WALLAGA UNIVERSITY

SCHOOL OF POSTGRADUATE STUDIES

COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

STANCE DETECTION FOR AFAAN OROMO TEXT USING DEEP LEARNING APPROACH

A THESIS SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE IN PARTIAL FULFILLMENT FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE

MSC THESIS

BY: WABI ETANA EBA

MAJOR ADVISOR: WAKGARI DIBABA (Ass.Prof).

June, 2025

Nekemte, Oromiya, Ethiopia

WALLAGA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
P.O. Box: 395, Nekemte, Ethiopia.

APPROVAL SHEET FOR SUBMITTING RESEARCH PROPOSAL

As members of the Board of Examining of the Final MSc thesis open defense, we certify that we have read and evaluated the thesis prepared by Wabi Etana Eba under the title "**Stance Detection for Afaan Oromo Text Using Deep Learning Approach**" and recommend that the thesis be accepted as fulfilling the thesis requirement for the Degree of Master of Science in Computer Science.

Submitted by:

_____	_____	_____
Name of Student	Signature	Date

Approved by:

1. _____	_____	_____
Name of Major Advisor	Signature	Date
2. _____	_____	_____
Department Head	Signature	Date
3. _____	_____	_____
Name of Chairman	Signature	Date
4. _____	_____	_____
Internal Examiner	Signature	Date
5. _____	_____	_____
External Examiner	Signature	Date

Final Approval and Acceptance Thesis Paper Approved by:

1. _____	_____	_____
PG College	Signature	Date
2. _____	_____	_____
Dean of College	Signature	Date
3. _____	_____	_____
Dean SGS	Signature	Date

STATEMENT OF THE AUTHOR

I Mr. **Wabi Etana Eba** hereby declare and affirm that the MSc thesis entitled "**Stance detection for Afaan Oromo Text Using Deep learning**" is my own work conducted under the supervision of **Wakgari Dibaba(Ass.Prof).** This thesis was submitted in partial fulfillment of the requirements for a Master of Science degree in Computer Science from the Post Graduate Studies at Wallaga University. I have complied with all ethical principles of scholarship in the data collection, preparation, analysis, and completion of this thesis; all scholarly material included in the thesis has been acknowledged through citation; I have properly cited and referenced all the original sources; and I declare that I have adhered to all principles of academic honesty and integrity as well as not misrepresented, fabricated, or falsified any idea, data, fact, or source in my thesis submission. I further declare that this thesis has not been submitted to any other institution anywhere for the award of any academic degree, diploma or certificate.

I understand that any violation of the above would be cause for disciplinary action by the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Name: **Wabi Etana Eba** Signature: _____ Date: _____

School: Informatics

Department: Computer Science

DECLARATION

This is to certify that this MSc thesis entitled "**Stance Detection for Afaan Oromo Text Using Deep Learning Approach**" was accepted in partial fulfillment of the requirements for the award of Degree of Master in Computer Science by the school of graduate studies of Wallaga University through the College of Engineering and Technology done by Wabi Etana Eba is a genuine work carried out by him under my guidance. The assistance and help received during the course of this investigation have been duly acknowledged. Therefore, I recommend that it can be accepted as fulfilling the MSc thesis requirement.

BIOGRAPHICAL SKETCH

Mr. Wabi Etana Eba was born on July 12, 1993 in East Wollega Zone, Sibu Sire woreda, from his father Ato Etana Eba and mother W/ro. Ijigayehu Temesgen. He completed his elementary school education at Cari Jarso and his high and preparatory school at Sibu Sire. The author joined Ambo University during the 2011/2012 academic year and graduated on June 01, 2015 with a Bachelor of Science degree in Computer Science.

The author has been working as a Database Administrator at Wallaga University *ICT* since July 27, 2019. He enrolled in the Wallaga University Postgraduate School for the 2020/2021 academic year as a graduate student pursuing a Master of Science in Computer Science.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to Almighty God for providing me with the strength, health, patience and giving immeasurable support in my journey to this success and God did what he promised me. All things are done by him I give glory to him.

Secondly, I would like to express my sincere and deepest appreciation to my advisor **Wakgari Dibaba(Ass.Prof)** for his advice, comments, guidance, suggestions and encouragement helped me throughout the preparation of the research study.

My gratitude also goes to the Department of Computer Science head, staffs and instructors assisted me in some way while I worked on this research.

Finally, I am grateful to my beloved wife, family and my friends directly or indirectly for their essential support and inspiration during the thesis preparation.

ACRONYMS AND ABBREVIATIONS

AOSD	:	Afaan Oromo Stance Detection
BLSTM	:	Bidirectional Long Short-Term Memory
CNN	:	Convolutional Neural Networks
GRU	:	Gated Recurrent Unit
IE	:	Information Extraction
IDE	:	Integrated Development Environment
LSTM	:	Long Short-Term Memory
NLTK	:	Natural Language Toolkit
NLP	:	Natural Language Processing
OBN	:	Oromia Broadcasting Network
OMN	:	Oromia Media Network
ONN	:	Oromia news network
RNN	:	Recurrent Neural Networks

Table of Contents

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The Afaan Oromo, often known as Oromo, is a significant African language that is spoken by the Oromo people in Ethiopia and some parts of Kenya. It is easy to determine its precise position among African languages since several factors, like the number of native speakers, geographic distribution, and cultural significance, cannot have a distinct impact on rankings.

Stance Detection is a task to automatically identify whether a particular news headline "Agrees" with, "Disagrees" with, "Discusses," or is Unrelated to a particular news article. So, in this study, we proposed to utilize the Deep learning algorithm (LSTM, BiLSTM and GRU) model with different feature extraction on Afaan Oromo text Stance Detection. Because, feature extraction is useful to reduce dimensions and get the best feature from the dataset by selecting and combining variables into features, thus, effectively reducing the amount of data.

In this research, we have used 12528 a newly collected and annotated pair of headline and body of Afaan Oromo text dataset to develop pre-trained word embedding model. The data train test splits (80/20) were used to choose the model that performed the best out of all the algorithms. Also, the text preprocessing activities of normalization, tokenization, text cleaning, and stop word removal were just a few of the NLP tasks that were carried out in this work.

In our experimentation, in terms of accuracy, precision, recall, and F1-score, the models created by the LSTM, BILSTM, and GRU algorithms produced more promising outcomes than other models. The results show that ordinary LSTM and GRU based modeling performs less than BILSTM-based modeling. BILSTM has great accuracy with 86.63% accuracy, 87% Precision, 89% recall, 87% f1-scores for the BILSTM model respectively, indicating the higher model's performance.

Key-words:

Stance detection, Afan Oromo Language, Deep learning, Social Media

Chapter one

Introduction

1.1. Background of the Study

Stance is defined as the expression of the speaker's standpoint and judgment toward a given proposition. Stance detection plays a major role in analytical studies measuring public opinion on social media, particularly on political and social issues. The nature of these issues is usually controversial, where in people express opposing opinions toward differentiable points. Generally, the stance detection process is also known as perspective and viewpoint detection, in which perspectives are identified by expressing stances toward an object of a controversial topic. (Worku, 2022)

Stance detection is the process of automatically determining the writers' position on a target, or topic of interest. It is dependent on the analysis of a written document and occasionally on the user's social media activity on debate websites. The term "stance detection" has several definitions. We first give various definitions of stance detection in the following section, after which we discuss some associated issues.

Stance detection is the task of automatically determining from the text whether the author of the text is in favor of, against, or neutral towards a proposition or target. The target may be a person, organization, government policy, movement or product etc. Stance detection is formalized as the task of assigning a stance label to a piece of the text concerning a specific target, i.e. whether a text is in favor of or against the given target, or neither of them(Du et al., 2017).The task of stance detection and classification is analogous to the three-way aspect-level sentiment analysis, which identifies the polarity of sentiment toward aspect terms.(Sobhani et al., 2017)

Social media platforms now play a significant role in a person's social interactions. These platforms are regarded as a powerful means of disseminating information in order to share opinions and express ideas. For real-time information and to stay in touch with the rest of the world, people rely on these tools as their main news source. They are helpful because they enable people to examine various facets of popular subjects, express their own viewpoints, receive prompt feedback, and discover the viewpoints of others. Because people utilize these platforms as their main communication tool, researchers can study a variety of aspects of online human behavior, including public sentiment(ALDayel & Magdy, 2021).

Stance detection is much similar to the sentiment analysis task. Opinion sentiment can be used to identify the stance of the document. However, the sentiment category does not simply

reflect the stance. Similarly, the need for automatic stance detection calls for attention because identifying news articles from real news becomes difficult from day today. This is, because of the dramatic rise of news on social media. The consequence of this problem leads the users to find a solution for quickly identifying author text from real one. (Chaudhry, A. K., Baker, D. & Thun-Hohenstein, 2017)

The phrase "Stance Detection" refers to the process of determining a subject's response to a primary actor's assertion. It is a fundamental component in a group of methods for evaluating fake news. More and more people are sharing the news piece on social media sites like Facebook, Telegram and Twitter. It falls into four general categories. agree if the body of text agrees with a headline, unrelated which means the body text talks a different issue than the headline, disagree which means the body text disagrees with the headline and discuss which means the body text discusses the same topic as the headline.(Girma, 2021)

Afaan Oromo news is text-based that is posted on social media, there is a debate on a particular headline. Due to this problem the main aim of stance detection in Afaan Oromo text was to explore the task of automatically determining the news of text published by social media that is posted on site like Facebook, Twitter and Telegram whether there is agree or disagree towards to a proposition or text target.

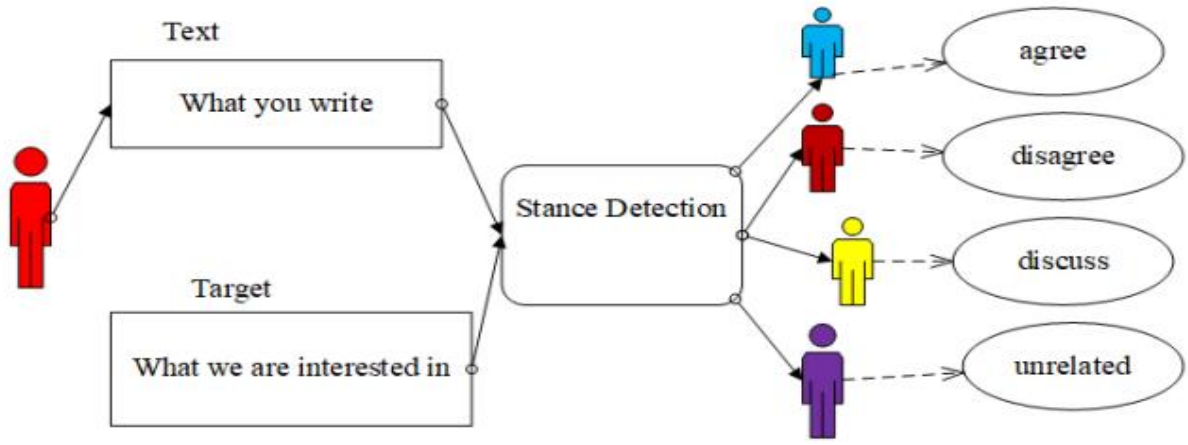

Figure 1.1.Schematic representation of stance detection procedures

1.2. Motivation

Developing stance detection from Afaan Oromo text have big role to ensure that speakers of the language are not left behind in the digital age and also enable them to contribute to the preservation and promotion of Afaan Oromo as inclusion of different languages in technological advances.

Stance classification was a newly emerging research area in the field of natural language processing. Since this research area is the recent and important concept in Afaan Oromo, which has limited works of literature exist, so different social media platforms are available today, and people attempt to express their stance toward a certain topic by using these platforms. Analyzing and classifying the stance helps to support decision-making by knowing the current topics and identifying stances toward those topics, whether they are agree, disagree, discuss or unrelated.

1.3. Statement of the problem

The use of social networks to communicate and express a wide range of viewpoints has increased as a result of recent developments in mobile computing and the Internet, which has improved interaction and idea sharing(Surafel, 2023).

The major purposes of stance classification have been to quantify public opinion toward an event or entity and to determine the attitude toward an entity under examination. Understanding and categorizing the communities' positions in favor of or against something can be accomplished by analyzing their social media posts.

These days, we have seen a variety of issues emerging in our nation, such as the displacement of people from various regions as a result of ethnic civil wars, unemployment, and the skyrocketing cost of living. Ignorance of people's interests or opinions regarding public policy and other issues raises these kinds of issues.

The majority of research on stance classification of social media to assess people's view point toward a particular topic has focused on the English language and European languages due to context-specific linguistic, cultural, and technological differences, particularly in social media communication. Therefore, it is difficult to simply apply models and techniques that are currently in use for the majority of languages with abundant resources to Afaan Oromo. The purpose of this study was to understand people's attitude and point of view around different issues by analyzing numerous Afaan Oromo comments submitted on the social media using deep learning techniques.

The statement problem of stance detection for Afaan Oromo using deep learning is:-

- There is a lack of annotated data for training and evaluating deep learning models. However, creating annotated data for low-resource languages can be challenging and time-consuming.

- A lack of pre-trained language models for Afaan Oromo. Pre-trained language models are large neural networks that learn representations of language from vast amounts of unlabeled text.
- It is usually considered as a sub-problem of sentiment analysis and aims to identify the stance of the text author towards a target (an entity, concept, event, idea, opinion, claim, topic, etc.) either explicitly mentioned or implied within the text.
- There is limited research on stance detection for Afaan Oromo text, an Afro-Asiatic language spoken by over 35 million people in Ethiopia and neighboring countries.

Traditional machine learning approaches for stance detection typically rely on handcrafted features such as n-grams and syntactic structures to identify the stance of a text. These approaches are limited by their inability to handle complex linguistic structures and model temporal dependencies in the text. Deep learning approaches have shown promising results in stance detection tasks, but there is limited research on their effectiveness for Afaan Oromo text. (Mohammad et al., 2016)

To address this gap, we have proposed a deep learning approach for stance detection in Afaan Oromo text using a recurrent neural network (RNN) such as long short-term memory (LSTM), Bidirectional long short-term memory (Bi-LSTM).Therefore, the problem that this research aims to address is how to develop effective deep learning models for stance detection that can handle noisy data and complex linguistic features while minimizing bias and improving performance on multi-lingual text.(Mohammad et al., 2016)

1.4. Research Questions/Hypothesis

To address the present exist issue, this study provided answers to the following questions:

1. What deep learning techniques are currently applied to the stance detection?
2. Which feature extraction technique is better representing Afaan Oromo stance detection dataset?
3. To what extent proposed model is efficient?
4. What would be the performance of the model for stance detection of Afaan Oromo text?
5. How does the proposed deep learning approach for stance detection in Afaan Oromo text compare to traditional machine learning approaches?

1.5. Objectives

1.5.1. General objective

The general objective of this research is to develop an effective deep learning model for stance detection in Afaan Oromo text.

1.5.2. Specific objectives

- Review literatures on the concepts of stance detection and assess different techniques, method and approaches used so far stance Detection.
- To collect and compile necessary dataset for training and testing.
- To apply preprocessing tasks on the collected dataset.
- To identify and apply different feature extraction mechanisms useful for classification detection in Afaan Oromo text using deep learning.
- To design system architecture of Afaan Oromo stance detection model.
- To develop a deep learning model that can accurately detect the stance of Afaan Oromo text towards a particular topic or headline.
- To compare and evaluate the generated classification model using different evaluation metrics.
- To test the models and report their results.

1.6. Scope and Limitations of the study

1.6.1. Scope

The scope of this study concerned on the modeling of Afaan Oromo stance detection text using deep learning and feature extraction techniques to build a detection model.

The focus of the research is on text stance recognition on Facebook, Twitter, and Telegram. Since the dataset gathered text of the stance we acquire is text-based, it does not include picture, video or audio data files. It is not multilingual and mostly focuses on identifying Afaan Oromo stance detection.

The development of stance detection for Afaan Oromo texts on social media was the main focus of this study. It would use a Deep learning model to identify whether the given texts is agree, disagree, discuss, or unrelated to the text target in social media like Facebook, Telegram and Twitter.

1.6.2. Limitations

In this research the absence of standard test corpus and evaluation tool for Afaan Oromo language would be a limitation though we prepared the corpus for the experimentation.

The study would be limited by the availability of labeled data for stance detection in Afaan Oromo text. It may be difficult to obtain large amounts of high-quality labeled data, which could impact on the performance of the deep learning model.

1.7. Significance of the study

The study aims to develop a deep learning-based model for detecting the stance of Afaan Oromo text. It used as input for researcher who wants to do further study and to automatically classify text comments with stance for better management. According to the number of publications published specifically since 2015, stance recognition has lately become a significant research subject in natural language processing (NLP). It is typically characterized as the detection of the stance (as positive, negative, or neutral) of the text producer towards a target and is a research area strongly related to sentiment analysis. Other study issues that are closely related to stance identification, besides sentiment analysis, include emotion recognition, sarcasm detection, controversy detection, opinion mining, and fake news detection and argument mining, among others.(Küçük& Can, 2022)

The study of stance detection for Afaan Oromo text using deep learning approach has several significant implications, including:

- Advancing the development of natural language processing tools and resources for under-resourced languages like Afaan Oromo: Developing a deep learning model for stance detection in Afaan Oromo text can help to fill the gap in natural language processing tools and resources for this language. This can enable more accurate analysis of Afaan Oromo text in various domains, including social media analysis, political discourse analysis, and market research.
- Contributing to the advancement of deep learning techniques for natural language processing: Deep learning has shown promising results in various natural language processing tasks, including sentiment analysis and stance detection. Developing a deep learning model for stance detection in Afaan Oromo text can contribute to the advancement of deep learning techniques for natural language processing.
- Promoting linguistic diversity and cultural understanding: Developing a deep learning model for stance detection in Afaan Oromo text can promote linguistic diversity and cultural understanding by enabling more accurate analysis of Afaan Oromo text.
- In order to address the issue of fake news, our study serves as a foundation for future research in the area of deep learning for the Afaan Oromo language.

1.8. Methodology

The methodology is the steps or procedure that the researcher follows to achieve the stated objectives. It is a road map that shows the direction of how the researcher is going to research to reach the end. The proposed methodology for stance detection in Afaan Oromo text using deep learning approach involves data collection, data preprocessing, model architecture, and model building, model evaluation and model testing.

1.8.1. Data Collection

To model Stance detection system for Afaan Oromo text by using deep learning and would collect lexicon data from various social media source data. The data will be collected from social media platforms such as Facebook, Twitter, Telegram of account pages like Afaan Oromo British Broadcasting Corporation (BBC), Afaan Oromo Voice of America (VOA), Afaan Oromo Fana Broadcasting Corporation (FBC), Oromia news network (ONN), Oromia Media Network (OMN), Oromia Broadcasting Network (OBN), individual activist page and verified personal account pages like Jawar Mohammad, Beekan Guluma, Olifan Merga, Taye Dendea and also different spiritual Facebook accounts and pages etc. The data was collected from selected Facebook pages by using Facepager and Facebook API scrapper tools.

To gather Afaan Oromo tweets that have been labeled with stance agree, disagree, discuss, and unrelated. Diverse viewpoints and opinions regarding a range of subjects should be represented in the dataset.

1.8.2. Data Preprocessing

The next step was to preprocess the data by removing noise, such as, URLs, hash tags, mentions and tokenizing the text into words or sub-words. This step also involves converting the text into numerical vectors that can be fed into a neural network.

Here are some common steps you can take to pre-process your data:

Tokenization is a module that, given the bounds of a written text, divides the text into a collection of tokens, often words. To identify the fundamental components of natural language, such as words, punctuation, and separators, it begins with a series of characters. By using tokenization, input texts provided by users are converted into a series of tokens. Tokenization is required in this study in order to choose contexts, halt word removal, and go on to next phases.

Normalization: - is a process of converting a list of letters or characters in words to a more uniform sequence of letters of words to improve text matching.

Stop Word elimination: - refers to removing of every frequent term from the indexing process, as they are poor in document contents indication.

Stemming: -is a process of reducing words to their word stem, base or root form.

1.8.3. Model Evaluation

The final step was to evaluate the performance of the model on a test set of Afaan Oromo tweets annotated with stance labels. The evaluation metrics include accuracy, precision, recall, and F1-score. The four main metrics used to evaluate a classification model are accuracy, precision, recall and f1-score.

Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

Accuracy = Correct Predictions / All Predictions

Precision is defined as the fraction of relevant examples (true positives) among all of the examples which were predicted to belong in a certain class.

Precision = True Positives / (True Positives + False Positives)

Recall is defined as the fraction of examples that were predicted to belong to a class with respect to all of the examples that truly belong in the class.

Recall = True Positives / (True Positives + False Negatives)

F1-score is balancing precision and recall on the positive class.

F1-score = 2 * (Precision * Recall) / (Precision + Recall).

1.8.5. Development Tools

The following tools will be used in the experiment to develop the prototype of the model.

NLTK: - NLTK is a leading platform for building Python programs to work with human language data. The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language.

Anaconda: - Anaconda is a free and open source distribution of the Python and R programming languages for data science and machine learning related applications (large scale data processing, predictive analytics, scientific computing), that aims to simplify package management and deployment.

Jupyter Notebook: - is an open source web application that allows creating and sharing documents that contain live code, equations, visualizations and narrative text. Uses include:

data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Python: - is a scripting language like PHP, Perl, Ruby and so much more. It can be used for web programming (django, Zope, Google App Engine, and much more). But it also can be used for desktop applications (Blender 3D, or even for games pygame).

Spyder: - is an open source cross platform integrated development environment (IDE) for scientific programming in the Python language.

Pandas: In computer programming, a panda is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. A panda is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

- Microsoft Office Word 2010 for preparation of Thesis proposal and whole documents.
- Power Point for preparation of Presentation slides

1.9. Organization of the Thesis

This study organized into six chapters and the remaining chapters are organized in the following ways.

Chapter two: presents the literature review on Afaan Oromo, Afaan Oromo Writing system, deep Learning Algorithms and related works on applications of Deep Learning algorithms on Afan Oromo Stance Detection.

Chapter three: deals with related works on stance detection for English, for Amharic, and for Afan Oromo.

Chapter Four: presents the detailed process and steps of our research Design. It presents the data sources, data collection, and preprocessing techniques.

Chapter five: present our experimental results and discussion for our proposed detection of stances in Afan Oromo Text using deep learning.

Chapter six: presents the conclusions and some suggestion or recommendations for future work.

Chapter Two

Literature Reviews

In this chapter we were reviewed and introduced a few works on various techniques related to stance detection. Literature review have been conducted on different research works which are related to stance detection and related fields to have understanding of the problem domain and the related researches that are done so far, which helps us for the better understanding to design a new model for Afaan Oromo language. Various literature studies are carried out focusing on various topics related to afaan Oromo stance identification and detection in a manner that highlights the gaps that currently exist. By employing a deep learning algorithm to concentrate on the Afaan Oromo text, this study is connected to these publications. Due to the lack of a standardized corpus to use as a data set, stance detection is a fundamental and novel idea for the Afaan Oromo language. This study offered a novel solution to the Afaan Oromo text stance identification problem as a result of that issue, taking into consideration the lessons we learned from the revision of linked literature works.

2.1. Afaan Oromo Languages

Afaan Oromo was one of the most widely spoken languages in Africa, surpassed only by Arabic and Hausa (Isokoski, 2004). Afaan Oromo is among the major languages that are widely spoken and used in Ethiopia and neighbor countries like Kenya and Somalia. Currently, it is an official language of Oromia regional state (which is the largest regional state in Ethiopia).It is used basically by Oromo people, who are the largest ethnic group in Ethiopia. Natural languages are tools for communication, and humans utilize them by mixing phonologies to produce words, words to build phrases, and phrases to form sentences. More than 33.8% of Ethiopians speak the Cushitic language known as Afaan Oromo. Like other natural languages, it has its own set of grammatical rules. The Latin alphabet Qubee, which was legally adopted in 1991, is used to write Afaan Oromo(Oliyad, 2021). By the late 1970s, the Oromo Liberation Front (OLF) and Oromo people living outside of Ethiopia had been using many iterations of the Latin-based spelling Heine (1986). In Oromia, Afaan Oromo is an official language of Oromia regional government.

2.2. An Overview of Afaan Oromo Writing System

Afaan Oromo writing system was a modification from Latin writing system. Thus, the language shares a lot of features with English writing with some modification (Abebe, 2010). The writing system of the language is known as “Qubee Afaan Oromo” is straightforward which designed based on the Latin script is. Thus letters in English language are also in

Afaan Oromo Language except the way it is written. That means, letters in English or Latin Alphabets are also found in Afaan Oromo except the ways they are combined in phonetic alphabets and the styles in which they are uttered (Martin, 2007). Afaan Oromo text is written from left to right and spaces between words use as demarcation (Wakshum, 2000). Afaan Oromo language uses Latin based script called Qubee, has been adopted and become the official script of Afaan Oromo since 1991 (Abebe, 2010). According to (Girma, 2013), Afaan Oromo is a phonetic language, which means that it is spoken in the way it is written. Unlike English or other Latin based languages there are no skipped or unpronounced sounds/alphabets in the language.

Qubee Afaan Oromo writing system has 33 letters that consists of all the 26 English letters with an addition of 7 combined consonant letters which are known as "Qubee Dachaa". These include ch, dh, sh, ny, ts, ph and zy. All the vowels in English are also vowels in "Afaan Oromo". These are: a, e, i o, and u. Vowels have two natures in the language and they can result in different meaning. The natures are short and long vowels. A vowel is said to be short if it is one. If it is two, which is the maximum, then it is called long vowel. Consider these words: lafa (ground), laafaa (soft), hara (lake), haaraa (new) etc. In a word where consonant is doubled the sounds are more emphasized. For example, dammee (sweety), damee (branch), fala (method), faallaa (opposite), badaa (bad), baddaa (highland, cold area). Afaan Oromo alphabet characterized by capital and small letters as in the case of the English alphabet. In Afaan Oromo language, as in English language, vowels are sound makers and are sound by themselves. Vowels in Afaan Oromo are characterized as short and long vowels. In order to get some insights into the writing system of Afaan Oromo, it is very important to look through the alphabets and sound systems of the language.

Description of Afaan Oromo Alphabets and Sound Systems

Afaan Oromo uses Latin character but with some modifications on sound of consonant and vowels. It has 26 letters called Qubee Afaan Oromo. However, later on a new letter" Z" was included in the alphabet as there are words which require the letter. For example: Amazon (Largest forest in south America), Zelaya (gold), Zeeytuuna (guava), Azoole (river in Arsii), Zeekkara (Opera), Zalmaaxaya (mess), Waziiza (fire place or fire work) and Zawii (insanity) are Afaan oromo words written using "Z" Additionally 'P and V 'are also added. 'P and V' letters are not Afaan Oromo letters because there is no Afaan Oromo word written by use of either of them (Wakshum, 2017). But they are included by considering the fact of handling borrowed terms from other languages like English. For example: Police, Piano, Television,

video, radio, mobile and etc. To sum up there are 33 letters of Afaan Oromo including ‘Z ‘, ‘P ‘, and ‘V ‘.

The Vowels Afaan Oromo Language (Dubbachiiftuu)

There are five vowels in Afaan Oromo Language.

These are: A, E, I, O and U (Upper Case Letter)

a, e, i, o, and u. (Lower Case Letter). They are similar to that of English, but they are uttered differently. Each vowel is pronounced in a similar way throughout its usage in every Afaan Oromo literature. In other words, there is no rule which could violate their pronouncing style in difference contexts.

The Consonants of Afaan Oromo Language (Sagaleewwan dubbifamtoota)

Most of Afaan Oromo constants do not differ greatly from Italian, but there are some exceptions and few special combinations.

We can categorize the consonant into two upper case letter and lower case letter. Those are:

1. B, C, D, F, G, H, J, K, L, M, N, P, Q, R, S, T, V, W, X, Y and Z (Upper Case Letter)
2. b, c, d, f, g, h, j, k, l, m, n, p, q, r, s, t, v, w, x, y and z (Lower Case Letter)
3. CH, DH, SH, NY, TS, PH and ZY (upper case letter of combined consonant letters)
4. ch, dh, sh, ny, ts, ph and zy (Lower case letter of combined consonant letters)

2.3. Afaan Oromo Language Parts of Speeches

2.3.1. Word class in Afaan Oromo

In line with the basic classifications of words in English language, Afaan Oromo words are categorized into eight grammatical categories (Noun, Verb, Adjective, Adverb, Adposition, Pronoun, Conjunction and Interjection); however, some researchers like MandefroLegesse (2012), Assefa (2005) and Getachew (2009) as cited in (Abebe, 2013) revealed that Afaan Oromo has five grammatical categories of words such as: nouns, verbs, adverbs, adjectives and Apposition.

2.3.2. Afaan Oromo Noun Class

Afaan Oromo nouns are words used to name or identify any member of a class of persons, places, or objects, much like in other languages. Five categories exist for nouns in Afaan Oromo: proper noun, common noun, collective noun, material noun, and abstract noun(Oliyad, 2021). A noun is a word that helps to identify the categories of things, people, places and ideas. Nouns in Afaan Oromo are inflected for gender, definiteness and number (Abebe, 2010).

