ETHIOPIAN INSTITUTE OF TECHNOLOGY-MEKELLE (EIT-M)
SCHOOL OF COMPUTING

POLITICAL STANCE DETECTION AND CLASSIFICATION ON TIGRIGNA TEXT USING DEEP LEARNING APPROACHES

By: Ngsti Gebrehiwot Shawul
Advisor: Guesh Dagnew(Ph.D.)

August, 2025
Mekelle, Ethiopia

POLITICAL STANCE DETECTION AND CLASSIFICATION ON TIGRIGNA TEXT USING DEEP LEARNING APPROACHES

A THESIS SUBMITTED TO SCHOOL OF COMPUTING
ETHIOPIAN INSTITUTE OF TECHNOLOGY-MEKELLE
MEKELLE UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF
MASTERS OF SCIENCE IN COMPUTER SCIENCE

NGSTI GEBREHIWOT SHAWUL

POLITICAL STANCE DETECTION AND CLASSIFICATION ON TIGRIGNA TEXT USING DEEP LEARNING APPROACHES

A THESIS SUBMITTED TO SCHOOL OF COMPUTING
ETHIOPIAN INSTITUTE OF TECHNOLOGY-MEKELLE
MEKELLE UNIVERSITY

BY
NGSTI GEBREHIWOT SHAWUL

APPROVAL SHEET

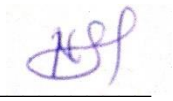
Name and signature of members of the examining Board

Role	Name	Signature	Date
Advisor	1. Guesh Dagnew (Ph.D.)		09/08/2025
Internal Examiner	2, Behailu **Getachew** (Ph.D.)		09/15/2025
Chair Person	3. Ferede Ali Tahir		25/08/2025

Declaration

I hereby declare that the work which is being presented in this thesis entitled "POLITICAL STANCE DETECTION AND CLASSIFICATION ON TIGRIGNA TEXT USING DEEP LEARNING APPROACHES" is original work of my own, has not been presented for a degree of any other university and all the resources used for the thesis have been duly acknowledged.

Name of the candidate: **Ngsti Gebrehiwot Shawul**

Signature: 

Date: 07/08/2025

Approval for Submission

This thesis, entitled " **POLITICAL STANCE DETECTION AND CLASSIFICATION ON TIGRIGNA TEXT USING DEEP LEARNING APPROACHES** " by **Ngsti Gebrehiwot Shawul**, has been submitted for examination for the degree of Master of Science in Computer Science with my approval as advisor in the School of Computing, EiTM, Mekelle University.

Name of Advisor: **Guesh Dagnew (Ph.D.)**

Signature: 

Date: 07/08/2025

Acknowledgment

Success is never achieved single-handedly. So, I must acknowledge all those who have provided helping hands in making this thesis successful. Firstly, I would like to thank ALMIGHTY GOD for supporting me in every way of my life.

I would also like to express my deep sense of gratitude towards my research advisor Dr. Guesh Dagnew for his invaluable guidance, cooperation, and approval in the successful completion of this work. Secondly, I am privileged to express my sense of gratitude to my respected instructors whose unparalleled knowledge, moral fiber and judgment along with their know-how, was an immense support in completing this thesis. Those who in one way or another have contributed to the success of the thesis are also recognized.

Last but not least, a great deal of appreciation and best wishes to all my friends for their encouragement during this work.

Dedicated to

This thesis is dedicated to my husband and children; to my parents for their continuous and consistent support by understanding my goals and for to my friend Kibrom Gidey who has supported me throughout the process; and to all of my friends who were helping me in many ways.

Table of Content

List of Tables

List of Figures

List of Acronyms

AI – Artificial Intelligence
BERT – Bidirectional Encoder Representations from Transformers
BI-LSTM – Bidirectional Long Short-Term Memory
BOW – Bag-Of-Words
CBOW – Continuous Bag-Of-Words
Collab – Collaboratory
CNNS – Convolutional Neural Networks
GPU – Graphical Processing Unit
GPT – Generative Pre-Trained Transformer
GRU – Gated Recurrent Unit
HAN – Hierarchical Attention Network
IDE – Integrated Development Environment
JSON – JavaScript Object Notation
LR – Logistic Regression
LSTM – Long Short-Term Memory
NLTK – Natural Language Toolkit
NLP – Natural Language Processing
RF – Random Forest
RNN – Recurrent Neural Networks
SG – Skip-Gram
SVM – Support Vector Machine
TF-IDF – Term Frequency-Inverse Document Frequency
TPLF – Tigray People's Liberation Front
TPU – TensorFlow Processing Unit
Word2Vec – Word to Vector

Abstract

The rise of social media has transformed public discourse, providing platforms for individuals to express their opinions on various topics, particularly political issues. Political stance detection, which identifies an individual's position on specific topics, has become increasingly important for policymakers, researchers, and organizations aiming to navigate complex social landscapes and make informed decisions. Despite its significance, most research in this area has focused on English and other European languages, with limited attention to Amharic and virtually none to Tigrigna, a language spoken by millions in Eritrea and Ethiopia. This gap is particularly critical given the ongoing socio-political challenges, such as unemployment and civil unrest, in Tigrigna-speaking communities. This study addresses the lack of research on political stance detection in Tigrigna by analyzing comments from the TPLF Facebook page. Data was collected using the Face-pager tool, and two feature extraction strategies—Bag of Words (BOW) and Skip-gram from Word2Vec—were employed to convert textual data into numerical representations suitable for machine learning. Advanced deep learning algorithms, including Gated Recurrent Unit (GRU), Transposed Gated Recurrent Unit (T-GRU), and Long Short-Term Memory (LSTM), were applied to classify political sentiments toward the TPLF party. The results demonstrate that the Transposed GRU model combined with the Skip-gram strategy achieved an accuracy of 82% and an F1-score of 0.8822, representing a significant advancement in political stance classification for low-resource languages. These findings highlight the effectiveness of deep learning approaches in analyzing Tigrigna text and provide a foundational methodology for future research. This study addresses a gap in the existing literature by providing a nuanced analysis of the socio-political dynamics within Tigrigna-speaking communities, which have been largely overlooked in political discourse research. By utilizing advanced techniques in stance detection, this research enhances our understanding of public sentiment and sets a precedent for scholarly inquiry into underrepresented languages. The contributions are threefold: it establishes a foundational dataset specifically tailored to Tigrigna-speaking contexts; it employs innovative natural language processing methods, such as transfer learning and alternative word embeddings; and it considers idiomatic expressions and the role of emojis, offering a more granular understanding of public sentiment. Looking ahead, future research should broaden the dataset to encompass a wider array of political topics and explore advanced machine learning techniques, thereby enriching the findings. This research lays the groundwork for subsequent studies and contributes to a more inclusive understanding of political discourse across diverse linguistic landscapes, ultimately fostering greater engagement with marginalized voices in the political arena.

Keywords: *Tigrigna language, Classification, Feature extraction, Detection, NLP, social media.*

CHAPTER ONE

1. INTRODUCTION

1.1. Background of study

In recent years, there has been growing interest in leveraging artificial intelligence (AI) technology on social media to determine user opinions. Social media platforms have become a key source of information and a space for sharing ideas and emotions. By analyzing the sentiments conveyed by users, valuable insights can be gained regarding public opinion, customer feedback, and brand perception. AI-driven stance detection methods use machine learning and natural language processing algorithms to automatically identify and classify sentiments. These algorithms analyze various text-based content, including social media posts, comments, and reviews.

According to DIGITAL 2022: ETHIOPIA, as of January 2022, the number of internet users in Ethiopia has reached a staggering 29.83 million. This growth is evident through the significant number of Ethiopians and Eritreans fluent in Tigrigna who have created personal profiles on Facebook – with 5.95 million in Ethiopia and 290.5 thousand in Eritrea as of January 2022[1]. As a result, there is a clear increase in the number of Facebook and social media users who are actively engaging in political discussions. Ethiopian and Eritrean communities are quickly embracing social media platforms as the primary means of exchanging political ideas.

Despite being relatively new forms of communication for political discourse and other societal topics, many users from these countries sink towards social networking sites with ease. It's worth noting that several prominent political parties and activists from both nations have established a strong online presence, amassing tens of thousands or more followers on popular platforms like Facebook and Twitter. The unrestricted ability to express one's thoughts without limitation is a crucial aspect of social media[2]

A speaker's stance represents their expressed opinion and evaluation of a given statement. In the context of social media, stance identification plays a critical role, particularly in analytical studies aimed at assessing public sentiment on political and social issues [3].

These topics are inherently argumentative, with individuals engaging in debates on various contentious issues. In social media discourse, stance detection has increasingly focused on subjects such as feminism and abortion. Similarly, political topics, including elections and referendums, have been extensively analyzed to assess public sentiment. Stance classification is influenced by multiple personal, cultural, and social factors. Moreover, the process is often referred to as point-of-view recognition, as it involves identifying perspectives through the expression of attitudes toward contentious subjects.

Stance identification is a complex process with multiple dimensions, closely tied to political positions, which are often shaped by empirical behavior and reflected in observable patterns. In the context of social media, users may explicitly express their opinions through posts on various topics or implicitly convey their attitudes through interactions and preferences[3],[4], [5]. Within the field of Natural Language Processing (NLP), stance detection has a wide range of applications, including the automated identification of community support or opposition toward specific political or religious viewpoints. Additionally, research has demonstrated its application in the development of recommendation systems and market forecasting models. Notably, stance identification has gained increasing significance in the detection of rumors and misinformation, highlighting its role in combating the spread of false information [3].

1.2. Motivation of the study

Nowadays, the increasing prominence of social media as a channel for political discourse has created new opportunities for understanding public opinion. Despite the widespread application of stance classification techniques in global languages such as English, little attention has been paid to resource-constrained languages such as Tigrigna. The unique linguistic and cultural characteristics of Tigrigna pose significant challenges, including a lack of annotated datasets and tools tailored to the language. These barriers have limited the development and application of AI-driven stance detection methods in Tigrigna is language.

The analysis of political stances in Tigrigna texts holds significant societal relevance, particularly in the context of the current political landscapes, where social media platforms such as Facebook serve as primary channels for political discourse. Despite the growing use of Tigrigna in digital communication, research NLP for this language remains limited. This study seeks to bridge existing gaps in Tigrigna NLP by enhancing the understanding of political stance detection, thereby facilitating more informed decision-making, fostering public engagement, and enabling more precise sentiment analysis.

