16,448	concatenate[0][0]
dropout_1 (Dropout)	(None, 64)	0	dense[0][0]
dense_1 (Dense)	(None, 3)	195	dropout_1[0][0]

Total params: 2,107,159 (8.04 MB)
Trainable params: 499,459 (1.91 MB)
Non-trainable params: 1,607,700 (6.13 MB)

Figure 12: Summary of LSTM model for AOTEC

4.2. Experimental setting

Models	Hyper parameter	Value
CNN LSTM BiLSTM	Activation function for output	Softmax function
	Optimizer	Adam
	Batch size	32, 64
	Embedding dimension	300
	Number of units in LSTM hidden units	128
	Pool-size in MaxPooling	2
	Dropout rate	0.5
	Number of units in Dense layer	64

	Number of classes in output layer	3
	Activation function in dense layer	ReLu
	Loss function	Categorical cross-entropy
	Learning rate	0.001
	Number of epochs	10, 20

Table 5: Summary of hyper parameters used

4.3. Training and Validating models

In order to assess the model's performance, our deep learning models require training on the Afaan Oromoo textual entailment classification dataset. 13060 pairs of sentences with labels have been gathered.

```
# Split the dataset into training, validation, and test sets
X = np.hstack((X_premise, X_hypothesis))
y = data['Label'].values
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.2, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
```

Figure 13: Splitting data into training, validation and test dataset

We used 80% of the gathered dataset for training, 10% for validation, and 10% for testing. Three models—CNN, LSTM, and BiLSTM—have been trained, and their accuracy has been evaluated.

Different scenarios have been considered during experiment by using both fasttext and word2vec for word embedding.

a. Batch size 64 with 10 epoch
b. Batch size 32 with 10 epoch
c. Batch size 64 with 20 epoch
d. Batch size 32 with 20 epoch