Proper nouns are nouns that are specifically relate to individuals, places, or things in Afaan Oromo. Bona, Oda, Arsi, Tolassa, sululta, Boorana and so on.

Nouns that are common, generic, non-specific categories are denoted by common nouns. Some examples of common nouns include Farda (horse), nama (person), Barsiisaa (teacher), biyya (country), laga (river), and so on.

Materials name that are used to make things are referred to as material nouns. Consider biddeena(injera), damma(honey). Book(kitaaba), food(nyaata), beds(siree),table (minjaala) and so forth.

Ideas that are intangible are called abstract nouns. They are not tangible objects that people can touch, see, smell, or hear. Typical instances include feelings, social ideas, political theories, and personal qualities (such as love, creativity, and democracy, jaalala and kallaqa). Abstract nouns include, among others, bilisumma (freedom), hiyyummaa (poor), dhukkuba (deasese), fayya (healthy), laafina (weakiness), jabeenya (strength), and so on.(Oliyad, 2021)

2.3.3. Afaan Oromo Verb class

A term like hear, become, or happen that is used to describe an action, state, or occurrence and that forms the majority of the sentence's predicate. verb that describes the actions and conduct of the subject in a sentence. Therefore, without a verb, no phrase can express all the information required. Afaan Oromo the main components of verbs are a stem that conveys the verb's lexical meaning and a suffix that denotes the verb's aspect, tense, and subject agreement.

Example in following sentences;

a) Caaltuun kaleessa uffata **bitatte**.

b) Boonaan konkolaataa haaraa **bite**.

c) Gadaan gara Finfinnee **deeme**.

In all above sentences the underline words “bitatte”,”bite” and “deeme” in respectively to sentences are verbs. Characters of these all words are to terminate the sentences within.

2.3.4. Afaan Oromo Adjective Class

An adjective was a word that describes or modifies a noun or pronoun. The primary benefits of adjectives are their ability to define nouns and provide a clear explanation for them. This implies that it can tell us more details about the objects that nouns and pronouns represent.

Adjectives are very important in Afaan Oromo because its structure is used in every day conversation. Afaan Oromo Adjectives are words that describe or modify another person or thing in the sentence (Debela, 2010).

Afaan Oromo adjectives can alter nouns or pronouns by coming after them.

Example in following sentences;

a. Gammadaan sangaa **diimaa** gurgure.

2.3.5. Afaan Oromo Adverb Class

An adverb was a word that modifies or limits the meaning of a verb, an adjective, another adverb, a preposition, a phrase, a clause, or a sentence. Adverbs are used to modify verbs that follow in the Afaan Oromo language. Adverbs usually precede the modified verb, but it's important to remember that words that appear before verbs aren't necessarily suitable candidates for being an adverb.

Example: Galataan **kaleessa** dhufe. (Geleta was come yesterday)

Caalaan **ariitiin** deeme. (Chala was went quickly)

In this example, the adverb kaleessa (yesterday) go before the verb dhufe (come) that it modifies.

In this example, the adverb ariitiin (quickly) go before the verb deeme (went) that it modifies.

2.3.6. Afaan Oromo Pre-Position and Post-Position Class

A preposition was a word or group of words that is used with a noun, pronoun, or noun phrase to show direction, location, or time, or to introduce an object.

Postpositions are a type of grammatical particle used to indicate the relationship between a noun or pronoun and other elements in a sentence.

Pre- and post- positions tie with nouns, pronouns and with other words in a sentence.

The main properties of pre- and post- positions are: they never use affixes and they don‘t contribute to form other words. A preposition in Afaan Oromo ties a noun to an action (e.g. achi irraa deemi) or to another noun (kitaabni minjaalarra jira). For the purpose of simplicity, this section will divide Afaan Oromo prepositions into two classes: prepositions and postpositions, with prepositions coming before the noun and postpositions coming after the noun they relay to. Ala (out, outside), bira (beside, with, around), booda (after), cinaa‘ (beside, near, next to) are example Postpositions. Some examples of prepositions are, gara (towards), erga (since, from, after), hanga (until), hamma (up to, as much as) and etc.

Afaan Oromo Conjunctions

A word that can be used to join or connect two words, phrases, clauses and sentences is known as a conjunction (Gazagn, 2012). Conjunctions can be divided into coordinating and subordinating conjunctions. Coordinating conjunctions are used to connect two independent clauses. Mostly these conjunctions are used when the speaker needs to lay emphasis on the

two sentences equally. Some of these conjunctions in Afaan Oromo include: garuu “but”, “moo” or“, kanaafuu “therefore”, haata’u malee “however/so”, ta’ullee “even though” etc.

Punctuation Marks

Punctuation was the act or system of using specific marks or symbols in writing to separate different elements from each other or to make writing more clear a mark, such as a full stop, comma, or question mark, used in writing to separate sentences and their elements and to clarify meaning.

Punctuation is included in texts to facilitate easy reading and to clarify content. Investigation of Afaan Oromo literature reveals that punctuation patterns in languages using the Latin writing system are similar for various types of punctuation. The following are some of the punctuation symbols that are often used in Afaan Oromo, much like in English.

Question Mark (?)

A question mark (Mallattoo Gaaffii) is used in interrogative statements or to conclude a straight query in Afaan Oromo. The compelling speech must conclude with this writing. To inquire about someone else, use the question mark. “Mallattoo Gaaffii”, Question mark (?): is used in interrogative or at the end of a direct question.

Example:

a) Magaalaa Finfinnee yoom deemtaa?

b) Gaheen hojii kee maalidha?

Full Stop (.)

The full stop (Tuqaa) in Afaan Oromo is a punctuation mark that is used in abbreviations and at the conclusion of sentences. “Tuqaa”, Full stop (.): Like English full stop is used at the end of a sentence and also in abbreviations. The sentence ends with a full stop every time. This punctuation mark can be used to convey important information.

Example

a) Boonsaan Amboo deeme.

b) lakkoofsa =Lakk.

c) Lakkoofsa saanduuqa poostaa=L.S. P 395

Exclamation mark (!)

In Afaan Oromo, Exclamation mark (Raajeffannoo) is used at the end of command and exclamatory sentences. We can use this exclamation mark to express our emotions to others.

Example

a) Badi asiitii!

b) Dafii koottu hin turiin!

c) Akkas malee korma koo!

Comma (,)

Comma (Qoodduu) is used to separate listing in a sentence or to separate the elements in a series. We can use this punctuation mark to separate a list of words. Example

a) Boqonnaa 3, fuula 250, bara 2000

b) Finfinnee, Oromiyaa

c) Guyyaa Roobii, Mudde 20, 1970.

d) Boontuun gabaa deemtee buna, ashaboo, zayita, barbaree fi raafuu bittee galte.

Colon (:)

In Afaan Oromo, Tuq-lamee (colon) is used to separate and introduce lists, clauses, and quotations, along with several conventional uses, and etc.

Example

a) Galaanaan kaleessaa sa'aatii 12:30 tti gale.

2.4. Sentences in Afaan Oromo

In Afaan Oromo, A sentence was group of words that are correctly structured to give one meaning. This definition is common in whichever language. But may be the structure is different depending on the language. In language like, English sentence must have subject, object, verb (S+O+V). But in Afaan Oromo, there is some dissimilarity.

Afaan Oromo Structure (Syntax)

Basically, every language has standardized word orders in sentences. For instance, (Abdi, 2015) stated that English and French languages have 'Subject-Verb-Object' (SVO) orders of words in their sentences. However, Afaan Oromo is different from these languages in its syntactic structure; it uses subject-object-verb (SOV) form which is similar to Amharic and Japanese languages (Charles, 1996). Subject-verb-object (SOV) is a sentence structure where the subject comes first, and the object and the verb are second and third elements of a sentence respectively. Afaan Oromo uses subject-object-verb (SOV) structure unlike English which has (SVO) structure.

There are different rules to word order in Afaan Oromo sentence construction understanding of this syntactic structure of sentence can help us to know the relationship between words which in turn leads us to categorize them correctly.

2.4.1. Types of Sentences

Fundamentally Afaan Oromo sentences can be classified in to different types based on structurally and functionally (Adugna, 2014).

Structurally, Afaan Oromo sentences can be classified in to four categories.

A) Simple sentence

In Afaan Oromo, A simple sentence contains a subject and a verb, and it may also have an object and modifiers. But, it contains only one independent clause.

Example

a) Caalaan deeme.

b) Margaan gara mana barumsaa deeme.

c) Bookaan buna dhuge.

B) Compound Sentence

In Afaan Oromo like other language, a compound sentence contains at least two independent clauses. These two independent clauses can be combined with a comma and a coordinating conjunction or with a semicolon in this language.

Example:

a) Bojaan gabaa deemee gale.

b) Boontuun ganama kaatee, buna danfistee, ciree ishee nyaattee mana barumsaa deemte.

C) Complex Sentence

In Afaan Oromo language, a complex sentence contains at least one independent clause and at least one dependent clause. This Dependent clause can refer to the subject (who, which) the sequence/time (since, while), or the causal elements (because, if) of the independent clause.

Example

a) Finfinnee yoo deemte, meeshaa naaf bittee galta.

b) Bokkaa cimaa waan roobeef, lolaan qabeenya hedduu mancaase.

c) Hoongeen waan dheerateef, uummatni beelaan miidhame.

D) Compound-Complex Sentences

Sentence types can also be combined from different types. In Afaan Oromo, A compound-complex sentence contains at least two independent clauses and at least one dependent clause.

Example

a) Bosonni yoo gubatuu fi biqilootni yoo manca'an faalama qilleensaatu dhufa.

b) Mirgaan yaabbattus, bitaaan yaabbattus walgahiin kooraa dhuma.

2.4.2 Functional Sentences

According to the basis of the meaning that their function, Afaan Oromo sentences can be divided into four main types. They are:

A) Declarative Sentence

A sentence tells a plain statement or expresses an opinion in Afaan Oromo is termed as a declarative sentence. In simple words, an ordinary statement is identified as a declarative sentence.

Example

a) Galmeen barnootaa eegalee jira.

b) Keeniyaan filannoo pireezidaantummaa gaggeessaa jirti.

B) Imperative Sentence

A sentence that in the form of order, command, instruction or a request, is known as an imperative sentence. Imperative sentence can end with a period or exclamation mark or a question mark.

Example

a) Bakka hojii kee deemi!

C) Interrogative Sentence

An Afaan Oromo sentence denotes an interrogative characteristic and ends with a question mark, is known as an interrogative sentence.

Example

a) Qajeelaan eessa jiraata?

b) Finfinnee yoom deemna?

D) Exclamatory sentence

In Afaan Oromo, A sentence that denotes different expressions like shock, surprise, anger, sad,happiness etc. is known as an exclamatory sentence. Exclamatory sentence should always end with an exclamation mark.

Example

a) Safuu dhabboo!

b) Yaa rabbi nu baasi!

c) Kun ajaa’iba rabbiiti!

2.5. Natural Language Processing

Natural Language Processing was an area of artificial intelligence that works to make computers more proficient at understanding human language. Data types with a lot of organization are databases. In contrast, the Internet has very few structural elements and is entirely unstructured. In this case, NLP's ultimate goal is to understand and simulate human

language. Natural Language Processing (NLP) is a field of study and application that looks into how computers can interpret and modify natural language text or speech in order to perform useful task. Natural Language Processing (NLP) refers to the use of computational methods to process spoken or written form of such free text which acts as a mode of communication commonly used by humans (Singh, 2018).

A collection of computational methods motivated by theory, natural language processing (NLP) is used to analyze and mechanically expressing human language. Currently, it is possible to evaluate millions of webpages in under a second. The study and practice of natural language processing, or NLP, examines how Natural language writing and speech can be interpreted and modified by computers to provide beneficial assignments (Girma, 2021).

Natural Language Processing (NLP) helps intelligent machines by improving their grasp of human language for linguistically based human-computer communication. Recent advances in computing power, as well as the availability of vast volumes of linguistic data, have increased the need for and demand for data-driven ways to automate semantic analysis (Girma, 2021).Using natural language features, natural language processing (NLP) attempts to convert unstructured data into computer-readable language. Machines decode any textual material using intricate algorithms to retrieve relevant information. Afterwards, robots are taught natural language reasoning using the acquired data. Using syntactic and semantic analysis, natural language processing allows machines to be guided by patterns in data.

2.6. Stance Detection Using Deep Learning Approach

Finding the viewpoint (either in favor of, against, or neither) toward a target in a given text is known as "stance detection." The growth of social media material has drawn more attention to this type of research. The traditional approach to stance detection involves turning it into challenges involving text classification. In order to solve such problems, deep learning models have already supplanted rule-based models and conventional machine learning models. There are two primary issues with current deep neural networks: the lack of labeled data and information in social media posts, as well as the inexplicable nature of deep learning models.

The definition of stance is "the expression of the speaker's standpoint and judgment toward a given proposition". In analytical research evaluating public opinion on social media, particularly on political and social matters, stance detection is a key component. These topics are typically contentious in nature, eliciting differing viewpoints over distinguishable aspects. Target themes for stance detection on social media have included feminism, abortion, and

climate change. Comparably, political subjects like elections and referendums have long been popular subjects for stance detection research on public opinion.(ALDayel & Magdy, 2021)

Human Language Technology and Computer Voice Synthesis are fields that deal with the use of natural language in interactions between people and machines. It enables computers to analyze natural language, aids in our understanding of human language, and it into a form that can be used for data representation.

NLP primarily focused on getting computers to perform significant and engaging tasks that were easily understood in human languages, like text-to-speech or speech-to-text conversion, natural language understanding, and machine translation to connect machines (computers) with people in their native tongue. An automated system for interpreting and representing human language is called natural language processing(Girma, 2021).

2.7. Overview of Afaan Oromo Language Stance Detection

Stance detection was a natural language processing (NLP) task that involves determining the stance or position expressed in a given text towards a particular topic or claim. It aims to classify the text as supporting, opposing, or neutral towards the target claim. The goal of stance detection was to understand the opinions, beliefs, and attitudes expressed in text data. It has various applications, including social media analysis, political discourse analysis, fake news detection, and sentiment analysis.

Stance detection using deep learning was a popular approach that leverages neural network architectures to classify the stance or position expressed in a given text towards a particular topic or claim. Deep Learning Models have shown promising results in capturing complex relationships and contextual information, making them well-suited for stance detection tasks.

"Stance Detection" is a term used to represent the extraction of a subject's reaction to a claim made by a primary actor. It is a core part of a set of approaches to fake news assessment. The news article is increasingly being shared via social media platforms like Twitter and Facebook. It can be categorized broadly into four groups. Agree, which is the body text agrees with the headline, disagree, which is disagrees: The body text disagrees with the headline, discusses, which was the body text discuss the same topic as the headline and unrelated, which is the body text discusses a different topic than the headline. Stance detection is the task of automatically determining from the text whether the author of the text is in agree of, disagree, discuss, or unrelated towards a proposition or target based on what to be interested. (Krejzl & Uk, 2018)

Analyzing a writer's point of view through their writing is the fundamental goal of stance detection in sociolinguistics. Using three criteria linguistic acts, social interactions, and human identity, stance detection seeks to identify the writer's inherent point of view from the text. Adverbs, lexical components, and adjectives are commonly used in conjunction with other language qualities in stance detection(Surafel, 2023).

Although they have certain similarities, sentiment analysis and stance identification are not the same. Find out if a text is neutral, negative, or favorable. Analyze the text to ascertain the speaker's point of view and the object of the speaker's opinion (the thing the opinion was directed towards). Contrarily, stance identification necessitates that systems estimate favorability toward a predefined target of interest. It's likely that neither the text's target of opinion nor its objective of interest is discussed in full.

2.8. Stance Detection According to Target

Stance classification was the problem of determining if a particular topic or entity has a stance in favor of or against it. Therefore, in order to ascertain the attitude towards a preset goal, stance detection requires the presence of the target. Three categories can be used to categorize stance detection depending on the type of target being assessed(Surafel, 2023).

Stance classification is the problem of determining if an entity or topic has an opinion that is in favor of or against it. Thus, posture detection requires the presence of a predetermined target to ascertain the attitude toward it. The three kinds of stance detection can be attributed to the type of target under consideration (Surafel, 2023).

As was previously said, in order to identify a target's stance, the stance identification task requires the target to be present. Three categories: single specified targets, multi-related targets, and claim-based targets can be used to aggregate stance detection in the literature now under publication based on the kind of assessment target(ALDayel & Magdy, 2021).

Target-Specific Stance Detection

Target-specific stance detection is the most fundamental type of stance detection on social media. The majority of earlier research concentrated on stance inference for a predetermined set of targets. The text (T) or the user (U) serves as the primary input in this sort of stance detection in order to anticipate the stance toward a particular preset single target (G)(ALDayel & Magdy, 2021).

In the task of stance identification, the dependent factor is the analysis's goal. One common approach to stance identification on social media is to deduce the stance only from the unprocessed text, which simplifies the problem to textual entailment work.

Stance (T, U|G) = {Favor, Against, None}

The main strategy for different types of stance research is target-specific attitude identification; even for benchmark datasets including many subjects, like SemEval stance 2016, most published work on this dataset trained a different model for each topic (target) separately.

Multi-related-targets stance detection

Multi-target stance recognition aims to comprehend the orientation of social media users toward two or more targets for a single issue simultaneously. The fundamental idea behind this kind of stance detection is that an individual's posture for one thing indicates how they feel about other targets that are connected to it(ALDayel & Magdy, 2021).

The goal of multi-target stance identification was to identify the author's attitude from a series of related targets and a text passage as an input for a classification issue. For every target, there is a stance classification that can be Favor, Against, or neither; furthermore, the classifications for each target may impact the other targets(Surafel, 2023).

Claim-based Stance Detection

In claim-based or open-domain attitude identification, the analysis centers around a claim made in a news article rather than an explicit entity like those described above. The primary objective statement in the dialogue sequence or supplied text should be identified as the first step in identifying the attitude. The claim (C) is the main input for the claim-based stance identification method. It could be the headline of the article or the post made by the rumor. Prediction label sets typically represent the statement as either true or false(ALDayel & Magdy, 2021).

Stance (T, C) = {Confirming, Denying, Observing}

2.9. Overview of stance classification

The definition of stance classification is the automatic identification of an author's support or opposition to a given objective in a piece of writing. The definition of stance is the way a speaker expresses their opinion and point of view regarding a particular proposition. In analytical research measuring public opinion on social media, stance detection is a key component.

According to (ALDayel & Magdy, 2021), a rising corpus of research has examined numerous social and political concerns on social media platforms by using the idea of stance recognition as a new frontier. Stance classification is typically regarded as a sub-problem of sentiment analysis and seeks to determine the author's position regarding a target (an entity,

concept, event, idea, opinion, claim, topic, etc.) that is either stated directly in the text or suggested in it. The way a person stands or positions themselves in relation to a target a person, an item, a concept, or an opinion is known as their stance. The position can be either agreed upon (e.g., by supporting someone or something, by opposing or criticizing someone or something opposed to the target, or by echoing the stance of somebody else); disagreed with (e.g., by opposing or criticizing someone or something, by supporting someone or something opposed to the target, or by echoing the stance of somebody else); or neither (e.g., by standing in a neutral position, or by not clearly expressing the stance within the passage). The process of automatically identifying from a text whether the writer agrees with, disagrees with, discusses, or has no connection to a proposition or target is known as "stance detection".

Defining Stance Detection

There are three primary categories of stance detection: Fake News stance detection, Rumor stance classification, and Generic stance detection. The Generic Stance Detection contain Multi-Target stance detection and Cross-Target stance detection which are the two popular methods for approaching generic stance identification(Roeters Van Lennep, 2021). Here are definitions for every term.

Generic Stance Detection

Recognizing Generic Stance: The most widely used type of stance detection is generic. It resembles a task involving classification. A paragraph and a goal make up the input. Next, based on how the text pertains to the target, the stance detection separates the text into three primary groups. "Favor," "Against," and "Neutral" are the most popular categories.

The position of the text is determined by these categories.

Multi-Target Stance Detection: Multi-target stance detection is a classification problem for an input consisting of a text passage and a set of related targets. The text author's stance is sought as a category label from this set: {Favor, Against, Neither} for each target. The classification of each stance (for each target) may impact the classification of the remaining targets. This definition will be used in this paper to explain multi-target stance detection. Future references to stance detection will only pertain to multi-target stance detection.

Cross-Target Stance Detection: In cross-target stance detection, a classification problem, the author's stance is searched for as a category label for a particular target from the following set: {Favor, Against, Neither}. This is done in situations where stance annotations are available for targets that are related but distinct from each other, meaning that there is insufficient stance-annotated training data for the target in question.

Rumour Stance Classification

A text passage and a pair of rumors are the inputs for rumor stance classification. This form determines whether the text's theme is consistent with the rumor or not. Supporting and Denying are the two primary groups. To expand on the two primary categories, some also include commenting and querying.

Fake News Stance Detection

In this type of posture identification, the news item's headline and text which need not be the same are used as input. The Fake News stance identification divides the article's content into four primary categories: Agree, Disagree, Discuss, and Unrelated.

Stance classification Vs. Sentiment analysis

Stance detection was not the same as sentiment analysis. Determining the speaker's point of view or whether a text is neutral, positive, or negative are examples of sentiment analysis tasks. In contrast, attitude identification necessitates that systems determine favorability toward a particular (pre-selected) object of interest. The text may not specifically name the object of interest or it may not be the text's target of opinion(Girma, 2021).

Liu(2010) states that the computational analysis of opinions, attitudes and emotions represented in text is known as sentiment analysis or opinion mining. We learn that the sentiment analysis problem was concerned with the sentiment without a specific target, which is anticipated by posture identification difficulties, from the work of Dilek Kucuk et al. (2020). The sentiment and posture (for a target) in the same sentence may not be at all aligned in addition to the target.

There are two sub-problems in sentiment analysis that are more similar to the stance classification problem. These two types of sentiment analysis are target-dependent and aspect-based. Their distinction is described as follows, nevertheless. Aspect-based sentiment analysis, as defined by García Pablos et al. (2015), takes into account the sentiment polarities towards a target entity (such as laptops) and several features (such as price) of this thing in a given input text. This topic is primarily utilized on review sites to create tables based on user ratings, not from free text reviews, but rather for a small set of specified aspects. Target dependent sentiment analysis, on the other hand, examines the text's sentiment polarity toward the target given a text and target pair(Girma, 2021).According to Tang et al. (2016), there are two primary distinctions between target-dependent sentiment analysis and stance detection:

(1) The stance target might not be specified explicitly in the input text, and

(2) The stance target might not be the sentiment in the text's target.

(3) Another distinction is that, in stance analysis, the target is often an entity or an aspect; in sentiment analysis, however, the target may be an event. Take a look at this sentence: "I bought a new mobile." The battery life is very short, despite the high storage. Given that the line presents a favorable assessment of high storage, "positive" sentiment polarity is expected. Sentiment polarity that is appropriate if the objective is to prolong battery life is "negative."

2.10. Stance Classification Using Machine Learning Algorithm

Machine Learning Algorithms

A subset of artificial intelligence (AI) is machine learning (ML), a specialized technology. By simulating human learning with data and algorithms, this field of study enables machines to get better over time and become more accurate at classifications, predictions, and data-driven insights. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves (Worku, 2022).

At the core of machine learning are algorithms, which are trained to become the machine learning models used to power some of the most impactful innovations in the world today.

Machine learning algorithms are computational models that allow computers to understand patterns and forecast or make judgments based on data without the need for explicit programming. These algorithms form the foundation of modern artificial intelligence and are used in a wide range of applications, including image and speech recognition, natural language processing, recommendation systems, fraud detection, autonomous cars etc.

A machine learning algorithm was a collection of guidelines or procedures that an artificial intelligence (AI) system uses to carry out tasks. These tasks are often related to either predicting output values from a given set of input variables or finding new data insights and patterns. In order to train a classifier using machine learning, the first step is feature extraction, which involves using a technique to convert each word into a vector representation of numbers. Sack of Words is one of the most popular methods; a vector indicates how often a term appears in a pre-established lexicon (Oliyad, 2021).

Types of Machine Learning Algorithms

There are four types of machine learning algorithms: supervised, unsupervised, semi-supervised, and reinforcement. Every type and variation has benefits of its own, depending on your budget and the demand for speed and precision.

Advanced machine learning algorithms require multiple technologies including deep learning, neural networks and natural language processing and are able to use both unsupervised and supervised learning.

The following are the most popular and commonly used algorithms.

1. Supervised Learning

This type of machine learning feeds historical input and output data in machine learning algorithms, with processing in between each input/output pair that allows the algorithm to shift the model to create outputs as closely aligned with the desired result as possible. Common algorithms used during supervised learning include Neural Networks, Decision Trees, Linear Regression, and Support Vector Machines.

Supervised learning can be separated into two types of problems when data mining: classification and regression.

Classification: uses an algorithm to accurately assign test data into specific categories. It recognizes specific entities within the dataset and attempts to draw some conclusions on how those entities should be labeled or defined. Common classification algorithms are linear classifiers, support vector machines (SVM), decision trees, K-nearest neighbor and random forest.

Regression: is used to understand the relationship between dependent and independent variables. It is commonly used to make projections, such as sales revenue for a given business. Linear regression, logistical regression, and polynomial regression are popular regression algorithms.

Supervised learning algorithms include:

Artificial neural networks: Also known as ANNs, neural networks or simulated neural networks (SNNs), are a subset of machine learning techniques and are at the heart of deep learning algorithms. The learner algorithm recognizes patterns in input data using building blocks called neurons, approximating the neurons in the human brain, which are trained and modified over time.

A Decision tree algorithm: is a supervised learning algorithm used for classification and predictive modeling tasks. It resembles a flowchart, starting with a root node that asks a specific question about the data. Based on the answer, the data is directed down different branches to subsequent internal nodes, which ask further questions and guide the data to subsequent branches. This process continues until the data reaches an end node, also known as a leaf node, where no further branching occurs.

Because of their simplicity and ease of handling complicated datasets, decision tree algorithms are widely used in machine learning. The structure of the algorithm facilitates an easy interpretation and understanding of the decision-making process.

Decision tree algorithms used for both predicting numerical values (regression problems) and classifying data into categories, decision trees use a branching sequence of linked decisions that may be represented with a tree diagram. One of the advantages of decision trees is that they are easy to validate and audit, unlike the black box of a neural network. This simplicity and interpretability make decision trees valuable for various applications in machine learning, especially when dealing with complex datasets.

K-nearest neighbor (KNN): is a supervised learning algorithm commonly used for classification and predictive modeling tasks. The name "K-nearest neighbor" reflects the algorithm's approach of classifying an output based on its proximity to other data points on a graph. Based on the majority of the labels among the K nearest neighbors, the algorithm assigns a classification to the new data point. Additionally, KNN can also be used for prediction tasks. Instead of assigning a class label, KNN can estimate the value of an unknown data point based on the average or median of its K nearest neighbors.

K-nearest neighbor also known as KNN, this non-parametric algorithm classifies data points based on their proximity and association to other available data. It assumes that similar data points can be found near each other. As a result, it seeks to calculate the distance between data points, usually through Euclidean distance, and then it assigns a category based on the most frequent category or average.

Linear regression: is a supervised machine learning technique used for predicting and forecasting values that fall within a continuous range, such as sales numbers or housing prices. It is a technique derived from statistics and is commonly used to establish a relationship between an input variable (X) and an output variable (Y) that can be represented by a straight line.

Linear regression is primarily used for **predictive modeling** rather than categorization. It is useful when we want to understand how changes in the input variable affect the output variable. By analyzing the slope and intercept of the regression line, we can gain insights into the relationship between the variables and make predictions based on this understanding.

Linear regression is used to identify the relationship between a dependent variable and one or more independent variables and is typically leveraged to make predictions about future outcomes. When there is only one independent variable and one dependent variable, it is known as simple linear regression.

Logistic regression, also known as "logit regression," is a supervised learning algorithm primarily used for binary classification tasks. It is commonly employed when we want to determine whether an input belongs to one class or another, such as deciding whether an image is a cat or not a cat.

Logistic regression predicts the probability that an input can be categorized into a single primary class. However, in practice, it is commonly used to group outputs into two categories: the primary class and not the primary class. To accomplish this, logistic regression creates a threshold or boundary for binary classification. For example, any output value between 0 and 0.49 might be classified as one group, while values between 0.50 and 1.00 would be classified as the other group.

Consequently, logistic regression is typically used for binary categorization rather than predictive modeling. It enables us to assign input data to one of two classes based on the probability estimate and a defined threshold. This makes logistic regression a powerful tool for tasks such as image recognition, spam email detection, or medical diagnosis where we need to categorize data into distinct classes.

Logistic regression**,** while linear regression is leveraged when dependent variables are continuous, logistic regression is selected when the dependent variable is categorical, meaning there are binary outputs, such as "true" and "false" or "yes" and "no." While both regression models seek to understand relationships between data inputs, logistic regression is mainly used to solve binary classification problems, such as spam identification.