The motivation for this study on political stance detection in Tigrigna text stems from significant gaps in existing deep learning (DL) and machine learning (ML) approaches, which have predominantly focused on resource-rich languages like English. Current models often rely on large, annotated datasets, making them ineffective for low-resource languages due to data scarcity. Traditional feature extraction methods like Bag of Words fail to capture the semantic nuances inherent in Tigrigna's rich morphology, while advanced models like LSTMs and GRUs struggle with overfitting when applied to small datasets. Moreover, existing research neglects the contextual understanding necessary for interpreting idiomatic expressions and cultural references

in Tigrigna. This study aims to bridge these gaps by developing a comprehensive annotated dataset, employing innovative feature extraction techniques, implementing robust model architectures like Transposed GRU, and incorporating cultural nuances, thereby enhancing the effectiveness of stance detection in Tigrigna and contributing to the advancement of NLP for low-resource languages.

1.3. Problem Statement

Social media has revolutionized communication and information sharing, providing a platform for individuals to express their views on diverse topics, including politics. This leads to an unprecedented level of public engagement in political discussions. However, this growth has also presented challenges in understanding and categorizing the vast array of opinions expressed online. In particular, stance classification, the process of determining whether a text expresses support, opposition, or neutrality toward a specific target, is an essential yet complex task in the field of Natural Language Processing (NLP).

While significant progress has been made in stance classification for widely spoken languages such as English and other European languages, resource-constrained languages such as Tigrigna remain unexplored [6],[7]. Tigrigna, spoken by millions in Ethiopia, Eritrea, and their diasporas, is deeply integrated into the cultural and political fabric of these regions. However, its unique linguistic structure and limited computational resources pose significant barriers to developing effective stance classification models. Unlike English, Tigrigna lacks large-scale annotated datasets essential for training machine learning models, which hinders advancements in NLP applications. The language also presents unique grammatical and syntactic structures, including its script derived from Ge'ez, phrasal verbs, and regional dialects, adding layers of complexity to computational processing[8]. Political discussions in Tigrigna are often loaded with implicit meanings, idiomatic expressions, and culturally specific references, which existing models trained on other languages fail to capture effectively[9].

Social networks are increasingly used to communicate and express a wide range of viewpoints due to advancements in mobile computing and internet accessibility. This trend has amplified public participation and the exchange of ideas across different demographics. Despite these advantages, most stance classification efforts have primarily focused on identifying attitudes toward specific topics and evaluating public opinion regarding events or subjects in resource-rich languages. Through the analysis of social media posts, it becomes feasible to understand and categorize communities' positions, whether in support or opposition. In Ethiopia and Eritrea, emerging

challenges such as ethnic and civil conflicts, unemployment, and the escalating cost of living highlight the critical need to analyze public sentiment effectively. Social media has become a vital avenue for citizens to voice their perspectives, making stance detection an invaluable tool for addressing societal issues. Ignorance of public interests and viewpoints worsens these problems, as policymakers lack the insights necessary to address underlying concerns.

Given the heightened level of political discourse on platforms like Facebook, where citizens express their thoughts on policies, parties, and events, the development of Tigrigna-specific NLP tools is essential. Current models, designed primarily for resource-rich languages, fail to account for the linguistic, cultural, and technological nuances of Tigrigna. This gap leaves critical questions unanswered and limits the ability to gauge public opinion effectively. The absence of robust stance detection models tailored to Tigrigna restricts the capacity of researchers, policymakers, and organizations to harness social media data for informed decision-making. Addressing these challenges requires advancing methodologies in data collection, annotation, and model development for Tigrigna, enabling the identification of public sentiment and fostering greater societal unity. The uniqueness of this study on stance classification in Tigrigna is highlighted by critical quantitative and comparative factors. Research shows that over 90% of NLP resources are dedicated to major languages like English, which benefit from extensive datasets such as the Common Crawl, offering over 25 terabytes of data, while Tigrigna has fewer than 1,000 annotated texts available for stance classification. Tigrigna's distinct grammatical structures, including its Ge’ez-derived script and rich morphology, present complexities not found in resource-rich languages. With approximately 29.83 million internet users in Ethiopia, about 70% engage in political discussions on platforms like Facebook, underscoring the urgent need for effective sentiment analysis tools tailored to this context. Existing models for widely spoken languages achieve accuracy rates exceeding 80%, yet preliminary tests reveal that Tigrigna-specific models yield accuracies below 50%, emphasizing the necessity for specialized approaches. Furthermore, the lack of Tigrigna-specific NLP tools constrains researchers and policymakers from effectively using social media data for informed decision-making. This study addresses these gaps by developing a model that incorporates the linguistic and cultural nuances of Tigrigna, establishing a foundational methodology for future research in low-resource languages.

To the best of the authors’ knowledge, there is no Tigrigna language stance detection and classification model in Ethiopia. Therefore, computing may be used to meet the need for stance detection and classification.

1.4. Research questions.

This study proposed to address the above problem by answering the following questions:

- What method can be developed to create annotated Tigrigna text datasets that effectively capture diverse political stances?
- Which feature extraction techniques best fits the linguistic and cultural characteristics of Tigrigna for accurate stance detection?
- Which classification algorithms achieve high accuracy in political stance detection for Tigrigna text?
- How does the performance and robustness of the proposed stance detection models compare to existing research for low-resource languages?
- What are the most effective evaluation metrics for assessing Tigrigna stance detection models in practical applications?

1.5. Objectives

1.5.1. General Objective

The General Objective of this research is to design and develop a robust stance detection and classification model capable of accurately classifying political stances in the Tigrigna text.

1.5.2. Specific Objectives

To achieve the general objective, the following specific objectives are identified;

- To develop a methodology for creating annotated Tigrigna text datasets that represent diverse political stances.
- To investigate and evaluate feature extraction techniques tailored to the linguistic and cultural nuances of Tigrigna.
- To develop and implement deep learning models capable of effectively classifying political stances in Tigrigna text.
- To compare the performance of different deep learning architectures on Tigrigna stance detection tasks.
- To evaluate the proposed stance detection models using established evaluation metrics

1.6. Scope of the Study

This research employs deep learning algorithms to accurately detect political stances in written text in the Tigrigna language. To construct a robust dataset, we mined posts and comments of the publicly available datasets from the years 2021-2023 on the TPLF Facebook page. These posts and comments can be categorized into two distinct groups: those in favor and those against the target party. Utilizing deep learning classifiers, we developed a stance detection and classification model and evaluated its performance using various metrics related to classification accuracy.

1.7. Limitation of the Study

The study on Tigrigna language-based political stance detection and classification faced several limitations that impact its findings. Notably, there was a lack of a shared public dataset, which restricts the ability to compare results with other studies and limits the generalizability of the findings. Additionally, the absence of pre-existing models tailored for Tigrigna necessitated the adaptation of existing models from other languages, which may not fully capture the linguistic nuances unique to Tigrigna. The study did not adequately account for idiomatic expressions, emojis, or Latin Amharic, nor did it explore multi-target attitudes, potentially missing nuanced insights. Furthermore, the research neglected to investigate additional themes, author identity, categorization justifications, and stance holder validity, all of which could provide deeper insights into the data. Moreover, inherent dataset biases may have skewed the results, as the dataset might not represent the full spectrum of Tigrigna speakers and their political views. These limitations highlight areas for future research and improvements in the field.

1.8. Significance of Study

The study can be important for different stakeholders; it is described as follows.

- Gauges public sentiment towards political entities and integrates cultural nuances.
- Facilitates informed discussions and detects misinformation, promoting healthier dialogue.
- Assists parties in crafting targeted messages and provides feedback on campaign effectiveness.
- Predicts potential conflicts through stance identification and fosters dialogue.
- Supplies quantitative data for studying political trends and informs policymakers
- Advances natural language processing for Tigrinya and enhances deep learning models

1.9. Organization of the Thesis

Chapter 1 covers the background, problem statement, motivation, objectives, scope, limitation and significance of the study. Chapter 2 presents the essential literature review and gap analysis. Chapter 3 comprises the research methodology, including research design, data preprocessing, feature extraction, activation functions, model overfitting handling, hyperparameter tuning, model evaluation techniques and Development Framework Tools. Chapter 4 presents the experimental setup and results of the proposed model, including data preprocessing, feature extraction, neural network selection, and word2vec generation. Chapter 5 presents the findings and discussion, describing the research results and evaluating the model using various metrics. Last but least, Chapter 6 comprises the concluding remarks, recommendations, future research directions.

CHAPTER TWO

2. LITERATURE REVIEW: THEORETICAL AND EMPIRICAL PERSPECTIVES

The overview of numerous kinds of literature is examined and discussed in this study's chapter, which also identifies research gaps and helps to establish research goals. It also provides a brief description of certain themes, their applicability, tools and procedures, and related issues.

2.1.The Tigrigna Language

The Tigrinya and Tigrayan communities communicate in Tigrinya (ትግርኛ, commonly spelled Tigrigna), an Ethiopian Semitic language predominantly used in Eritrea and the Tigray Region of northern Ethiopia[10]. This language is also spoken by the global diaspora from these areas.

Tigrinya literature is significantly shaped by Ge'ez but has notable differences. It employs phrasal verbs and typically positions the main verb at the end of a sentence rather than at the start. This is particularly evident in expressions related to Christian life and Biblical references. Very recently, due to its standing in Ethiopian society and its straightforward form, Ge'ez served as a literary medium [[11], [12]].

The oldest known written record in Tigrinya is a 13th-century manuscript of local laws found in the Logosarda area of the Debub Region in Southern Eritrea. During the British occupation of Eritrea, the Ministry of Information published a weekly newspaper in Tigrinya for five cents, which was sold in five thousand copies per week. It was said to be the first of its sort at the time[13].

The oldest known Following Amharic, Oromo, and Somali, Tigrinya is the most widely spoken language in Eritrea and ranks as the fourth most spoken language in Ethiopia. Additionally, it is spoken by many immigrant communities in countries such as Sudan, Saudi Arabia, Israel, Germany, Italy, Sweden, the United Kingdom, Canada, and the United States, among others.

Tigrinya is one of the languages that the multicultural Special Broadcasting Service in Australia plays on public radio[14]. Dialects of Tigrinya vary in terms of phonetics, lexicon, and grammar. It doesn't seem like any dialect is acknowledged as the norm.

Consonant phonemes

The phonemes of Tigrinya are relatively typical for an Ethiopian Semitic language. It features the standard seven-vowel system, along with a series of ejective consonants. Tigrinya also retains two pharyngeal consonants and includes a voiceless velar or uvular ejective fricative.

These features help distinguish spoken Tigrinya from related languages like Amharic. However, they do not set it apart from Tigre, which has also preserved the pharyngeal consonants [15]. This contrasts with many other modern Ethiopian Semitic languages.

2.2.Tigrigna writing system

The script (the Tigrigna scripts used for this study is written at the last page in appendix A) used to write Tigrinya was initially designed for Ge`ez. The characters of the Ethiopic script are grouped based on both the consonant and the vowel, and each symbol represents a syllable consisting of a consonant and a vowel[16]. The seven vowels of Tigrinya are represented by columns in the table below, which shows them in the customary sequence. Once more, the consonants are given to the rows in the conventional sequence.