A. Model 1: BiLSTM

a) BiLSTM with fasttext with 20 epoch, batch size 64

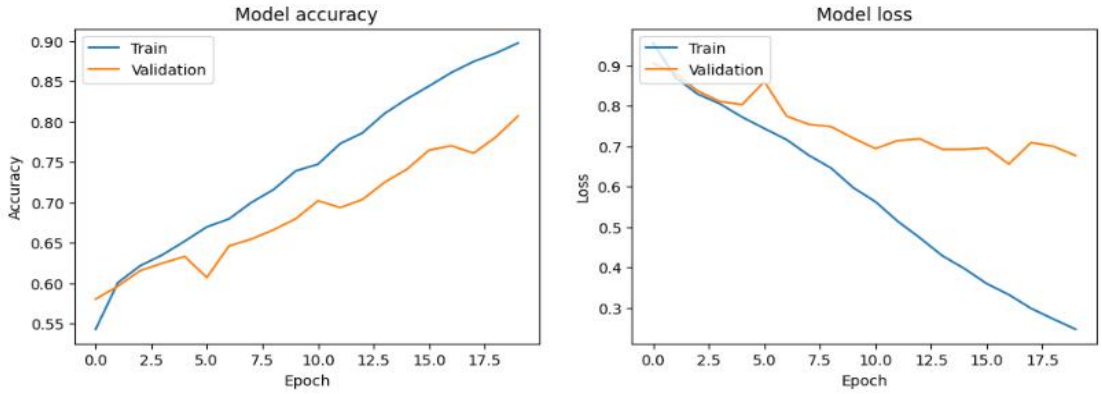


Figure 14: BiLSTM with fasttext with 20 epoch and batch size 64

b) BiLSTM with word2vec with 20 epoch, batch size 64

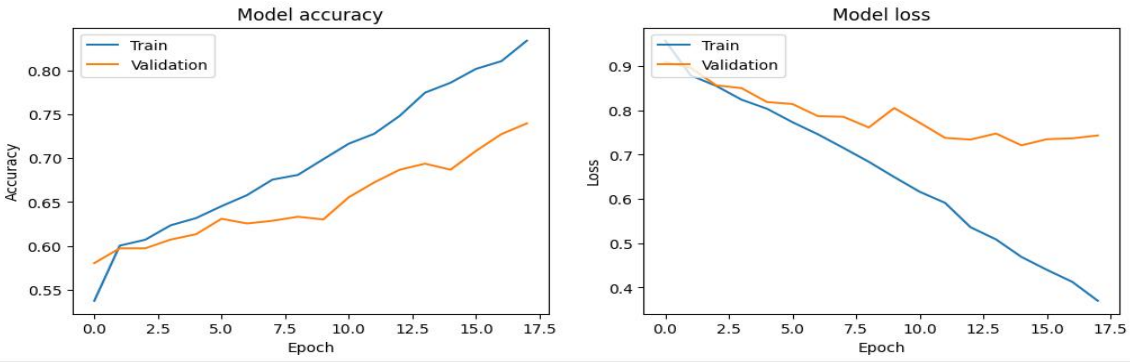


Figure 15: BiLSTM with word2vec with 20 epoch and batch size 64

c) BiLSTM with fasttext with epoch 20 and 32 batch size

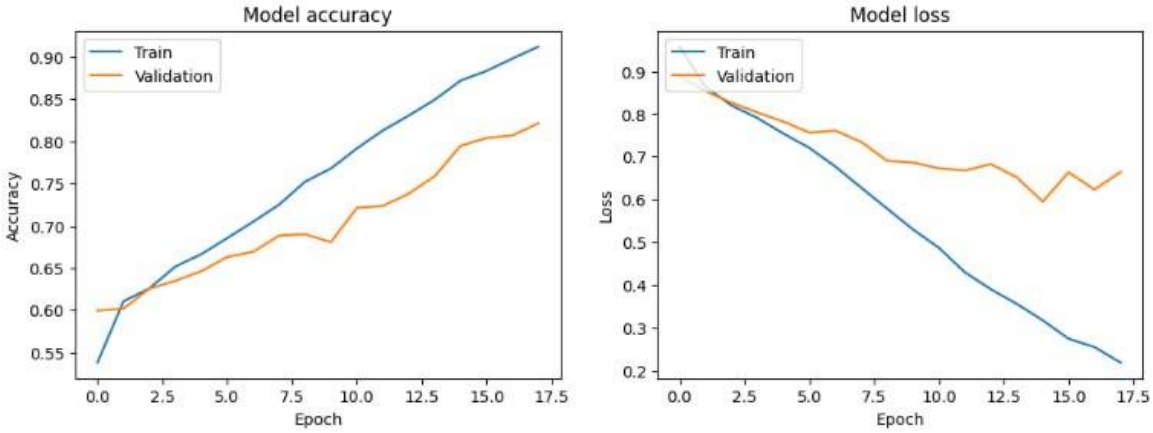


Figure 16: BiLSTM with fasttext with 20 epoch and batch size 32

d) BiLSTM with word2vec with epoch 20 and 32 batch size

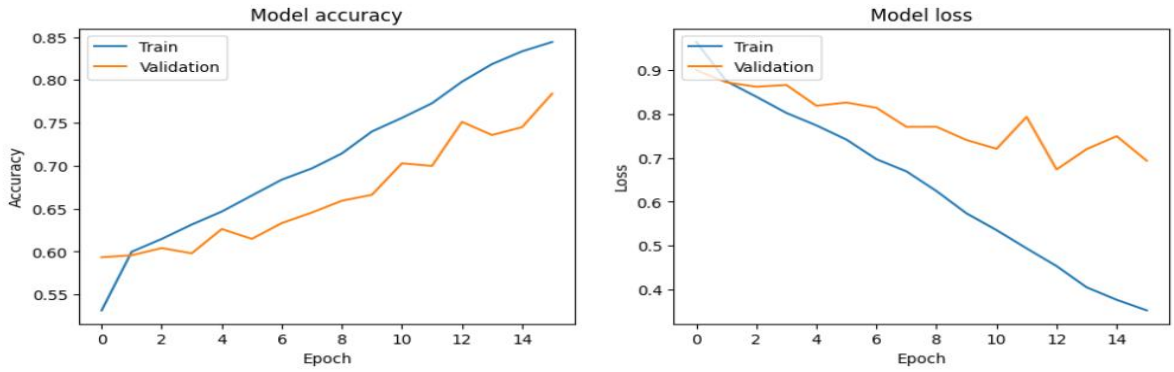


Figure 17: BiLSTM with word2vec with epoch 20 and 32 batch size

e) BiLSTM with fasttext with 10 epoch and 32 batch size

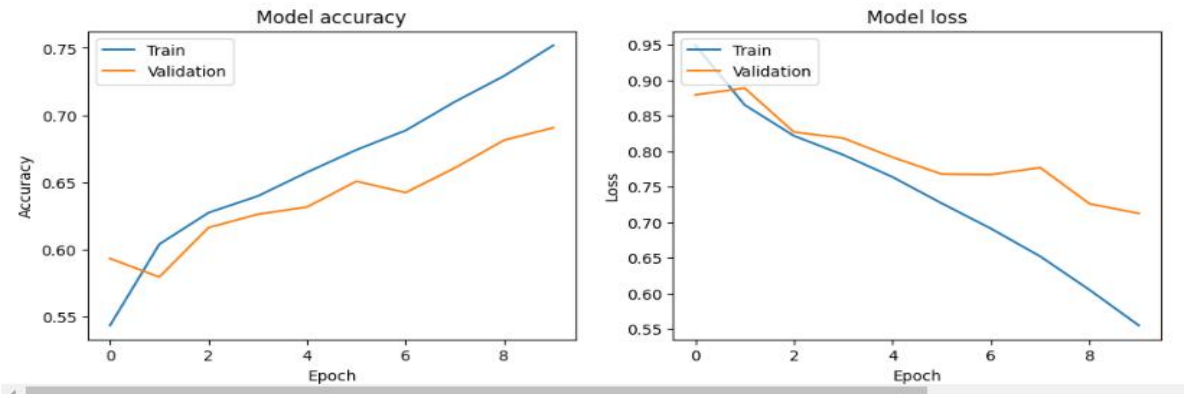


Figure 18: BiLSTM with fasttext with 10 epoch and batch size 32

f) BiLSTM with word2vec with 10 epoch and 32 batch size

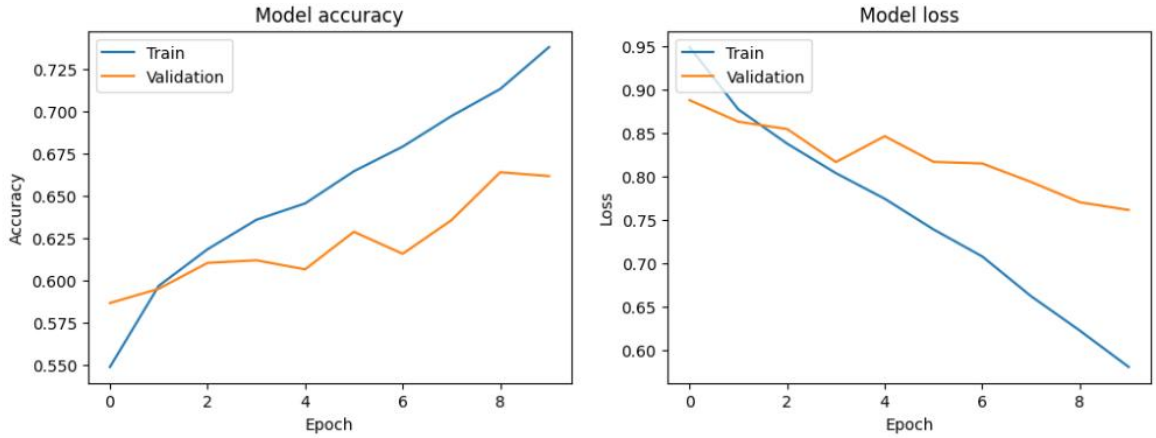


Figure 19: BiLSTM with word2vec with 10 epoch and 32 batch size

g) BiLSTM with fasttext with 10 epoch and 64 batch size

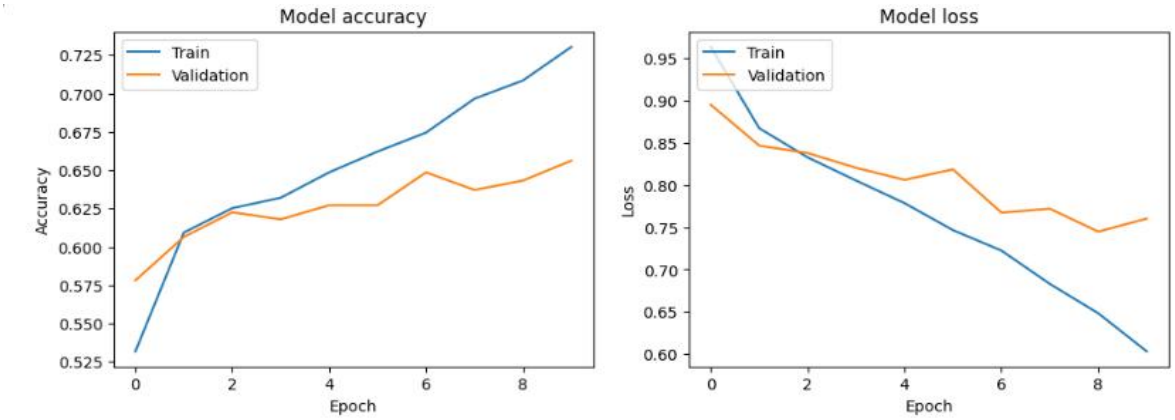


Figure 20: BiLSTM with fasttext with 10 epoch and batch size 64

h) BiLSTM with word2vec with 10 epoch and 64 batch size

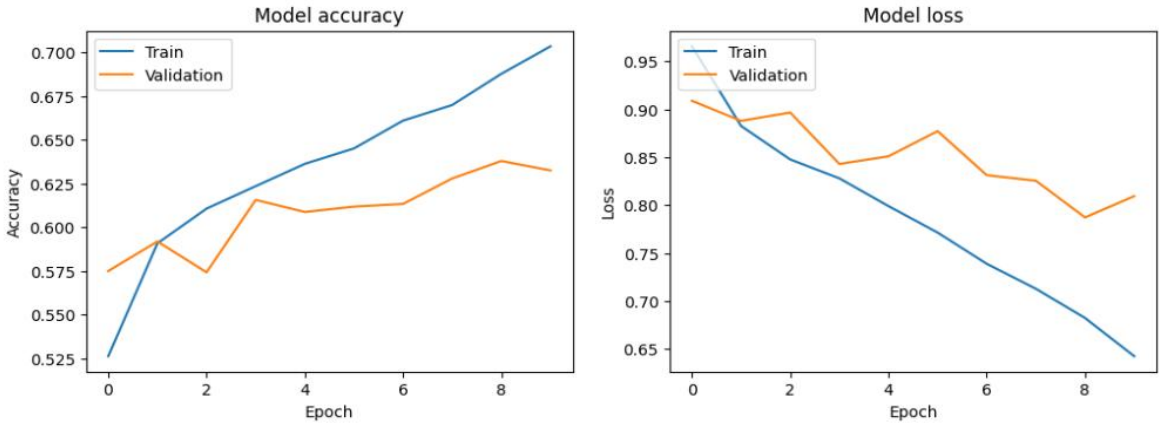


Figure 21: BiLSTM with word2vec with 10 epoch and 64 batch size

I. BiLSTM with fasttext summary

Model	Epoch	Batch size	Learning rate	Training dataset accuracy	Validation dataset accuracy	Test dataset accuracy
BiLSTM	10	32	0.001	75.20%	69.06%	70.21%
	10	64	0.001	73.03%	65.62%	68.98%

	20	32	0.001	91.23%	82.15%	80.47%
	20	64	0.001	89.74%	80.70%	78.79%

Table 6: Summary of BiLSTM model accuracy using fasttext

II. BiLSTM with word2vec summary

Model	Epoch	Batch size	Learning rate	Training dataset accuracy	Validation dataset accuracy	Test dataset accuracy
BiLSTM	10	32	0.001	73.74%	66.15%	66.30%
	10	64	0.001	70.32%	63.24%	64.31%
	20	32	0.001	84.42%	78.42%	74.57%
	20	64	0.001	83.39%	73.96%	70.29%