Defining Neural Networks

A neural network is structured like the human brain and consists of artificial neurons, also known as nodes. These nodes are stacked next to each other in three layers:

- The input layer
- The hidden layer(s)
- The output layer

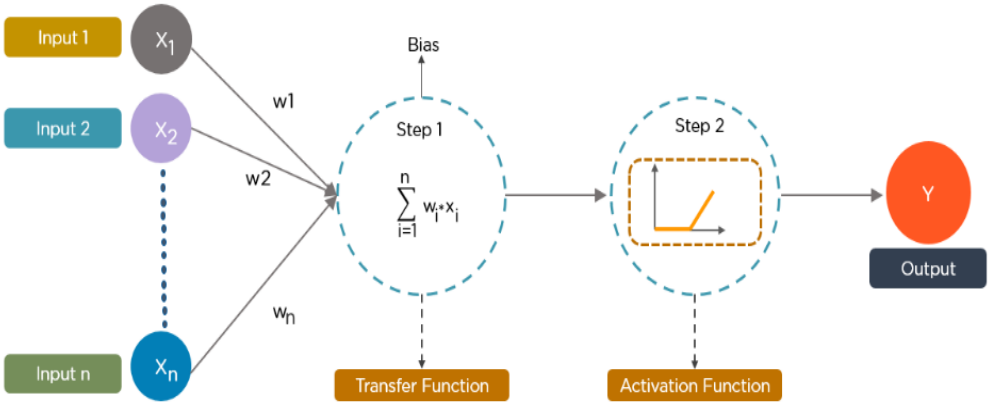

Figure 2.1.Neural Network (By Avijeet Biswal, 2024)

Data provides each node with information in the form of inputs. The node multiplies the inputs with random weights, calculates them, and adds a bias. Finally, nonlinear functions, also known as activation functions, are applied to determine which neuron to fire (Avijeet, 2024).

Naive Bayes: is a set of supervised learning algorithms used to create predictive models for binary or multi-classification tasks. It is based on Bayes' Theorem and operates on conditional probabilities, which estimate the likelihood of a classification based on the combined factors while assuming independence between them.

Naïve Bayes adopts the principle of class conditional independence from the Bayes Theorem. This means that the presence of one feature does not impact the presence of another in the probability of a given outcome, and each predictor has an equal effect on that result. There are three types of Naïve Bayes classifiers: Multinomial Naïve Bayes, Bernoulli Naïve Bayes and Gaussian Naïve Bayes. This technique is primarily used in text classification, spam identification and recommendation systems.

The Naive Bayes method is capable of handling big datasets with efficiency because it makes use of the premise of factor independence, which streamlines computations. Tasks where the components may be taken into consideration independently while still contributing to the total classification, such as document classification, email spam filtering, sentiment analysis, and many more, are especially well-suited for it.

Random Forest: A random forest algorithm is an ensemble of decision trees used for classification and predictive modeling. Instead of relying on a single decision tree, a random forest combines the predictions from multiple decision trees to make more accurate predictions. Once trained; the random forest takes the same data and feeds it into each decision tree. Each tree produces a prediction, and the random forest tallies the results. The

most common prediction among all the decision trees is then selected as the final prediction for the dataset.

In a random forest, the machine learning algorithm predicts a value or category by combining the results from a number of decision trees. The "forest" refers to uncorrelated decision trees, which are assembled to reduce variance and enable more accurate predictions. Random forests address a common issue called "over fitting" that can occur with individual decision trees. Over fitting happens when a decision tree becomes too closely aligned with its training data, making it less accurate when presented with new data.

A support vector machine (SVM): is a supervised learning algorithm commonly used for classification and predictive modeling tasks. SVM algorithms are popular because they are reliable and can work well even with a small amount of data. The method of binary categorization assigns all of the elements (tweets) to one of two classes (Worku, 2022).

The goal of SVM is to find the best possible decision boundary by maximizing the margin between the two sets of labeled data.

2. Unsupervised Learning

While supervised learning requires users to help the machine learn, unsupervised learning algorithms don't use the same labeled training sets and data. Instead, the machine looks for less obvious patterns in the data. Unsupervised machine learning is very helpful when you need to identify patterns and use data to make decisions. Common algorithms used in unsupervised learning include Hidden Markov models, k-means, hierarchical clustering, and Gaussian mixture models.

Unsupervised algorithms are widely used to create predictive models. Common applications also include clustering, which creates a model that groups objects together based on specific properties, and association, which identifies the rules between the clusters.

Unsupervised Learning Algorithms

Unsupervised learning is the process of teaching a machine using data that is neither neutrally classed nor labeled, and then allowing the algorithm to operate on that data unsupervised. Without any prior dataset training, the machine's objective in this case is to categorize unsorted data based on similarities, patterns, and differences (Oliyad, 2021). Unlike supervised learning, unsupervised learning uses unlabeled data. This is particularly useful when subject matter experts are unsure of common properties within a data set. Unsupervised, machine learning problems further grouped into clustering and association problems.

Clustering: In the context of unsupervised learning, clustering is an important idea. Its main focus is on identifying a structure or pattern inside an unclassified data collection. If there are any natural groupings in our dataset, clustering techniques will find them(Oliyad, 2021). Clustering algorithms will process our dataset and find natural clusters if they exist in the dataset. We can also modify the number of clusters that our algorithms should detect. It permits us to change these groups' level of granularity. Various kinds of clustering algorithms exist. - Singular value decomposition, Principal Component Analysis, K-means Clustering, K-NN (k nearest neighbors), Hierarchical Clustering, and Independent Component Analysis. Clustering algorithms are particularly useful for large datasets and can provide insights into the inherent structure of the data by grouping similar points together. It has applications in various fields such as customer segmentation, image compression, and anomaly detection.

Association: Association rules allow establishing associations between data objects inside large dataset. This unsupervised technique is about discovering interesting relationships between variables in large dataset.

3. Semi-Supervised Learning: Have characteristics from both supervised and unsupervised learning. It may be used to avoid the costly labeling process or when there is insufficient labeled data for a supervised learning algorithm. A semi-supervised machine learning model is situated in between unsupervised and supervised learning. Partially labeled datasets, in which most of the data is unlabeled, are acceptable for semi-supervised learning. Like unsupervised learning, semi-supervised learning looks for patterns in the data points and then uses the labeled data to highlight those patterns. Lastly, using the recently applied labels, the entire model is trained (Oliyad, 2021). It is not a good idea to train the model using only a limited amount of labeled data because it is generally known that training results will be less accurate when the ratio of training sample numbers to feature measurements is low. To increase performance, labeled and unlabeled data must be combined by the model throughout the training process. For preparing the labeled data, including identifying innate structure in the domain, or density estimates, the unlabeled data can be used. That is, the model uses the annotated data to identify patterns, which it then automatically applies to the unannotated data to classify it. Consequently, the training's entry data are labeled. While the results may be on par with those of a supervised machine learning approach, less human labor is required.

4. Reinforcement learning

Reinforcement learning is the closest machine learning type to how humans learn. The algorithm or agent used learns by interacting with its environment and getting a positive or

negative reward. Common algorithms include temporal difference, deep adversarial networks, and Q-learning.

2.11. Stance Classification Using Deep Learning Algorithm

Deep Learning Algorithm

Deep learning was a machine learning and artificial intelligence technique that uses certain human brain functions to make judgments with the goal of intimidating people and their behavior. This is a crucial component of data science, channeling modeling based on data-driven methods under statistical and predictive modeling. In order for an organism to possess the capacity to learn, adapt, and behave like a human, it must be driven by powerful forces known as algorithms. Neural networks having more than two layers are referred to as deep learning networks. It has evolved from a simple neural network in several aspects, such as having more neurons and more intricate methods of interconnecting layers and automated feature extraction(Meron, 2020).

Deep learning was a subfield of machine learning that solves problems using artificial neural networks at a hierarchical level. With neuron nodes connected to one another like a web, artificial neural networks are constructed similarly to the structure of the biological brain. While most programs examine data in a linear fashion, deep learning systems' hierarchical function enables machines to interpret data in a nonlinear manner.

Deep learning algorithms are dynamically constructed to run across multiple layers of neural networks. Afterwards, each of them moves on to the next layer after passing via straightforward layered representations. On datasets containing hundreds of features or columns, machine learning is typically trained to perform fairly well.

Neural networks are just a collection of pre-trained decision-making networks that are designed to do specific tasks. The process of teaching computers to perform tasks that come effortlessly to humans is known as deep learning. Understanding how the human nervous system functions provided inspiration for the methodology and designs, which imitate human vision. Many research on the subject of posture categorization use deep neural networks, like RNNs and CNNs (Girma, 2021).

Artificial Neural Network

Artificial neural networks come in the simplest form, known as Multi-Layer Perceptrons (MLPs). It combines several different perceptron models. Inspired by the workings of the human brain, perceptrons attempt to mimic its functionality in order to solve issues.

ANNs are computer systems inspired by the biological neural networks found in human brains, where simple components operate independently of one another without the need for a centralized control unit. Within neural networks, the weights between the units aid in the storage of information.

By adjusting their weights, neural networks will pick up new data and patterns (Patterson & Gibson, 2017). ANNs often learn to do tasks without the need for task-specific rules to be written; instead, they learn by analyzing instances(Meron, 2020).

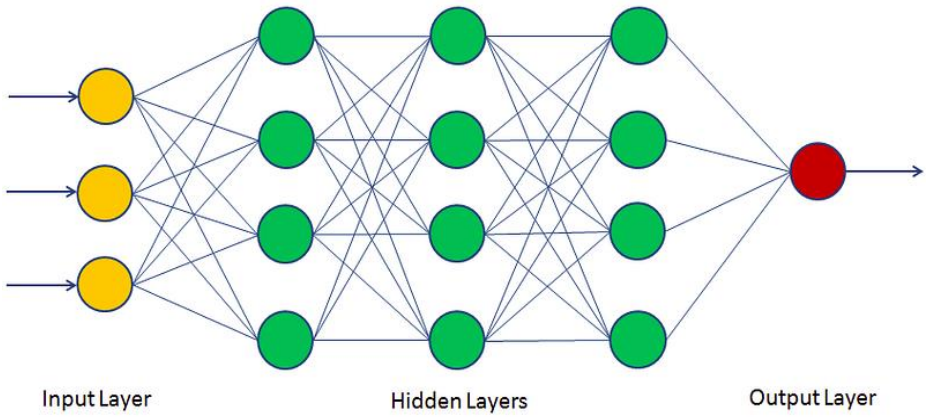

Figure 2.2.Multi-Layer Perceptron Neural Network (Avinash Navlani, 2021)

Figure above depicts a feed forward artificial neural network, commonly referred to as layered perceptron. It receives inputs from outside sources. It is made up of an input layer, output layer, and hidden layers. Additionally, there is output that leaves the output layer. The signal transmission for the hidden layers is the input layer's function. After doing computational analysis, the output layer will receive the result from the hidden layers.

An artificial neural network (ANN) is made up of neurons, which are computer units that are linked to one another in an intricate communication network. This allows the brain to perform extremely complicated computations.

The network architecture of neural networks influences their behavior. The number of neurons, the number of layers, and the kinds of connections between layers can all be used to characterize a network's architecture.

Feed-forward Artificial Neural Network

An artificial neural network that is feed forward is one kind in which a cycle is not produced by connections between nodes. The input layer was the first layer and the output layer was the last layer in their layer-by-layer arrangement. We can use in the middle concealed layers that support the learning model's enhanced performance. There is no communication between the outside world and these concealed layers. An artificial neural network that is composed of

hidden layers is referred to as a multilayer perceptron, deep feed forward neural network, or just deep neural network. Many learning strategies are used by multilayer perceptrons, with the back propagation method being the most often used. Gradient descent is used in back propagation to minimize error on the network's output by adjusting the connection weights(Meron, 2020).

It was the most widely recognized and simple to comprehend (Patterson & Gibson, 2017). Every layer was entirely connected to every other layer, and each layer may include a varying number of neurons. The performance of the prediction will be determined by the correct values of the weight sand biases. The method of fine-tuning the weights and biases from the input data is known as training the Neural Network. Each iteration of the training process will have the following steps:

- Calculating the predicted output ŷ, known as feed forward
- Updating the weights and biases, known as back propagation.

Neural networks are utilized in speech recognition, facial identification, marketing, healthcare, and other fields. Artificial Neural Networks (ANNs) aim to solve any given (data-driven) problem by imitating the way the human brain operates. A neural network is made up of several layers of perceptrons. The various layers of Perceptrons process the input data you feed into the neural network in order to produce the desired output.

Architecture of Neural Network

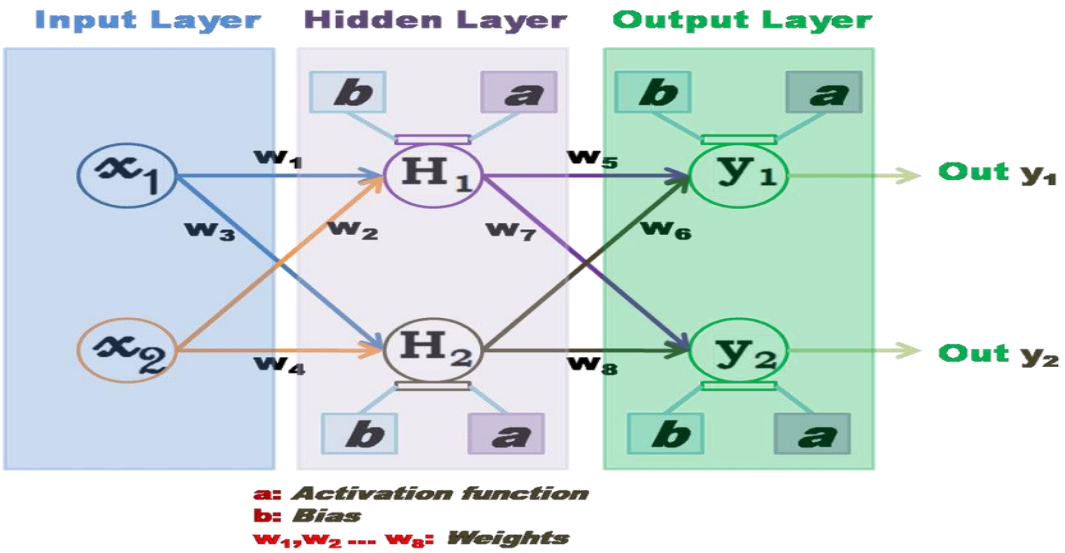

Figure 2.3.Architecture of Neural Network

A neural network consists of three layers:

1. Input Layer: In this layer, input data needs to feed. Input of input layer goes to the hidden layer
2. Hidden Layer: Locate between input and output layer. The input of hidden layer is output of input layer. In the real-world example, there can be multiple hidden layers. To explain neural network, I am using one hidden layer in this article
3. Output Layer: Output of hidden layer goes to output layer. This layer generates predicted output of Neural Network. In the above picture and for this article I am considering two class Neural Network (Out y_1, Out y_2)

Neural Network Formation

Before listing down all equations of a simple neural network, let me clear you that, an artificial neural network equation consist of three things:

1. Linear function
2. Bias
3. Activation function

Output of any layer is the combination of bias and activation function with a linear function.

For example

Input of H_1 (or Output of x_1) = $x_1w_1 + x_2w_2 + b_1$

Here

$\mathbf{x_1w_1 + x_2w_2}$ is the linear function

$\mathbf{b_1}$ is the bias (constant)

Activation function is required to calculate output of any layer.

Steps to train Neural Network

There are three steps to train a Neural Network

- Forward Propagation
- Error Calculation
- Back Propagation

A Recurrent Neural Network (RNN)

A recurrent neural network (RNN) was able to process a sequence of arbitrary length by recursively applying a transition function to its internal hidden state vector of the input sequence. According to P. Liu et al. (2016), a recurrent neural network (RNN) may process a sequence of any length by iteratively applying a transition function to the input sequence's internal hidden state vector. Several gates are employed in RNNs as history handlers to manage the preceding sequential data for the subsequent sequences.

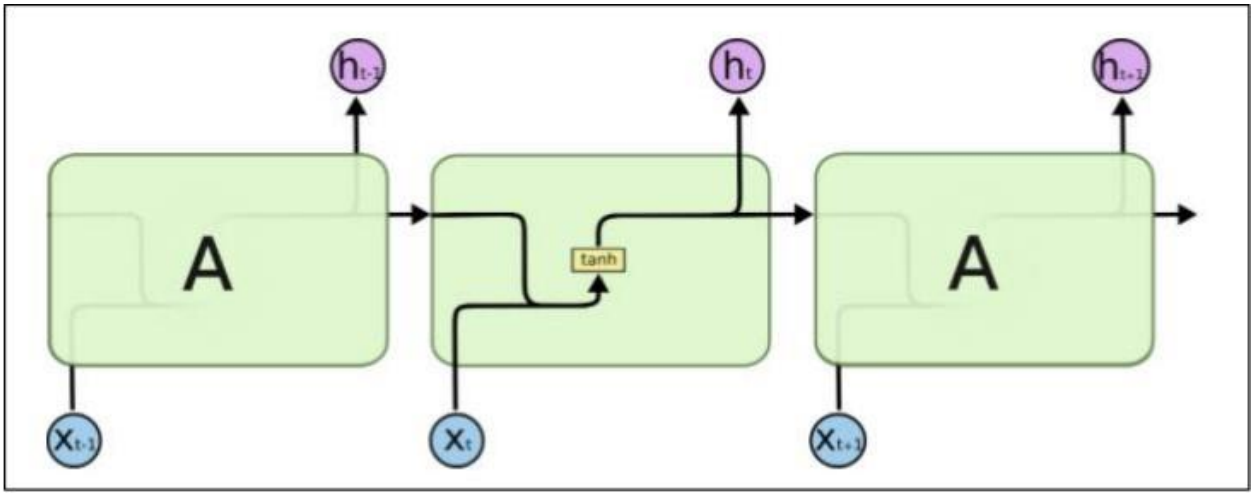

Figure 2.4.Recurrent neural network (Source: (Girma, 2021)**)**

Recurrent neural networks are a particular kind of neural network in which the input from one stage was passed into the next. All of the inputs and outputs of standard neural networks are independent of one another, but in certain situations, such as when it's necessary to forecast the next batch of data from a given dataset, the prior data is needed, and it will be necessary to retain the prior data. Thus, RNN was created, and it used a hidden layer to tackle this problem. Hidden state, which retains some information about a sequence, is the fundamental and most significant component of RNN (Medsker & Jain, 2021). Natural language processing, speech synthesis, time-series prediction, and other applications benefit from the usage of RNN.

This RNN has drawback of exploding and vanishing gradient descent.

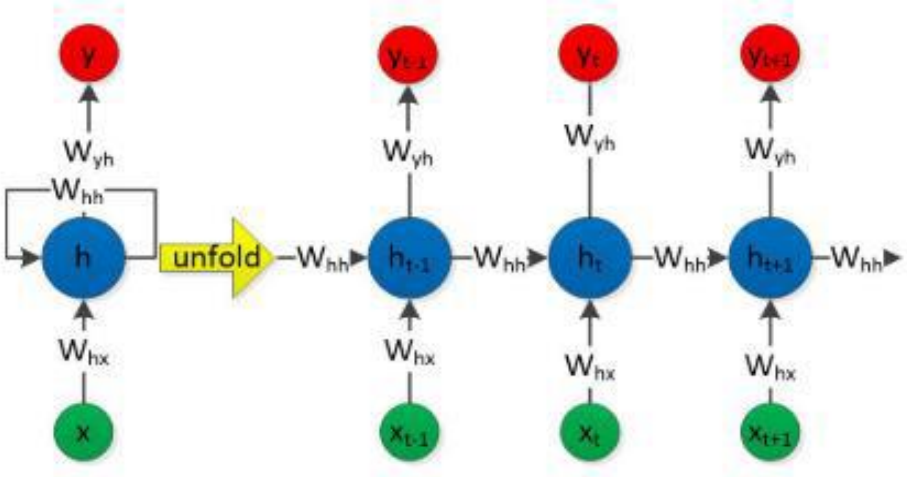

Figure 2.5.Recurrent neural network(Meron, 2020) shows the architecture of RNN.

Where:

x: is input

h: hidden state

y: output

w: weight

The RNN cell takes two different inputs; the first one is the outputs of the last hidden stat and the second one is observation at time t in the hidden state. Then the RNN will predict the next output at a time t+1.

RNN is a class of artificial neural networks where connections between nodes form a directed graph along with a temporal sequence. This allows it to exhibit temporal dynamic behavior.

RNN takes input from x (input variable) and from the previous hidden state. This behavior helps to remember the previous state and to predict the next outcome (temporal prediction) even if it suffers from exploding and vanishing gradient descent.

Recurrent Neural Networks (RNNs)

RNNs are designed to recognize patterns in data sequences, such as time series or natural language. They maintain a hidden state that captures information about previous inputs.

How it Works

- Hidden State: At each time step, the hidden state is updated based on the current input and the previous hidden state. This allows the network to maintain a memory of past inputs.
- Output: The hidden state generates an output at each time step. The network is trained using back propagation through time (BPTT) to minimize prediction error.

Long Short-Term Memory

LSTMs are a special kind of RNN capable of learning long-term dependencies. They are designed to avoid the long-term dependency problem, making them more effective for tasks like speech recognition and time series prediction.

Learning long-term dependencies was a unique capability of LSTM, a type of RNN. Reaction links are present. The full data sequence can also be processed by it. LSTM has been used to address the shortcomings of RNN. Long memory allows LSTM to recall prior processes and orders. As faults are propagated across time and layers, it assists in preventing them (Brownlee, 2017). Back propagation through time was the method most commonly used to train RNN. Due to the parameters capturing short-term dependencies as the information from older time steps decays, this gives rise to the problem of the disappearing gradients. Exploding gradients operate in reversal of diminishing gradient descent, causing the error to increase significantly with each time step. By use the gates to selectively remember pertinent

knowledge and delete irrelevant information. The issue of vanishing gradient descent is solved with LSTM.

Recurrent neural network (RNN) architecture known as Long Short-Term Memory (LSTM) was created specifically to describe temporal sequences and their long-range relationships more accurately than traditional RNNs (Lee et al. 2019). Unlike RNNs, LSTMs are specifically made to avoid the long-term reliance issue. Recalling information for extended periods of time is essentially the default behavior for these networks. Figure 3 depicts the LSTM's architecture.

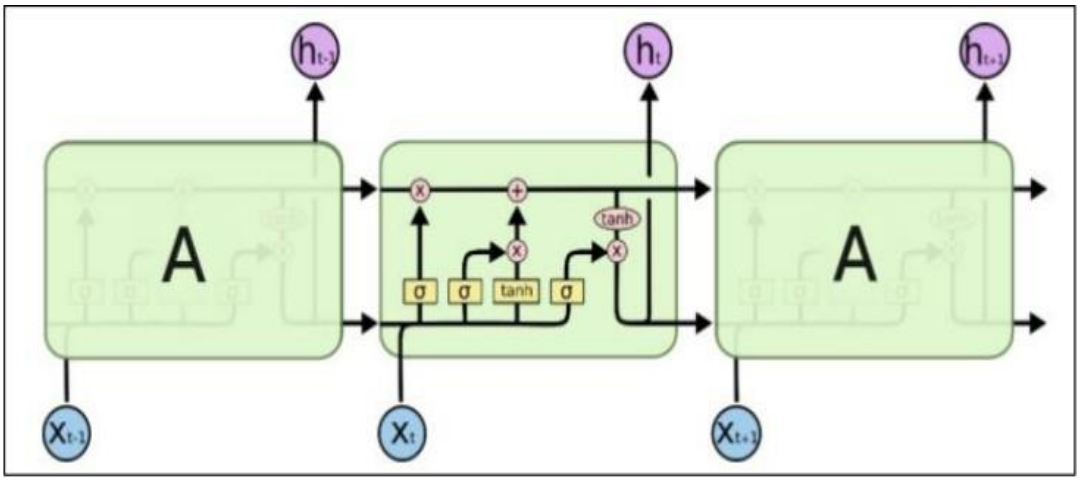

Figure 2.6.Long Short-Term Memory (Girma, 2021)

LSTM networks outperform straightforward RNNs in the analysis of sequential data due to their reduced sensitivity to temporal gaps(Meron, 2020).

The Mechanism how it Works

- Cell State: LSTMs have a cell state that runs through the entire sequence and can carry information across many steps.
- Gates: Three gates (input, forget, and output) control the flow of information:
- Input Gate: Determines which information from the current input should be updated in the cell state.
- Forget Gate: Decides what information should be discarded from the cell state.
- Output Gate: Controls the information that should be outputted based on the cell state.

In a gated cell, LSTMs store data independently of the RNN's regular flow.

Similar to data in a computer's memory, information can be stored in, read from, and written to cells. Through gates that open and close, the cell makes decisions about what to store, when to allow reads and writes, and removals. However, unlike computer digital storage, which was completely in the range of 0 to 1, these gates are analog and were applied with element-wise duplication by sigmoid. Predicting, classifying, learning, and processing

sequential data are all areas in which LSTMs thrive. It can be used for things like stock market prediction, weather forecasting, video analysis, and caption creation (Greff et al., 2017).

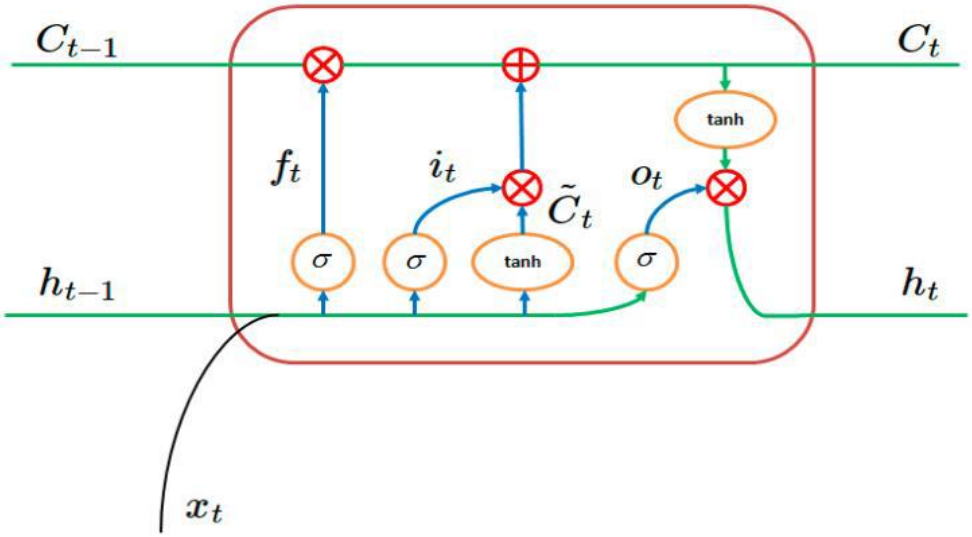

Figure 2.7.LSTM-RNN shows the architecture of LSTM-RNN(Meron, 2020).

Where;

Xt: input at time t

ht: hidden state at time t

ct: cell state at time t

ft: forget gate at time t

It: input gate at time t

ot: output gate at time t

LSTM cells function as neurons in certain of the layers of an LSTM-RNN, a particular kind of RNN. While various machine learning models have typically attempted to achieve this, LSTM cells help the neural network learn about temporal features in data, in a similar manner to how Convolutional Layers help a neural network learn about picture features.

A full matrix is the input for LSTM Neural Networks. There are typically two inputs for an LSTM-RNN. They are the output of the preceding LSTM cell (which provides it with some insight into the preceding input) and the input column of the new matrix(Meron, 2020).

The amount that a new value enters the cell is managed by the input gate's weights and biases. The amount of time a value stays in the cell is determined by the weights and biases of the forget gate. According to Brownlee (2017), the weights and biases of the output gate regulate how much of the value in the cell is used to calculate the output activation of the LSTM block.

Long-term memory is the common word for cell state. The looping arrows in an LSTM cell, which enable the storage of previously recorded information, signify the recursive nature of the cell (Hussein and others, 2018).

$$ct = ft * ct-1 + it * \hat{c}t$$

The forget gate modifies the cell state and adjust by the input modulation gate. It realizes that there might be change in the context after encounter its first loop.

$$ft = (wf\ [ht-1] + bt)$$

The input gates determine which information should enter to the cell state. The input gate has a range of [0, 1] and is a sigmoid function. The sigmoid function will only add memory and not to be able to remove/forget memory.

$$it = (wi\ [ht-1] + bi)$$

$$\hat{c}t = tanh\ (wc\ [ht-1] + bc)$$

Output gate that is usually called focus vector highlights which information should go to the next hidden state.

$ot = (wo\ [ht-1] + bo)$

$ht = ot * tanh\ (ct)$ (2.7)

Bidirectional Long Short-Term Memory (Bi-LSTM)

Bi-LSTM was proposed to overcome the limitations of a regular RNN. Bi-LSTMs are recurrent neural networks that can be trained with all of the input data that has been and will be accessible for a given time period. The Bi-LSTM method extracts context from both ends of a sequence by merging the results of two RNNs that process data in opposite directions (Girma, 2021). Target-specific stance detection was achieved by combining a two-stage deep attention neural network with Bi-LSTM in tweet stance detection. The model uses classic bidirectional LSTM to encode target tokens and densely connected BI-LSTM to encode tweet tokens. The model's ability to process contextual information that supported the stance identification results was aided by the attention mechanism determined by comparing the target and tweet tokens' semantic similarity.