An abugida features an unmarked symbol for each consonant, typically followed by an inherent or canonical vowel. In the Ethiopic abugida, the canonical vowel is ä, represented in the first column of the table. The symbols in this column for those consonants are pronounced with the vowel a, similar to the fourth column, since the pharyngeal and glottal consonants in Tigrinya (and other Ethiopian Semitic languages) cannot be followed by this vowel.These superfluous symbols, which are displayed in the table with a dark gray backdrop, are becoming less and less common in Tigrinya[16]. The consonant+ə form (symbol in the sixth column) is used to signify a consonant when no subsequent vowel is present.

Each of the consonants ḥ›, s›, and s’› in Tigrinya features two rows of symbols, indicating a complexity in the phonetic system of the language. This complexity arises from the loss of certain distinctions that were present in Ge'ez, with at least one distinction becoming obsolete in modern Tigrinya. Specifically, in Eritrea, the differences between the sounds represented by s and s’ are no longer recognized in everyday usage. For example, both "he approached" (qärräbä) and "he was near" (qäräbä) are spelled ቀረበ, which does not indicate any phonetic variation. Fortunately, Tigrinya speakers typically do not find this lack of distinction problematic, as such minimal pairs are quite rare in the language. Context usually clarifies meaning, allowing effective communication despite the phonetic overlap, showcasing the adaptability of Tigrinya and its speakers.

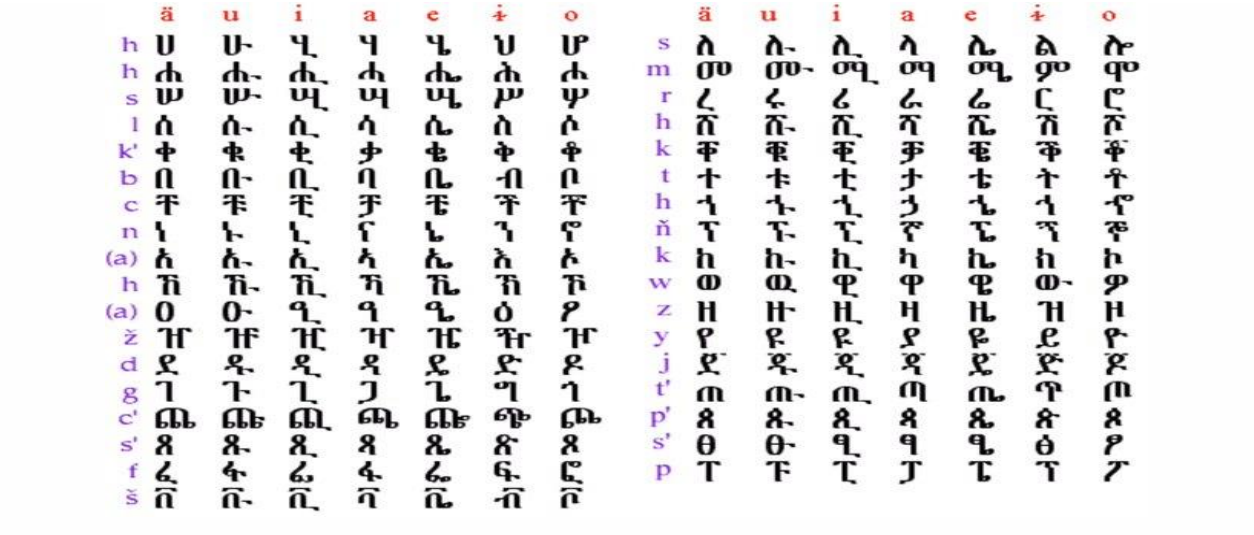


Figure 1. Ge`ez Alphabets

2.3.Natural Language Processing

Natural Language Processing (NLP) is a multidisciplinary field dedicated to enabling computers to understand, interpret, and generate human language. By integrating techniques from linguistics, computer science, and artificial intelligence, NLP serves as a bridge between human communication and machine comprehension. It facilitates a variety of tasks, including language translation, sentiment analysis, text classification, and speech recognition, leading to valuable applications in areas such as information retrieval, healthcare, and business intelligence. Central to NLP is the development of algorithms and models that empower computers to process and analyze natural language text effectively. Among its various applications, text classification stands out as a key component of NLP's capabilities.It involves automatically categorizing or classifying text documents into predefined categories or topics based on their content[17]. The description below is how NLP is used in text classification:

Preprocessing: Before classification, text data typically undergoes preprocessing, which includes several important steps. These steps involve removing punctuation, converting text to lowercase, eliminating stop words (common words such as "ን" or "እቲ" that lack significant meaning; a list of stop words for this study can be found in Appendix B), and performing tokenization. This preprocessing is essential for improving the quality and efficiency of the classification process.

Feature Extraction: NLP techniques are used by extracting relevant features from the text. These features capture the characteristics of the text that are informative for classification. Common techniques include the bag-of-words model, where each document is depicted as a vector that counts word frequencies or indicates the presence or absence of words. Another technique is TF-IDF (Term Frequency-Inverse Document Frequency), which assesses the significance of a word in a document compared to a broader set of documents

Text classification has a wide range of practical applications. Some examples include spam detection, which classifies emails as spam or non-spam. In sentiment analysis, text is classified as positive, negative, or neutral to gauge customer sentiment or public opinion. Topic classification involves categorizing news articles or social media posts into different topics or domains[17]. Intent recognition identifies the intent or purpose behind user queries in chatbots or virtual assistants. In document classification, organizing documents into categories for efficient retrieval and management[17].

2.4. Stance Detection and classification

In sociolinguistics, stance detection and classification focus on analyzing the writer's perspective as expressed in their writing. This involves connecting the writer's inherent viewpoint to three key elements: linguistic acts, social interactions, and aspects of human identity. By exploring these relationships, researchers can gain insights into how writers convey their opinions and attitudes through language.

Stance detection seeks to identify the writer's stance from the text. Adjectives, adverbs, and lexical components are commonly used along with other language qualities in stance detection[18].

The task of attitude identification involves classifying a text input and target combination. It identifies the author's stance by selecting a category label from the set: Favor, Against, or Neither. The aim may or may not be directly specified in the text. Neutral is sometimes added to the list of stance types [[4], [5],[19]].

Though they have certain similarities, sentiment analysis, and stance identification are not the same. Ascertain if a text is neutral, negative, or favorable. From the text, determine the speaker's point of view and the object of their opinion. Conversely, attitude identification requires systems to identify favorability toward a predefined target of interest. Often, neither the text's target of opinion nor its purpose is fully discussed[5],[19].

2.5. Stance detection according to target

position categorization is the problem of determining if a certain topic or entity has a position in favor of or against it. Therefore, to ascertain the attitude towards a preset goal, stance detection requires the presence of the target. Depending on the type of target being assessed, stance detection may be categorized into three groups[4], [5],[19].

2.5.1. Multi-related-targets stance detection

Understanding social media users' simultaneous orientation toward two or more targets for a single issue is the aim of multi-target stance identification. This kind of stance recognition is based on the fundamental idea that a person's stance on one target indicates how they feel about other related targets[20]. Multi-target stance detection is a classification issue where an input consisting of a

text passage and a series of associated targets is used. From this set of targets, the author's attitude is wanted as a category label. For every target, there are three possible stance classifications: favor, against, or neither. The classifications for each target may affect the classifications for the other targets[4].

Equation 1, represents a stance detection scenario, where the stance of a text T, given a topic or subject U and some group Gn, is modeled in terms of opposing or favoring political positions.

Stance (T/U, Gn) = {(FavorG1, Against Gn+1), (Favor Gn+1, Against G1)} *..... Equation (1)*

The variables given in Equation 1 is described as follows.

- **T**: This represents the text or document in which the stance is being analyzed. It can be any written content, such as an article, post, or statement.
- **U**: This denotes the topic or subject that the text T is addressing. It serves as the context in which the stance is evaluated.
- **G_n**: This indicates a specific group or entity that is relevant to the stance being analyzed. The subscript n can represent a specific instance or category of a group within a broader context.
- **G_1**: This represents another group or entity, distinct from G_n. It is involved in the comparison of stances.
- **G_{n+1}**: This denotes a group or entity that is being contrasted with G_n. It follows the same contextual numbering, indicating it is another group that may be favored or opposed.
- **Favor G_1 / Favor G_{n+1}**: These terms indicate that the stance expressed in text T is supportive of the respective group, suggesting a positive alignment or endorsement.
- **Against G_{n+1} / Against G_1**: These terms indicate that the stance expressed in text T is oppositional to the respective group, suggesting a negative alignment or disapproval.

2.5.2. Target-specific stance detection

By utilizing the characteristics of the social actor, the basic version of stance detection on social media may be developed. Thus, Text T, or user U, and Given Target G are the two main inputs in this type of stance identification. The dependent factor in the stance identification task is the analysis's goal. Stance identification on social media is commonly approached by assuming the stance just from the raw text, which simplifies the problem to textual entailment work[18].

Stance (T, U|G) = {Favor, Against, None} …… equation (2)

Target-specific attitude identification is the main strategy used in different stance research; even in benchmark datasets including many subjects, most published work on the dataset trained a

different model for each target separately[18], the target-specific stance detection for the target employed in this work is demonstrated in the examples below.

Target: TPLF Party

Comment: “ከምቲ አብ ስትራተጂአ ዝአወጀቶ ግብራዊ ሰላም ንምርግጋጽ ዓብይ ወፈያ ገይራ እያ።”

“It has been a big dedication ensuring practical peace as it is declared in its strategy.”

Stance: In favor

Target: TPLF Party

Comment: “እቲ ውድብ አብ ትልሚ ፖለቲካዊ ትሕዝትኡ ምንም ዓይነት ሰላም ናይምፍጥር ዛዕባ ዘለዎ አይኮነን።”

“The party has never been a peace strategy for the people of Tigray in its life, rather always chaos is its reflection”

Stance: Against

2.5.3. Claim-based stance detection

Claim-based analysis sometimes referred to as open-domain attitude identification, centers on a claim made in a news article rather than an explicit entity like those described above. Finding the main target statement in the speech sequence or supplied text is the first step in identifying the attitude. The main input to the claim-based attitude identification algorithm is claim (C), which might be the headline of the article or the post made by the report. Typically, the prediction label sets represent the statement as either true or false[18].

$$Stance\ (T, C) = \{Confirming, Denying, Observing\} \quad \text{Equation (3)}$$

2.6.Stance Detection using a deep learning approach

Deep learning approaches have been successfully applied to stance detection and classification and have achieved state-of-the-art performance in various natural language processing (NLP) tasks. Deep learning can be used for stance detection and classification by leveraging neural network architectures that can capture the semantic meaning and context of the text. Here is a common approach to detect using deep learning[1].