Table 7: Summary of BiLSTM model with word2vec

B. Model 2. LSTM

a) LSTM with fasttext with 20 epoch, batch size 64

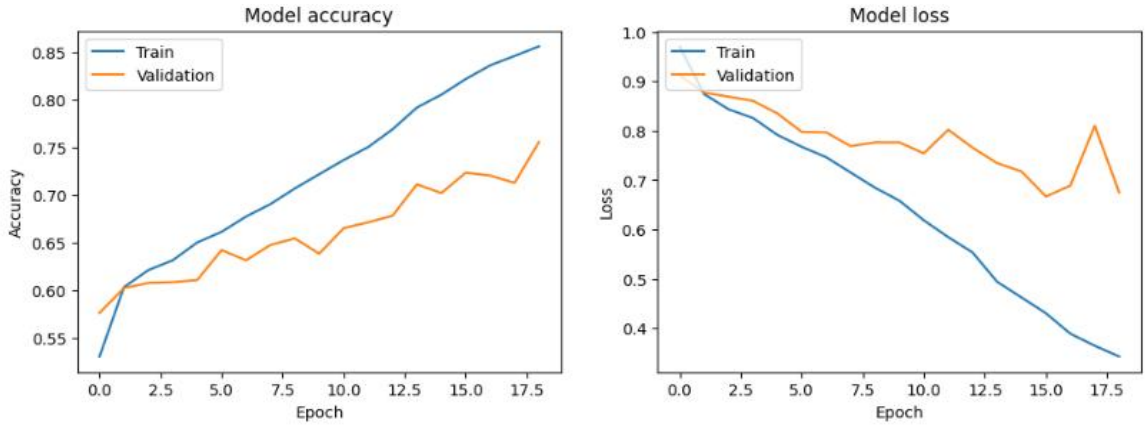


Figure 22: LSTM with fasttext with 20 epoch and batch size 64

b) LSTM with word2vec with 20 epoch, batch size 64

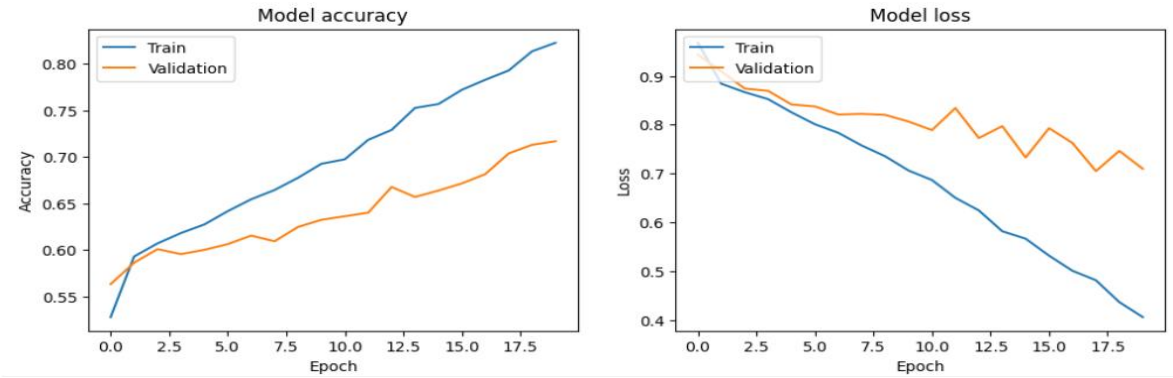


Figure 23: LSTM with word2vec with 20 epoch and batch size 64

c) LSTM with fasttext with 20 epochs, 32 batch size

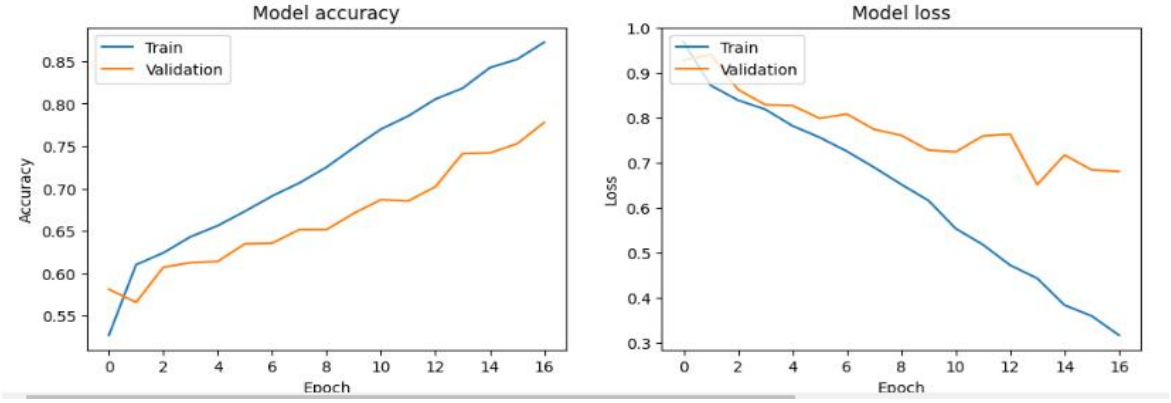


Figure 24: LSTM with fasttext with 20 epoch and batch size 32

d) LSTM with word2vec with 20 epochs, 32 batch size

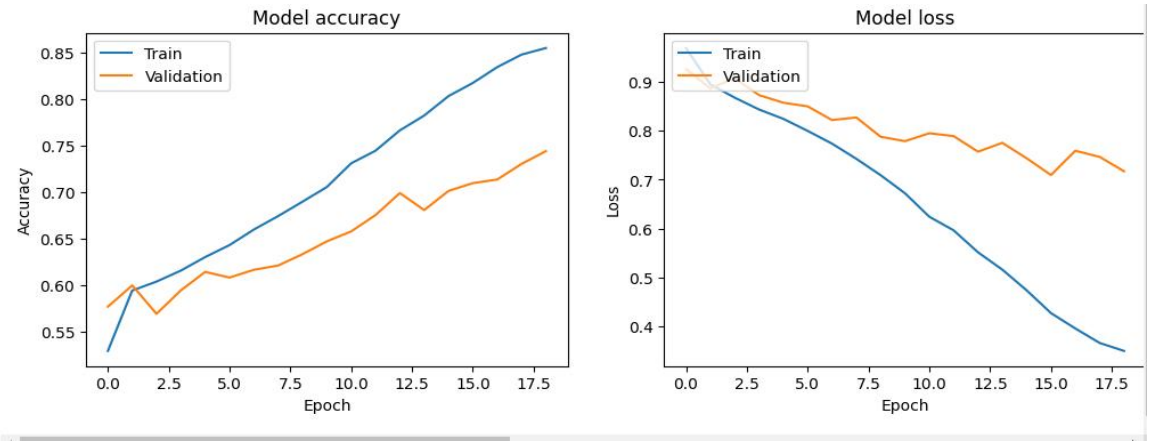


Figure 25: LSTM with word2vec with 20 epochs, 32 batch size

e) LSTM with fasttext with 10 epoch, batch size 64

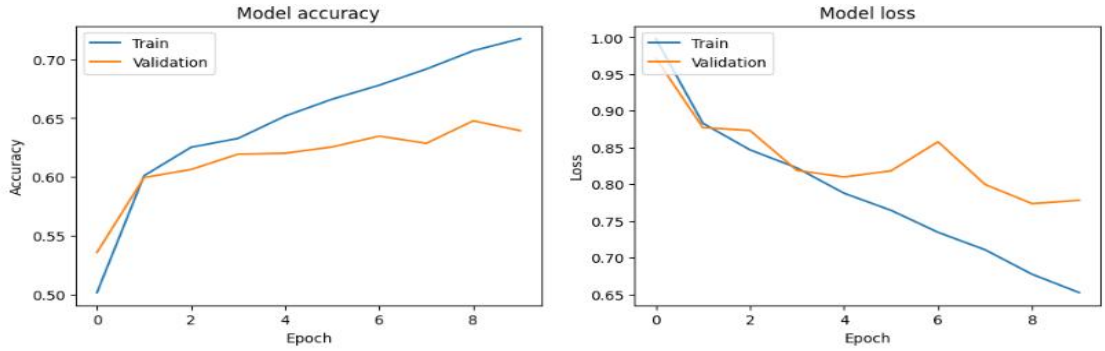


Figure 26: LSTM with fastttext with10 epoch and batch size 64

f) LSTM with word2vec with 10 epoch, batch size 64

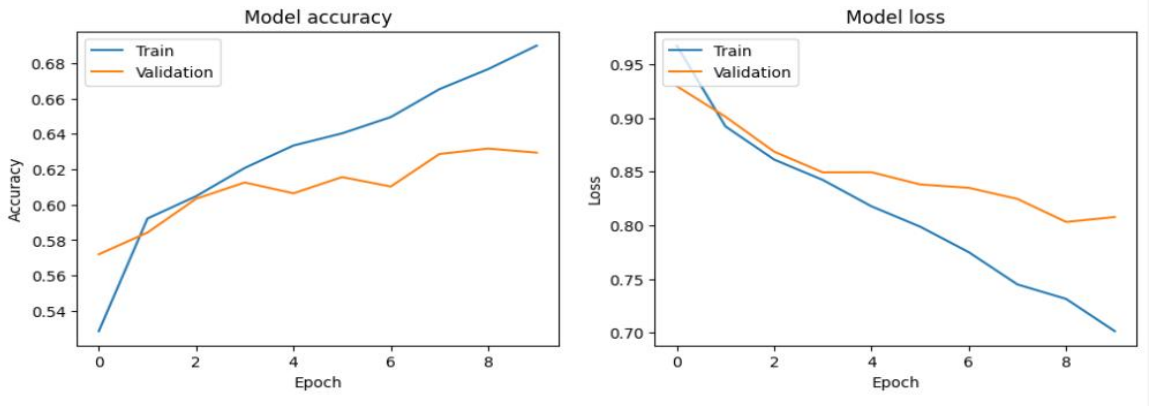


Figure 27: LSTM with word2vec with 10 epoch and batch size 64

g) LSTM with fasttext with 10 epoch, and 32 batch size

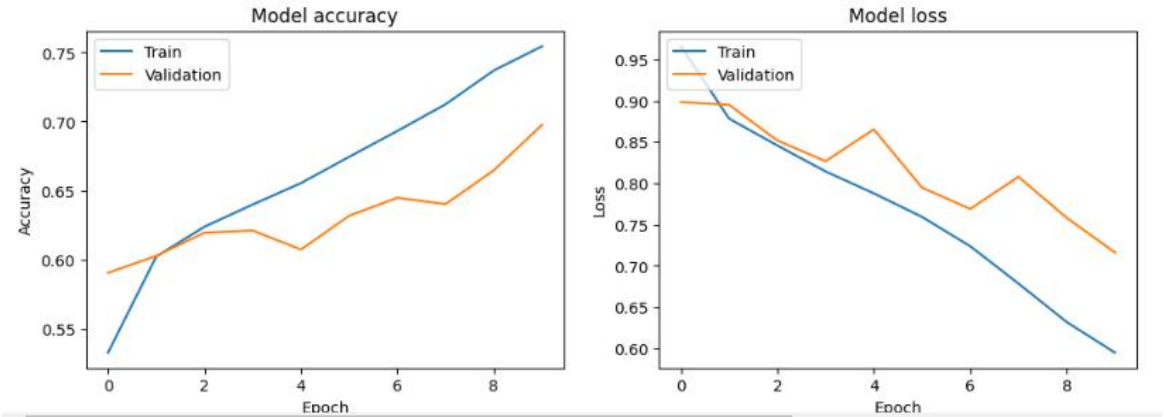


Figure 28: LSTM with fasttext with 10 epoch and batch size 32

h) LSTM with word2vec with 10 epoch, and 32 batch size

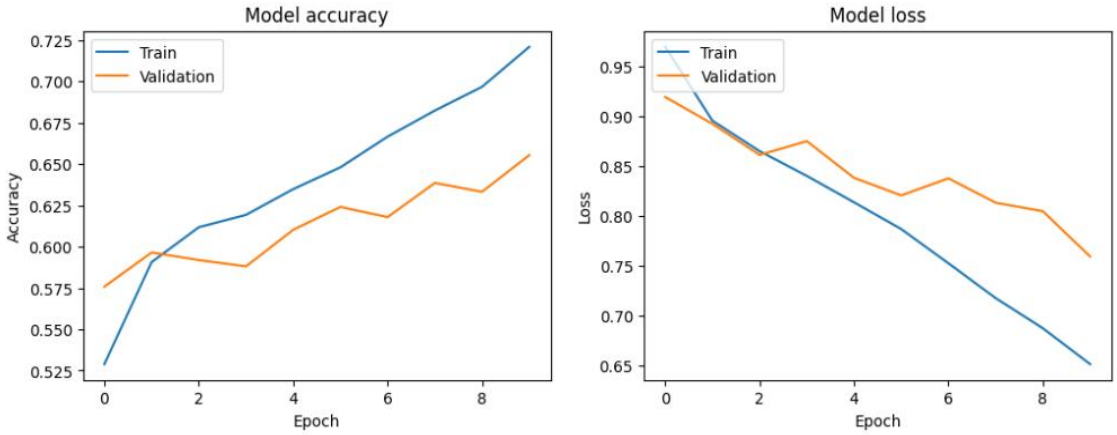


Figure 29: LSTM with word2vec with 10 epoch, and 32 batch size

I. LSTM Model summary with fasttext

Model	Epoch	Batch size	Learning rate	Training dataset accuracy	Validation dataset accuracy	Test dataset accuracy
LSTM	10	32	0.001	75.44%	69.75%	70.29%
	10	64	0.001	71.75%	63.9%	69.52%
	20	32	0.001	87.25%	77.79%	75.57%
	20	64	0.001	85.59%	75.57%	72.43%

Table 8: LSTM model summary with fasttext

II. LSTM Model summary with word2vec

Model	Epoch	Batch size	Learning rate	Training dataset accuracy	Validation dataset accuracy	Test dataset accuracy
LSTM	10	32	0.001	72.09%	65.54%	66.92
	10	64	0.001	68.98%	62.94%	64.93%