In the work of Zhu et al. (2018), Bi-LSTM was employed in conjunction with CNN for text sentiment classification. Based on this research, the model built with CNN and Bi-LSTM algorithms considers the correlation between various textual aspects. According to their research, Bi-LSTM categorizes text emotions based on the semantics of the text sequence, whereas CNN can extract the latent semantic information of text by convolving the text of embedding.

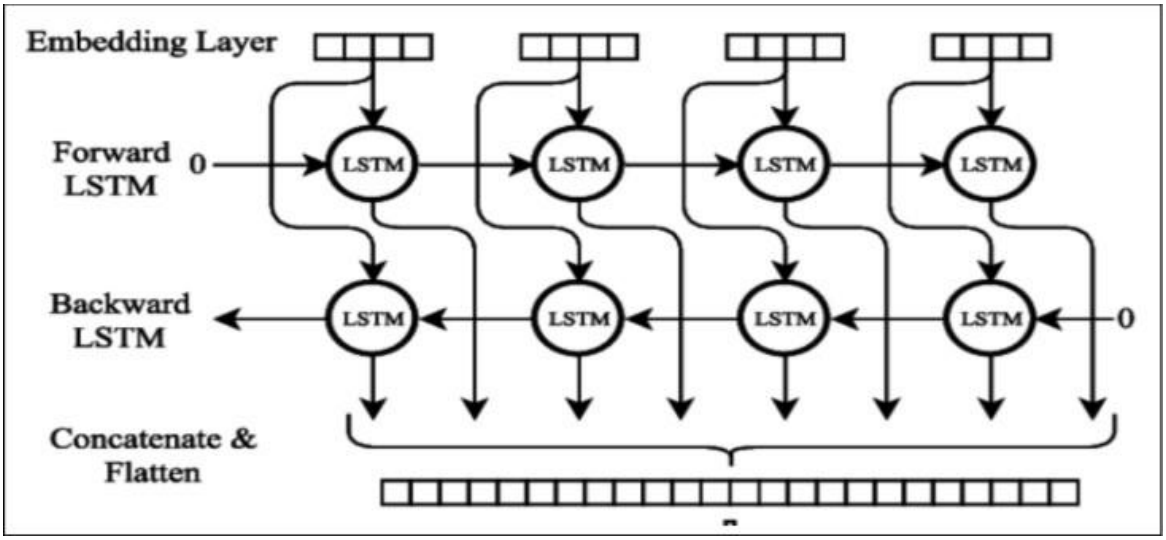

Figure 2.8.An illustration of the Bi-LSTM architecture (Girma, 2021)

Convolutional Neural Network (CNN)

CNNs are a type of deep neural network that are typically used for tasks related to image processing. It collects fundamental properties from objects, including edges that run horizontally or diagonally, using a unique method known as convolution.

CNN is a multilayer network in which one layer's output serves as the layer's subsequent input. According to Rahanoui et al. (2019), it consists of an input, multiple hidden layers, and an output. Each CNN convolution will fire when a specific pattern is observed, such as word n-grams in the case of natural language processing. By altering the kernels' sizes and concatenating their outputs, patterns of several sizes adjacent words are found. It was found that CNN performed exceptionally well when it came to sequential data analysis(Girma, 2021).For NLP tasks CNN uses a neural-based approach which represents a feature function that is applied to constituting words or n-grams to extract higher-level features.

In the study by Wei et al. (2016), CNN was suggested as a way to identify viewpoint in tweets. According to this author, distinct filter matrices that are discriminatively sensitive to various patterns can be used to extract patterns. In this case, a sentence's particular words in order form the pattern. A CNN was trained for sentence-level classification tasks using pre-trained word vectors, thanks to Kim's (2014) research. Based on this research, the CNN models outperform the current best practices for tasks involving question classification and sentiment analysis.

The mechanism how it Works:-Convolutional Layer: This layer applies a set of filters (kernels) to the input image, where each filter slides (convolves) across the image to produce a feature map. This helps detect various features such as edges, textures, and patterns.

- Pooling Layer: This layer reduces the dimensionality of the feature maps while retaining the most essential information. Common types include max pooling and average pooling.
- Fully Connected Layer: After several convolutional and pooling layers, the output is flattened and fed into one or more fully connected (dense) layers, culminating in the output layer that makes the final classification or prediction.

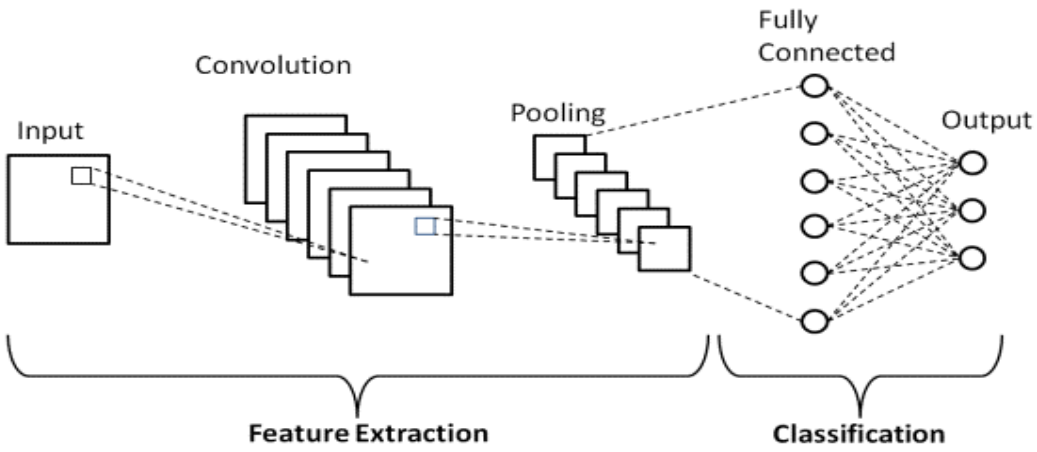

Figure 2.9.Convolutional Neural Network

Gated recurrent unit (GRU)

A specific kind of recurrent neural network called a Gated Recurrent Unit (GRU) employs gating techniques to regulate the flow of data into and out of the network. The GRU has two gates that control how much of the prior hidden state should be forgotten and how much of the current input should be used to update the hidden state. These gates are known as the reset gate and the update gate. The updated hidden state is used to calculate the GRU's output.

The final model is becoming more and more popular because it is simpler to understand than previous LSTM iterations. The GRU cell simply has two gates: a reset gate and an update gate. One gating signal and its associated parameters might therefore be saved. The single GRU cell loses some of its effectiveness compared to the original LSTM since one gate is not full. According to Yu et al. (2019), the GRU is not comprehensible for counting or for solving context-free language problems. It is also not suitable for translation. On the other hand, information is modulated within a Gated Recurrent Unit, like the LSTM, eliminating the need for an additional memory cell (Rana, 2016).

Challenges in Deep Learning

Deep learning has made significant advancements in various fields, but there are still some challenges that need to be addressed. Here are some of the main challenges in deep learning:

Data availability: It requires large amounts of data to learn from. For using deep learning it's a big concern to gather as much data for training.

Computational Resources: For training the deep learning model, it is computationally expensive because it requires specialized hardware like GPUs and TPUs.

Time-consuming: While working on sequential data depending on the computational resource it can take very large even in days or months.

I**nterpretability:** Deep learning models are complex; it works like a black box and it is very difficult to interpret the result.

Over-fitting: when the model is trained again and again, it becomes too specialized for the training data, leading to over fitting and poor performance on new data.

Advantages of Deep Learning:

1. **High accuracy:** Deep Learning algorithms can achieve state-of-the-art performance in various tasks, such as image recognition and natural language processing.
2. **Automated feature engineering:** Deep Learning algorithms can automatically discover and learn relevant features from data without the need for manual feature engineering.
3. **Scalability:** Deep Learning models can scale to handle large and complex datasets, and can learn from massive amounts of data.
4. **Flexibility:** Deep Learning models can be applied to a wide range of tasks and can handle various types of data, such as images, text, and speech.
5. **Continual improvement:** Deep Learning models can continually improve their performance as more data becomes available.

Disadvantages of Deep Learning:

1. **High computational requirements:** Deep Learning AI models require large amounts of data and computational resources to train and optimize.
2. **Requires large amounts of labeled data**: Deep Learning models often require a large amount of labeled data for training, which can be expensive and time-consuming to acquire.
3. **Interpretability:** Deep Learning models can be challenging to interpret, making it difficult to understand how they make decisions.

4. **Over-fitting:** Deep Learning models can sometimes over fit to the training data, resulting in poor performance on new and unseen data.
5. **Black-box nature**: Deep Learning models are often treated as black boxes, making it difficult to understand how they work and how they arrived at their predictions.

Activation Functions

In neural networks, the activation function produces output decision boundaries and was used to improve performance of the model. The activation function is a mathematical expression that decides whether the input should pass through a neuron or not based on its significance. It also provides non-linearity to networks. Without activation function, the neural network becomes a simple linear regression model (Awan, 2023).

There are several types of activation functions:

- Tanh
- ReLU
- Sigmoid
- Linear
- Softmax

Loss Function

The loss function is the difference between actual and predicted values. It allows neural networks to track the model's overall performance. Depending on specific problems, we chose a certain type of function, for example, mean squared error (Awan, 2023).

Loss = Sum (Predicted - Actual) ^{2}

The most used loss functions in deep learning are:

- Binary cross-entropy
- Categorical hinge
- Mean squared error
- Sparse categorical cross-entropy

Forwarding propagation and Back propagation

In forwarding propagation, we initialize our neural network with random inputs to produce an output that is random too. To make our model perform better, we adjust weights randomly using back propagation. To track the model's performance, we need a loss function that will find global minima to maximize the model's accuracy.

Stochastic Gradient Descent

Gradient descent is used to optimize loss function by changing weights in a controlled way to achieve minimum loss. Now we have an objective, but we need direction on whether to

increase or decrease the weights to achieve better performance (Awan, 2023). The derivative of the loss function will give us direction and we can use it to update the weights of the network. In stochastic gradient descent, samples are divided into batches instead of using the entire dataset to optimize gradient descent. This is useful if you want to achieve minimum loss faster and optimize computational power.

Hyper-parameter

Hyper-parameters are the tunable parameters adjusted before running the training process. These parameters directly affect model performance and help you achieve faster global minima (Awan, 2023).

List of most used hyper-parameters:

- **Learning rate:** step size of each iteration and can be set from 0.1 to 0.001. In short, it determines the speed at which the model learns.
- **Batch size:** number of samples passed through a neural network at a time.
- **Number of epochs:** an iteration of how many times the model changes weights. Too many epochs can cause models to over fit and too few can cause models to under fit, so we have to choose a medium number.

2.12. Feature Extraction Methods

A feature is a numerical representation of a given data set because most machine learning algorithms and deep learning algorithms demand a vector representation of a text as input. As a result, we must convert the text into a numerical representation since both machine learning models and the deep learning model can only interpret numerical data. Features are significant aspects of the data that can be utilized in analysis, and this characterization is one of the largest outputs. Additionally, by generating a meaningful future, features teach us valuable information about the stored dataset.

2.12.1. Bag-of-Words Model

A text is represented as a bag of its words in the bag-of-words (BOW) model, which is a simplified representation. Term frequency is the quantity of terms that occur in the text is the name given to this kind of feature. Notwithstanding its simplicity and clarity, the BOW approach is not without flaws. The model is unable to capture the semantic or syntactic relationship between two words, much like when it ignores word order and spatial information (grammar). Document-term-matrix is what is produced.

2.12.2. N-Gram Model

An n-gram model, which is a probabilistic language model, uses a (n-1)-order Markov model to predict the subsequent item in a sequence. These days, n-gram models are widely used in computational linguistics, probability, and communication theory. The N-Gram model can contain spatial information in the text and is an alternative to the BOW model. N-gram models typically perform better than BOW models since they store word co-occurrence information.

2.12.3. TF-IDF

Term frequency was the quantity of terms that are present in the text; term frequency-inverse document frequency (TF-IDF) is another statistically based representation of text. The TF-IDF model incorporates Inverse Document Frequency, which quantifies how much information a word delivers. The BOW and N-gram models only take term frequency into account. TF-IDF lessens the impact of stop words or frequently used words by combining term frequency with inverse document frequency.

2.12.4. Word2vec

In 2013, Tomas Mikolov and other Google researchers developed the word embedding technique known as Word2vec to address challenging natural language processing problems. It was capable of continually reading through lengthy texts to look for word relationships or dependencies.

To find word similarities, word2vec employs the cosine similarity measure. The cosine angle of one signifies that there is word overlap. The words are independent of one another and have no contextual relationship if the cosine angle is 90. It associates words with vector representations that are similar.

Chapter Three

Related Works

Automatic stance detection has emerged as one of the most attractive areas of research in Natural Language Processing due to the rapid rise of information transmission methods. In addition, the fields of information identification and natural language processing have been quite interested in it lately. While some research was conducted in multiple languages, the majority of research was conducted in English. The Spanish language has also made progress for certain other languages spoken throughout the world, such as Catalan. The majority of them use network-based stance detection, whereas some languages use hybrid (linguistic and network) stance detection. To the best of the researcher's knowledge, no more researches have been done in this area for the majority of local languages, such as Afaan Oromo and Amharic. The work that has been done in the field of automatic stance identification was shown here in rough form, with the most pertinent work being briefly summarized. In this section, related work on stance identification in local languages and foreign languages from a global perspective is described.

3.1. Stance Detection on Foreign Languages

The release of a seminal study on automatic stance detection was many years ago. The practical necessities for automatic stance detection has grown significantly over the years, which is why there have been several urgent papers written on the subject in various languages. Internationally published Stance Detections articles written by various academics are arranged below based on the language category, order of publication, and method they employed.

3.1.1. Stance detection for English Language

In the field of natural language processing, stance recognition as a subtask of sentiment analysis and opinion mining is quite recent. Supervised machine learning has always been the primary method for stance categorization. A fresh dataset comprised of labels for both stance and sentiment was also generated by the challenge organizers (Thomas, Pang, & Lee, 2006). The results for Spanish and English SVM classifiers (TF-IDF and Fast Text features) are similar, with an F-score of 69%. As it was during the cross-validation phase, Flair models' results have dramatically decreased.

3.2. Stance Detection on Local Languages

3.2.1. Stance Detection for Amharic Language

According to Girma Bewketu Research (2021), in the study they use the approach of multi-task learning to build Amharic stance classification model. Their model jointly learns sentiment classification, target identification and stance classification tasks at the same time. They collect Amharic corpus from social media by employing web-scraping. To prepare the dataset, they filter the collected corpus using keywords to extract comments or tweets that more describe four selected targets or topics (Abiy Ahmed, Green Legacy, Tigray War, and New Currency) of the study. For data annotation they build an android stance annotator application that has cloud-based data storage. Using the application, they annotate the filtered dataset with five annotators. For text representation, they build a Word2Vec embedding model using the collected corpus. To build the model they use a combination of the CNN and Bi-LSTM algorithm.

The performance of the multi-task model is better than that of the single task deep learning models, which require many separate models for every task. For tasks 1 (target identification), 2 (stance classification), and 3 (sentiment classification), the model's F1-score is 86%, 65%, and 50%, respectively.

3.2.2. Stance Detection for Afaan Oromo Language

Worku Bacha Research (2022) states that the study's automation of the Afaan Oromo text stance detection model was presented, and that feature extraction and supervised machine learning techniques were used to develop the detection model. The necessary information was gathered from the Facebook public page only, and based on text body ID, it was manually categorized as agree, disagree, discuss, and unrelated to the headline. The dataset comprises 1692 total Afaan Oromo news texts from both datasets, of which 862 were total stances and 830 were total bodies that were prepared. Using 1692 of the total datasets, the experiment was carried out. For classification, three supervised machine learning algorithms LR, RF, and MNB have been used. At 81%, 69%, and 48% accuracy, on average, was the researcher's achievement. Through outperforming all other classifiers and achieving the greatest outcomes in terms of improved accuracy (81%), the experimental results demonstrate that the LR outperformed the other models. Thus, to sum up, the optimal technique for stance identification and classification for Afaan Oromo text was found using Logistic Regression (LR), which provides researchers and other users with a useful tool

Chapter Four

Research Methodology and System Design

4.1. Introduction

This chapter covered system architecture, research methods, dataset construction, building the model, evaluating and predicting the result and strategies for achieving study goals and providing a solution to the research question. The study's methodology is described and supported in the section that follows in this chapter.

4.2. The proposed Afaan Oromo Stance Detection Architecture

This section presents a deep learning technique-based solution for Afaan Oromo text stance detection that can effectively address the topic under investigation. The primary component of the suggested method to identify stance on social media is derived from these studies creation of stance detection datasets for news text.

The suggested architecture, which is based on the architecture depicted in figure below, is utilized to identify news material posted by Afaan Oromo. It uses the Afaan Oromo datasets as input and preprocesses them according to the language's requirements, eliminating punctuation, normalizing, tokenizing, and other fundamental preprocessing steps. Following complete preprocessing, word embedding is used for feature extraction. An essential feature vector of the dataset used to train and test the model is the task's output.

4.3. System Design Architecture for Stance Detection

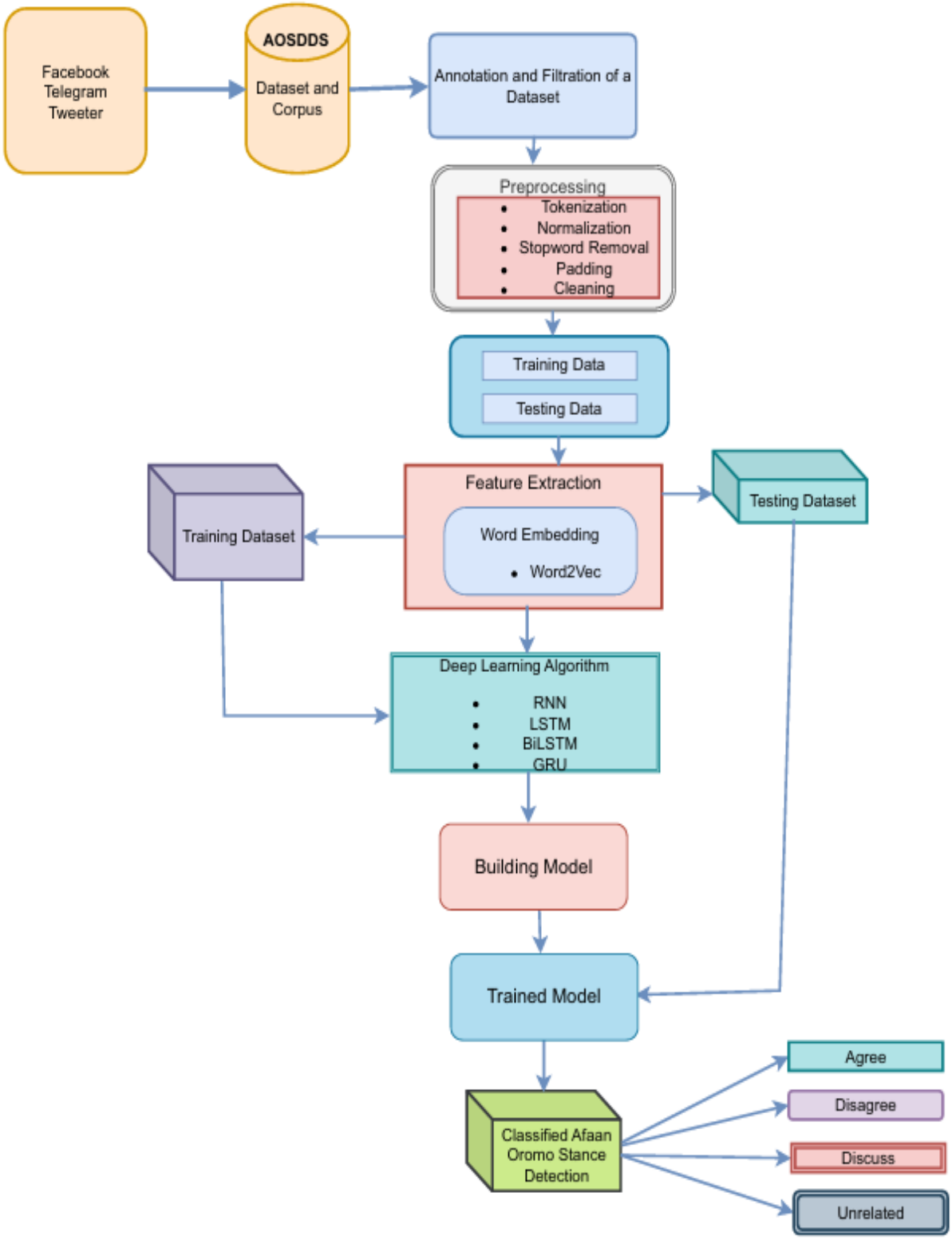

Figure 4.1.System Design Architecture for Stance Detection

4.4. Data Collection and Corpus Preparation

The data used collected from different social media platform. Several social media networks, including Facebook, Telegram, and Twitter, are the sources of the data used in our study. The study compiles textual news posts in Afaan Oromo text that are made on social media. Afaan Oromo Voice of America (VOA), Afaan Oromo Fana Broadcasting Corporation (FBC), Afaan Oromo British Broadcasting Corporation (BBC), Oromia Media Network (OMN), Oromia Broadcasting Network (OBN), and other social media account sites were the sources of news datasets. There is a dataset available that has a headline and some text in it. The system's output would be the body of text's position in relation to the headline. We used two .csv datasets that we have used to implement the system. Those are:

Train stance data set: Contains Body ID, Headline, Domain and Stance which has number of records 10022. Test stance data set: Contains Body ID, Headline, Domain and Stance which has number of records 2506.A total of 12528 news datasets as Train stance data set and Test stance data set was used; out of these 10022 were Train stance data set and 2506 were Test stance data set.

4.4.1. Annotation Category

Based on the description of stance detection, the instruction was developed by interviewing experts, going over past research annotation rules, and analyzing the psychology of stance. The annotator uses these guidelines to categorize each post texts headline as follows: "agree," "disagree," "discuss," and "unrelated" based on body or of texts.

4.4.2. Dataset Preparation

Following the collection of the data, the files or data tables representing train stances and train bodies are created by gathering, cleaning, filtering, and combining the data. MS Excel and other preparation programs were employed in this process. The next step involves annotating the post-news text in the dataset and creating a training model for Afaan Oromo text posture recognition. This is where filtering and cleaning the data comes in. An overview of the architecture of the suggested Afaan Oromo text Stance Detection is given in the ensuing sections.

4.4.3. Data Consolidation

Data was produced in a variety of formats and from a wide range of sources. Consolidation of data is the act of gathering all of the data from many sources, eliminating duplicates, and fixing mistakes before storing it in a single area, such as a data warehouse or dataset. In

general, data consolidation is the act of gathering all of your data from many sources across your company, organizing it, and consolidating it into one place, like a cloud data warehouse.

4.4.4. Dataset Annotation

A fundamental step for researchers who are creating classification models was an annotation of data. The lack of a labelled or annotated corpus was one of the factors contributing to the poor resource status of the Afaan Oromo text. The majority of earlier studies on material written in Afaan Oromo and other local language required manual annotations. People are required to manually annotate an Afaan Oromo text corpus using word processing and spreadsheet programs like Microsoft Excel. Thus, the process of adding information to a document or set of data at any level is called annotation. This text holds both the training and testing data of our model. In this instance, labeling a post-news text was necessary throughout the annotation step in order to construct a stance detection dataset.

Annotated Dataset is a collection of text collected from Facebook, Telegram and Twitter. This text holds both the training and testing data of our model. Unlike other languages for Afaan Oromo, there are no any standardized annotated publicly available corpora like English or other languages for stance detection. Targeting at classifying the stance level across Facebook, Telegram and Twitter for Afaan Oromo language users, with the help of linguistic and legal professionals, we have prepared a corpus of text collected from Facebook, Telegram and Twitter of Afaan Oromo social media, Individual Politicians, Activist, Radio Broadcast and etc. These pages typically post discussions passing across a variety of domain like Health, Agriculture, Sport, Business, Religious, Technology, Education topics. The content of the comments from Facebook posts, Telegram and Twitter were using Face-pager and Comment extractor to save them to a CSV/Excel file. Facebook, Telegram and Twitter platforms are selected to collect data for the subsequent reasons. Comparative studies have revealed how in countries with inadequate Internet service, like Ethiopia, specifically Facebook has become nearly a synonym for the Internet, a platform through which user's access information, services, and take part in online communications. The entire number of collected dataset after removing duplicates and irrelevant tweets is 12528 texts. The collected data were annotated by two annotators one from legal profession and the other is from Afaan Oromo profession. Based on the overall perceived meaning of the tweet (or Facebook post and comments) they annotated into agree, disagree, discuss and unrelated. The total number of agree tweets is 3660, the number disagree tweets is 1748, while the number discuss tweets is 3589 and unrelated tweet is 3525.

Table 4.1.Summary of classes and Dataset Source

Stance	Domain								
	Health	Agriculture	Sport	Business	Religious	Political	Technology	Education	Total
Agree	684	130	234	864	898	266	442	148	3666
Disagree	334	81	141	230	334	321	223	84	1748
Discuss	417	127	250	728	773	456	632	206	3589
Unrelated	487	130	210	660	926	474	507	131	3525
Total	1922	468	835	2482	2931	1517	1804	569	12528

Texts written by users on social network websites like Facebook and Telegram contain lots of noise that can significantly affect the results of the stance detection procedure. In social network, occasionally users use informal language while articulating their opinion. Typically, natural language like English uses automatic techniques to correct spelling and grammar, translate from one language to another language. Though, for Afaan Oromo such platforms and resource do not exist online even if developed for the needs of fulfillment for the Degree. So that, we considered this problems into account and addressed several concerns such as; correct spelling errors and grammar of some text manually, correct short form or contraction and replication of characters, because some of them may useful in the tweets label, and a word written in other language mixed with Afaan Oromo was converted into Afaan Oromo by linguistic professionals. Afaan Oromo is a lowland east Cushitic language which has tens of millions of native speakers in Ethiopia and in neighboring countries such as Kenya and Somalia. Afaan Oromo is dialectic language. A dialect is a variety of Afaan Oromo which is associated with a particular region and or social class. To state the comprehensible, speakers from different geographical regions speak Afaan Oromo somewhat differently. While we are collecting dataset from social media, we considered the different variety of Afaan Oromo dialects. So, we have collected our dataset from different classes of Afaan Oromo dialects. And then, to train the deep learning in demand to handle the complexities of Afaan Oromo dialects, which makes the stance detection system to a certain extent challenging.

Table 4.2.Afaan Oromo text Stance Dataset

Stance Category	No of each category	In percentage	Description
Agree	3666	29	The headline agrees the assertion in the news article.
Discuss	3589	29	Headline discusses the same topic as the news article but is not exactly agree
Disagree	1748	14	The headline disputes the news article's assertion.
Unrelated	3525	28	A news article's topic is not covered by the headline.
Total	12528	100	

4.5. Dataset preprocessing

Preprocessing was the first step once we have collected and annotated the dataset. After the dataset has been gathered and annotated, the initial stage is preprocessing. Manual labeling is done on annotated datasets. The input text is formatted in this preparation step so that it is ready for additional analysis. The dataset's cleaning, normalization, tokenization, stop word removal, and encoding processes are all part of the preparatory phases of the architecture.

4.5.1. Data Cleaning

Stance detection was preceded by text preparation, which entails cleaning the retrieved data. It entails locating and removing non-textual content from the textual dataset as well as any information that might raise privacy concerns, such as the source URL, the commenter's name, location, comment date, and the number of likes. It also involves removing numbers, whitespace, punctuation, and stop words the most common stop words in Afaan Ormo documents being fi (And), gara (to), ani (I), keessa (in), isa (he), ishee (she), kankoo (mine), isaan (they), and nuti (we). Also completed throughout the cleaning procedure is the list of tasks that follow.

4.5.2. Tokenization

Tokenization was the process of splitting input text into units called tokens, each of which can be a word or anything else like a number or a punctuation mark. Tokenization is applied to the input text based on the Afaan Oromo language characteristics used to retrieve the documents. Tokenization is a module that, given the bounds of a written text, divides the text into a set of tokens, typically words. By using tokenization, input texts provided by users are converted into a series of tokens. The smallest unit that will be taken out of the input sentence prior to carrying out a stance detection task is called a token.

Among the Cushitic families, the Afaan Oromo employ Latin script for textual purposes. In written documents, they use a whitespace character to divide words from one another (Gezehagn, 2012).

Tokenization Algorithm

Input *Words (sequence of characters)*

Output *Words*

Process

I. Read each sentence from list

II. Then split them in to words

III. Return list of words

4.5.3. Normalization

Normalization was a process of converting a list of letters or characters in words to a more uniform sequence of letters of words to improve text matching. We have normalized several characters in this study into lowercase because they are occasionally represented in the corpus and user input as either uppercase or lowercase. The most popular kind of normalization is case folding, which lowercases every word. Normalization involves process of handling problem related with variation of cases (Uppercase or lower case or Mixed Cases). Normalization Algorithm:-

Start

Input: *Text document corpus (F) and list of corresponding*
Normalizers (N) homophones (n) characters
Output: *Normalized text document*
split words from file (F) in character level
Find homophones (n) from F
If n is in F
Replace it by the corresponding normalizer (N)
Else
navigate through all words in F
Save F in new file as new file
Stop

4.5.4. Stop word Removal

We eliminated the stop words from our document corpus after tokenizing and normalizing it because they had no bearing on word meaning. Not every term in the document has the same meaning to the texts in which it appears. For example, the conjunctions: - fi, akkasumas, kana malees, achiis, yeroo booda, dursa, and nouns like kana and sana. The text file "stoplist.txt" contains the Afaan Oromo stop word list. After reading the files, the algorithm stores the information in a variable.