2.6.1. Using Convolutional Neural Networks (CNN) based Models

CNNs have shown promising results in stance detection and classification tasks. CNNs are widely used in computer vision tasks, but they have also been successfully applied to natural language processing tasks, including stance detection. In this research discussion, we explored the use of

[1] I. Augenstein, T. Rocktäschel, A. Vlachos, and K. Bontcheva, "Stance Detection with Bidirectional Conditional Encoding," *Proc. 2016 Conf. Empirical Methods in Natural Language Processing (EMNLP)*, Austin, TX, USA, 2016, pp. 876–885.

CNNs for stance detection and classification and discussed their advantages and challenges[20].

CNNs out perform at capturing local patterns and dependencies in data through the application of convolutional filters. In the context of text classification, the input to a CNN is typically a sequence of word embeddings or character embeddings. The convolutional filters slide over the input, extracting local features or n-gram representations. These features are then combined and processed by subsequent layers, such as pooling layers and fully connected layers, to make predictions[21].

One advantage of using CNNs for stance detection is their ability to capture local contextual information. By applying convolutional filters of various sizes, CNNs can detect meaningful patterns and combinations of words that are indicative of specific stances. For example, certain linguistic patterns or phrases may be strongly associated with a particular stance, and CNNs can learn to recognize and leverage these patterns effectively[20].

Additionally, CNNs can automatically learn hierarchical representations from raw text data. Lower-level filters capture local word combinations, while higher-level filters capture more abstract and global representation[22]. This hierarchical feature extraction allows CNNs to capture both fine-grained and high-level features, which can be beneficial for stance detection. It enables the model to learn representations of varying roughness, capturing both detailed linguistic signs and broader contextual information.

Furthermore, CNNs can handle variable-length input sequences, making them suitable for stance detection tasks where the length of the text can vary. By applying filters of different sizes and utilizing padding techniques, CNNs can process inputs of different lengths and extract relevant features[23].

Despite their advantages, CNNs for stance detection also face challenges. One challenge is the limited ability of CNNs to capture long-range dependencies in the text. Stance detection often requires understanding the overall context and discourse, which may involve capturing dependencies between distant words or sentences. While CNNs can capture local dependencies effectively, they may struggle to model long-range dependencies as effectively as other architectures like recurrent neural networks (RNNs) or transformer-based models[24].

Another challenge is the need for labeled training data. CNNs typically require a large amount of labeled data to learn meaningful representations and achieve good performance. However, acquiring labeled stance detection datasets can be time-consuming and expensive, as it often requires domain expertise and manual annotation. As a result, the availability of large-scale labeled datasets for stance detection can be limited.

To address these challenges, researchers have explored various approaches, such as combining

CNNs with other models (e.g., RNNs or transformers) to capture both local and global dependencies effectively. Additionally, transfer learning techniques, such as pre-training on large-scale general language tasks or domain adaptation, have been investigated to mitigate the data scarcity issue[21].

In conclusion, CNNs have shown promise in stance detection and classification tasks by capturing local patterns and hierarchical representations in text data. They can effectively capture local linguistic cues and leverage them for stance detection. However, challenges exist in modeling long-range dependencies and acquiring labeled training data. Ongoing research focuses on addressing these challenges and exploring hybrid architectures that combine CNNs with other models to enhance the performance of stance detection systems.

2.6.2. Recurrent Neural Networks (RNNs) based Models

Stance detection and classification, particularly using Recurrent Neural Networks (RNNs), is an active area of research in natural language processing (NLP). Stance detection aims to determine the attitude or perspective expressed towards a particular target or topic in a given text, while stance classification involves assigning predefined stance categories to the detected stances. In recent years, RNNs have shown promising results for stance detection and classification tasks due to their ability to capture sequential dependencies in text data[25].

One common approach for stance detection using RNNs is to represent text as sequences of word embedding. Word embedding are dense vector representations that capture semantic similarities between words. These embeddings are fed into an RNN model, such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU). The RNN processes the sequential information and learns to capture the context and dependencies between words. The final hidden state of the RNN is then used for stance detection or classification[26].

Researchers have explored different variations of RNN architectures for stance detection. For instance, Bidirectional RNNs incorporate both forward and backward information by processing the text in both directions. This allows the model to capture not only the preceding words but also the subsequent words, improving the understanding of the context. Attention mechanisms have also been employed to give more weight to relevant words or phrases in the text, enabling the model to focus on the most informative parts[25].

In addition to the architectural choices, the availability and size of labeled datasets play a crucial role in training accurate stance detection models. Researchers have created various annotated datasets specific to different domains or topics for training and evaluating stance detection models. However, labeled data for stance detection may be limited, leading to challenges in training deep learning models effectively. Transfer learning techniques, such as pre-training on large corpora or

leveraging external resources like pre-trained language models, have been explored to alleviate the data scarcity issue and improve performance[27].

Evaluation of stance detection models is typically done using metrics such as accuracy, precision, recall, and F1-score. Researchers have also investigated the impact of different factors on model performance, including the size of the training data, the choice of word embedding, the architecture of the RNN, and the presence of noise or bias in the labeled data.

While RNNs have shown success in stance detection and classification, there are ongoing research efforts to further enhance their performance. This includes exploring more advanced architectures, incorporating contextualized word representations (such as BERT or GPT), addressing bias and fairness issues, and adapting the models to handle multilingual or cross-domain settings[26].

In conclusion, recurrent neural networks have proven to be effective for stance detection and classification tasks. Their ability to capture sequential dependencies in text data makes them well-suited for modeling the context and understanding the stance expressed towards a target or topic. Ongoing research aims to improve the performance of these models and address various challenges in stance detection, ultimately advancing the field of natural language processing.

2.6.3. Transformer-based Models

Transformer-based models, such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-Trained Transformer), have achieved remarkable success in various NLP tasks, including stance detection[28]. These models rely on a self-attention mechanism to capture contextual information from the entire text. By pre-training on large amounts of text data and then fine-tuning on the specific task, these models can effectively capture the relationships between words. This allows them to achieve high performance on stance detection[29].

Transformer-based models have emerged as powerful architectures for various natural language processing (NLP) tasks, including stance detection and classification. The Transformer model, introduced by Vaswani[30], revolutionized the field by leveraging self-attention mechanisms. These mechanisms capture global dependencies between words in a text sequence. This architecture has been widely adopted and achieved state-of-the-art performance in many NLP tasks, including stance detection.

Stance detection using this model involves encoding the input text sequence into contextualized representations. This is done by employing multi-head self-attention mechanisms. The model attends to different parts of the input sequence to capture both local and global dependencies effectively. This ability is crucial for stance detection, as understanding the context plays a significant role in determining the attitude or perspective expressed towards a particular target or

topic[30].

One popular variant of the Transformer model used for stance detection is BERT (Bidirectional Encoder Representations from Transformers). BERT pretrains a large-scale Transformer model on a massive corpus to learn contextualized word representations. These pre-trained representations can then be fine-tuned on specific stance detection tasks using supervised learning. By leveraging large-scale pretraining, BERT-based models can effectively capture the semantics and context of the input text. This leads to improved performance in stance detection[31].

Researchers have explored different strategies for incorporating Transformer-based models into stance detection frameworks. For instance, BERT-based models can be used as feature extractors. In this case, the contextualized word representations generated by BERT are fed into additional layers or classifiers for stance detection. Alternatively, the entire BERT model can be fine-tuned for the stance detection task end-to-end. This allows the model to learn task-specific representations.

Moreover, researchers have investigated techniques to handle stance detection in a multi-label or multi-class setting. Stance classification involves assigning predefined stance categories to the detected stances. For multi-label classification, where multiple stances can be assigned to a single text, modifications to the model architecture or loss functions are applied. These modifications accommodate multiple labels and enable the model to handle cases where a text expresses multiple perspectives towards a target or topic[31].

Evaluation of Transformer-based models for stance detection is typically done using standard metrics such as accuracy, precision, recall, and F1-score. Researchers have also explored techniques to address challenges such as data scarcity, model bias, and domain adaptation when applying Transformer-based models for stance detection.

In summary, Transformer-based models, particularly BERT, have demonstrated significant advancements in stance detection and classification tasks. Their ability to capture global dependencies and contextual information has improved the understanding of the context and the identification of stances. Ongoing research focuses on fine-tuning strategies, handling multi-label classification, addressing bias and fairness issues, and adapting the models to specific domains or languages. These efforts contribute to the continuous improvement and application of Transformer-based models in stance detection research and applications.

2.6.4. Hybrid Approaches

Hybrid approaches for stance detection and classification combine multiple techniques or models to leverage their complementary strengths and improve overall performance[32]. These approaches often integrate both traditional feature-based methods and deep learning-based models

to capture different aspects of the stance detection task effectively.

One common hybrid approach involves combining rule-based or feature-based methods with deep learning models. Traditional feature-based methods rely on handcrafted features such as n-grams, syntactic patterns, or lexical resources to represent the text and identify stances. These features capture linguistic cues and domain-specific knowledge that can be useful for stance detection[33]. Deep learning models, on the other hand, outshine in capturing complex patterns and representations from raw text data.

In this hybrid approach, the traditional feature-based methods extract relevant features from the input text, which are then combined with the learned representations from deep learning models. For instance, the handcrafted features can be concatenated with the output of a recurrent neural network (RNN) or a transformer-based model. This combination allows the deep learning model to benefit from both the learned representations and the explicit linguistic features, leading to improved performance.

Another hybrid approach involves ensemble methods, where multiple models are trained independently and their predictions are combined to make the final decision. Ensemble methods can combine different types of models, such as traditional machine learning algorithms, deep learning models, or even rule-based systems. Each model in the ensemble may be trained using different architectures, features, or hyperparameters, providing diverse perspectives on the stance detection task[32].

The predictions of individual models can be combined using various techniques, such as majority voting, weighted voting, or stacking. Ensemble methods have been shown to enhance the robustness and generalization capability of the stance detection system by reducing the impact of individual model biases or weaknesses.

Furthermore, hybrid approaches can also incorporate domain-specific knowledge or external resources. For instance, domain-specific lexicons or sentiment dictionaries can be used to augment the features or embedding of the input text. These resources provide additional information about the sentiment or opinion expressed towards specific terms or entities, which can improve the accuracy of stance detection.

Evaluation of hybrid approaches for stance detection and classification involves assessing the performance of the combined models or techniques using standard evaluation metrics such as accuracy, precision, recall, and F1-score[34]. Researchers often compare the performance of hybrid approaches with individual methods to demonstrate the effectiveness of the combination.