	20	32	0.001	85.52%	74.40%	74.12%
	20	64	0.001	82.22%	71.66%	70.64%

Table 9: LSTM model summary with word2vec

C. Model 3: CNN

a) **CNN with fasttext with 20 epoch, batch size 64**

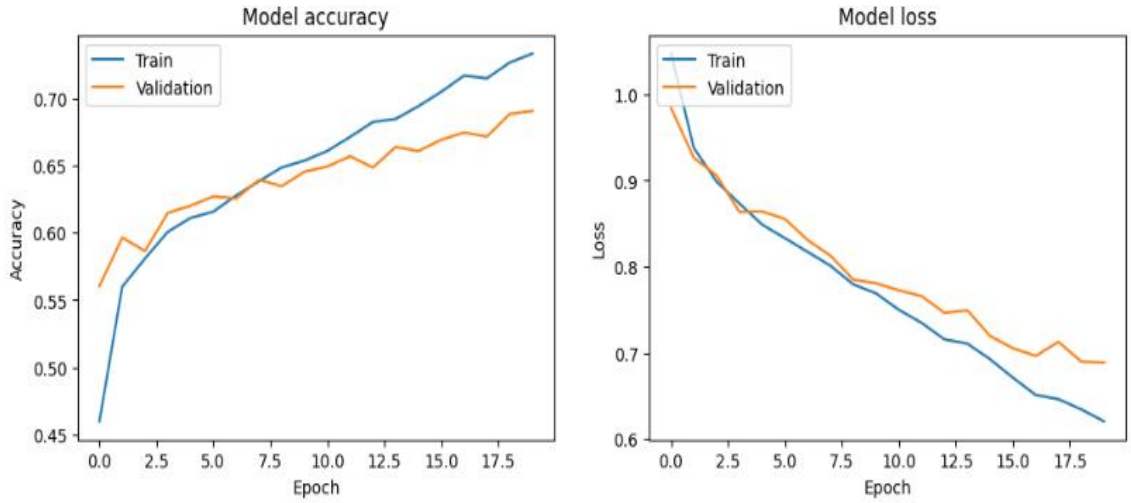


Figure 30: CNN with fasttext with 20 epoch and batch size 64

b) **CNN with word2vec with 20 epoch, batch size 64**

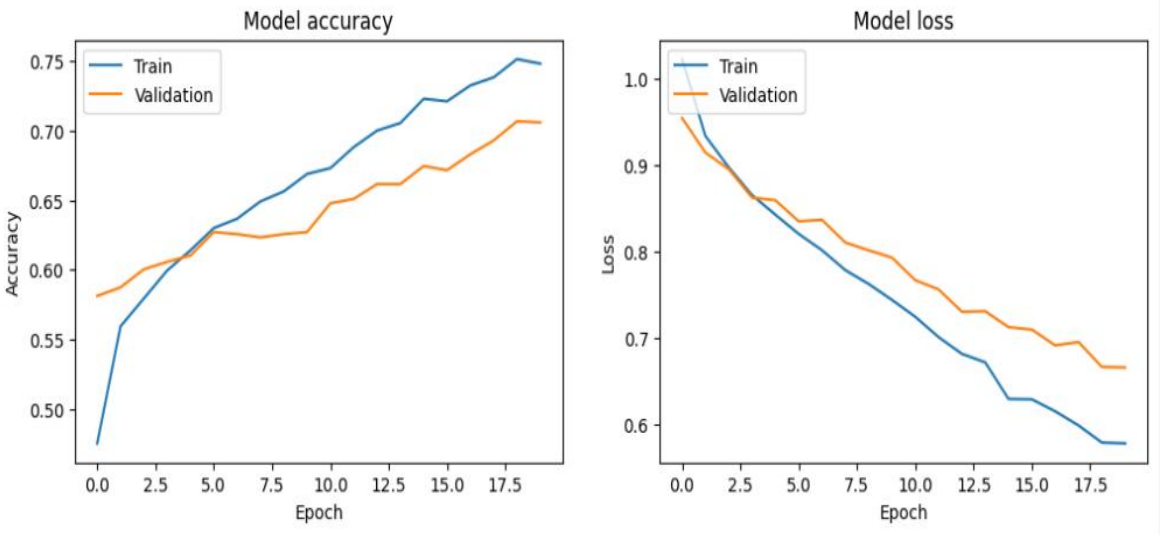


Figure 31: CNN with word2vec with 20 epoch and batch size 64

c) CNN with fasttext 20 epoch, 32 batch size

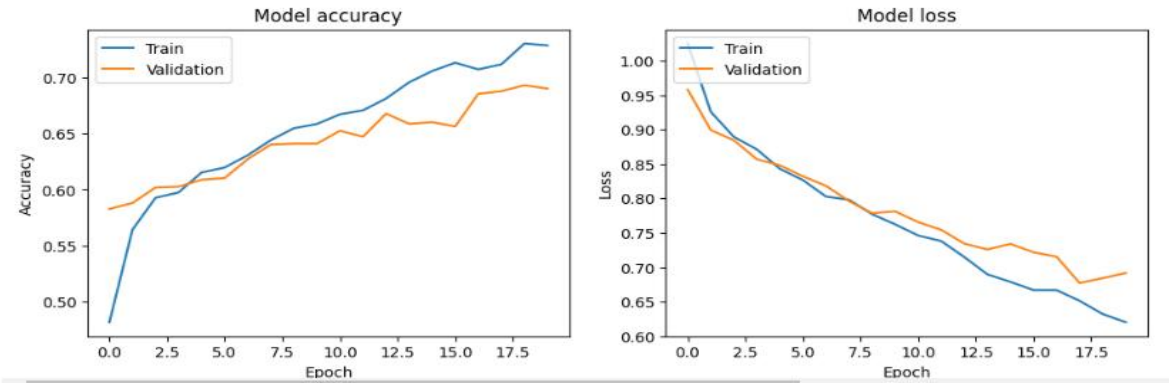


Figure 32: CNN with fasttext with 20 epoch and batch size 32

d) CNN with word2vec 20 epoch, 32 batch size

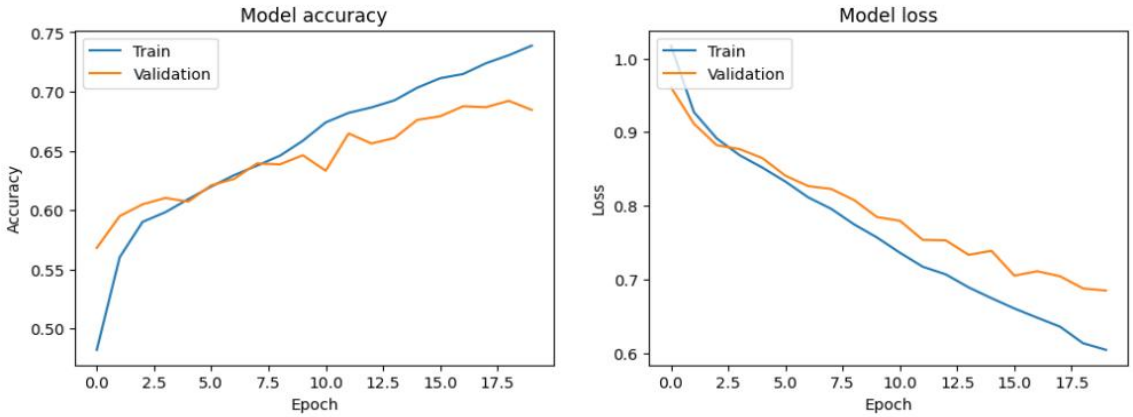


Figure 33: CNN with word2vec 20 epoch, 32 batch size

e) CNN with fasttext 10 epoch, batch size 64

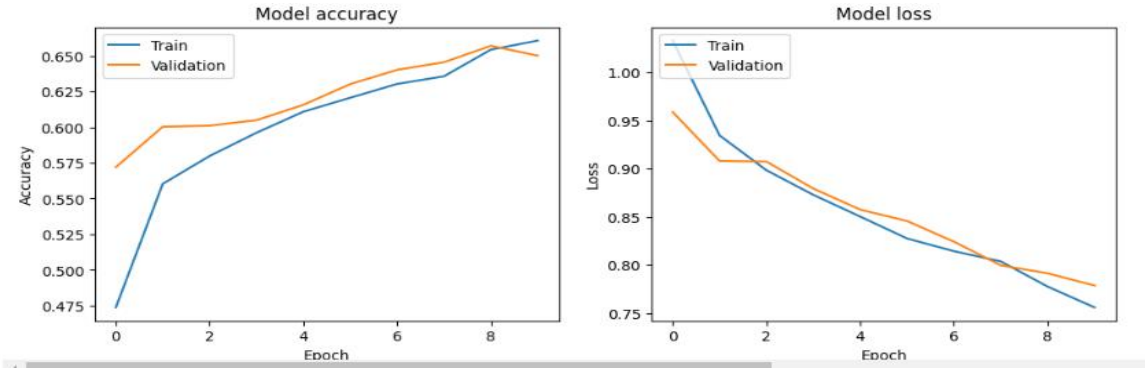


Figure 34: CNN with fasttext with 10 epoch and batch size 64

f) CNN with word2vec 10 epoch, batch size 64

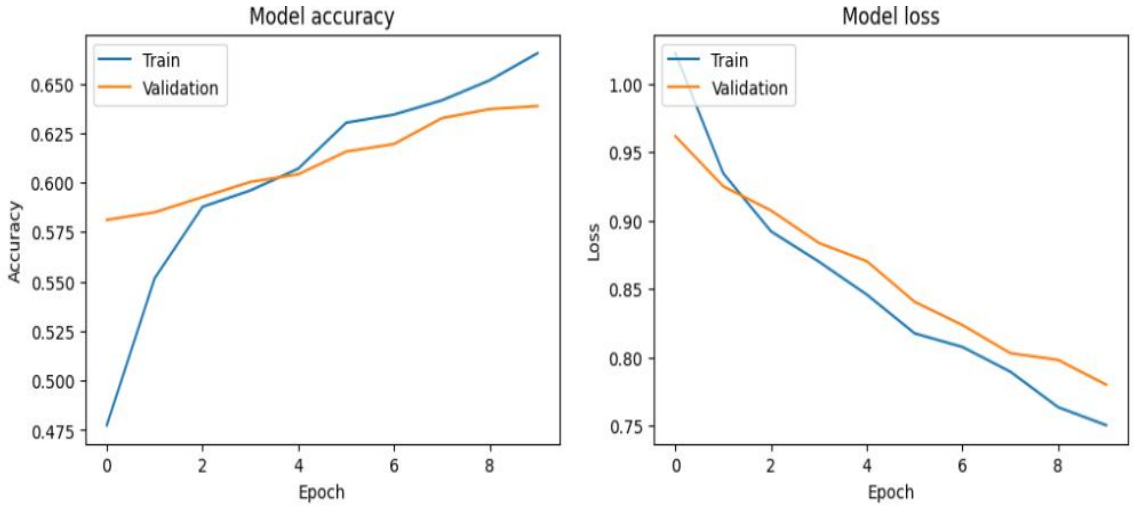


Figure 35: CNN with word2vec 10 epoch and batch size 64

g) CNN with fasttext 10 epoch, 32 batch size

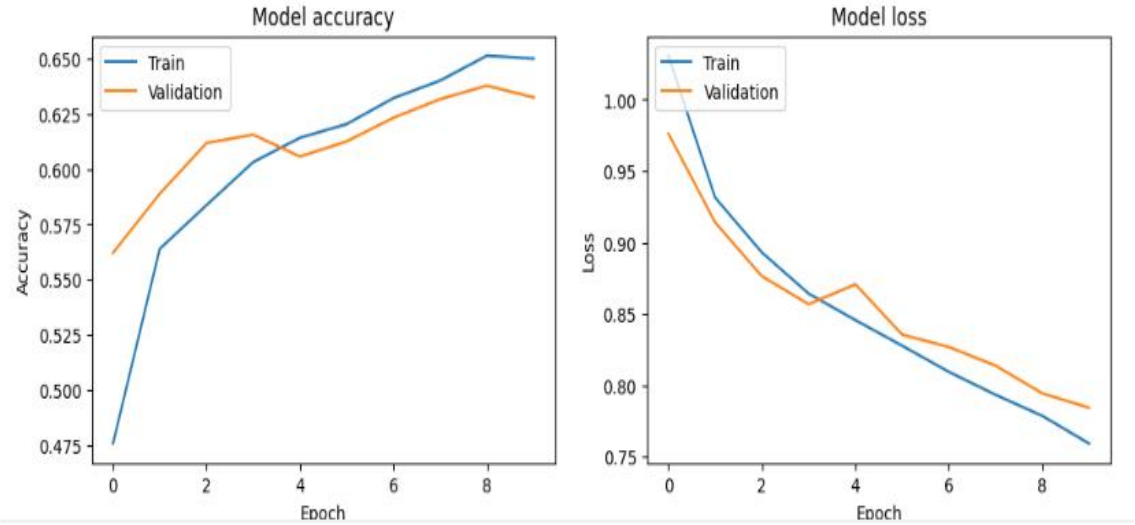