Afaan Oromo stop word list is saved in text file stoplist.txt. The algorithm reads the files and saves it on variable. Then tokenize and normalize the terms which are available in the files and check the terms whether different from the terms of stop words.

Dictionary lookup was employed for this study also. For the purpose of this research work, list of around 500 stop words that is compiled from Afaan Oromo books during implementation of a stemmer by Debela Tesfaye (Debela, 2010) was used.

Stop word removal Algorithms

Start

Input: Text document corpus (F) and list of stop words(s)
Output: Text documents free from stop words
Find stopword(s) from file (F)
If S is in F
Remove S from the file F
else
Navigate through all words in F
Save F in new file as new file
Stop

Padding

All input documents for the deep learning network must be homogeneous matrices with the same size and shape. But not every text document in the pre-process has the same length. Put another way, it makes sense that some of the texts have a higher or lower word count than others. Therefore, in order to make the text size uniform, some of the parts of the less frequent document are removed and a zero is placed at the end of the shorter document matric to make the documents padded into the same size.

Start

Input: Tokenized and numerically represented corpus

Output: Uniformly padded tokenized document

Count number of tokens (t) from each file

Find maximum and minimum tokens

Determine the pad size n

Pad the tokens (t) to defined size n

If t<n

Add n-t zero pads to t

Else

Remove t-n tokens from t

Stop

4.6. Feature Extraction and Model Building

Feature extraction was the process of increasing information density by retrieving and extracting key properties from a large volume element of data. It is the technique of extracting important properties from a bigger data element in order to increase information density is called feature extraction. Numerous variables that demand a lot of processing power to process are typical of these huge datasets. The process of choosing and/or combining variables into features, which efficiently reduces the quantity of data that needs to be processed while properly and fully characterizing the original dataset, is known as feature extraction.

The feature extraction procedure comes in handy when we need to cut down on processing resources without sacrificing pertinent or crucial data. It might also lessen the quantity of data that is unnecessary for a certain analysis. Additionally, the machine learning process's learning and generalization stages are accelerated by the reduction of data and the machine's efforts in creating variable combinations, or features(Kabada, 2020).

Both Bidirectional Long Short-Term Memory (Bi-LSTM) and Long Short-Term Memory (LSTM) use an embedding layer between the input and the Bi-LSTM and LSTM layers to build vector representations for incoming words. Similarly, for word embedding, Feed Forward Neural Network has an own embedding layer. The embedding layer weights can be initialized with random values or, more frequently, using third-part word embedding, such as word2vec, GloVe, and fast text. It is optional to fine-tune these weights during training. Nevertheless, the embedding layer of the neural network performs the initialization of the weight in this study with a random value.

Word Embedding

Word Embedding was a language modeling technique used for mapping words to vectors of real numbers. It represents words or phrases in vector space with several dimensions. Word embedding can be generated using various methods like neural networks, co-occurrence matrix, probabilistic models, etc. Word embedding is a type of word representation that allows words with similar meaning to have a similar representation. They are a distributed representation for text that is perhaps one of the key breakthroughs for the impressive performance of deep learning methods on challenging natural language processing problems.

Word Embedding Algorithms

Word embedding methods learn a real-valued vector representation for a predefined fixed sized vocabulary from a corpus of text. The learning process is either joint with the neural network model on some task, such as document classification, or is an unsupervised process, using document statistics. This section reviews three techniques that can be used to learn a word embedding from text data.

Embedding Layer

An embedding layer, for lack of a better name, is a word embedding that is learned jointly with a neural network model on a specific natural language processing task, such as language modeling or document classification. It requires that document text be cleaned and prepared such that each word is one-hot encoded. The size of the vector space is specified as part of the model, such as 50, 100, or 300 dimensions. The vectors are initialized with small random numbers. The embedding layer is used on the front end of a neural network and is fit in a supervised way using the Back propagation algorithm.

The one-hot encoded words are mapped to the word vectors. If a multilayer Perceptron model is used, then the word vectors are concatenated before being fed as input to the model. If a recurrent neural network is used, then each word may be taken as one input in a sequence.

This approach of learning and embedding layer requires a lot of training data and can be slow, but will learn and embedding both targeted to the specific text data and the NLP task.

Word2Vec

Word2Vec was a statistical method for efficiently learning a standalone word embedding from a text corpus. It was developed by Tomas Mikolov, et al. at Google in 2013 as a response to make the neural-network-based training of the embedding more efficient and since then has become the de facto standard for developing pre-trained word embedding.

Additionally, the work involved analysis of the learned vectors and the exploration of vector math on the representations of words.

Word2Vec was a popular technique in natural language processing (NLP) for learning word embedding, which are dense numerical representations of words in a continuous vector space and can be used in python. These embedding capture semantic relationships between words, making them useful for various NLP tasks like machine translation, text classification, and sentiment analysis.Word2Vec embedding have proven to be very useful due to their ability to capture semantic similarities and relationships between words. Similar words often end up with similar vector representations, and algebraic operations on these vectors can produce meaningful results.Word2Vecconsists of models for generating word embedding. These models are shallow two layer neural networks having one input layer, one hidden layer and one output layer. Word2Vecutilizes two architectures:

Two different learning models were introduced that can be used as part of the word2vec approach to learn the word embedding; they are:

- Continuous Bag-of-Words or CBOW model.
- Continuous Skip-Gram Model.

CBOW (Continuous Bag of Words): CBOW model predicts the current word given context words within specific window. The input layer contains the context words and the output layer contains the current word. The hidden layer contains the number of dimensions in which we want to represent current word present at the output layer.

The CBOW model learns the embedding by predicting the current word based on its context. The continuous skip-gram model learns by predicting the surrounding words given a current word.

Skip Gram: Skip gram predicts the surrounding context words within specific window given current word. The input layer contains the current word and the output layer contains the context words. The hidden layer contains the number of dimensions in which we want to represent current word present at the input layer.

The continuous skip-gram model learns by predicting the surrounding words given a current word. The continuous skip-gram model learns by predicting the surrounding words given a current word.

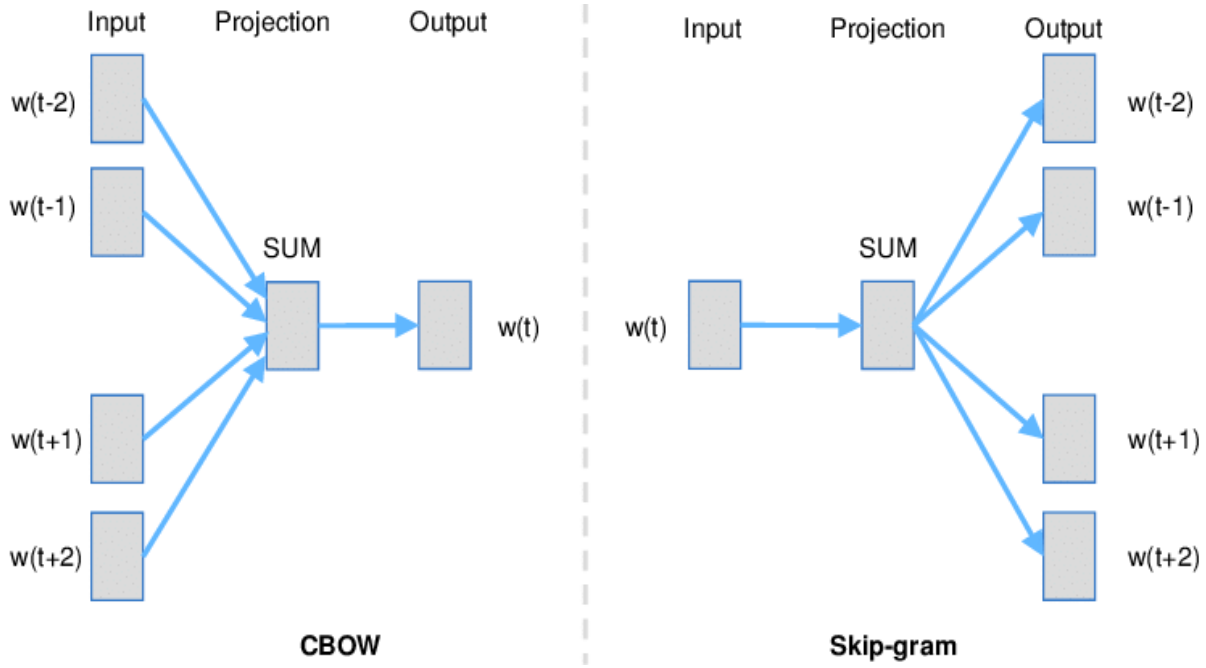

Figure 4.2.CBOW and SKIP GRAM (Daniel Braun, 2017)

Both models are focused on learning about words given their local usage context, where the context was defined by a window of neighboring words. This window was a configurable parameter of the model. The key benefit of the approach is that high-quality word embedding can be learned efficiently (low space and time complexity), allowing larger embedding to be learned (more dimensions) from much larger corpora of text (billions of words).

GloVe

The word2vec technique, which was created by Stanford's Pennington et al. to effectively train word vectors, is extended by the Global Vectors for Word Representation, or GloVe, algorithm.

Word representations in classical vector space were created by applying matrix factorization techniques like Latent Semantic Analysis (LSA), which perform well when applying global text statistics. However, they are not as effective as learned techniques like word2vec when it comes to capturing meaning and proving it on tasks like calculating analogies (see the King and Queen example above, for example).

GloVe was a method that combines local context-based learning in word2vec with the global statistics of matrix factorization techniques like LSA.

GloVe which stands for Global Vectors for Word Representation was a popular word embedding technique that captures semantic relationships between words in a vector space. It's designed to address some limitations of traditional methods like Word2Vec. GloVe

produces word embedding by analyzing the global co-occurrence statistics of words in a large corpus. We can get word embedding from glove using python for any sentence.

Splitting Training and Testing Dataset

The performance of the deep learning method was affected by the division of the training and testing datasets. In order to generalize unseen datasets, artificial neural networks must learn more from training datasets. A small training set is one of the causes of the artificial neural network's inability to generalize and over fit. Thus, you may determine whether or not your model is generalized by testing it on a large dataset. However, the majority of the time, the amount of dataset you have determines how much is divided between training and testing(Kabada, 2020).

Data splitting was the act of partitioning Annotated dataset into two parts, usually for cross-validator purposes. In this research, we split our dataset into training and testing data. Training portion of the dataset is used to train a predictive model. And the other to evaluate the stance detections model performance. We used sky learn python library to split our dataset into training and testing data.

Prediction

The aim of this study was to construct and model an Afaan Oromo stance classification model that can recognize the target or issue and the stance of the Afaan Oromo tweet or comment regarding that topic. During the prediction stage, we put the model to the test. Using the trained model, we projected the target and stance of any new tweets or comments during this phase. The model was loaded when Afaan Oromo's tweet or comment has been preprocessed and sent into it. The model initially indicates the intended audience or subject of the provided tweet or remark. The model presents the stance for the specified comment or tweet next to the target. Lastly, the model shows the tweet or comment's stance class.

4.7. Evaluation metrics

Evaluation Metrics in Deep Learning

Evaluation metrics are crucial in assessing the performance of deep learning model. They provide quantitative measures that guide the selection of models and the tuning of hyper-parameters. Different tasks require different metrics, and understanding which metric to use is the key to interpreting model results effectively. The metrics used are namely accuracy, precision, recall, and f1-score.

True Positives (TP) is the cases when the actual class of the data point was true and the predicted is also true.

True Negatives (TN): are instances where the predicted class was incorrect and the actual class of the data point was incorrect.

False Positives (FP): are instances where the projected class is accurate but the actual class of the data item is false.

False Negatives (FN): are the cases where the data point's real class was true but the anticipated value was false.

Confusion Matrix: A confusion matrix is a table that is used to describe the performance of a classification model on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm.

In classification tasks, where the output is a discrete label, common evaluation metrics include:

Accuracy: Accuracy is the simplest evaluation metric for classification. It is the ratio of correctly predicted observations to the total observations and provides a quick measure of how often the model is correct.

The accuracy of our algorithm can be calculated as:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. It is also known as the positive predictive value.

$$precison = \frac{TP}{TP + FP}$$

Recall or sensitivity:-measures the ratio of correctly predicted positive observations to all actual positives. These metrics are particularly useful when dealing with imbalanced datasets.

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F1 Score:-The F1 Score is the harmonic mean of precision and recall. It is a balance between the two metrics and is particularly useful when you need to take both false positives and false negatives into account.

$$\text{F} - \text{measure} = \frac{2 * \text{precision} * \text{recall}}{\text{Precision} + \text{recall}}$$

ROC Curve: The Receiver Operating Characteristic (ROC) curve is a graphical plot that illustrates the diagnostic ability of a binary classifier as its discrimination threshold is varied.

AUC: The Area under the Curve (AUC) represents the measure of the ability of the classifier to distinguish between the classes and is used as a summary of the ROC curve.

Regression Metrics

For regression tasks where the model predicts continuous values, common metrics include:

Mean Absolute Error (MAE): MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation.

Mean Squared Error (MSE): MSE measures the average of the squares of the errors that is, the average squared difference between the estimated values and the actual value.

RMSE: is the square root of the mean of the squared errors. It has the useful property of being in the same units as the response variable and can be more appropriate than MSE when errors are particularly undesirable. It is also common to use multiple metrics to get a more holistic view of the model's performance. For example, in a classification task, one might look at both the accuracy and the F1 score to understand both the overall correctness and the balance between precision and recall.

4.8. Development Tools

Python: Python is a high-level, object-oriented, interpreted programming language for general purposes. Python is a robust text processing language and is easy to obtain NLP packages for. It offers a wide range of modules and functions thanks to its sizable standard library.

Anaconda Navigator: Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them.

Spyder: - Spyder is a Python development environment with many features for working with Python code, such as a text editor, debugger, profiler, and interactive console.

Jupyter Notebook: Jupyter Notebook is a web-based application that allows us to create and share documents that contain live code, equations, visualizations, and narrative text.

Python Libraries for Deep Learning

1. Tensor-Flow: Tensor-Flow is widely considered one of the best Python libraries for deep learning applications. Developed by the Google Brain Team, it provides a wide range of

flexible tools, libraries, and community resources. Beginners and professionals alike can use Tensor-Flow to construct deep learning models, as well as neural networks.

2. Keras:- Kears is yet another notable open-source Python library used for deep learning tasks, allowing for rapid deep neural network testing. Keras provides you with the tools needed to construct models, visualize graphs, and analyze datasets. On top of that, it also includes pre-labeled datasets that can be directly imported and loaded.

3. NumPy:- One of the other well-known Python libraries, NumPy can be seamlessly utilized for large multi-dimensional array and matrix processing. It relies on a large set of high-level mathematical functions, which makes it especially useful for efficient fundamental scientific computations in deep learning.

4. Scikit; - Learn: Scikit Learn were originally a third-party extension to the SciPy library, but it is now a standalone Python library on Github. Scikit-Learn include DBSCAN, gradient boosting, support vector machines, and random forests within the classification, regression, and clustering methods.

5. SciPy: - Scipy is a free and open-source library based on Numpy. SciPy is one of the best Python libraries out there thanks to its ability to perform scientific and technical computing on large datasets. It is accompanied by embedded modules for array optimization and linear algebra.

6. Pandas: - One of the open-source Python libraries mainly used in data science and deep learning subjects is Pandas. The library provides data manipulation and analysis tools, which are used for analyzing data. The library relies on its powerful data structures for manipulating numerical tables and time series analysis.

7. Microsoft CNTK:-Another Python library for deep learning applications is Microsoft CNTK (Cognitive Toolkit), which is formerly known as Computational Network ToolKit. The open-source deep-learning library is used to implement distributed deep learning and machine learning tasks.

8. Matplotlib: Matplotlib is a Python computer language data visualization and graphical plotting package used for 2D visuals. It can be utilized with different graphical user interface toolkits, web application servers, shell scripts, and Python scripts. Plotting several states while the model is being trained is done with it.

9. Genism: Gensim is an open source Python library for natural language processing, with a focus on topic modeling. It is also efficient multicore implementations for various algorithms to increase processing speed.

Chapter Five

Experimentation and Implementation

5.1. Introduction

This chapter explains implementation, experimentation and evaluation methods of the entire works and results are described in detail. In this Section we discuss the overall implementation of our system, Stance Detection for Afaan Oromo language.

Next also we would describe the data collection and preparation, data annotation, dataset preparation, data cleaning and preprocessing, data analysis, feature extraction and language modeling, experimental setup, system prototyping tools and programming language, hardware and software used, word embedding, model building and implementation, experimentation, evaluation processes and results with detail discussion.

5.2. Dataset Collection and Preparation

To conduct the experimentation and obtain better results and a well-performing model for the categorization of Afaan Oromo texts stance detection, a substantial amount of data was needed. By its very nature, the deep learning approach works best with large datasets and needs a lot of data. Texts from Afaan Oromo gathered from several sources such as facebook, tweeter and telegram which are written by Afaan Oromo. To create the corpus variety, documents are gathered from various news articles and other internet resources, such as media from Oromia Broadcasting Network (OBN), Voice of America (VOA), Fana Broadcasting Corporate (FBC), BBC Afaan Oromo News, Bariisaa Afaan Oromo newspapers, and others. The data are tagged or labeled with the appropriate class once they have been gathered. In order to determine if the data are labeled to the correct class or not, the experts review the labeled papers. Experts are people who have studied the language, can communicate well in it, and whose mother tongue is Afaan Oromo.

We gathered Afaan Oromo text Stance dataset which is organized into 12528 tuples from sources with a sizable following on Facebook and Telegram. Therefore, human-posted texts on various websites and media contain noise, missing values, inconsistent information, and incomplete information, all of which have an impact on model performance. These repercussions make preprocessing datasets an essential step before developing a model. Because pre-processing was required to increase dataset quality to some degree and transform the data into a format that the model can comprehend. This suggests that those who perform study in the field are among the experts.

Loading Afaan Oromo Text Stance Detection Dataset

With the growing usage of deep learning to handle real-time problems, it was imperative to reduce the time required to develop dependable deep learning models, that is, the time from algorithm design to implementation to generate the desired model. In order to accomplish this, the first and most important stage in modeling a deep learning problem is dataset loading. As explained about data format, we load Afaan Oromo Text Stance Detection Dataset CSV file format using python standard library which provides built-in modules CSV module and reader () function. Besides, we used Pandas and pandas.read_csv () function because it's a very flexible function to load data as DataFrame.

```
data = pd.read_csv('/Users/WB/Desktop/prepare dataset 4 SD/Stance_SD_dataset1.csv', encoding = 'latin1')
data.head()
```

	Body ID	Headline	Domain	Stance
0	1	Dhibamaa kaansarii kan ta'e namni ganna 46, ta...	Fayyaa	disagree
1	2	Weerara Koleraa Godiina Shawaa Bahaa Aanaa Boo...	Fayyaa	discuss
2	3	Qorannoon akka agarsiisutti amalli keenya fayy...	Fayyaa	disagree
3	4	Hariiroo sukkaaraa fi dhukkuba sukkaaraa giddu...	Fayyaa	agree
4	5	Miila keessan qaxxaamursuun oomisha isparmii i...	Fayyaa	agree

```
print("Number of tags: {}".format(len(data.Stance.unique())))
frequencies = data.Stance.value_counts()
frequencies
```

```
Number of tags: 4
Stance
agree        3666
discuss      3589
unrelated    3525
disagree     1748
Name: count, dtype: int64
```

Figure 5.1.Loaded data and Information of Afaan Oromo Text Stance Data set

Preprocessing Data

Preprocessing was the initial stage of data processing and analysis, whereby text data must be prepared for use in additional analysis and interpretation before being used for application purposes. Data with varying forms and representations are referred to as unstructured data. The act of applying any kind of computation to unstructured raw data in order to convert it into a format that can be handled more quickly and efficiently in a subsequent step is known as document pre-processing. Following the selection and collection of the documents,

document pre-processing actions are implemented to ensure that the papers are formatted appropriately for indexing and search ability.

The Afaan Oromo text corpora gathered from several domains serve as the source material for this thesis project. Various text pre-processing activities are included in Afaan Oromo Stance detection procedure before documents containing detections are retrieved from Afaan Oromo corpus. We have mostly employed text tokenization, normalization, stop word removal, and stemming as pre-processing methods.

```
# DATA PREPROCESSING
# Tokenization and Normalization
# load doc into memory
def load_doc(filename):
    # open the file as read only
    file = open(filename, 'r', encoding='latin1')
    # read all text
    text = file.read()
    # close the file
    file.close()
    return text
```

```
# load document
in_filename = "/Users/WB/Desktop/prepare dataset 4 SD/Stance_SD_dataset1.csv"
doc = load_doc(in_filename)
print(doc[:10000000])
```

```
Body ID,Headline,Domain,Stance
1,"Dhibamaa kaansarii kan ta'e namni ganna 46, tapha lootorii Powerball jedhamuun doolaara biiliyoona 1.3 injifate. ",Fayyaa,disagree
2,Weerara Koleraa Godiina Shawaa Bahaa Aanaa Boosat keessatti mudateen namoota sadii du'anii heddu dhibichaan qabamaniiru.,Fayyaa,discuss
3,Qorannoon akka agarsiisutti amalli keenya fayyaa sammuu keenyaa irratti dhiibbaa guddaa qaba.Amaloota fayyaa sammuu keenyaa miidhaniifi akkamitti ak
ka ittisuu qabnu gorsa ogeeyyii kana dubbisaa.,Fayyaa,disagree
4,Hariiroo sukkaaraa fi dhukkuba sukkaaraa gidduu jiru beekuun gaaridha jedha dhaabbanni fayyaa oromiyaa.,Fayyaa,agree
5,Miila keessan qaxxaamursuun oomisha isparmii irratti dhiibbaa uumuu akka danda'u ragaaleen ni mul'isu.Miila walirra qaxxaamursanii taa'uun miidhaa f
ayyaa biroo kam fa'aa qaba.,Fayyaa,agree
6,Sagaleen meeshaalee manaa fayyaa keessan akka miidhu quba qabduu?Sagaleen nama jeequ hundi kaaniitti homaa fakkaachuu dhiisuu danda'a. Ogeeyyiin fay
yaa garuu sagaleen nama jeequ dhukkuba onneetiif kan nama saaxilu ta'uu himu. ,Fayyaa,agree
7,Itoophiyaa keessatti of ajjeesuun akka dhimma yaaddessaa ta'eetti ilaalamu isaa dhaabbanni fayyaa gabaase.,Fayyaa,discuss
```

```python
import string
# turn a doc into clean tokens
def clean_doc(doc):
    # replace '--' with a space ' '
    doc = doc.replace('--', ' ')
    # split into tokens by white space
    tokens = doc.split()
    # remove punctuation from each token
    table = str.maketrans('', '', string.punctuation)
    tokens = [w.translate(table) for w in tokens]
    # remove remaining tokens that are not alphabetic
    tokens = [word for word in tokens if word.isalpha()]
    # make lower case
    tokens = [word.lower() for word in tokens]
    return tokens
```

```python
# clean document
tokens = clean_doc(doc)
print(tokens[:10000000])
print('Total Tokens: %d' % len(tokens))
print('Unique Tokens: %d' % len(set(tokens)))
```

```
['body', 'idheadlinedomainstance', 'kaansarii', 'kan', 'tae', 'namni', 'ganna', 'tapha', 'lootorii', 'powerball', 'jedhamuun', 'doolaara', 'biiliyoon
a', 'injifate', 'fayyaadisagree', 'koleraa', 'godiina', 'shawaa', 'bahaa', 'aanaa', 'boosat', 'keessatti', 'mudateen', 'namoota', 'sadii', 'duanii',
'heddu', 'dhibichaan', 'qabamaniirufayyaadiscuss', 'akka', 'agarsiisutti', 'amalli', 'keenya', 'fayyaa', 'sammuu', 'keenyaa', 'irratti', 'dhiibbaa',
'guddaa', 'qabaamaloota', 'fayyaa', 'sammuu', 'keenyaa', 'miidhaniifi', 'akkamitti', 'akka', 'ittisuu', 'qabnu', 'gorsa', 'ogeeyyii', 'kana', 'dubbisa
afayyaadisagree', 'sukkaaraa', 'fi', 'dhukkuba', 'sukkaaraa', 'gidduu', 'jiru', 'beekuun', 'gaaridha', 'jedha', 'dhaabbanni', 'fayyaa', 'oromiyaafayya
aagree', 'keessan', 'qaxxaamursuun', 'oomisha', 'isparmii', 'irratti', 'dhiibbaa', 'uumuu', 'akka', 'dandau', 'ragaaleen', 'ni', 'mulisumiila', 'walir
ra', 'qaxxaamursanii', 'taauun', 'miidhaa', 'fayyaa', 'biroo', 'kam', 'faaa', 'qabafayyaaagree', 'meeshaalee', 'manaa', 'fayyaa', 'keessan', 'akka',
'miidhu', 'quba', 'qabduusagaleen', 'nama', 'jeequ', 'hundi', 'kaaniitti', 'homaa', 'fakkaachuu', 'dhiisuu', 'dandaa', 'ogeeyyiin', 'fayyaa', 'garuu',
'sagaleen', 'nama', 'jeequ', 'dhukkuba', 'onneetiif', 'kan', 'nama', 'saaxilu', 'tauu', 'himu', 'fayyaaagree', 'keessatti', 'of', 'ajjeesuun', 'akka',
'dhimma', 'yaaddessaa', 'taeetti', 'ilaalamu', 'isaa', 'dhaabbanni', 'fayyaa', 'gabaasefayyaadiscuss', 'dhabuurratti', 'hojii', 'dhabuu', 'zuleeykaa',
'aadam', 'jedhamti', 'qaroo', 'dhabeettidha', 'sabbataatti', 'beelaaf', 'daaru', 'obsitee', 'akka', 'baratte', 'haasofti', 'egasiis', 'yunivarsitii',
'seentee', 'eebbifamte', 'fayyaaagree', 'xiyyaara', 'keessaa', 'nama', 'dheeraa', 'awwaluu', 'buxxuulaa', 'xiyyaaraan', 'gara', 'dubaayitti', 'imaluu
f', 'rakkadheen', 'ture', 'jedhe', 'fayyaaagree', 'koo', 'rabbi', 'si', 'haa', 'fayyisufayyaaagree', 'qulqulluu', 'fi', 'naannawaa', 'mijataa', 'uumuu
n', 'lammiileen', 'dhibeewwan', 'daddarboof', 'akka', 'hin', 'saaxilamne', 'qooda', 'qabaachuu', 'dubbataniiru', 'ministirri', 'ministeera', 'fayyaa',
'dooktar', 'maqdas', 'dhaabaan', 'ibsanfayyaaagree', 'duulli', 'talaallii', 'ittisa', 'dhibee', 'saree', 'maraattee', 'eegale', 'fayyaaagree', 'muuzii
n', 'fayyaa', 'namaaf', 'qabufinfinnee', 'fulbaana', 'fbc', 'muuziin', 'kuduraa', 'baayee', 'barbaachisu', 'keessaa', 'isa', 'tokkoofi', 'qabiyyee',
'albuuda', 'pootaasiiyeemii', 'jedhamun', 'beekamuudha', 'fayyaaagree', 'tokko', 'dhukkubsataan', 'rifeensa', 'muru', 'ispeeshaalistii', 'mormaa', 'ol
```