In conclusion, hybrid approaches for stance detection and classification leverage the strengths of different techniques or models to improve overall performance. By combining traditional feature-

based methods with deep learning models or using ensemble methods, these approaches can enhance the accuracy and robustness of stance detection systems [32][33]. Additionally, incorporating domain-specific knowledge or external resources further augments the performance. Ongoing research focuses on exploring new combinations, refining integration strategies, and addressing challenges in hybrid approaches to advance the field of stance detection.

2.7.Feature Extraction Methods in Stance Detection

Feature extraction methods play a crucial role in stance detection and classification tasks by transforming raw text data into meaningful representations that capture relevant information for identifying stances. These methods aim to capture linguistic cues, syntactic patterns, or semantic information that can discriminate between different stances expressed in the text[35].

2.7.1. N-gram

One common feature extraction method is based on n-grams, which are contiguous sequences of n words. By considering different n-gram sizes (e.g., unigrams, bigrams, trigrams), these methods capture local word interactions and can identify specific word patterns or combinations that are indicative of certain stances. N-gram features can be used directly as binary indicators, frequency counts, or transformed using techniques such as term frequency-inverse document frequency (TF-IDF) to give more weight to informative n-grams[36].

2.7.2. Syntactic patterns or grammatical structures

Another feature extraction method is based on syntactic patterns or grammatical structures. These methods leverage syntactic parsers or dependency parsers to extract syntactic relationships between words in the text [37]. Features such as the presence of specific syntactic patterns, the position of words in the parse tree, or the dependency relations between words can provide valuable information for stance detection. These features can be represented as binary indicators or transformed into numerical representations using techniques like one-hot encoding or word embeddings.

2.7.3. Lexical resources and sentiment analysis

Lexical resources and sentiment analysis techniques can also be employed as feature extraction methods for stance detection. Sentiment lexicons or emotion dictionaries contain lists of words associated with specific sentiments or emotions[31]. By identifying the presence of words from these lexicons in the text, it is possible to capture the sentiment or emotional tone expressed towards a target or topic. Sentiment analysis techniques, such as the calculation of sentiment scores, can also be used as features to quantify the overall sentiment polarity of the text.

In addition to these traditional feature extraction methods, advanced techniques based on deep learning models have gained popularity in recent years. Models such as recurrent neural networks

(RNNs) or transformer-based architectures (e.g., BERT) can capture contextual information and learn expressive representations from raw text data. These models can be used to extract embedded representations of words or entire sentences, which can then be used as features for stance detection. The advantage of deep learning-based feature extraction methods is their ability to capture complex patterns and dependencies in the data, leading to improved performance in stance detection tasks[37].

The choice of feature extraction methods depends on various factors, including the available resources, the characteristics of the dataset, and the specific requirements of the stance detection task. Researchers often explore different feature extraction techniques and compare their performance to identify the most effective methods. The evaluation of feature extraction methods is typically done by training classifiers using the extracted features and assessing their performance using standard evaluation metrics such as accuracy, precision, recall, and F1-score[31].

In summary, feature extraction methods are crucial for stance detection and classification tasks as they transform raw text data into meaningful representations. N-grams, syntactic patterns, lexical resources, and deep learning-based models are among the commonly used techniques. The choice of feature extraction methods depends on the characteristics of the data and the specific requirements of the task. Ongoing research focuses on exploring new feature extraction techniques, combining different methods, and adapting them to specific domains or languages to advance the field of stance detection[38].

2.8.Stance detection applications

Stance detection applications companies on technologies that aim to determine the position or stance expressed in a given text or statement. Stance detection can be applied to various domains, including social media analysis, political discourse, customer feedback analysis, and more [4][5].

2.8.1. Social Media Monitoring

Stance detection can be used to analyze social media posts and comments to understand public sentiment toward specific topics, products, or events. For example, a company might use stance detection to monitor social media conversations about their brand and identify whether the sentiment expressed is positive, negative, or neutral[38].

2.8.2. Political Analysis

Stance detection can be applied in political analysis to understand the positions expressed by politicians or political parties on various issues. It can be used to track public opinion and sentiment toward specific policies or to analyze political debates and speeches[39].

2.8.3. Fake News Detection

Stance detection can play a role in identifying misinformation and fake news. Analyzing the stance expressed in an article or post can determine whether the information is biased, misleading, or based on false premises[40].

2.8.4. Customer Feedback Analysis

Companies can use stance detection to analyze customer feedback and reviews. It can help identify customers' positive or negative sentiments towards products or services, allowing companies to gain insights and make improvements accordingly[39][40].

2.8.5. Opinion Mining

Stance detection can be used in opinion mining to determine the stance expressed by individuals or groups on various topics. It can be valuable in understanding public opinion trends, identifying emerging issues, or predicting future preferences[38],[39].

2.9. Experimental research review

Twitter sentiment analysis is the subject of SemEval-2013 Task 2, which is presented in the work by S. Mohammad et al[41]. A wealth of information for comprehending social media emotions, the dataset consists of annotated tweets divided into three sentiment categories: positive, negative, and neutral. The writers used a variety of machine learning techniques, such as ensemble methods and support vector machines, to successfully categorize the feelings. Important discoveries show that emoticons and linguistic characteristics greatly improve sentiment categorization accuracy. However, the study finds shortcomings in handling the subtleties of tweet context and the influence of irony and sarcasm on emotion, underscoring the need for more reliable models that can account for these intricacies.

Augenstein et al[42]. address the challenge of detecting stances in political speeches, specifically focusing on speeches from the UK Parliament. The authors make a significant contribution by presenting a well-structured dataset and proposing a machine-learning approach that utilizes various features, such as n-grams, part-of-speech tags, and sentiment analysis. One notable strength of the study is its exploration of domain adaptation techniques, which have the potential to improve the generalization of stance detection models across different political contexts. However, while the study introduces innovative methods, it leaves room for further investigation into the scalability of these techniques across larger and more diverse datasets. Overall, this work adds valuable insights into the development of more robust stance detection models, though it could benefit from addressing the limitations in applying domain adaptation beyond the specific context of UK political speeches.

Nguyen et al[43] tackle the complex task of stance detection in online debates, where numerous users contribute opinions on various topics. The authors propose a novel approach using a

hierarchical attention network (HAN), which effectively captures the interactions between users and their respective contributions to the overall stance expressed. A key strength of this work lies in its ability to model the hierarchical structure of debates, distinguishing between individual posts and the collective stance that emerges from the discussion. The use of attention mechanisms allows the model to focus on the most relevant parts of user interactions, enhancing the overall performance instance classification. However, the study primarily focuses on debates within a specific context, leaving open questions about the generalizability of the proposed model across different domains or more diverse debate structures. Additionally, while the hierarchical attention network shows promise, the complexity of the model may raise concerns regarding computational efficiency when applied to larger datasets or real-time applications. Despite these limitations, this work contributes significantly to the field by presenting an advanced method for capturing stances in multi-user debates, opening avenues for further research in domain adaptation and efficiency improvements.

Kiritchenko et al[44] present a detailed investigation into the stance classification of political tweets, leveraging a supervised learning approach that combines lexical, syntactic, and semantic features. The inclusion of tweet-specific features, such as hashtags and emoticons, is a notable aspect of the research, reflecting the unique linguistic characteristics of Twitter. This aspect strengthens the study by demonstrating how social media-specific elements contribute to the performance of stance detection models. However, one of the potential limitations of the study is its reliance on a single platform such as Twitter which may hinder the applicability of the findings across other social media networks with different communication norms. Additionally, while the proposed method shows promising results, it could benefit from a deeper exploration of how these features interact with one another or how domain adaptation techniques might improve performance in different political contexts. Despite these limitations, the research offers valuable insights into how supervised learning models can be tailored for stance classification in short, informal texts such as tweets, setting a solid foundation for future work in this area.

Zadeh et al [45].investigate the application of deep learning techniques for stance detection in political tweets, introducing an ensemble approach that leverages multiple models, including convolutional neural networks (CNNs) and long short-term memory (LSTM) networks. This combination enables the model to capture both local contextual information, such as word-level features through CNNs, and global context via LSTMs, which track the sequence of words over time. The ensemble approach is a notable strength of the study, as it blends the strengths of different architectures to improve the overall stance detection performance. However, one limitation of the study is the potential computational complexity introduced by combining multiple

deep learning models, which may hinder scalability for large datasets or real-time applications. Additionally, while the proposed method shows improved accuracy, the research could further explore how external information, such as user profiles or tweet metadata, might enhance the stance classification process. Despite these challenges, the study makes a significant contribution by showcasing the potential of deep learning ensembles in tackling stance detection in politically charged social media environments, paving the way for more robust models in this domain.

In another work[46], a support vector machine (SVM)-based learning system is developed, functioning in two phases. In the first phase, the system assesses whether the tweets express subjective or objective positions. In the second phase, the tweets are further categorized into negative, positive, or neutral stances. Various models were employed to build these stance detection systems, including unigram and feature-based models. The performance, as measured by F-scores, reached 72.4% for the unigram model and 68% for the feature-based model, demonstrating the effectiveness of these techniques. Additionally, the authors proposed a tree kernel-based approach for tweet classification, leveraging tree representations to better capture the structure of tweets. Their kernel and senti-features model achieved a maximum F-score of 60.83%, indicating the model's comparative performance, though it leaves room for further improvement in handling tweet classification accuracy.

Wang et al.[47] present a study on political stance detection in news articles, introducing a supervised learning approach that incorporates a range of lexical, syntactic, and semantic features. A key strength of this work is its focus on domain adaptation techniques, aimed at improving the generalization of stance detection models across various news domains. This approach addresses the challenge of models often overfitting to specific datasets, making them less effective when applied to articles from different sources. By employing domain adaptation, the authors enhance the model’s ability to generalize, making it more robust for stance detection across diverse political contexts. However, while the study demonstrates notable improvements in model adaptability, it may benefit from further exploration of how more complex deep learning architectures could enhance performance compared to traditional supervised learning methods. Overall, this work contributes to the growing body of research on political stance detection by emphasizing cross-domain generalization, a critical issue for real-world applications.

The other work, Surafel Tadesse [1], focused on analyzing posts and comments from the Prosperity Party Facebook page. The study employed traditional machine learning techniques, including Support Vector Machine (SVM), Logistic Regression (LR), and Random Forest (RF). Using TF-IDF for feature extraction, the SVM model achieved an accuracy score of 0.82. However, the study did not explore other feature extraction techniques, nor did it utilize deep

learning approaches. Additionally, the dataset used was limited, which may affect the generalizability of the findings.

Lastly, Bewketu Molla [48], utilized web-scraped social media data and applied deep learning models, specifically Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory (BI-LSTM). The study achieved an F1-score of 86% for target identification, 65% for stance classification, and 50% for sentiment classification. Despite these results, the study did not explore other word embedding techniques, identify additional targets, or examine support and opposing stance labels from mixed stance tweets.