Figure 36: CNN with fasttext with 10 epoch and batch size 32

h) CNN with word2vec with 10 epoch, 32 batch size

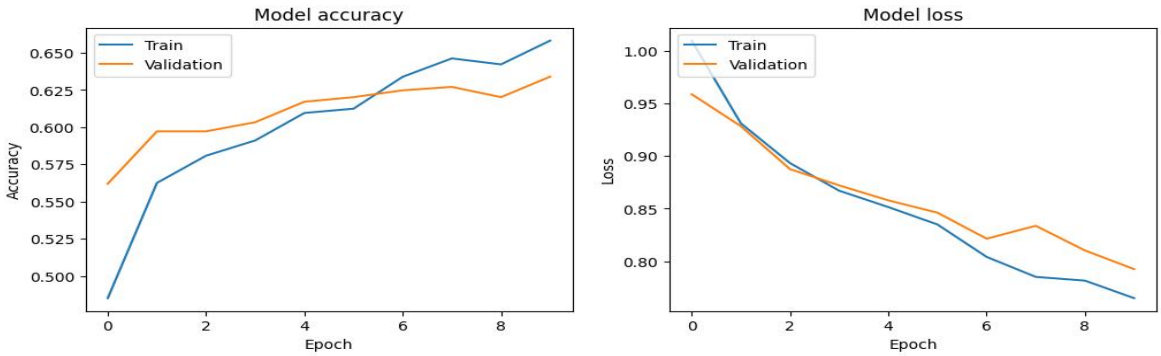


Figure 37: CNN with word2vec with 10 epoch and 32 batch size

I. Summary of CNN model with fasttext

Model	Epoch	Batch size	Learning rate	Training dataset accuracy	Validation dataset accuracy	Test dataset accuracy
CNN	10	32	0.001	65.01%	63.23%	66.04%
	10	64	0.001	66.06%	65.01%	65.54%
	20	32	0.001	72.83%	68.98%	72.8%
	20	64	0.001	73.3%	69.06%	71.9%

Table 10: Summary of CNN model accuracy using fasttext

II. Summary of CNN model with word2vec

Model	Epoch	Batch size	Learning rate	Training dataset accuracy	Validation dataset accuracy	Test dataset accuracy
CNN	10	32	0.001	65.82%	63.39%	64.77%
	10	64	0.001	66.52%	63.85%	64.85%
	20	32	0.001	73.87%	68.45%	70.21%
	20	64	0.001	74.81%	70.59%	73.58%

Table 11: Summary of CNN model with word2vec

4.4. Discussion

When we get to the discussion section, the following is a summary of all the models together with the corresponding epochs, batch size used, learning rate, training accuracy, validation accuracy, and test accuracy.