[45]:
```python
import re
with open('/Users/WB/Desktop/prepare dataset 4 SD/Stance_SD_dataset1.csv', encoding = 'latin1') as f, open("D:\stopwords.txt") as st:
    f_content = f.read()
    processed = re.split('[^a-zA-Z]', f_content)
    processed = [x.lower() for x in processed]
    #processed = [x.upper() for x in processed]
    processed = [(x) for x in processed]
    #print(processed)

    st_content = st.read()
    st_list = set(st_content.split())

    clean_text = [x for x in processed if x not in st_list]
    print (clean_text)
```

```
['body', 'id', 'headline', 'domain', 'stance', '', '', '', 'dhibamaa', 'kaansarii', 'ta', 'e', 'namni', 'ganna', '', '', '', '', 'tapha', 'lootorii',
'powerball', 'jedhamuun', 'doolaara', 'biiliyoona', '', '', '', '', 'injifate', '', '', '', 'fayyaa', 'disagree', '', '', 'weerara', 'koleraa', 'godii
na', 'shawaa', 'bahaa', 'aanaa', 'boosat', 'mudateen', 'namoota', 'du', 'anii', 'heddu', 'dhibichaan', 'qabamaniiru', '', 'fayyaa', 'discuss', '', '',
'qorannoon', 'agarsiisutti', 'amalli', 'fayyaa', 'sammuu', 'keenyaa', 'dhiibbaa', 'guddaa', 'qaba', 'amaloota', 'fayyaa', 'sammuu', 'keenyaa', 'miidha
niifi', 'akkamitti', 'ittisuu', 'qabnu', 'gorsa', 'ogeeyyii', 'dubbisaa', '', 'fayyaa', 'disagree', '', '', 'hariiroo', 'sukkaaraa', 'dhukkuba', 'sukk
aaraa', 'beekuun', 'gaaridha', 'jedha', 'dhaabbanni', 'fayyaa', 'oromiyaa', '', 'fayyaa', 'agree', '', '', 'miila', 'qaxxaamursuun', 'oomisha', 'ispar
mii', 'dhiibbaa', 'uumuu', 'danda', 'u', 'ragaaleen', 'ni', 'mul', 'isu', 'miila', 'qaxxaamursanii', 'taa', 'uun', 'miidhaa', 'fayyaa', 'fa', 'aa', 'q
aba', '', 'fayyaa', 'agree', '', '', 'sagaleen', 'meeshaalee', 'manaa', 'fayyaa', 'miidhu', 'quba', 'qabduu', 'sagaleen', 'nama', 'jeequ', 'hundi', 'k
aaniitti', 'homaa', 'fakkaachuu', 'dhiisuu', 'danda', 'a', '', 'ogeeyyiin', 'fayyaa', 'sagaleen', 'nama', 'jeequ', 'dhukkuba', 'onneetiif', 'nama', 's
aaxilu', 'ta', 'uu', 'himu', '', '', 'fayyaa', 'agree', '', '', 'itoophiyaa', 'ajjeesuun', 'dhimma', 'yaaddessaa', 'ta', 'eetti', 'ilaalamu', 'dhaabba
nni', 'fayyaa', 'gabaase', '', 'fayyaa', 'discuss', '', '', '', 'qaroo', 'dhabuurratti', 'hojii', 'dhabuu', '', 'zuleeykaa', 'aadam', 'jedhamti', '',
'qaroo', 'dhabeettidha', '', 'sabbataatti', 'beelaaf', 'daaru', 'obsitee', 'baratte', 'haasofti', '', 'egasiis', 'yunivarsitii', 'seentee', 'eebbifamt
```

```
# STOPWORD REMOVAL
import re
with open('/Users/WB/Desktop/prepare dataset 4 SD/Stance_SD_dataset1.csv', encoding = 'latin1') as f, open("D:\stopwords.txt") as st:
    f_content = f.read()
    #splitting text.txt by non alphanumeric characters
    processed = re.split('[^a-zA-Z]', f_content)
    st_content = st.read()
    #splitting stopwords.txt by new line
    st_list = re.split('\n', st_content)
    #print(st_list) to check it was working

    #what I'm trying to do is: traverse through the text. If stopword appears,
    #remove it. otherwise keep it.
    for word in st_list:
        f_content = f_content.replace(word, "")
        #print(f_content)
```

```
[41]: st_list
```

```
'achuma',
'adda',
'addaatti',
'afoo',
'agarsiisoo',
'al',
'ala',
'alatti',
'alla',
'akka',
'akkam',
'akkamii',
'akkamiitu',
'akkana',
'akkataa',
'akkataan',
'akkas',
'akkasitti',
```

Figure 5.2.Implementation of all Preprocessing phase

Sample of AOTSDD after Preprocessed

In Figure 5.3 we gave 5 samples of preprocessed AOTSDD to show whether or not the loaded dataset was preprocessed. Especially the dataset was normalized as shown below and we have 5 columns like Body ID, Headline, Domain, Stance, and removed_punc with corresponding values.

```
[27]: # Remove all punctuations from the text

def remove_punct(text):
    return ("".join([ch for ch in text if ch not in str.punctuation]))
```

```
[29]: data['removed_punc'] = data['Headline'].apply(lambda x: remove_punct(x))
data.head()
```

[29]:

	Body ID	Headline	Domain	Stance	removed_punc
0	1	Dhibamaa kaansarii kan ta'e namni ganna 46, ta...	Fayyaa	disagree	Dhibamaa kaansarii kan tae namni ganna 46 taph...
1	2	Weerara Koleraa Godiina Shawaa Bahaa Aanaa Boo...	Fayyaa	discuss	Weerara Koleraa Godiina Shawaa Bahaa Aanaa Boo...
2	3	Qorannoon akka agarsiisutti amalli keenya fayy...	Fayyaa	disagree	Qorannoon akka agarsiisutti amalli keenya fayy...
3	4	Hariiroo sukkaaraa fi dhukkuba sukkaaraa giddu...	Fayyaa	agree	Hariiroo sukkaaraa fi dhukkuba sukkaaraa giddu...
4	5	Miila keessan qaxxaamursuun oomisha isparmii i...	Fayyaa	agree	Miila keessan qaxxaamursuun oomisha isparmii i...

Figure 5.3.Sample of Preprocessed AOTSDDS

As we expressed in figure 5.1., we have 12528 tuples in our collected dataset. Also, Figure 5.4 indicates the distribution of claims in each stance. As shown, that all stance label has different count. Therefore, it indicates that the collected dataset is balanced.

```
[51]: data.Stance.unique()
```

```
[51]: array(['disagree', 'discuss', 'agree', 'unrelated'], dtype=object)
```

```
[53]: words = list(set(data['tokens'].values))
words.append("ENDPAD")
num_words = len(words)

tags = list(set(data["Stance"].values))
num_tags = len(tags)

import plotly.express as px
fig = px.histogram(data[~data.Stance.str.contains("O")], x="Stance",color="Stance")
fig.show()
```

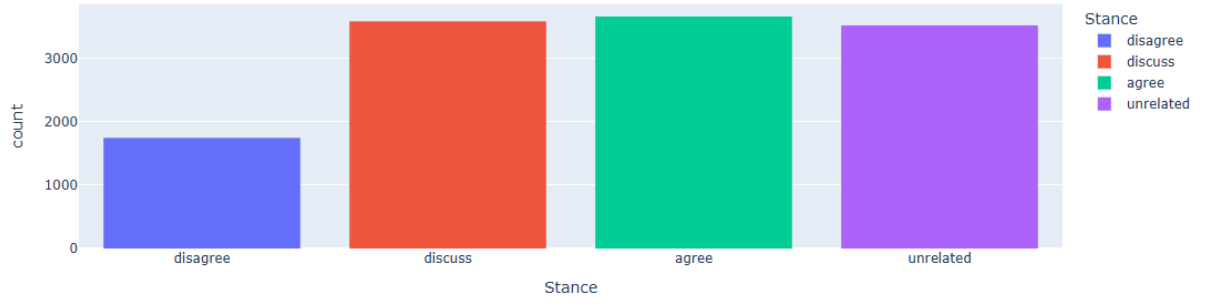

Figure 5.4.Number of Claims in each Stance

5.3. Feature Extraction

The process of transforming unprocessed text into a neural network process-able format for stance recognition through deep learning include feature extraction, which collects significant

patterns and relationships. Recurrent neural network (RNN) layers such as LSTM and Bi-LSTM are commonly used in conjunction with embedding layers in order to extract features.

5.3.1 Word Embedding

Word embedding are low-dimensional, dense representations of words that map words with similar meanings to neighboring points in vector space, allowing deep learning models to take advantage of these semantic relationships during training. Word embedding is derived from word context in a corpus of text.

In stance detection using deep learning models LSTM, Bi-LSTM and GRU word embedding play a crucial role in transforming textual data into numerical vectors that these models can process effectively. Word embedding capture semantic relationships and contextual meanings of words, enhancing the model's ability to understand and classify text based on stance (e.g., agree, disagree, discuss, unrelated).

Word2vec was a class of models that conveys related words closer to each other by representing a word in a huge text corpus as a vector in n-dimensional space (usually of several hundred dimensions, with each unique word in the corpus being allocated a matching vector in the space). Word2vec trains words against other words that are nearby them in the input corpus, which is similar to how an auto-encoder trains words. However, instead of training against the input words through reconstruction, word2vec trains words against other words. Word2vec does this in one of two ways: either by predicting a target word using context (a method called continuous bag of words, or simply CBOW), or by predicting a target context using a word which a method known as skip-gram (Oliyad, 2021).

The Word2Vec model for word embedding has since been developed. To enhance performance and achieve more precise classification outcomes, word embedding generated with word2vec are employed in text classification tasks like stance detection. The Gensim library was used to produce the pre-trained data for Afaan Oromo. Taking cues from CBOW and Skip-gram, this library is rather simple to utilize.

Model Implementation Procedures

As we have discussed above, after the data preprocessing step, the model is built by LSTM, Bi-LSTM and GRU deep learning algorithms. There is merge layer, dropout layer, activation function, loss function, optimizers and evaluation metrics which we used to implement and evaluate the model.

Training Phase: we have developed LSTM, Bi-LSTM and GRU models. The LSTM is neural network model of deep learning. It consists of a sequence of dense layer, LSTM layer,

activation, and dropout layers. The LSTM neural network will learn the pattern from the given dataset. Dropout is also applied to prevent over fitting by altering the network architecture at training time. It ensures that no single node in the networks is responsible to learn a pattern.

A **Bidirectional LSTM**, or **Bi-LSTM**, is a sequence processing model that consists of two LSTMs: one taking the input in a forward direction, and the other in a backwards direction. Bi-LSTMs effectively increase the amount of information available to the network, improving the context available to the algorithm.

Validation Phase: different techniques such as dropout at early stages and batch normalization between network layers are applied to better characterize or learn features and have higher accuracy. In this phase, our aim is to increase the performance of our model by increasing the accuracy or by decreasing the loss. Combinations of parameters (weight and bias) that provide higher accuracy are used to classify testing datasets.

Testing Phase: The learnt model has been given testing text that is distinct from training datasets in order to assess the system's adaptability to new datasets. The prediction model should be trained using training data once it has been constructed. The target attributes needs to be present in the training set. In order to regulate learning parameters, training parameters and data attributes including the target variable are also crucial. To train the model, we utilized the fit () function from the keras. Training data, training target, batch size, epochs, and validation data are all included.

Batch Size: The quantity of samples that must be processed before the internal model parameters are updated is known as the batch size. Predicting outcomes by iterating through one or more samples is what batch computing is all about. Upon completion of the batch, the errors are computed by comparing the predictions with the expected output variables.

We used 16 & 32 in number for batch size. We have tried to the models with different batch values and we got better result with batch size of 32.

Epochs: the number of epochs determines the time that the learning algorithm will work through the entire training. It’s iteration over the entire training data and training target provided. The model is trained until the epoch of index epochs is reached. In this work by number 5 and 12 epochs are used to train and validate the dataset.

Merge Layer: A layer called the Merge Layer can be found in the Keras library. Two layers are combined using it. There are various methods for combining layers. Concatenate is a handy tool for combining two layers' input into a single layer. Moreover, it follows a sequence.

Activation Function: The input layer of a neural network does not employ the activation layer since it lacks computation among the three levels. There is no communication between hidden layers and the outside world. It performs all types of calculations on features that are transferred as output through the output layer and entered through the input layer. Activation functions are typically applicable to output and hidden layers.

Activation function helps to decide, whether a neuron should be activated or not. It will be determined by calculating weighted sum and further adding bias with it. The importance of the activation function is to introduce non-linearity into the output of a neuron. Neural networks have neurons that work in correspondence of weight, bias and their respective activation function. In a neural network, we need to update the weights and biases of the neurons on the basis of the error at the output. This process is known as back-propagation. Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases.

Dropout: Dropout is the term for disregarding units or neurons; in the training process, a randomly selected group of neurons will be dropped out or ignored. Keras, which is supplied by the dropout core layer, supports this regularization technique. Preventing over fitting is the primary goal of dropout. In order to avoid over fitting and maintain model accuracy, dropout regularization of 20% is typically applied. 20% of dropouts have also been used.

Optimizer: The aim of deep learning is to reduce the difference between the predicted output and the actual output which is also called as a Cost function or Loss function. In order to minimize the cost function, we need to find the optimized value for weights by achieving much iteration with different weights, which is called gradient descent. Generally, gradient descent is an iterative machine learning optimization algorithm to reduce the cost function. Optimizers are algorithms or methods that are used to change the attributes of the neural network such as weights and learning rate in order to reduce the losses.

Adam is an acronym for Adaptive Moment Estimation and is a widely utilized optimization technique in deep learning and machine learning. Using training data from deep learning models, Adam is an alternative optimization algorithm that may be used to update network weights iteratively in place of the traditional stochastic gradient descent process. The optimization approach can manage sparse gradients on noisy data while preserving a per-parameter learning rate. It does this by combining the best features of the Root Mean squared Propagation (RMSProp) and Adaptive gradient (AdaGrad) algorithms.

Hyperparameter and Experiment Setup used for Train the Model

No	**Model**	**Epoch**	**Batch Size**	**Learning Rate**	**Dropout**	**Optimizer**	**Activation Function**
1	LSTM	5	16, 32	0.0001	0.2	adam	Softmax
		12	16, 32	0.0001	0.2	adam	softmax
2	BILSTM	5	16, 32	0.0001	0.2	adam	softmax
		12	16, 32	0.0001	0.2	adam	softmax
3	GRU	5	16, 32	0.0001	0.2	adam	softmax
		12	16, 32	0.0001	0.2	adam	softmax

Embedding Layer

We are worked to develop our model now that we know how to load the dataset. As the initial layer of the model, we will employ an embedding layer. In natural language processing applications, embedding is a commonly utilized technique that has shown promise in mapping categorical variables, such as words, to a vector of continuous integers.

More specifically, we will employ pre-trained Word2vec and GloVe word vectors that is, vectors that have already been trained to map each word to a particular size. In this model, we employ 200 embedding size. To be more precise, we will train an Embedding Layer to learn word representations and a Long Short-Term Memory (LSTM) recurrent neural network to learn word predictions that are contingent on context.

The news's headline and body are embedded independently in the embedding layer since they undergo separate preprocessing and tokenization, are represented as a vector, and are subjected to analysis by several deep learning models. We utilize the Keras Embedding layer to accomplish this by setting the parameter.

Dense Layer

A neural network layer known as the dense layer is tightly linked, meaning that every neuron in it receives input from every other neuron in the layer before it. It is discovered that the layer in the models that is employed the most frequently is the dense layer. The completely connected layer known as the dense layer uses an activation function to convert the deep representation of the input data into the final output class. The number of classes as an output vector dimension and the activation function are the two fundamental parameters of the model's dense layer. In the hidden layer, we employ the sigmoid activation function. Moreover, the SoftMax activation function produces an output with values between 0 and 1.

Our output layer transfers the result into four vector dimensions for each of the other four groups, which include agree, disagree, discuss, and unrelated. In order to detect attitude for text classification models, it is utilized as the last layer. N, the number of categories in our dataset, should be represented by the dense layer with N neurons that makes up the final layer.

5.3.2 Implementation of Proposed Model by Deep Learning Models

Deep learning model techniques must be trained when we obtain the preprocessed Afaan Oromo Text Stance detection dataset, embedded vector, encoded and padded, and the vectorized article body and headline. In order to determine which model performs best on the Afaan Oromo Text Stance detection dataset, we provided some hints regarding implementations, architecture, how to begin training and making predictions, and layers of the models.

5.4. Deep Learning Model Training

We have developed a corpus of text (posts and comments) for this study that we gathered from Afaan Oromo newspapers' public Facebook, telegram and Twitter pages. The general number of collected tweets after removing duplicates and irrelevant tweets are 12528. The tweets were categorized into four categories: agree, disagree, discuss and unrelated depending on the overall interpretation of the tweets.

The total number of agree tweets is 3666, while the number of disagree tweets is 1748 and the number of discuss tweets is 3589 and the number of unrelated tweets are 3525. After that, we have shuffled the data using the skylearn library in order to mix the dataset. After we have shuffled the data we have spitted 80% (10022) for training and 20% (2506) for testing data by using percentage split method of the skylearn python library.

5.4.1. Train-Test Splitting

To select the suitable data splitting amount, we tried splitting dataset: 80%/20% for a train/test splitting ratio. The experiment is conducted with several other parameters held constant, including the activation function, batch size, maximum iteration, and the number of hidden layers. The test set in each experiment was 80%/20% of the data splitting, while the training set was the remaining portion. Therefore, 80% of the data set is used for model training, while 20% is used for testing the learned model.

The result of the above train-test split ratio is shown in Table 5.1 below.

Classes	Number of Stance Text dataset		
	Train set	Test set	Total
Disagree	1398	350	1748
Discuss	2871	718	3589
Agree	2933	733	3666
Unrelated	2820	705	3525
Total	10022	2506	12528

Table: 5.1. Number of Stance Text dataset train-test split ratio

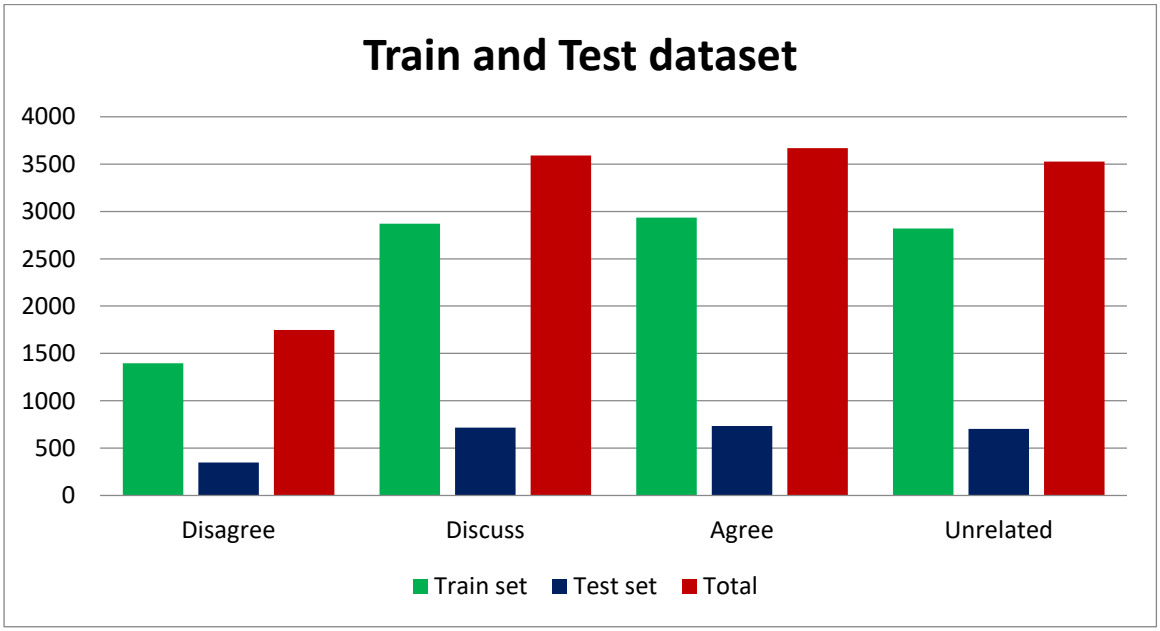

Figure: 5.5. Classification accuracy with the new system on different train-test splits ratios

5.5. Deep Learning Models

LSTM Models

A deep learning model is needed to train a model after we split the training and testing datasets, encoded and padded portions of the headline and body, and transformed the class of vectors (integers) into a binary class matrix. In order to accomplish this, we created LSTM in our experiment to identify assertions made by Afaan Oromo Headline and Body News. These statements fall into the following categories: Agree, Disagree, Discuss, and Unrelated. The most widely used RNN architecture, LSTM, is developed to maximize the capture of long-term dependencies.

LSTM uses three gates (input, output, and forget) to control the flow of data into and out of the cell and a memory cell to hold values over variable time intervals in order to solve the gradient fading or exploding issues that vanilla RNNs have.

An input layer in the embedding layer of a language model would initially learn the relationship between the headline and body. Then, in order to control the input dimension of every neuron, the extra layer inserts several elements in a ReLU layer. The outputs of an additional layer are processed to alter the size of an LSTM layer, which is made up of flattenlayer LSTM cells. The outputs of the flatten-layer are then processed and sent through a sigmoid activation function for classification or prediction, and the model's performance is assessed using the Adam and categorical cross-entropy loss.

[109]:
```
from tensorflow.keras import Model, Input
from tensorflow.keras.layers import LSTM, Embedding, Dense
from tensorflow.keras.layers import TimeDistributed, SpatialDropout1D, Bidirectional
```

[111]:
```
from tensorflow.keras import Model, Sequential, Input
from tensorflow.keras.layers import Embedding, Dense, LSTM
from tensorflow.keras.layers import Bidirectional, TimeDistributed, SpatialDropout1D
```

[113]:
```
model = Sequential()

model.add(Embedding(input_dim = input_dim, output_dim = output_dim, input_length = input_length))

model.add(SpatialDropout1D(0.2))

model.add(LSTM(units = 200, recurrent_dropout = 0.2, return_sequences = True))

model.add(TimeDistributed(Dense(units = n_tags, activation = 'softmax',kernel_regularizer=keras.regularizers.l1(0.001))))
keras.layers.Dropout(0.2)

#model.add(Dense(units=16,activation='relu'))

model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
#model.add(Dense(1, activation='sigmoid'))

#model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])

model.summary()
```

```
Training LSTM model...
Epoch 1/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 55ms/step - accuracy: 0.2611 - loss: 1.3812
Epoch 2/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 16s 51ms/step - accuracy: 0.4940 - loss: 1.2573
Epoch 3/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 16s 52ms/step - accuracy: 0.6931 - loss: 0.7866
Epoch 4/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 19s 57ms/step - accuracy: 0.7684 - loss: 0.5688
Epoch 5/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 52ms/step - accuracy: 0.7924 - loss: 0.4767
Epoch 6/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 16s 52ms/step - accuracy: 0.8120 - loss: 0.4221
Epoch 7/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 56ms/step - accuracy: 0.8183 - loss: 0.3826
Epoch 8/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 19s 52ms/step - accuracy: 0.8352 - loss: 0.3494
Epoch 9/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 20s 51ms/step - accuracy: 0.8403 - loss: 0.3280
Epoch 10/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 51ms/step - accuracy: 0.8525 - loss: 0.3011
Epoch 11/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 16s 51ms/step - accuracy: 0.8579 - loss: 0.2957
Epoch 12/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 16s 51ms/step - accuracy: 0.8600 - loss: 0.2783
```

Figure 5.6.LSTM Model building and Training

Bi-LSTM Model

Recurrent neural networks, such as the bidirectional LSTM (Bi-LSTM), are commonly used when context information about the input was required. In a unidirectional LSTM, data flows from the past to the present. A bi-directional LSTM, on the other hand, uses two hidden states to function both forward and backward in addition to forward and backward. Consequently, Bi-LSTMs are more equipped to understand the context. Bi-LSTM was used to elevate the input data slice. When evaluating words, the Bi-LSTM model can consider both their prior and subsequent context. Bidirectional long short-term memory is comprised of two layers in our model, followed by a flattening layer and a layer that is linked to a dense layer.

A number of the parameters employed in each layer of the Bi-LSTM are displayed in below.

[54]:
```
from tensorflow.keras import Model, Input
from tensorflow.keras.layers import Bidirectional, Embedding, Dense
from tensorflow.keras.layers import TimeDistributed, SpatialDropout1D, Bidirectional
```

[55]:
```
from tensorflow.keras import Model, Sequential, Input
from tensorflow.keras.layers import Embedding, Dense, Bidirectional
from tensorflow.keras.layers import Bidirectional, TimeDistributed, SpatialDropout1D
```

[58]:
```
models = Sequential()
models.add(Embedding(input_dim = input_dim, output_dim = output_dim, input_length = input_length))
models.add(SpatialDropout1D(0.2))
models.add(Bidirectional(LSTM(units = 200, recurrent_dropout = 0.2, return_sequences = True)))
models.add(TimeDistributed(Dense(units = n_tags, activation = 'softmax',kernel_regularizer=keras.regularizers.l1(0.001))))
keras.layers.Dropout(0.2)
#model.add(Dense(units=16,activation='relu'))
models.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
#model.add(Dense(1, activation='sigmoid'))
#model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
models.summary()
```

```
Training BiLSTM model...
Epoch 1/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 32s 60ms/step - accuracy: 0.2640 - loss: 1.3776
Epoch 2/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 54ms/step - accuracy: 0.4655 - loss: 1.2396
Epoch 3/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 54ms/step - accuracy: 0.7031 - loss: 0.7816
Epoch 4/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 55ms/step - accuracy: 0.7725 - loss: 0.5535
Epoch 5/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 55ms/step - accuracy: 0.8002 - loss: 0.4656
Epoch 6/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 54ms/step - accuracy: 0.8238 - loss: 0.4048
Epoch 7/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 55ms/step - accuracy: 0.8318 - loss: 0.3688
Epoch 8/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 54ms/step - accuracy: 0.8374 - loss: 0.3406
Epoch 9/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 54ms/step - accuracy: 0.8528 - loss: 0.3022
Epoch 10/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 56ms/step - accuracy: 0.8554 - loss: 0.2909
Epoch 11/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 18s 58ms/step - accuracy: 0.8535 - loss: 0.2844
Epoch 12/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 54ms/step - accuracy: 0.8663 - loss: 0.2642
```

Figure 5.7.Bi-LSTM Model building and Training

GRU Model

```
[173]: from tensorflow.keras.layers import Embedding,GRU,LSTM,Bidirectional,SimpleRNN

*[176]: model_GRU = Sequential()
model_GRU.add(Embedding(input_dim = input_dim, output_dim = output_dim, input_length = input_length))
model_GRU.add(SpatialDropout1D(0.2))
model_GRU.add(GRU(units = 200, recurrent_dropout = 0.2, return_sequences = True))
model_GRU.add(TimeDistributed(Dense(units = n_tags, activation = 'softmax',kernel_regularizer=keras.regularizers.l1(0.001))))
keras.layers.Dropout(0.2)
#model.add(Dense(units=16,activation='relu'))
model_GRU.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
#model.add(Dense(1, activation='sigmoid'))
#model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
model_GRU.summary()
```

Figure 5.8.GRU Model building and Training

5.6. Performance Evaluation and Experimental Result

In this section, we have conducted many experiments to evaluate the efficacy of the suggested paradigm. We have conducted experiments using neural network models such as LSTM, BI-LSTM and GRU. The epoch number, learning rate, embedding dimension, and train-test dataset splitting ratio parameters in this experiment remain constant during all observations.

Lastly, we assessed the experiment's performance using the f1-score performance assessment metrics, research findings of all observations, and analysis of the effect based on evaluation metrics including accuracy, precision, and recall.

We have implemented learning rate and early stopping function as a training regulator. If a measurement has stopped improving, the learning rate value is decreased using the preceding one. Callbacks have been used as a scheduling tool. After a predetermined amount of reading, the scheduler reduces the number of epochs if the learning rate does not increase. The final one is used to halt training in the event that the model's errors (validation loss) increase. Training will be hampered when the counter reaches the early stop patience value and min_delta if a steady check is performed while the current loss is not precisely the typical loss.

5.7. Afaan Oromo Text Stance Detection Using LSTM

In this experiment, we classified stance from Afaan Oromo social media text comments using an LSTM deep learning neural network model. This was done because social media is likely the most popular platform for sharing information, and stance analysis enables users to access content that more closely matches their preferences. Included in this network model are the

output layer, LSTM layer, dropout layer, embedding layer, and spatial dropout layer. Using the CBOW pre-trained word representation, the embedding layer is carried out.

5.7.1. Performance Evaluation Metrics of the Model Using LSTM

Under this experiment, the model is done by applying the above hyper-parameters and dataset distribution. The performance of the LSTM model is analyzed by using evaluation metrics, namely precision, recall, f1-score, and accuracy. Under this session, we can achieve the required precision of 86%, a recall of 89%, and an f1-score of 87%. Accuracy can be generalized as it is shown in Fig 5.9 below; we can achieve 86% of training accuracy. This demonstrates that to increase accuracy, we need to clean the data more and increase the data size, especially the training data size unless the classified accuracy of the fewer common words is prevailed upon by the majority common word class.

Classification Report for LSTM:

	precision	recall	f1-score	support
disagree	0.69	0.96	0.81	1402
discuss	0.91	0.87	0.89	2873
agree	0.93	0.86	0.89	2898
unrelated	0.92	0.86	0.89	2820
accuracy			0.87	9993
macro avg	0.86	0.89	0.87	9993
weighted avg	0.89	0.87	0.88	9993

Figure.5.9.Performance Evaluation metrics report for Stance detection using the LSTM approach

A confusion matrix is used to assess how well the proposed method performs in terms of accuracy. It consists of a row and a column, with the row representing actual categories and the column representing the estimated number of textual posts categorized into each class. The diagonal components are used to indicate the appropriate number of forecasts by the classifier and the classifier incorrectly anticipates those components outside the diagonal. In this case, we have used 80% (10022) of the datasets for training purposes and 20% (2506) dataset for test. As shown in Fig. 5.9 below, the first row indicates that out of training datasets 1390 Disagree, 1350 textual Posts are classified correctly as the category of Disagree and the remaining textual Posts are categorized into other classes wrongly. In the next rows, the total 2871 Discuss class are correct labels are 2489. Agree class (2933) the correct labels are 2486, Unrelated class (2820) the correct labels are 2414. This demonstrates that these classes are becoming more common with each other. As shown from the confusion matrix

details, 8739 textual Posts out of 10022 are correctly classified under its Stance class category and the average percent accuracy is 86%, and the remaining 14% classes are wrongly classified. Fig. 5.9 illustrates the results in diagrammatic form, including a true label with predicate labels of data distribution.