In this revised literature review, we critically analyze relevant studies in political stance detection, highlighting their methodologies and limitations to demonstrate the uniqueness of our research on Tigrigna texts. Mohammad et al. (2013) focused on political debates but struggled with nuances like sarcasm, which our study addresses by incorporating idiomatic expressions specific to Tigrigna. Augenstein et al. (2016) emphasized domain adaptation techniques yet did not explore scalability across diverse datasets; we overcome this by applying tailored feature extraction methods. Nguyen et al. (2018) introduced a hierarchical attention network but faced generalization challenges, which we address by focusing on the cultural context of Tigrigna discourse. Kiritchenko et al. (2014) investigated political tweets but were limited to Twitter, while our analysis includes Facebook comments for broader insights. Zadeh et al. (2017) utilized ensemble techniques, facing scalability issues, whereas our Transposed GRU model balances performance with efficiency. Surafel Tadesse (2023) achieved an accuracy of 82% using traditional methods, while we employ advanced deep learning techniques and feature extraction strategies like Word2Vec to enhance robustness. Lastly, Bewketu Molla (2021) applied CNNs and BI-LSTMs but did not explore alternative embeddings or multi-target labels, gaps our work fills by using a comprehensive dataset and innovative NLP techniques. This critical comparison establishes the significant contributions of our research to the field of political stance detection in low-resource languages.

Table 1. Summary of related works

Related Work	**Dataset**	**Approach**	**Key Findings**	**Research Gaps**
Mohammad, Saif, et al. (2013)[41]	U.S. Senate debates	Supervised learning using various features	Introduces a dataset of political debates and explores different features and	Limited evaluation of the generalization of the models to new debates and the impact of bias in the dataset

Related Work	Dataset	Approach	Key Findings	Research Gaps
			algorithms for stance classification	
Augenstein, Isabelle et al. (2016) [48]	UK Parliament speeches	Machine learning with lexical, syntactic, and semantic features	Investigates the detection of stance in political speeches and the impact of domain adaptation techniques	Limited exploration of cross-domain generalization and the influence of speaker-specific characteristics in stance detection
Nguyen, Thien Huu et al. (2018) [43]	Online debate platform	Hierarchical attention network (HAN)	Addresses stance detection in online debates and proposes a hierarchical attention network to capture user interactions	Limited investigation of the performance of the proposed HAN approach in the presence of noisy and unstructured user-generated content
Kiritchenko, Svetlana et al. (2014) [44]	Political tweets	Supervised learning with lexical, syntactic, and semantic features	Focuses on stance classification of political tweets and explores tweet-specific features such as hashtags and emoticons	Limited analysis of the impact of contextual information in tweets and the generalization of models to different political domains
Zadeh, Amir et al. (2017) [45]	Political tweets	Ensemble approach with deep learning models	Explores the application of deep learning techniques, including CNNs and LSTMs, for stance detection in political tweets	Limited exploration of model interpretability and the impact of adversarial examples in deep learning-based stance detection
Luciano Barbosa (2010)[46]	Twitter from Biased and Noisy Data	SVM-based learning, unigram, and feature-based models	The F-scores obtained for unigram and feature-based models were 72.4 and 68 %	does not delve deeply into the impact of bias on sentiment detection models, does not extensively explore techniques to handle noisy and unstructured content, does not investigate the generalization of models to different domains or languages
Wang, Wei et al. (2017)[47]	News articles	Supervised learning with lexical,	Investigates political stance detection in news articles and	Limited analysis of the impact of news source bias on stance detection and the

Related Work	Dataset	Approach	Key Findings	Research Gaps
		syntactic, and semantic features	explores domain adaptation techniques for improved model generalization	scalability of domain adaptation techniques to large-scale news corpora
Surafel Tadesse (2023)[1]	Prosperity party Facebook page posts and comments	SVM, LR and RF	Achieved accuracy score of **0.82** using TF-IDF feature extraction and SVM**.** based on these results,	Not checked using various feature extraction technique, not checked using deep learning approach and not utilizing enough datasets
Bew Ketu Molla (2021)[48]	social media by employing web-scraping	CNN, BI-LSTM	Model achieves F1-score of 86% for task 1 (target identification), 65% for task 2 (stance classification) and 50% for task 3 (sentiment classification)	The study did not explore other word embedding techniques, identify additional targets, or examine support and oppose stance labels from mixed stance tweets.

2.9.1. Research Gap Analysis

The related works on political stance detection and classification encompass a variety of approaches and datasets. Mohammad[41] introduces a dataset of political debates and explores various features and algorithms for stance classification, emphasizing the necessity for further evaluation of model generalization and potential dataset bias. Augenstein[49] focuses on stance detection in political speeches, highlighting the importance of domain adaptation techniques and the need for cross-domain generalization, along with consideration of speaker-specific characteristics. Nguyen[43] addresses stance detection in online debates through a hierarchical attention network but notes that further investigation is needed to effectively handle noisy and unstructured user-generated content. Kiritchenko[44] concentrates on the stance classification of political tweets, examining tweet-specific features while considering generalization across different political domains. Zadeh[45] explores deep learning techniques for stance detection in tweets, calling for additional research on model interpretability and the impact of adversarial examples. Wang[47] investigates political stance detection in news articles and the application of domain adaptation techniques, suggesting that more analysis is required regarding news source bias and the scalability of domain adaptation to large-scale news corpora. The other work[1], lacks exploration of advanced feature extraction techniques and deep learning models, as well as a more extensive dataset. The last paper[48] did not investigate alternative word embeddings, expand

target identification, or analyze mixed stance tweets. Collectively, these works contribute significantly to the field. They provide valuable datasets and explore various features and algorithms. Additionally, they identify critical research gaps that warrant further investigation. These gaps include model generalization, dataset bias, contextual information, interpretability, and the scalability of domain adaptation techniques. This research utilizes both machine learning and deep learning approaches.

CHAPTER THREE

3. RESEARCH METHODOLOGY

3.1.Introduction

This chapter outlines the proposed approach for detecting and classifying political stances through deep learning techniques. In this research, we created a new dataset of social media stances, which serves as the main resource for developing our political stance categorization method. We will detail the data collection methods, data analysis and preparation, design and experimental procedures, evaluation measures, and implementation strategies used in the study.

3.1.1. Research Design Methods

This study utilizes an experimental research design to systematically analyze political stances in Tigrigna text. This design is particularly effective for exploring quantitative research techniques, including data collection, cause-and-effect relationships, evaluation metrics, and statistical analysis to support or refute a theory. The experimental approach facilitates controlled testing and iterative refinement of models, ensuring that the results are both robust and reproducible. Specifically, the study involves systematically gathering and processing Tigrigna text data from social media to develop and evaluate deep learning models for political stance detection and classification. The research process is organized into distinct stages to maintain a scientifically valid approach.

The first step is data collection, where text data is gathered from the TPLF official Facebook page using Face pager. This involves extracting posts, comments and replies relevant to political discussions. Once collected, the data undergoes preparation and annotation, which includes text preprocessing, noise removal, and manual labelling of political stances into categories such as Pro, Neutral, and Anti.

Following annotation, the text is processed using morphological analysis, which is crucial for Tigrigna due to its complex linguistic structure. Tokenization, stemming, and handling of prefixes and suffixes are performed to extract meaningful root words. After morphological processing, feature extraction techniques are applied, where word embeddings (Fast Text or Word2Vec), TF-IDF, and n-gram representations are utilized to capture the contextual meaning of the text.

For model selection, deep learning algorithms such as LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Units) are chosen due to their effectiveness in sequential text classification. During this phase, hyperparameter tuning is performed to optimize model performance. The model training and architectural development process involves implementing Bi-LSTM/GRU networks that process input sequences and classify political stances.

Finally, the trained models undergo evaluation using standard performance metrics such as accuracy, precision, recall, and F1-score. Additionally, cross-validation is conducted to ensure the

generalizability and robustness of the models. The entire research workflow, depicted below, is visually represented in the flow diagram, which illustrates each step from data collection to model evaluation.

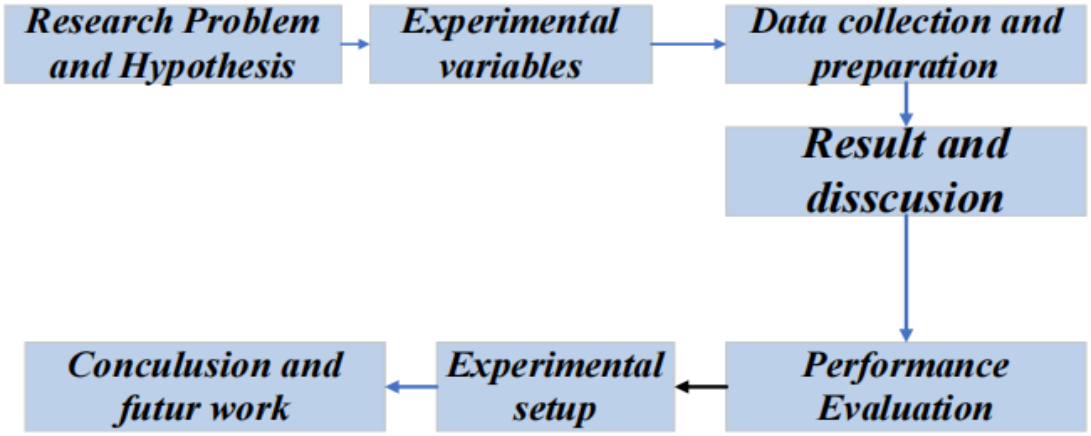


Figure 2. Research design procedures

3.2.Corpus Collection and preparation

The data for this study was gathered from the official Facebook page of the TPLF Party, which serves as the main platform for sharing posts and engaging with the party's community. We utilized the Face pager tool, a specialized service that allows users to retrieve publicly accessible data from social media platforms and other websites through web scraping and APIs. The data was collected on the time interval of 2021 GC to 2023 GC about the Tigray war. This method allows for the systematic gathering of relevant information, ensuring that our dataset is comprehensive and accurately reflects the party's online presence. The dataset collected is used for developing the word2vec embedding model and training.

3.2.1. Dataset Annotation

Data annotation is the process of enriching a document or dataset with additional information at various levels, facilitating the development of effective deep learning models. In this context, the annotation of an attitude dataset involves systematically labeling textual remarks to enhance the accuracy and reliability of stance detection models. The study chooses which comments to annotate using a straightforward random selection method. This method involves selecting a subset of comments in an unbiased and random manner to ensure the dataset is representative of the entire commentators. It also gives each page's filtered comments an equal chance of becoming an annotation. The annotation was carried out using the researcher's instruction guidelines. For the data annotation techniques, the spreadsheet tool Excel has been used.