I. For word embedding with fasttext

Model	Epoch	Batch size	Learning rate	Training dataset accuracy	Validation dataset accuracy	Test dataset accuracy
CNN	10	32	0.001	65.01%	63.23%	66.04%
	10	64	0.001	66.06%	65.01%	65.54%
	20	32	0.001	72.83%	68.98%	72.8%
	20	64	0.001	73.3%	69.06%	71.9%
LSTM	10	32	0.001	75.44%	69.75%	70.29%
	10	64	0.001	71.75%	63.9%	69.52%
	20	32	0.001	87.25%	77.79%	75.57%
	20	64	0.001	85.59%	75.57%	72.43%
BiLSTM	10	32	0.001	75.20%	69.06%	70.21%
	10	64	0.001	73.03%	65.62%	68.98%
	20	32	0.001	91.23%	82.15%	80.47%
	20	64	0.001	89.74%	80.70%	78.79%

Table 12: Summary of models with fasttext

II. For word embedding with word2vec

Model	Epoch	Batch size	Learning rate	Training dataset	Validation dataset	Test dataset

				accuracy	accuracy	accuracy
CNN	10	32	0.001	65.82%	63.39%	64.77%
	10	64	0.001	66.52%	63.85%	64.85%
	20	32	0.001	73.87%	68.45%	70.21%
	20	64	0.001	74.81%	70.59%	73.58%
LSTM	10	32	0.001	72.09%	65.54%	66.92
	10	64	0.001	68.98%	62.94%	64.93%
	20	32	0.001	85.52%	74.40%	74.12%
	20	64	0.001	82.22%	71.66%	70.64%
BiLSTM	10	32	0.001	73.74%	66.15%	66.30%
	10	64	0.001	70.32%	63.24%	64.31%
	20	32	0.001	84.42%	78.42%	74.57%
	20	64	0.001	83.39%	73.96%	70.29%

Table 13: Summary of models with word2vec

Based upon the experimental analysis:

- BiLSTM model with fasttext word embedding performs better than other two models (With 91.23% training accuracy, 82.15% validation accuracy, and 80.47% test dataset accuracy, the BiLSTM model outperformed the other models).
- LSTM model performed good when we compare with CNN model
- CNN performs less when compared with other two models.

4.5. Error Analysis between Models

The fact that a single text can have multiple interpretations and those distinct texts can have the same meaning is one of the difficulties with natural language inference. One of the features of natural language is the variety of ways it might convey an idea. For instance, the opposing concepts are described in the following two sentences: "Tolaan buna dhugaa

hin jiru" and "Tolaan buna dhugaa jira." We have therefore performed mistake analysis based on Afaan Oromoo's various sentence forms, keeping these notions in mind. Here, the following tables describes more.

Types of Sentences	Model	Predicted label	Actual label
1. Paraphrase **Premise**: Ijoolleen kolfaa jiru. **Hypothesis**: daa'imman ganmadaa jiru	CNN	Entailment	Entailment
	LSTM	Entailment	
	BiLSTM	Entailment	
2. Long sentence **Premise**: Murteessaan tokko dirree tokko irratti taphattoota tokko tokko waliin haasa'aa jira. **Hypothesis**: abbaan murtii rukuttaa adabbii kennan irratti taphattootaa waliin mari'achaa jiru	CNN	Entailment	Neutral
	LSTM	Entailment	
	BiLSTM	Entailment	
3. Conditional **Premise**: Garee saawwanii laga tokko keessa dhaabbatan. **Hypothesis**: saawwan dirree keessa fiigaa jiru.	CNN	Neutral	Contradiction
	LSTM	Contradiction	
	BiLSTM	Contradiction	
4. Active/Passive **Premise**: namoonni mana ijaaruudhaaf simintoo fayyadamu.	CNN	Neutral	Neutral
	LSTM	Neutral	

Hypothesis: simmintoon mana ijaaruuf ni fayyada	BiLSTM	Neutral	
5. Quantifier **Premise**: Saroonni lama ala jiru. **Hypothesis**: Saroonni afur mana keessa jiru.	CNN	Contradiction	Contradiction
	LSTM	Contradiction	
	BiLSTM	Contradiction	

Table 14: Error Analysis between models

We can infer from table 10 above that our model works better for the various sentence kinds given. CNN was unable to anticipate the proper value for the conditional sentence type. Yet, LSTM and BiLSTM made accurate predictions.

Overall, our study provided a response to the research question posed in Chapter 1 Section 1.3, which is as follows.

RQ1: Which deep learning approach performs better for Afaan Oromoo textual entailment classification?

For our study, we selected three deep learning approaches based on their effectiveness for local patterns, sequential dependencies and contextual information extraction. These three approaches are CNN, LSTM and Bi-LSTM. By using similar hyperparameter for these models, BiLSTM model achieved better performance than other two models (CNN and LSTM).

RQ2: How can Afaan Oromoo textual entailment be classified using the chosen approach?

Several steps have been followed. First, we collected pair of Afaan Oromoo texts dataset, we preprocessed the collected dataset, we used both word2vec and fasttext to make comparison for word embedding, we implemented the BiLSTM to capture the dependencies from both directions (premise and hypothesis). Then, we concatenated the output of BiLSTM layer. We used the dense layer for further processing the concatenated outputs. At the end, we also used softmax function to classify the output into Entailment,

Neutral and Contradiction. The model is evaluated based on the given validation and test dataset.

RQ3: To what extent the model is effective?

The chosen model, BiLSTM model is more effective than CNN and LSTM for Afaan Oromoo textual entailment classification as we tested on different sentence types we have listed before for Afaan Oromoo. As we checked for conditional sentences, BiLSTM predicted the correct value.

CHAPTER: FIVE

CONCLUSSIONS AND RECOMMENDATION

This is the conclusion and recommendation part, that discusses the activities done in this research by presenting how much the intended objectives are achieved and problems are addressed. The future works are also discussed.

5.1. Conclusion

The researcher performed a comparison analysis between three deep learning models, CNN for Afaan Oromoo textual entailment classification, LSTM for Afaan Oromoo textual entailment classification and BiLSTM for Afaan Oromoo textual entailment classification.

We have collected 13060 pair of sentences dataset manually from different Afaan Oromoo texts and also, we translated the dataset from Standard Natural Language Inference dataset available for English. Then, CNN, LSTM, and BiLSTM are the three deep learning models that we used to train the dataset. For each model, we used varying epochs and batch sizes during training, and we noted variations in test, validation, and training accuracy. With 91.23% training accuracy, 82.15% validation accuracy, and 80.47% test dataset accuracy, the BiLSTM model outperformed the other models. We carried out error analysis on various sentence types, including conditional, long, active/passive, and paraphrase sentences, in order to characterize how the model handles Afaan Oromoo sentence structures.

The work's contributions are:

- Afaan Oromoo textual entailment dataset has been prepared.
- Used fasttext and word2vec for comparison of word embedding.
- Used BiLSTM, LSTM and CNN models for comparison.
- Used drop out with BiLSTM, LSTM and CNN models for sentence embedding
- Concatenated the embedded sentences
- The prepared Afaan Oromoo textual entailment model can be used as the benchmark for Afaan Oromoo question answering, text summarization, machine translation and etc.

5.2. Recommendation

For this thesis, we used deep learning models for Afaan Oromoo textual entailment classification. To compare it with other languages, we used less dataset, but it is recommended to use huge dataset. Here are some of the recommendations.

- Prepare the huge dataset
- Deep learning models other than CNN, LSTM, and BiLSTM can be used for Afaan Oromoo textual entailment classification. Various deep learning models have applications in textual entailment. Other models are available to any researcher.

REFERENCES

[1] R. Navigli and V. R. Elena, “Natural Language Understanding : Instructions for (Present and Future) Use,” pp. 5697–5702, 2003.

[2] K. M. Verspoor and K. B. Cohen, “Synonyms,” no. June, 2018, doi: 10.1007/978-1-4419-9863-7.

[3] M. H. Haggag and A. Mohammed, “Different Models and Approaches of Textual Entailment Recognition,” vol. 142, no. 1, pp. 32–39, 2016.

[4] A. Poliak, “A Survey on Recognizing Textual Entailment as an NLP Evaluation”.

[5] A. Mishra, “Deep Learning Techniques in Textual Entailment”.

[6] L. Bentivogli, I. Dagan, H. T. Dang, D. Giampiccolo, B. Magnini, and R. Gan, “The Fifth PASCAL Recognizing Textual Entailment Challenge”.

[7] R. Gan, L. P. Group, O. M. Way, H. Language, and F. B. Kessler, “Recognizing textual entailment : Rational , evaluation and approaches,” vol. 15, no. 4, 2009, doi: 10.1017/S1351324909990209.

[8] L. Bentivogli, P. Clark, I. Dagan, D. Giampiccolo, and R. Gan, “The Seventh PASCAL Recognizing Textual Entailment Challenge,” no. i, 2008.

[9] P. Eleftheriadis, I. Perikos, and I. Hatzilygeroudis, “Evaluating Deep Learning Techniques for Natural Language Inference,” *Appl. Sci.*, vol. 13, no. 4, 2023, doi: 10.3390/app13042577.

[10] W. Tegegne, “The Development of Written Afan Oromo and the Appropriateness of Qubee , Latin Script , for Afan Oromo Writing,” *Issn 2224-3178*, vol. 28, no. 1976, pp. 8–14, 2016, [Online]. Available: https://www.iiste.org

[11] C. Lyu, Y. Lu, D. Ji, and B. Chen, “Deep Learning for Textual Entailment Recognition,” 2015, doi: 10.1109/ICTAI.2015.35.