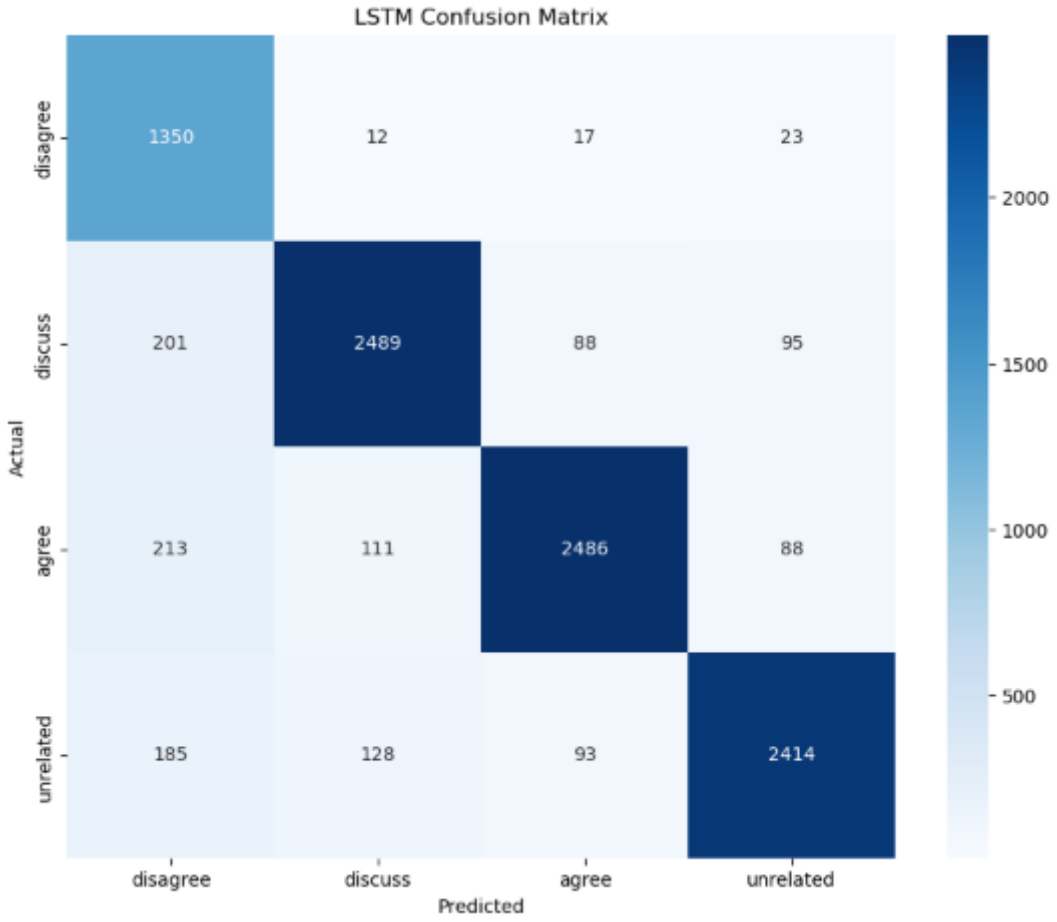

Figure: 5.10. Confusion metrics for Stance classification using the LSTM model

As demonstrated in Fig.5.11 and Fig. 5.12, we also illustrated the loss values during the model's training and validation phases. The training accuracy increases roughly and linearly, but the validation accuracy fluctuates a lot and eventually achieves a good accuracy by applying the Softmax activation function. During experimental work, we train our model utilizing Adam optimizer for various epochs, especially in contrast to the results, and revert our model on more epochs with selected parameters and the most promising validation accuracy was registered at the 12^{th} epoch using 80% and 20% of training and test sets of synthetically produced Afaan Oromo social media text comment dataset. It is the outcome of Adam overshooting in the optimum direction or overshooting of low-loss regions. The training loss drops sharply to 1.4, but the validation loss fluctuates up and down.

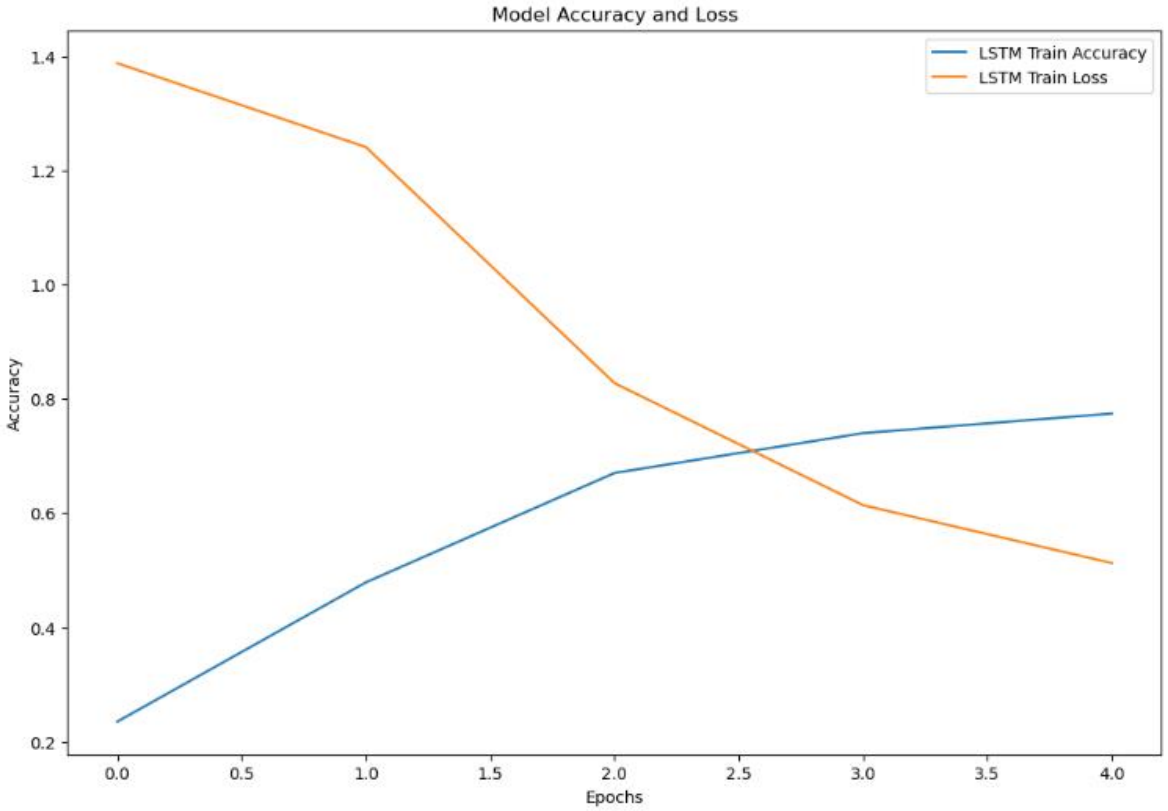

Figure: 5.11.Training Accuracy and loss using LSTM

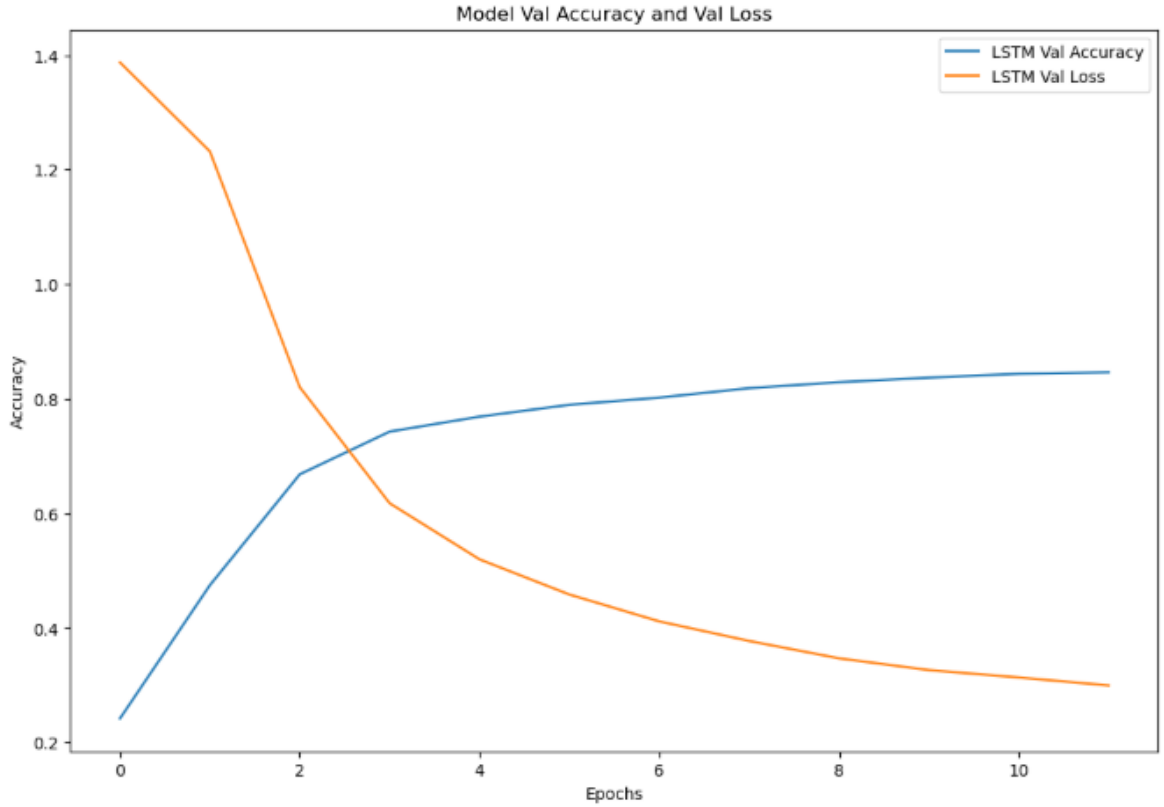

Figure: 5.12. Validation Accuracy and loss using LSTM

5.8. Afaan Oromo Text Stance Detection Using BILSTM

As explained in chapter two, Bi-LSTM is one of the simple algorithms of deep learning used to classify Offensives using text comments. The Bi-LSTM approach is both the former and resulting logical data, and the data obtained by Bi-LSTM can be viewed as two diverse text-based interpretations. The test results for this experiment used for the Bi-LSTM classifier are discussed in the following sections.

5.8.1. Performance Evaluation metrics of the Model Using BILSTM

In the same case as that of experiment one, the detailed accuracy by the class result of the Stance detection using text component using BILSTM describes the precision, recall, and F-measure for each subcategory of the Stance text classifier. The performance of the BILSTM model is analyzed by using evaluation metrics, namely precision, recall, f1-score, and accuracy. Under this session, we can achieve the required precision of 87%, a recall of 89%, and an f1-score of 87%. Accuracy can be generalized as it is shown in Fig 5.13 below; we can achieve 86.63% of training accuracy. This demonstrates that to increase accuracy, we need to clean the data more and increase the data size, especially the training data size unless the classified accuracy of the fewer common words is prevailed upon by the majority common word class.

Classification Report for BiLSTM:

	precision	recall	f1-score	support
disagree	0.72	0.97	0.83	1402
discuss	0.89	0.89	0.89	2873
agree	0.92	0.87	0.89	2898
unrelated	0.94	0.84	0.89	2820
accuracy			0.88	9993
macro avg	0.87	0.89	0.87	9993
weighted avg	0.89	0.88	0.88	9993

Figure: 5.13. Performance evaluation metrics of stance detection using BI-LSTM

The confusion matrix result describes the variations between expected and actual labels by applying the BI-LSTM model. It consists of a row and a column, with the row representing actual categories and the column representing the estimated number of textual Posts categorized into each class. The diagonal components are used to indicate the appropriate number of forecasts by the classifier and the classifier incorrectly anticipates those components outside the diagonal. In this case, we have used 80% (10022) of the datasets for training purposes. As shown in figure below, the first row indicates that out of training

datasets 1357 textual posts are classified correctly as the category of Disagree and the remaining textual posts are categorized into other classes wrongly. In the next rows, out of 2871 Discuss class the correct labels are 2549. Agree class out of (2933) the correct labels are 2511 and finally, from the total Unrelated class (2820) the correct labels are 2376.

This demonstrates that these classes are becoming more common with each other. As shown from the confusion matrix details, 8728 textual posts out of 10022 are correctly classified under its Stance detection class and the average percent accuracy is 86.63% and the remaining 13.36 % classes are wrongly classified. Figure: 5.14, depicts the test accuracy results for each class in numeric form as a means of approximating each text value to proper

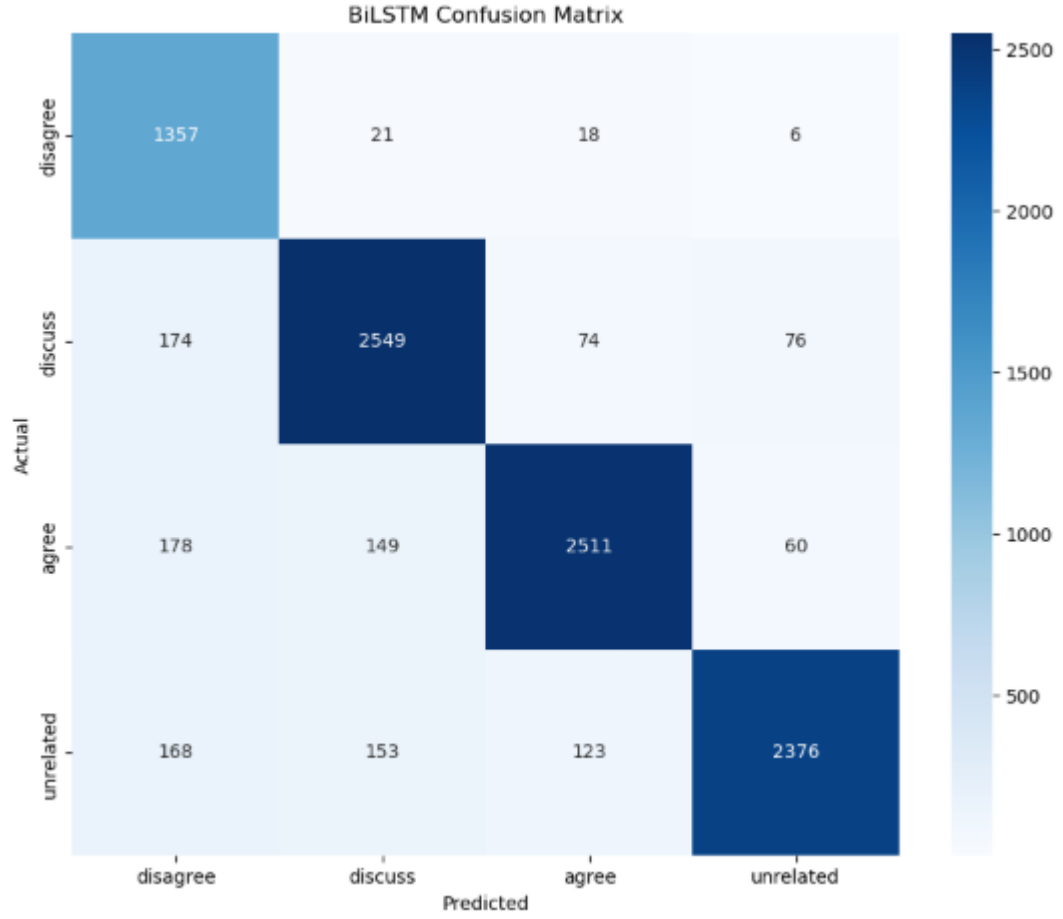

Figure: 5.14. Confusion metrics for Stance classification using the BiLSTM model

Accuracy and loss performance using BILSTM

The training accuracy and loss result is shown in Fig.5.15 and 5.16 clearly that training and validation accuracy increases while training and validation loss reduces almost continuously. There is some evidence that over fitting is common. The Adam optimization technique, Early Stopping, and dropout regularization are used to address this. Because each component is independent and can be eliminated, dropout regularization diminishes confidence in a trait

that is only valuable when combined with other distinctive features. Up to the specified epoch, however, training and validation accuracy coexist; yet, training loss significantly declines while validation loss declines more gradually. This is a result of modifications made to the layer input data preparation.

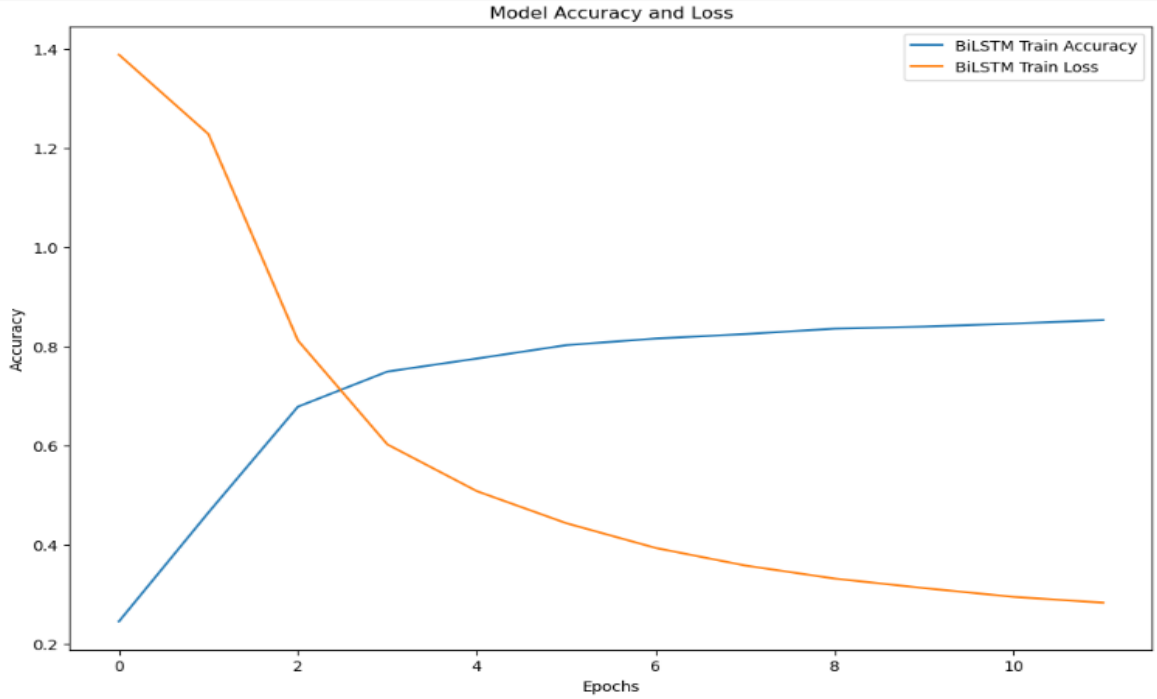

Figure: 5.15.Training Accuracy and loss using BILSTM

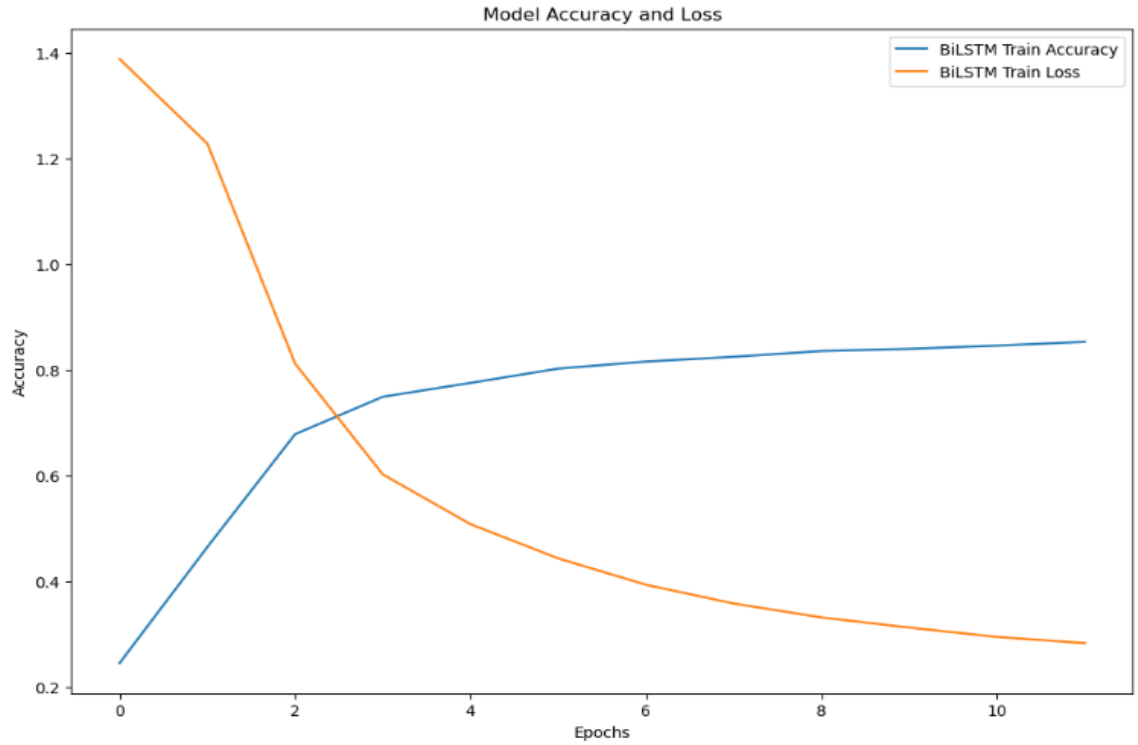

Figure: 5.16. Validation Accuracy and loss using BiLSTM

5.9. Stance Classification in Afaan oromo Text Using GRU

In this experiment, we classified Stance from Afaan oromo social media text comments using a GRU deep learning neural network model, which consists of an embedding layer, spatial dropout layer, GRU layer, dropout layer, and output layer. This is because social media is likely the most popular information-sharing website and Stance analysis enables users to access content that more accurately matches their preferences. The embedding layer uses CBOW pre-trained word representation.

5.9.1. Performance Analysis of the Model Using GRU

The aforementioned hyperparameters and dataset distribution are applied to the model in this experiment, and evaluation metrics such as accuracy, precision, recall, and f1-score are used to assess the GRU model's performance. Under this session, we can achieve the required precision of 87%, a recall of 89%, and an f1-score of 87%. Accuracy can be generalized as it is shown in Fig 5.17 below; we can achieve 86.5% of training accuracy.

This indicates that in order to improve accuracy, we must clean the data more and increase the amount of the data, particularly the training data size, until the majority common word class outperforms the classified accuracy of the fewer common words. Fig 4.7 shows the detailed performance values of each class.

Classification Report for GRU:

	precision	recall	f1-score	support
disagree	0.73	0.95	0.83	1396
discuss	0.91	0.87	0.89	2890
agree	0.92	0.86	0.89	2896
unrelated	0.90	0.87	0.89	2811
accuracy			0.88	9993
macro avg	0.87	0.89	0.87	9993
weighted avg	0.89	0.88	0.88	9993

Figure: 5.17.Performance Evaluation Report for Stance Classification Using the GRU Approach

A confusion matrix, which is composed of a row and a column, is used to evaluate the accuracy of the suggested method. The diagonal components indicate the proper number of forecasts made by the classifier, and the classifier incorrectly anticipates those components outside the diagonal. The row represents actual categories, and the column represents the estimated number of textual posts categorized into each class.

In this case, we have used 80% (10022) of the datasets for training purposes. As shown in figure 5.18 below, the first row indicates that out of training datasets 1301 textual posts are

classified correctly as the category of Disagree and the remaining textual posts are categorized into other classes wrongly. In the next rows, out of 2871 Discuss class the correct labels are 2544. Out of Agree class (2933) the correct labels are 2564 and finally, from the total unrelated class (2820) the correct labels are 2470. This indicates that there is a growing prevalence of these classes. According to the details of the confusion matrix, 8879 text posts out of 10022 are correctly classified under their Stance class, with an average accuracy of 86.5%; the remaining 13.5% of posts are incorrectly classified.

The results are shown diagrammatically in Figure 5.18, which also includes a true label and predicate labels of the data distribution.

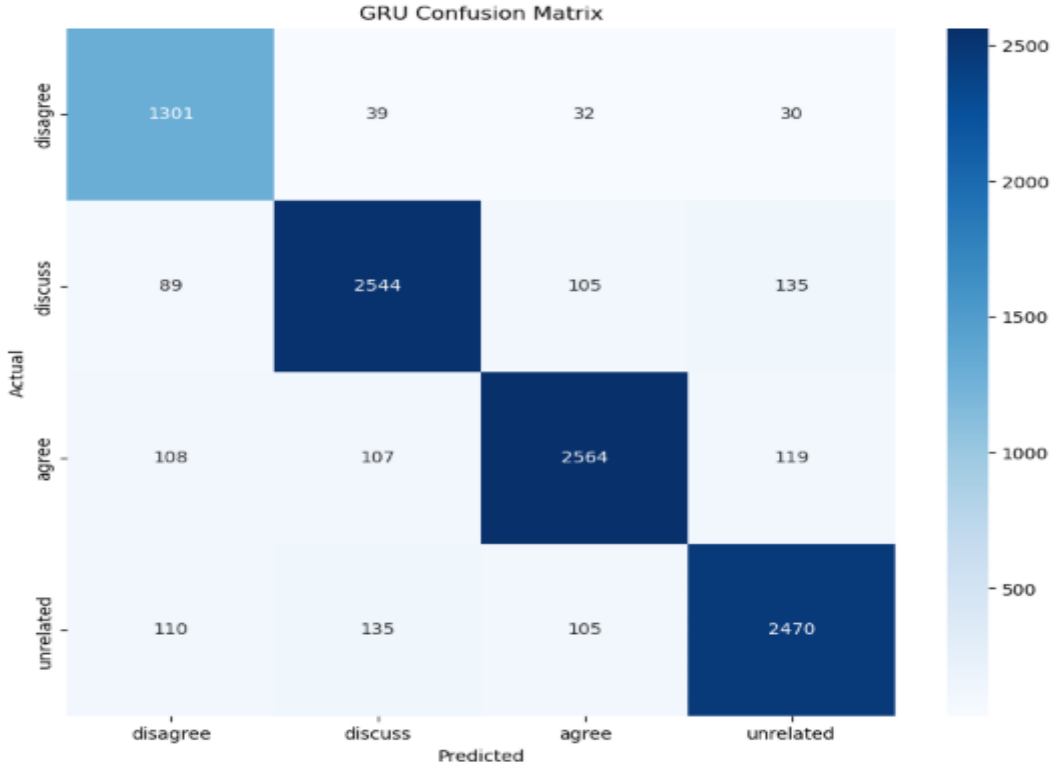

Figure: 5.18. Confusion metrix for Stance classification using the GRU model

We additionally depicted the loss values throughout the model's training and validation stages, as shown in Figure 5.19 and 5.20 the validation accuracy varies greatly and eventually reaches the optimum accuracy by using the Softmax activation function, while the training accuracy rises approximately linearly. We trained our model using the Adam optimizer for a variety of epochs during the experimental work, particularly in contrast to the results, and then reply our model on additional epochs with specific parameters. The most promising validation accuracy was recorded at the 12th epoch using 80% and 20% of training, validation, and test sets of a synthetically generated Afaan oromo social media text comment dataset. The training loss drops sharply to 1.4, but the validation loss fluctuates up and down.