3.2.1.1. Stance Annotation Guideline

The criteria that annotators were given to evaluate stance. We just take the comment into account when labeling for position toward a certain target in our annotation process. Take into account if the comment expressly supports or contradicts the aim.

Target: [Target entity]

Comment: [Facebook comment]

Question: Based on the remarks, which of the following statements about the user's attitude toward the target is most likely to be true?

It appears from the comment that the user supports the target. The reason for this might be any of the following: If the comment does not explicitly support the target but contains facts that lead us to believe that the person is in favor of the target, we can still conclude that the user supports the target. If the comment does not explicitly support the target but supports the position of someone who is aligned with the target, we can conclude that the user supports the target. If the comment does not support the target but supports, the position of someone who is not the target.

The comment shows that the user is against the objective.

Any of the following may be the reason for this: If the comment is clearly against the target; if it is opposed to something or someone who is aligned with the target, which leads us to believe that the user is against the target; If the comment is neutral but contains information that leads us to believe that the user is against the target; If we are unable to ascertain the user's position but the comment is endorsing the position of another.

Below is a sample data annotation process undergoing.

Annotation Format

1. Columns for the Dataset:
    - Comment ID: Unique identifier for each comment.
    - Comment Text: The actual text of the comment in Tigrigna.
    - Stance Label: The stance of the comment (e.g., Support, Oppose, Neutral).
    - Annotator Notes: Optional field for annotators to add notes or explanations.
2. Stance Categories:
    - Support: The comment expresses support for TPLF or its actions.
    - Oppose: The comment expresses opposition to TPLF or its actions.

Here’s an example of a small dataset with Tigrigna comments and their corresponding stance labels used during annotation:

Table 2. Analysis of Political Stances on TPLF Comments in Tigrigna

Comment ID	***Comment Text (Tigrigna)***	***Stance Label***	***Annotator Notes***
1	*"ህወሓት ንደቂ ህዝቢ ኤርትራ ኣዝዩ ጽቡቕ እዩ።"*	*In favor*	*Positive sentiment towards TPLF.*
2	*""ህወሓት ኣብ ልዕሊ ህዝቢ ትግራይ ሕማቅ ዝገበረ እዩ።"*	*Against*	*Criticism of TPLF's actions.*
3	*"ብሰንኪ ህወሓት ዝተወለዐ ኩናት ህዝቢ ትግራይ ኣዝዩ ተጎዲኡ እዩ።"*	*Against*	*General statement, no clear stance.*
4	*"ህወሓት ንህዝቢ ትግራይ ኣይሕግዝን እዩ።"*	*Against*	*Direct opposition to TPLF.*
5	*"ህወሓት ንህዝቢ ትግራይ ኣዝዩ ጽቡቕ እዩ።"*	*In favor*	*Positive sentiment toward TPLF.*

Annotation Steps

1. Define Annotation Guidelines:
    - Provide clear instructions to annotate the comments into Support or Oppose.
    - Include examples for each category to ensure consistency.
2. Pre-process the Data:
    - Remove irrelevant comments (e.g., spam, non-Tigrigna text).
    - Normalize the text (e.g., remove special characters, correct spelling errors).
3. Assign Annotators:
    - Use multiple annotators to ensure reliability. Each comment should be annotated by at least two annotators.
    - Resolve disagreements through discussion or a third annotator.
4. Annotate the Dataset:
    - Annotators read each comment and assign a stance label based on the guidelines.
    - Add notes if the stance is ambiguous or requires clarification.

Example Annotation Process

Comment 1:

Comment Text: "ህወሓት ንደቂ ህዝቢ ኤርትራ ኣዝዩ ጽቡቕ እዩ።"

Annotation:

- Stance Label: In favor
- Annotator Notes: The comment praises TPLF for its actions toward the Eritrean people.

Comment 2:

Comment Text: "ህወሓት ኣብ ልዕሊ ህዝቢ ትግራይ ሕማቅ ዝገበረ እዩ።"

Annotation:

- Stance Label: Against
- Annotator Notes: The comment criticizes TPLF for its actions against the people of Tigray.

Comment 3:

Comment Text: " ብሰንኪ ህወሓት ዝተወለዐ ኩናት ህዝቢ ትግራይ ኣዝዩ ተንዲኡ እዩ።"

Annotation:

- Stance Label: Against
- Annotator Notes: The comment criticizes TPLF for being a reason to start the war as damaged the people of Tigray

3.3.Data Preprocessing

Data preparation is the most crucial stage in stance detection and classification, as many studies have discussed. Since it gets the data ready for the model, it is the first and most crucial stage in creating a learning model. It enhances the quality of the dataset. Since the real-world data is messy and contains noise, it cannot be used directly by the model. It is inappropriate for the model. After annotation, the datasets need to undergo data preprocessing to make the fully cleaned data suitable for training the model. Before creating the deep learning model, data pretreatment techniques are applied. Several data preparation approaches, including tokenization, normalization, stop word removal, and the removal of extraneous items and punctuation, were employed in the study.

3.3.1. Data Cleaning

The technique of eliminating noise from a dataset such as punctuation to improve the model's comprehension is known as data cleaning. Before stance identification, the extracted data must be cleaned as part of the text preparation process. 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 name, location, and number of likes of a comment, comments that are irrelevant to the subject of the study, comments made in languages other than Tigrigna, numbers removed, whitespace stripped, punctuation removed, and stop words removed. The most often used stop-words in Tigrigna texts were eliminated from the textual dataset: ን (And), ናብ (to), ናይ (of), ኣነ (I), ውሽጢ (in), ንሱ (he), ንሳ (she), ናተይ (mine), ንሶም (they), ንሕና (we). The Tigrigna language may utilize numerals [0–9] (Numerals listed in appendix 3)and has a numbering system that looks like this: [፩, ፡-, ፫, ፬, ፭, ፮, ፯, ፱…]. Determining whether those numbers were exits was thus the first task of this investigation. If they are there, take them out of the text. In addition, the duties on the following list are completed when cleaning.

Data Cleaning Algorithm (appear in appendix D)

Input Sentence (sequence of characters)

Output Filtered Sentences

Process

I. Go over each word in sentences.

II. Divided among tokens

III. Replace the token if it is a stop word, number, punctuation, or an unimportant Unicode character.

IV. Join the words to create the initial phrases.

V. Restore clear sentences

3.3.2. Data Tokenization

Tokenization is the most basic operation performed on text data when working with it. The patterns must be tokenized into tokens using spaces during the text preprocessing step. Tokenization is applied to the input text based on the Tigrigna language characteristics used to retrieve the documents.

Example፡ “ብሔር ተኮር ዝኮነ ጽልኢ ዘረባ ጠጠው ክብል አለዎ።”, which mean “race-based heat speech should be banned?”. This sentence is then tokenized as ‘ብሔር’, ‘ተኮር’, ‘ዝኮነ’, ጽልኢ’, ‘ዘረባ’, ‘ጠጠው’, ‘ክብል’, አለዎ’, ‘?’As deep learning models do not work directly with text data, the fully cleaned data must be converted into numerical data using text feature extraction or word embedding methods.

Tokenization Algorithm

Input Words (sequence of characters)

Output Words

Process

I. Go over each sentence in the list

II. Next, break them up into individual words.

III. Word list return

3.3.3. Data Normalization

A group of related activities called "text normalization" are intended to level the playing field for all text. The Tigrigna text is normalized through the usage of the language's writing system. This is not the case with the Ethiopian Tigrigna language. Consequently, the Tigrigna language normalizer enlarges Tigrigna language short forms that are divided by either period (ር.ምምሕዳር) or slash (ር/ምምሕዳር) to indicate ርእሰ ምምሕዳር 'head of state'. This way, the results of a search utilizing ር/ምምሕዳር (ር.ምምሕዳር) and ርእሰ ምምሕዳር would be identical in quantity and kind.

One character representation for the same sound is required to remove such differences while processing the Tigrigna text; this representation was utilized for the text that was part of the experiment.

Normalization Algorithm

Input Words

Output Normalized Characters

Process

I. Go over each word.

II. Dividing the text into characters

III. Replace "ጎ" with "ሀ" if the character is ጎ.

IV. If the character is ሠ, then substitute ሰ; if the character is ዓ, ዐ, then substitute;

V. if the character is ፅ, then substitute ጸ.

VI. Come to an end if

VII. Give the normalized phrase

3.4.Morphological Processing Techniques

Vowel and consonant combinations are represented by a single sign, the alphabet because the Tigrigna language is syllabic[50],[51]. Tigrigna displays the root and stem pattern morphological phenomena as a result of its rich morphology. The process of identifying a word's constituent morphemes is known as morphological processing. We employ HORNMORPHO[52], a Python software, to do this work. It generates words and analyzes morphology in three languages of the Horn of Africa: Tigrigna (ትግርኛ), Amharic (አማርኛ), and Oromo (Afaan Oromoo, Oromiffa). In other words, it breaks down words into their component morphemes, or meaningful bits, and then creates new words based on a given root, stem, and grammatical structure representation.

Morphological Processing Algorithm

Input Words

Output Root word

Process

I. Go over each word.

II. Enter Horn Morphology with the term

III. From the segmented word produced by Horn morpho, choose the root word.

IV. Give the word back.

3.5.Feature Extraction Techniques

After the text has been cleaned up by eliminating extraneous symbols, punctuation, and text normalization, the next stage is feature extraction. deep learning algorithms are not able to operate directly on unprocessed text when working with textual input. Therefore, the text should be transformed into a feature matrix (or vector) using feature extraction algorithms. Preprocessed text is the input for the automated feature extraction procedure. Large amounts of labeled data are entered; before being utilized for training, the data is tokenized, preprocessed, and features are extracted. At this point, the dataset is converted into a numerical vector[2].

Word2vec

This kind of word embedding technique turns each word in a text into a vector that expresses the word's semantic meaning. One use of neural networks for unsupervised learning is Word2vec. It consists of a fully connected layer and a projection layer that are trained by backpropagation and stochastic gradient descent. The word in the context of an n-gram is transformed into continuous vectors by the projection layer. Words that occur simultaneously or repeatedly in the context of an N-gram are more likely to be activated by the same weight, which leads to a correlation between words.

For the proposed, the research chose and applied word2vec algorithms to translate text input into numerical values. The word2vec approach scans the whole corpus and creates vectors by first identifying the words that the target word appears with most frequently. This reveals the semantic proximity of the words to one another[38]. The word2vec technique is used in neural network models to learn word connections from a big text sample. Two architectural models may be utilized in word2vec: Skip-Gram and Continuous Bag-of-Word (CBOW).

The CBOW model word2vec predicts the target word only inside a window and utilizes words that come before and after it.

The window-limited word is used by the SKIP-GRAM model to anticipate words that come before and after it. To obtain input and target words, a window is employed as a kernel. The window is moved from the start of the text to the finish. As an example, when the window size is given at 2, then word2vec will consider 2 words in before and 2 words after a word associated with it[53].