[12] P. Chithra and M. Henila, “International Journal of Computer Sciences and Engineering Open Access,” *Int. J. Comput. Sci. Eng.*, vol. 6, no. 10, pp. 628–632,

2019, doi: 10.26438/ijcse/v6i1.161167.

[13] D. Z. Korman, E. Mack, J. Jett, and A. H. Renear, “Defining textual entailment,” *J. Assoc. Inf. Sci. Technol.*, vol. 69, no. 6, pp. 763–772, 2018, doi: 10.1002/asi.24007.

[14] J. Diego, R. Katrin, and E. Greg, “X-PARADE: Cross-Lingual Textual Entailment and Information Divergence across Paragraphs ◊,” 2021.

[15] P. B. Debarghya Majumdar, “Lexical Based Text Entailment System for Main Task of RTE6,” *Proc. TAC*, 2010, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.232.1092

[16] E. Lien and M. Kouylekov, “UIO-Lien: Entailment Recognition using Minimal Recursion Semantics,” *8th Int. Work. Semant. Eval. SemEval 2014 - co-located with 25th Int. Conf. Comput. Linguist. COLING 2014, Proc.*, pp. 699–703, 2014, doi: 10.3115/v1/s14-2125.

[17] F. M. Zanzotto, M. Pennacchiotti, and A. Moschitti, “A machine learning approach to textual entailment recognition,” *Nat. Lang. Eng.*, vol. 15, no. 4, pp. 551–582, 2009, doi: 10.1017/S1351324909990143.

[18] J. M. Czum, “Dive Into Deep Learning,” *J. Am. Coll. Radiol.*, vol. 17, no. 5, pp. 637–638, 2020, doi: 10.1016/j.jacr.2020.02.005.

[19] M. H. Haggag and A. M. Ahmed, “Recognizing Textual Entailment based on Deep Learning Approach,” vol. 181, no. 43, pp. 36–41, 2019.

[20] D. Bijiga, T. The, and O. Writing, “Kent Academic Repository University of Kent,” 2015.

[21] I. Bedane, “The Origin of Afaan Oromo : Mother Language,” vol. 15, no. 12, 2015.

[22] W. Tesema and D. Tamirat, “American Journal of Computer Investigating Afan Oromo Language Structure and Developing Effective File Editing Tool as Plug-in into Ms Word to Support Text Entry and Input Methods”.

[23] H. Tamiru, G. Addis, and A. Ethiopia, “Afan Oromo Text Summarization With

Deep Learning Approach," 2022.

[24] "Afaan Oromo Multi-Label News Text Classification Using Deep Learning Approach," pp. 1–12, 2023.

[25] H. Mesfin, "College of Natural Sciences Helina Mesfin Advisor : Fekade Getahun (PhD)," 2020.

[26] M. R. Costa-jussà, "Recognizing Textual Entailment using Deep Learning Techniques Supervisor :," 2017.

[27] P. Pakray, S. Bandyopadhyay, and A. Gelbukh, "A Hybrid Textual Entailment System using Lexical and Syntactic Features".

[28] D. Rudrapal, A. Das, and B. Bhattacharya, "Recognition of Partial Textual Entailment for Indian Social Media Text," vol. 23, no. 1, pp. 143–152, 2019, doi: 10.13053/CyS-23-1-2816.

[29] Q. Nhat and M. Pham, "A Machine Learning based Textual Entailment Recognition System of JAIST Team for NTCIR9 RITE".

[30] M. Khader, A. Awajan, and A. Alkouz, "Textual Entailment for Arabic Language based on Lexical and Semantic Matching," no. October, 2016, doi: 10.21700/ijcis.2016.109.

[31] N. Almarwani and M. Diab, "Arabic Textual Entailment with Word Embeddings," pp. 185–190, 2017.

[32] M. B. W. Bakari and M. Neji, "Recognizing Textual Entailment for Arabic using seman- tic similarity and Word Sense Disambiguation," vol. 3, no. 1, pp. 1–10.

[33] S. Roy, P. Sharma, K. Nath, D. K. Bhattacharyya, and J. K. Kalita, "Pre-processing: A data preparation step," *Encycl. Bioinforma. Comput. Biol. ABC Bioinforma.*, vol. 1–3, no. April 2020, pp. 463–471, 2018, doi: 10.1016/B978-0-12-809633-8.20457-3.

[34] A. K. Bhattacharjee, A. Mallick, and A. Dey, "Data Cleaning in Text File," vol. 9, no. 2, pp. 17–21, 2013.

[35] M. Sule, “Vol-9 Issue-1 2023,” no. 1, pp. 1873–1892, 2023.

[36] J. Gavhane, R. Prasad, and R. Kumar, “Graph embeddings for natural language processing,” *Graph Learn. Netw. Sci. Nat. Lang. Process.*, vol. 7148, pp. 77–95, 2022, doi: 10.1201/9781003272649-4.

[37] M. Mohammadi, R. Mundra, R. Socher, and L. Wang, “CS224n: Natural Language Processing with Deep Learning,” pp. 1–13, 2017.

[38] Jeffrey Pennington, “GloVe: Global Vectors for Word Representation Jeffrey,” *AES J. Audio Eng. Soc.*, vol. 19, no. 5, pp. 417–425, 1971.

[39] B. Li, A. Drozd, T. Liu, and X. Du, “Subword-level Composition Functions for Learning Word Embeddings,” pp. 38–48, 2018, doi: 10.18653/v1/w18-1205.

[40] A. Jacovi and O. S. Shalom, “Understanding Convolutional Neural Networks for Text Classification,” 2016.

[41] A. Ghosh, A. Sufian, F. Sultana, A. Chakrabarti, and D. De, *Fundamental Concepts of Convolutional Neural Network*, no. June. 2020. doi: 10.1007/978-3-030-32644-9.

[42] R. C. Staudemeyer and E. R. Morris, “– Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks,” no. September, 2019.

[43] B. Jang, M. Kim, G. Harerimana, S. Kang, and J. W. Kim, “applied sciences Bi-LSTM Model to Increase Accuracy in Text Classification : Combining Word2vec CNN and Attention Mechanism,” 2020.

[44] M. V. Koroteev, “BERT: A Review of Applications in Natural Language Processing and Understanding,” no. March, 2021, doi: 10.48550/arXiv.2103.11943.

[45] K. Banerjee, C. Vishak Prasad, R. R. Gupta, K. Vyas, H. Anushree, and B. Mishra, “Exploring alternatives to softmax function,” *Proc. 2nd Int. Conf. Deep Learn. Theory Appl. DeLTA 2021*, pp. 81–86, 2021, doi: 10.5220/0010502000810086.

[46] Z. Karimi, “Confusion Matrix.” [Online]. Available: https://www.researchgate.net/publication/355096788