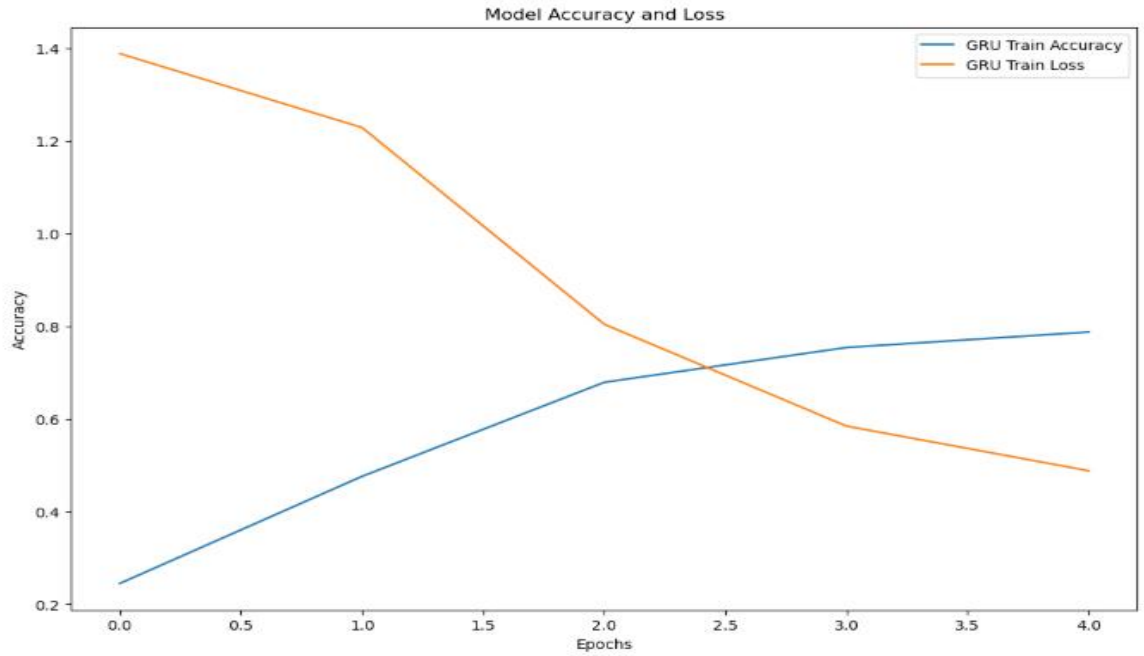

Figure: 5.19.Training Accuracy and loss using GRU

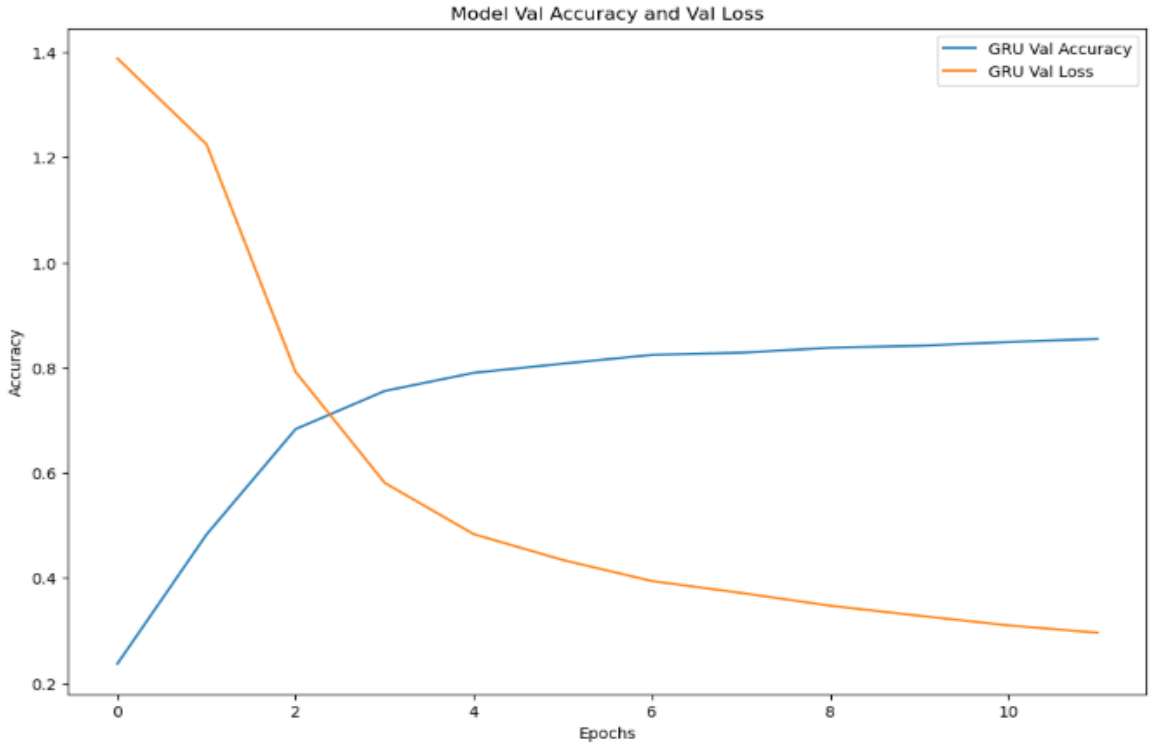

Figure: 5.20. Validation Accuracy and loss using GRU

5.10. Discussion of the Result

In this study, we have demonstrated the model for classifying Stance in Afaan Oromo social media text comments, which was enabled by 12528 Afaan Oromo text comments. The model was tested and trained using 10022 comments for training and 2506 for testing. The experiment shows that the training-test splitting ratio fits the best accuracy at 80% of training and 20% of the testing dataset, indicating that the amount of training and testing data determines the model's performance. The parameters that we have changed to fit the best model accuracy for Stance classification using Afaan oromo text include the single dense output layer, multiple dense output layers, and changing activation function.

This suggests that every parameter has been set following a number of experiments. The comparison of performance outcomes is due to these experimental configurations. After the final feature is developed, an experiment is carried out using five deep learning algorithms to assess the performance of training and testing the dataset. The first experiment was conducted by adopting the LSTM network model. The second experimental analysis was done by the bidirectional LSTM model, which can learn backward and forward contextual information from the text. Using GRU with the same optimizer and activation function, the third experiment was conducted.

As demonstrated by the examination of every experiment, all stance types that fall under a particular category experience misclassifications. The different experimental settings can be used to explain the variations in experimental effects. 86%, 86.63% and 86.5%, respectively, are the training accuracy for the LSTM, BILSTM, and GRU algorithms.

The results of the experiment, as can be seen in Table 5.2 below, show that the deep learning algorithms can achieve comparable performance.

Algorithms	Classes	precision	Recall	F1-score	Train Accuracy
LSTM	Disagree	0.69	0.96	0.81	86%
	Discuss	0.91	0.87	0.89	
	Agree	0.93	0.86	0.89	
	Unrelated	0.92	0.86	0.89	
	Macro Ave	0.86	0.89	0.87	
BILSTM	Disagree	0.72	0.97	0.83	86.63%
	Discuss	0.89	0.89	0.89	
	Agree	0.92	0.87	0.89	
	Unrelated	0.94	0.84	0.89	
	Macro Ave	0.87	0.89	0.87	
GRU	Disagree	0.73	0.95	0.83	86.5%
	Discuss	0.91	0.87	0.89	
	Agree	0.92	0.86	0.89	
	Unrelated	0.90	0.87	0.89	
	Macro Ave	0.87	0.89	0.87	

Table: 5.2. The result of the experiment shows that the deep learning algorithms can achieve comparable performance.

Generally, BILSTM is superior to the other classifiers for classifying stance in Afaan oromo social texts. As we covered in the previous chapter, this is because BILSTM can produce crucial features. The BILSTM model's accuracy needs to be improved for the reasons listed below. As a result, word vectors generated with CBOW can yield suitable features, and text features generated with BILSTM can effectively generate high-level text features. Because of its efficacy and efficiency, this study chooses the BILSTM classification model as the Stance classification model.

Additionally, as demonstrated by the experiment's findings, BILSTM is preferred.

Chapter Six

Conclusion and Recommendation

6.1. Conclusion

In this study, deep learning techniques for stance detection in Afaan Oromo social media writings are examined. Human-computer interaction and text synthesis are among the domains that rely on text-based stance detection. We used a new dataset for Stance Detection tasks that was collected from several Afaan Oromo social media platforms, including Facebook pages, Telegram and Twitter, and categorized into four groups agree, disagree, discuss, and unrelated to assess the algorithm.

The core model for planning and completing the entire task are summarization, fitting, and model development (word embedding, deep learning technique, output determination, and model assembly). Pre-processing techniques such as, tokenization, normalization, stop-word removal and stemming were used. For the Stance categorization model in Afaan Oromo literature, this work has provided an acceptable technique and architecture.

For our study data was used to carefully build various stance model combinations for our investigation and assess their efficacy. We have used 12528 stance detection texts from various authenticated social media pages that have an impact on social media as well as from indigenous books and other sources to evaluate the models' performance and train them. The models were then trained using the prepared dataset.

The Model was developed by applying the three algorithms i.e. LSTM, BILSTM and GRU. The model performance achieved the following result for LSTM, BILSTM and GRU. LSTM scored 86% accuracy and 87% F-score whereas BILSTM achieved 86.63% accuracy and 87% F1 score and GRU achieved 86.5% accuracy and 87% F1-score. The experimental results show that BILSTM model achieved the better performance than LSTM and GRU models.

6.3. Contribution of the study

This study offers many contributions; the main contributions of this work are summarized as follows:

- We have prepared around 12528 pairs of headlines and bodies of Afaan Oromo text.
- The corpus or data set generated during the current study's data preprocessing effort can be readily made available for use in any future natural language processing studies requiring Afaan Oromo texts.
- Three deep learning techniques are evaluated for performance, and the optimal deep learning model for stance detection from Afaan Oromo texts is determined.

6.2. Recommendation

The development of a stance identification model using Afaan Oromo text provides a lot of room for further research, and further effort or analysis is required. Even though the study's findings are intriguing, they are not without flaws, and more investigation is required to determine these. For future researchers, the following instructions are presented to make it more practical and suitable for users who perform better than this work.

- We suggest adding more stance expressions, such speech and picture, to enhance stance classification tasks.
- Future studies can compare the findings with those of the current study using different methods, such as the BERT model.
- Further studies are advised to gather data sets of various sizes and shapes in order to determine whether the existing model can be improved.
- Gathering data from more social media platforms, such Twitter, TikTok, and You Tube, in order to compile texts and create a dataset.
- Our model was trained on a small dataset; it could be a good idea to expand the study to include a larger dataset.
- Further researchers are advised to conduct the research on stance detection by using Bilingual or Multilingual languages.

References

ALDayel, A., & Magdy, W. (2021). Stance detection on social media: State of the art and trends. *Information Processing and Management*, *58*(4). https://doi.org/10.1016/j.ipm.2021.102597

Chaudhry, A. K., Baker, D. & Thun-Hohenstein, P. (2017). Stance Detection for the Fake News Challenge: Identifying Textual Relationships with Deep Neural Nets. *Stanford*, 1–10.

Du, J., Xu, R., He, Y., & Gui, L. (2017). Stance classification with target-specific neural attention networks. *IJCAI International Joint Conference on Artificial Intelligence*, 3988–3994.

Girma, B. (2021). *Amharic stance classification using deep learning*. 16–80.

Kabada, S. (2020). Emotion Detection for Afaan Oromo Using Deep Learning. *New Media and Mass Communication*, *92*, 1–14. https://doi.org/10.7176/nmmc/92-01

Krejzl, I. P., & Uk, A. (2018). *Stance detection and summarization in social networks PhD Study Report CORE View metadata, citation and similar papers at core Stance detection and summarization in social networks*. http://www.kiv.zcu.cz/publications/

Küçük, D., & Can, F. (2022). A tutorial on stance detection. *WSDM 2022 - Proceedings of the 15th ACM International Conference on Web Search and Data Mining*, 1626–1628. https://doi.org/10.1145/3488560.3501391

Meron, T. (2020). *Crime Analysis and Prediction System (CAPS)*.

Mohammad, S. M., Kiritchenko, S., Sobhani, P., Zhu, X., & Cherry, C. (2016). SemEval-2016 task 6: Detecting stance in tweets. *SemEval 2016 - 10th International Workshop on Semantic Evaluation, Proceedings*, 31–41. https://doi.org/10.18653/v1/s16-1003

Oliyad, S. (2021). *JIMMA INSTITUTE OF TECHNOLOGY FACULTY OF COMPUTING AND INFORMATICS Online Hate Speech Detection for Afaan Oromo Using Deep Learning OLIYAD SEBOKA MESKELE A THESIS SUBMITTED TO FACULTY OF COMPUTING AND INFORMATICS OF*.

Roeters Van Lennep, J. (2021). *Deep Learning for Stance Detection: A Review and Comparison of the State-of-the-Art Approaches*. 1–19. https://www.zdnet.com/article/online-fake-news-costing-us-78-billion-globally-each-year/

Singh, S. (2018). *Natural Language Processing for Information Extraction*. 1–24. http://arxiv.org/abs/1807.02383

Sobhani, P., Inkpen, D., & Zhu, X. (2017). A Dataset for Multi-Target Stance Classification. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, *2*, 551–557.

Surafel, T. (2023). POLITICAL STANCE DETECTION ON AMHARIC TEXT USING MACHINE LEARNING. *Journal of Engineering Research*.

Worku, B. (2022). *SCHOOL OF GRADUATE STUDIES STANCE DETECTION OF AFAAN OROMO TEXT USING SUPERVISED MACHINE LEARNING APPROACH Master ' s Thesis By Worku Bacha February 2022 Nekemte , Ethiopia*. *February*.

ALDayel, A., & Magdy, W. (2021). Stance detection on social media: State of the art and trends. *Information Processing and Management*, *58*(4). https://doi.org/10.1016/j.ipm.2021.102597

Chaudhry, A. K., Baker, D. & Thun-Hohenstein, P. (2017). Stance Detection for the Fake News Challenge: Identifying Textual Relationships with Deep Neural Nets. *Stanford*, 1–10.

Du, J., Xu, R., He, Y., & Gui, L. (2017). Stance classification with target-specific neural attention networks. *IJCAI International Joint Conference on Artificial Intelligence*, 3988–3994.

Girma, B. (2021). *Amharic stance classification using deep learning*. 16–80.

Kabada, S. (2020). Emotion Detection for Afaan Oromo Using Deep Learning. *New Media and Mass Communication*, *92*, 1–14. https://doi.org/10.7176/nmmc/92-01

Krejzl, I. P., & Uk, A. (2018). *Stance detection and summarization in social networks PhD Study Report CORE View metadata, citation and similar papers at core Stance detection and summarization in social networks*. http://www.kiv.zcu.cz/publications/

Küçük, D., & Can, F. (2022). A tutorial on stance detection. *WSDM 2022 - Proceedings of the 15th ACM International Conference on Web Search and Data Mining*, 1626–1628. https://doi.org/10.1145/3488560.3501391

Meron, T. (2020). *Crime Analysis and Prediction System (CAPS)*.

Mohammad, S. M., Kiritchenko, S., Sobhani, P., Zhu, X., & Cherry, C. (2016). SemEval-2016 task 6: Detecting stance in tweets. *SemEval 2016 - 10th International Workshop on Semantic Evaluation, Proceedings*, 31–41. https://doi.org/10.18653/v1/s16-1003

Oliyad, S. (2021). *JIMMA INSTITUTE OF TECHNOLOGY FACULTY OF COMPUTING AND INFORMATICS Online Hate Speech Detection for Afaan Oromo Using Deep Learning OLIYAD SEBOKA MESKELE A THESIS SUBMITTED TO FACULTY OF COMPUTING AND INFORMATICS OF*.

Roeters Van Lennep, J. (2021). *Deep Learning for Stance Detection: A Review and Comparison of the State-of-the-Art Approaches*. 1–19. https://www.zdnet.com/article/online-fake-news-costing-us-78-billion-globally-each-year/

Singh, S. (2018). *Natural Language Processing for Information Extraction*. 1–24. http://arxiv.org/abs/1807.02383

Sobhani, P., Inkpen, D., & Zhu, X. (2017). A Dataset for Multi-Target Stance Classification. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, *2*, 551–557.

Surafel, T. (2023). POLITICAL STANCE DETECTION ON AMHARIC TEXT USING MACHINE LEARNING. *Journal of Engineering Research*.

Worku, B. (2022). *SCHOOL OF GRADUATE STUDIES STANCE DETECTION OF AFAAN OROMO TEXT USING SUPERVISED MACHINE LEARNING APPROACH Master ' s Thesis By Worku Bacha February 2022 Nekemte , Ethiopia*. *February*.

APPENDIXES

Appendix I: Manual data source

Body ID	Headline	Domain	Stance
1	Dhibamaa kaansarii kan ta'e namni ganna 46, tapha lootorii Powerball jedhamuun doolaara biiliyoona 1.3 injifate.	Fayyaa	disagree
2	Weerara Koleraa Godiina Shawaa Bahaa Aanaa Boosat keessatti mudateen namoota sadii du'anii heddu dhibichaar	Fayyaa	discuss
3	Qorannoon akka agarsiisutti amalli keenya fayyaa sammuu keenyaa irratti dhiibbaa guddaa qaba.Amaloota fayyaa	Fayyaa	disagree
4	Hariiroo sukkaaraa fi dhukkuba sukkaaraa gidduu jiru beekuun gaaridha jedha dhaabbanni fayyaa oromiyaa.	Fayyaa	agree
5	Miila keessan qaxxaamursuun oomisha isparmii irratti dhiibbaa uumuu akka danda'u ragaaleen ni mul'isu.Miila wa	Fayyaa	agree
6	Sagaleen meeshaalee manaa fayyaa keessan akka miidhu quba qabduu?Sagaleen nama jeequ hundi kaaniitti hom	Fayyaa	agree
7	Itoophiyaa keessatti of ajjeesuun akka dhimma yaaddessaa ta'eetti ilaalamu isaa dhaabbanni fayyaa gabaase.	Fayyaa	discuss
8	Qaroo dhabuurratti hojii dhabuu. Zuleeykaa Aadam jedhamti, qaroo dhabeettidha. Sabbataatti beelaaf daaru obsit	Fayyaa	agree
9	Mudannoo xiyyaara keessaa nama dheeraa Awwaluu Buxxuulaa Xiyyaaraan gara Dubaayitti imaluuf rakkadheen tu	Fayyaa	agree
10	Lammeechaa koo Rabbi si haa fayyisu.	Fayyaa	agree
11	Biyya qulqulluu fi naannawaa mijataa uumuun lammiileen dhibeewwan daddarboof akka hin saaxilamne qooda qa	Fayyaa	agree
12	Finfinneetti duulli talaallii Ittisa dhibee Saree maraattee eegale.	Fayyaa	agree
13	Faayidaa Muuziin fayyaa namaaf qabuFinfinnee, Fulbaana 10, 2017 (FBC) Muuziin kuduraa baay'ee barbaachisu kee	Fayyaa	agree
14	Gaaf tokko- "Dhukkubsataan rifeensa muru ispeeshaalistii mormaa oliiti nuun jedhe" Mudannoo Ogeeyyii fayyaa.	Fayyaa	unrelated
15	Komii Kenna Tajaajila Fayyaa Fooyya'e.	Fayyaa	agree
16	Ameebaan Dhukkuba Mar'imaanii Qofa Miti.	Fayyaa	agree
17	Dooktar Abrahaam Bakka Bu'aa Sagantaa Nyaata Addunyaa waliin mari'atan Finfinnee, Hagayya 23, 2016 (FBC)Minis	Fayyaa	discuss
18	Fayyaan Faaya!Fayyaan sektara caasaa guddaa fi fedhii tajaajilamaa bal'dhaa qabudha.	Fayyaa	agree
19	Weerara koleeraa Sudaan keessatti torban muraasa darban keessa jalqabeen hoo xinnaate namoonni 20 ol du;anii	Fayyaa	unrelated
20	Akka gabasa dhaabbata mootummoota Gamtoomaniitti samuuda ji'a waxabajjii keessa naannoo westwater irraa fu	Fayyaa	agree
21	Jaarmayaan Fayyaa Addunyaa (WHO)n torban dabree Rippabiliika Dimookiraatawaa Koongoo (DRC) fi biyyoota Afr	Fayyaa	discuss
22	Rakkoo onnee Daa'iimmanii ilaalchisee Yunivarsitii Haawaasaatti speeshaalistii yaala daa'iimmanii fi ogeessa qora	Fayyaa	agree
23	Israa'el hojii gargaarsa namoomaa akkasumas duula talaallii pooliyoo bal'inaan kennite.	Fayyaa	agree

Appendix II:Sample data loading

```
data = pd.read_csv('/Users/WB/Desktop/prepare dataset 4 SD/Stance_SD_dataset1.csv', encoding = 'latin1')
data.head()
```

	Body ID	Headline	Domain	Stance
0	1	Dhibamaa kaansarii kan ta'e namni ganna 46, ta...	Fayyaa	disagree
1	2	Weerara Koleraa Godiina Shawaa Bahaa Aanaa Boo...	Fayyaa	discuss
2	3	Qorannoon akka agarsiisutti amalli keenya fayy...	Fayyaa	disagree
3	4	Hariiroo sukkaaraa fi dhukkuba sukkaaraa giddu...	Fayyaa	agree
4	5	Miila keessan qaxxaamursuun oomisha isparmii i...	Fayyaa	agree

Appendix III:Stopword removal

'ilaalchisee',	'al',	'akkataa',	'ammoo',
'gaggeeffame',	'ala',	'akkataan',	'ala',
'aga',	'alatti',	'akkas',	'alatti',
'achi',	'alla',	'akkasitti',	'alla',
'achuma',	'akka',	'akkasumas',	'an',
'adda',	'akkam',	'akkuma',	'ana',
'addaatti',	'akkamii',	'akksumas',	'anee',
'afoo',	'akkamiitu',	'amma',	'ani',
'agarsiisoo',	'akkana',	'ammaa',	'asham',

'asitti',
'attam',
'ati',
'awu',
'bira',
'biraa',
'biratti',
'biroo',
'biroon',
'biroos',
'biyyam',
'booda',
'booddee',
'bukkee',
'cinaa',
'dabalatees',
'dhaa',
'dhaan',
'dudduuba',
'dugda',
'duuba',
'dura',
'duraa',
'eega',
'eegana',
'eegasii',
'ennaa',
'eenuu',
'eenyu',
'eennu',
'eenyufaa',
'eenyuun',
'eennuun',
'eenyuuf',
'eennuuf',
'eenyufaa',
'eenyufaadhaan',
'eenyufaaf',
'eenyufaarraa',
'eenyufatti',
'eenyurra',
'eennurraa',
'eenyurraa',
'eennurraa',
'eenyurratti',
'eennuratti',
'eenyuuf',
'eenyuree',
'eennuree',
'eenyutti',
'eennutti',
'eenyum',
'eessa',
'ega',
'eessatti',
'eessarraa',
'eessaaf',
'eessaan',
'erga',
'ergii',
'f',
'faa',
'faallaa',
'fagaatee',
'faati',
'faatidhaa',
'faatiidhaa',
'fakkaatanitti',
'fakkaatti',
'fakkaatu',
'fakkaattu',
'fakkeenyaa',
'fakkeenyaaf',
'fedhetuu',
'fi',
'fk',
'fkf',
'fkn',
'fknf',
'fuuldura',
'fundura',
'fullee',
'fuullee',
'gaa',
'gad',
'gadi',
'gaditti',
'gahaa',
'gajjallaa',
'gala',
'galan',
'galani',
'galuu',
'gama',
'gamam',
'gar',
'gara',
'garam',
'garamiin',
'garamitti',
'garana',
'garas',
'isiniin',
'isiniis',
'isinirraa',
'ittaanee',
'itti',
'ittillee',
'ittumallee',
'ittiin',
'ituu',
'ituullee',
'jala',
'jara',
'jechaan',
'jechoota',
'jechuu',
'jechuun',
'jedhu',
'jedhus',
'jira',
'jiraadha',
'jiraadhe',
'jiraanne',
'jiraate',
'jiraattee',
'jiraatta',
'jiraatti',
'jiraanna',
'jiraatan',
'jiraattan',
'jiran',
'jiranii',
'jiranirra',
'jiranu',
'jirre',

Appendix IV: Models

LSTM

[109]:
```
from tensorflow.keras import Model, Input
from tensorflow.keras.layers import LSTM, Embedding, Dense
from tensorflow.keras.layers import TimeDistributed, SpatialDropout1D, Bidirectional
```

[111]:
```
from tensorflow.keras import Model, Sequential, Input
from tensorflow.keras.layers import Embedding, Dense, LSTM
from tensorflow.keras.layers import Bidirectional, TimeDistributed, SpatialDropout1D
```

[113]:
```
model = Sequential()

model.add(Embedding(input_dim = input_dim, output_dim = output_dim, input_length = input_length))

model.add(SpatialDropout1D(0.2))

model.add(LSTM(units = 200, recurrent_dropout = 0.2, return_sequences = True))

model.add(TimeDistributed(Dense(units = n_tags, activation = 'softmax',kernel_regularizer=keras.regularizers.l1(0.001))))
keras.layers.Dropout(0.2)

#model.add(Dense(units=16,activation='relu'))

model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
#model.add(Dense(1, activation='sigmoid'))

#model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])

model.summary()
```

```
Training LSTM model...
Epoch 1/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 55ms/step - accuracy: 0.2611 - loss: 1.3812
Epoch 2/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 16s 51ms/step - accuracy: 0.4940 - loss: 1.2573
Epoch 3/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 16s 52ms/step - accuracy: 0.6931 - loss: 0.7866
Epoch 4/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 19s 57ms/step - accuracy: 0.7684 - loss: 0.5688
Epoch 5/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 52ms/step - accuracy: 0.7924 - loss: 0.4767
Epoch 6/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 16s 52ms/step - accuracy: 0.8120 - loss: 0.4221
Epoch 7/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 56ms/step - accuracy: 0.8183 - loss: 0.3826
Epoch 8/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 19s 52ms/step - accuracy: 0.8352 - loss: 0.3494
Epoch 9/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 20s 51ms/step - accuracy: 0.8403 - loss: 0.3280
Epoch 10/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 51ms/step - accuracy: 0.8525 - loss: 0.3011
Epoch 11/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 16s 51ms/step - accuracy: 0.8579 - loss: 0.2957
Epoch 12/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 16s 51ms/step - accuracy: 0.8600 - loss: 0.2783
```

BILSTM

[54]:
```
from tensorflow.keras import Model, Input
from tensorflow.keras.layers import Bidirectional, Embedding, Dense
from tensorflow.keras.layers import TimeDistributed, SpatialDropout1D, Bidirectional
```

[55]:
```
from tensorflow.keras import Model, Sequential, Input
from tensorflow.keras.layers import Embedding, Dense, Bidirectional
from tensorflow.keras.layers import Bidirectional, TimeDistributed, SpatialDropout1D
```

[58]:
```
models = Sequential()
models.add(Embedding(input_dim = input_dim, output_dim = output_dim, input_length = input_length))
models.add(SpatialDropout1D(0.2))
models.add(Bidirectional(LSTM(units = 200, recurrent_dropout = 0.2, return_sequences = True)))
models.add(TimeDistributed(Dense(units = n_tags, activation = 'softmax',kernel_regularizer=keras.regularizers.l1(0.001))))
keras.layers.Dropout(0.2)
#model.add(Dense(units=16,activation='relu'))
models.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
#model.add(Dense(1, activation='sigmoid'))
#model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
models.summary()
```

```
Training BiLSTM model...
Epoch 1/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 32s 60ms/step - accuracy: 0.2640 - loss: 1.3776
Epoch 2/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 54ms/step - accuracy: 0.4655 - loss: 1.2396
Epoch 3/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 54ms/step - accuracy: 0.7031 - loss: 0.7816
Epoch 4/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 55ms/step - accuracy: 0.7725 - loss: 0.5535
Epoch 5/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 55ms/step - accuracy: 0.8002 - loss: 0.4656
Epoch 6/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 54ms/step - accuracy: 0.8238 - loss: 0.4048
Epoch 7/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 55ms/step - accuracy: 0.8318 - loss: 0.3688
Epoch 8/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 54ms/step - accuracy: 0.8374 - loss: 0.3406
Epoch 9/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 54ms/step - accuracy: 0.8528 - loss: 0.3022
Epoch 10/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 56ms/step - accuracy: 0.8554 - loss: 0.2909
Epoch 11/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 18s 58ms/step - accuracy: 0.8535 - loss: 0.2844
Epoch 12/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 54ms/step - accuracy: 0.8663 - loss: 0.2642
```

GRU

[173]:
```
from tensorflow.keras.layers import Embedding,GRU,LSTM,Bidirectional,SimpleRNN
```

*[176]:
```
model_GRU = Sequential()
model_GRU.add(Embedding(input_dim = input_dim, output_dim = output_dim, input_length = input_length))
model_GRU.add(SpatialDropout1D(0.2))
model_GRU.add(GRU(units = 200, recurrent_dropout = 0.2, return_sequences = True))
model_GRU.add(TimeDistributed(Dense(units = n_tags, activation = 'softmax',kernel_regularizer=keras.regularizers.l1(0.001))))
keras.layers.Dropout(0.2)
#model.add(Dense(units=16,activation='relu'))
model_GRU.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
#model.add(Dense(1, activation='sigmoid'))
#model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
model_GRU.summary()
```

```
Training GRU model...
Epoch 1/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 22s 49ms/step - accuracy: 0.2340 - loss: 1.3901
Epoch 2/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 52ms/step - accuracy: 0.5092 - loss: 1.2439
Epoch 3/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 16s 52ms/step - accuracy: 0.7068 - loss: 0.7650
Epoch 4/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 16s 50ms/step - accuracy: 0.7734 - loss: 0.5382
Epoch 5/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 53ms/step - accuracy: 0.8011 - loss: 0.4630
Epoch 6/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 16s 51ms/step - accuracy: 0.8326 - loss: 0.3898
Epoch 7/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 16s 52ms/step - accuracy: 0.8393 - loss: 0.3629
Epoch 8/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 20s 51ms/step - accuracy: 0.8503 - loss: 0.3284
Epoch 9/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 17s 55ms/step - accuracy: 0.8547 - loss: 0.3196
Epoch 10/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 18s 55ms/step - accuracy: 0.8541 - loss: 0.3067
Epoch 11/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 16s 52ms/step - accuracy: 0.8592 - loss: 0.2897
Epoch 12/12
313/313 ━━━━━━━━━━━━━━━━━━━━ 16s 50ms/step - accuracy: 0.8654 - loss: 0.2779
```