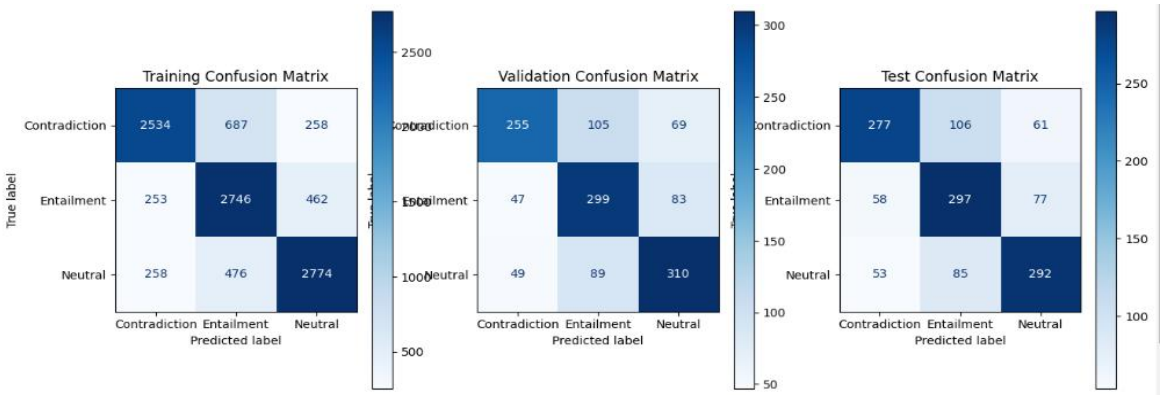
APPENDIXES

Appendix A: Afaan Oromoo stopwords

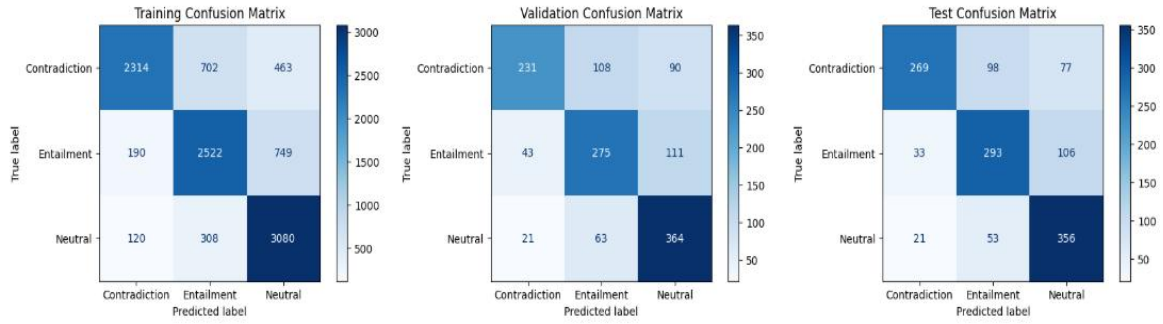
Immoo	haa	akka	mee	akkasi	Ishii
Utuu	kan	kee	hanga	jechuun	Ol
Waan	akkam	oliif	waggaa	akkasumas	Hoggaa
Kanaaf	oliin	akkuma	hogguu	kanaafi	Yammuu
Kanaafu	osoo	yemmuu	illee	otoo	Otumalle
Ani	innaa	keenyaa	booddee	isaa	Saniif
Koo	simmoo	silaa	malee	sun	Yoom
Eegasii	ta`ullee	yookinimmoo	ennaa	garuu	Itumalle
Nuti	tanaafi	ituullee	ofii	hanga	woo
Hogguu	oo	jara	isinitti	eega	tanaaf
Ta'us	akkum	natti	isatti	kanatti	Kanaan
Hennas	moo	jechuuf	narraa	hoggus	Isee
Jechuun	duras	ta'a	isaatiin	isinittis	Dabalatee
Isaanirraa	hamma	dhaan	Ta'uyyuu	wajjin	Oggaa
Innaasan	kanaafillee	alatti	keessatti	an	Sana
Saniif	irra	fullee	isaanis	isiniinis	Nuy
Nuu	booddees	boodas	kuniyyuu	Waa'ee	Giddu

Appendix B: Confusion matrix of the models

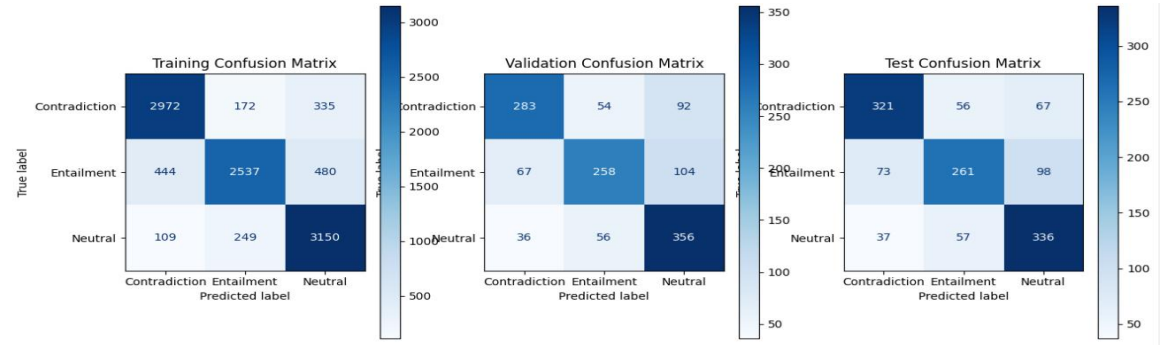
i. BiLSTM model with word2vec confusion matrix table with 10 epoch and 32 batch size



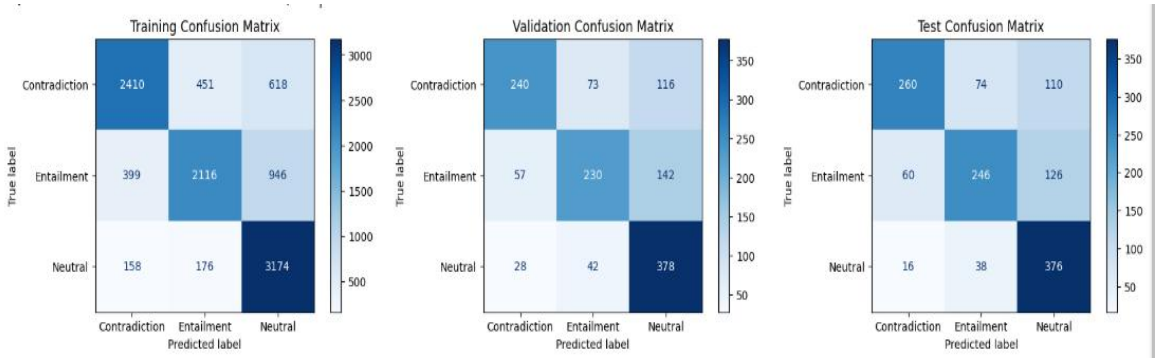
ii. BiLSTM model with fasttext confusion matrix table with 10 epoch and 32 batch size



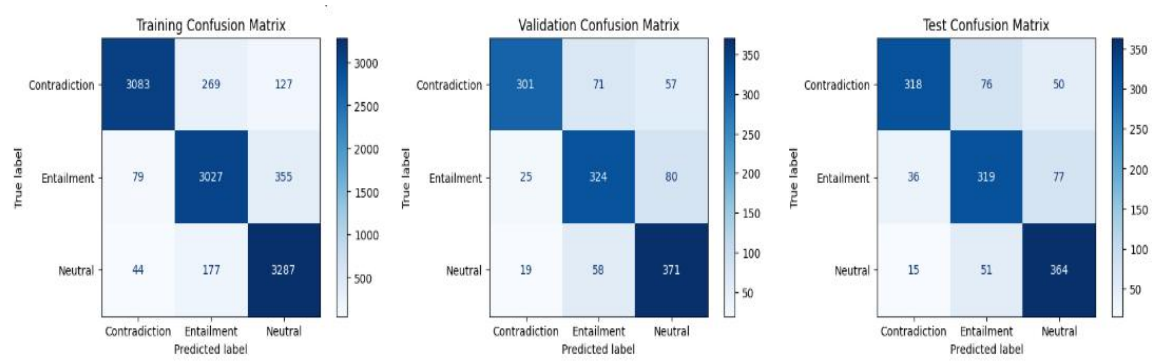
iii. BiLSTM model with word2vec confusion matrix table with 10 epoch and 64 batch size



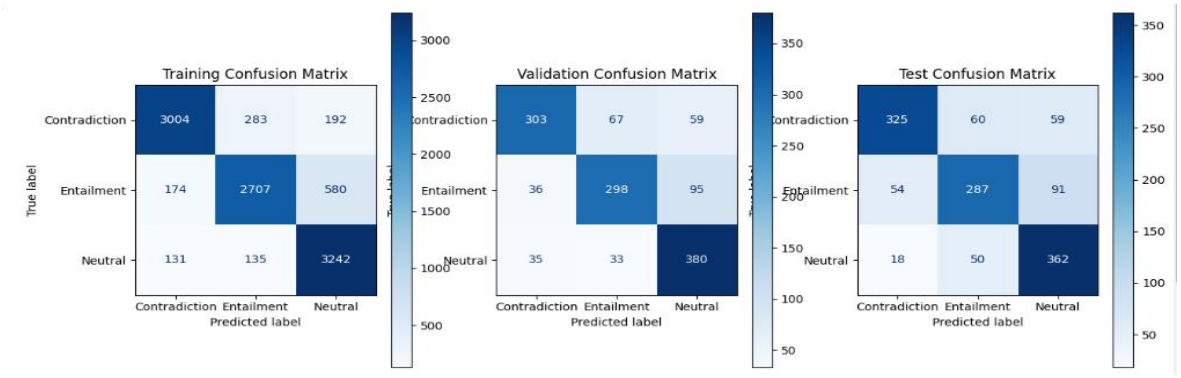
iv. BiLSTM model with fasttext confusion matrix table with 10 epoch and 64 batch size



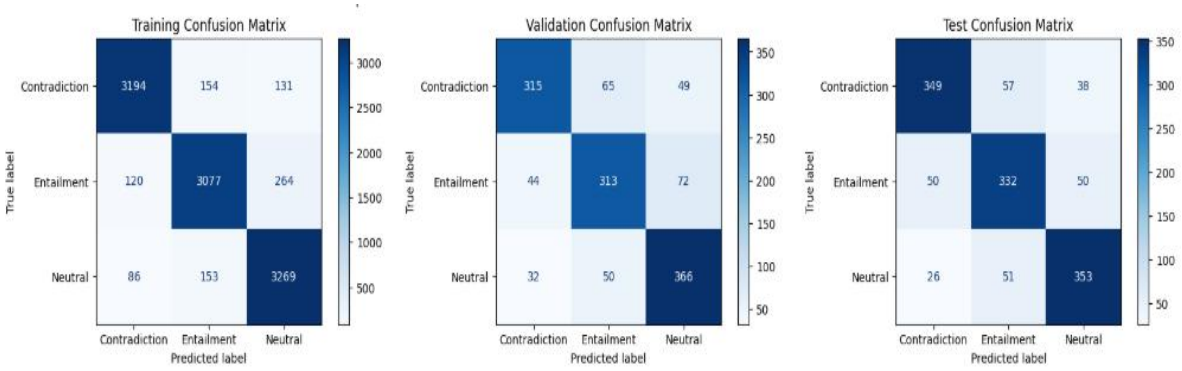
v. BiLSTM model with fasttext confusion matrix table with 20 epoch and 32 batch size



vi. BiLSTM model with word2vec confusion matrix table with 20 epoch and 32 batch size



vii. BiLSTM model with word2vec confusion matrix table with 20 epoch and 64 batch size



viii. BiLSTM model with fasttext confusion matrix table with 20 epoch and 64 batch size