Word2Vec Algorithm

Input Words

Output Set of vectors

[2] J. Brownlee, "A Gentle Introduction to Vectorization of Text for Deep Learning," *Machine Learning Mastery*, Aug. 7, 2017. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-vectorization-text-deep-learning/

Process

I. The comments will be split down to the word level, and word-level morphological processing will be carried out
II. One-hot vectors representing the word will be provided; the length of the vectors will be comparable to the vocabulary length. The vector is all zeros, except the index, which is allocated one, and represents the word we wish to represent.
III. The word embedding serves as the weight for the hidden layer that the single hot encoded vector passes through. Two edges exist between nodes (words) only if their associated vocabulary co-occurs in a window of length K, where K is the window's size. The weights are changed by minimizing the loss function.
IV. Produces the potential key terms from the lexicon.

3.6.Algorithm selection

Many cutting-edge classification algorithms, such as deep learning and machine learning models, are available for stance categorization. However, LSTM and its version GRU were shown to be the most successful for various datasets in the majority of research on stance identification using RNN models[18].

3.7.Classification Tasks

The comments are a dataset used in the investigation. We employed our data to gradually improve our model's ability to forecast whether the supplied comments are in support of or opposition to the specified target during the training phase. For training (in favor, against), we need to provide the deep learning model with the feature-extracted comment data together with their labels. After that, the program tries to understand the patterns required to classify the remark to our target (output). Next, unknown data will be given into the model to help it predict the required label. Hidden remarks will be encoded as features, which the machine learning model will use to categorize.

3.7.1. Training set

The real dataset that was used to train the model is known as the training set. The model watches and learns from this data to predict the outcome or make the best judgments. The primary training data in this study came from Facebook, which was then sorted and preprocessed to make sure the model worked as it should. The type of training data has a big impact on how well the model can generalize. The higher the ability and variety of the training data, the better the model will function.

3.7.2. Validation Set

The model's hyperparameters are adjusted using the validation set, which is thought of as a part of the model training process. This data is just used for evaluation by the model, which does not learn

from it, allowing an impartial and objective assessment. To minimize bias and variance, the validation dataset also be utilized for regression by stopping model training when the validation dataset's loss is greater than the training dataset's loss. The data ranges from 10% to 20%.

3.7.3. Testing set

This dataset is distinct from the training set but has some similarities with it in terms of the probability distribution of the classes. It is only utilized once the model has completed training. A well-structured dataset with data from several situations that the model is likely to face in practice is sometimes referred to as a testing set. Often using the validation and testing set together as a testing set is not considered best practice. If the model's accuracy on training data exceeds that on testing data, it is said to have been overfitted.

3.8. Evaluation Metrics

Evaluation measures that are commonly used are accuracy, recall, and F-score (or F-measure). Precision (P) and recall (R) may be used to calculate the F-score, a combination measure that allows you to give these two measures various weights.

The two-way stance identification process also frequently uses this F-score assessment metric (in one of the two classes: (Favor or against,). The most popular form of the F-score is determined by taking the macro-average of the F-scores for the categories that are classified as favorable and negative.

The F1 score for the "In Favor" class would be calculated as:

$$F1\_In\_Favor = \frac{2 * (Precision\_In\_Favor * Recall\_In\_Favor)}{(Precision\_In\_Favor + Recall\_In\_Favor)} \quad \text{..... equation (4)}$$

Similarly, the F1-score for the "Against" class would be:

$$F1\_In\_Against = \frac{2 * (Precision\_In\_Against * Recall\_In\_Against)}{(Precision\_In\_Against + Recall\_In\_Against)} \quad \text{..... equation (5)}$$

The overall F1 score for the stance classification task can be calculated as the macro average of the individual F1 scores for each class:

Accuracy

Another statistic for stance detection is Accuracy, which is determined as follows:

Accuracy= Correct Classification / All classifications

$$Accuracy = (TP + TN) / (TP + TN + FP + FN \text{..... Equation (6)}$$

Confusion Matrix

An accurate and inaccurate prediction amount produced by a classifier is compiled into a table called a confusion matrix. It is used to assess the performance of a classification model. A classification model's efficacy may be evaluated by calculating performance metrics such as accuracy, precision, recall, and F1-score [[54], [55]].

For the first two prediction classes of the classifiers, the matrix is a 2*2 table; for the following three classes, it is a 3*3 table, and so on. The projected values and actual values, as well as the total number of forecasts, make up the two dimensions of the matrix. The actual values are the actual values for the given data; the anticipated values are those that the model predicts. A 2*2 matrix was used to represent the confusion matrix because there were only two classes in this study [[54], [55]].

Table 3. Confusion matrix for binary classification problem

Predicted / **Actual**	**In favor**	**Against**
In favor	TP	FP
Against	FN	TN

True Negative: The actual value was also against the forecast made by the model.

True Positive: Both the actual number and the model's prediction were accurate.

False Negative: Also known as Type-II error, this occurs when the model predicts something negative but the actual result is positive.

False Positive: Although the model predicted a positive result, the actual value was negative. Another name for it is a Type-I error.

The following work flow diagram summarizes the idea described theoretically.

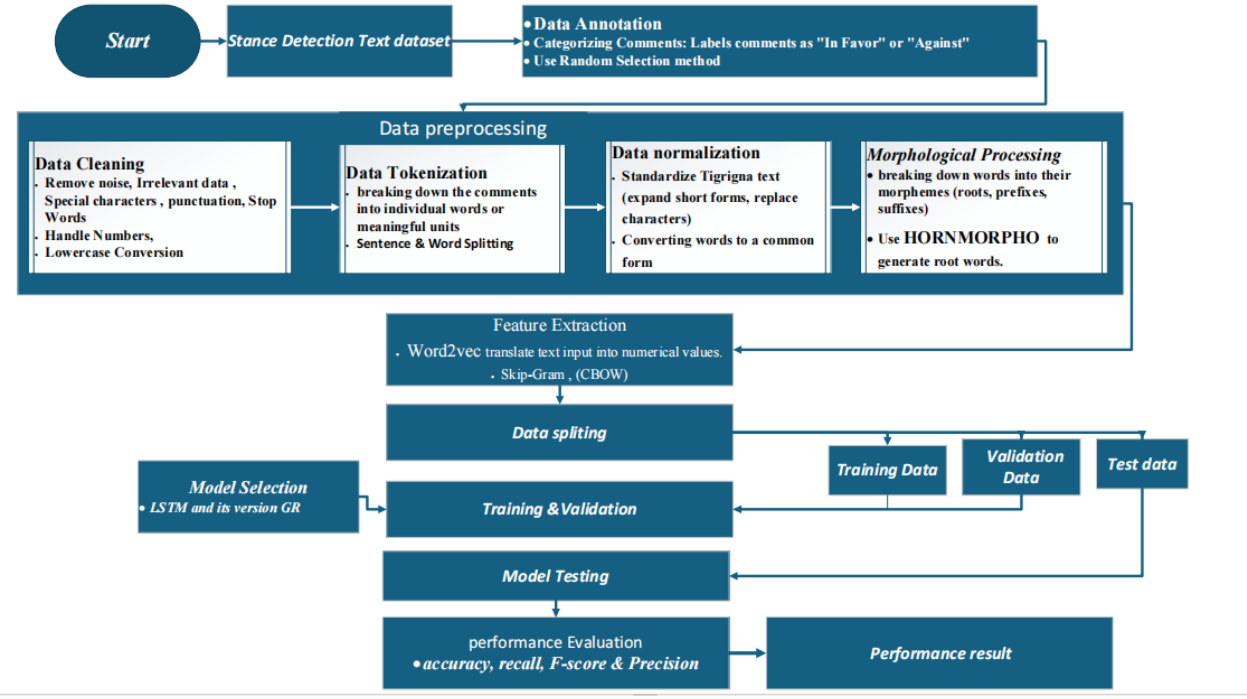


Figure 3. work flow diagram

3.9. Software Development Framework Tools

The study assesses the model's context as well as the implementation environment before deciding to develop the research. Python is our chosen programming language for designing, implementing, and testing the model. There are numerous tools, but Python is interactive with its accessible packages, and it's not a tough language to learn. The Natural Language Toolkit (NLTK) is a well-known Python library for working with natural language. Tokenization, stemming, lemmatization, tagging, word count, and other features are available in this library. NLTK is designed to support natural language processing and closely related areas such as machine learning, artificial intelligence, information retrieval, and linguistics. In this study, the NLTK library is used to work with and preprocess the datasets. The first is Keras[56], which allows for quick experimentation with the overall purpose of offering a standard and easy-to-use interface for a variety of deep learning frameworks such as TensorFlow[57]. Theano is a program that allows us to efficiently define data, optimize, and evaluate mathematical formulas involving multi-dimensional arrays, among other things. On top of the TensorFlow library, the conversational bot employs Keras. Pandas[58] is another useful package, which efficiently provides data structures and operations for handling numerical tables and time series. It comes in handy when addressing monumental amounts of data. The sci-kit-learn library[59], which was intended to function alongside the Python programming language numerical and scientific libraries, NumPy and SciPy, is tiny (compared to this implementation) but important. This is used for automatic utilities like partitioning a dataset into train, validation and test sets in this context. The proposed approach is implemented with the

Anaconda application version 1.9.7 with 64-bit support, which is used to install the latest version of Python together with all its numerous modules and IDEs. Additionally, the Jupyter Notebook is the most widely used and convenient IDE for working with Python among AI and deep learning researchers. Colab, also known as Collaboratory, is a free Jupyter notebook that runs on the cloud and stores data on Google Drive. It allows the users to run deep learning code that takes a long time within a short time by allowing users to use GPU and TPU. The study used this web application for the visualization of data, processing data, writing the code, and running the code. Finally, Notepad++ is a free source code editor that is used in this work to prepare a Tigrigna text dataset. It is a simple text editor to prepare datasets and save them with a JSON file extension. The dataset is prepared in the form of JSON format using the Notepad++ code editor.

CHAPTER FOUR

4. EXPERIMENTAL SETUP, IMPLEMENTATION AND RESULTS

4.1. Introduction

This study employs a design science methodology. In this chapter, we apply the strategy discussed in Chapter 3 to the suggested solution for Tigrigna Political Stance detection using deep learning algorithms. We also identify the best combination of learning algorithm and feature extraction for the final models to build the artifacts. Additionally, we used datasets on political stances that include binary labels (in favor or against).

4.2. Preparing Dataset for model creation and validation

Facebook is the most widely utilized medium. News stories can be shared on Facebook pages to reach a larger audience. The corpus was gathered manually and includes comments left on posts published from 2021 to 2023 on the official Facebook page of the TPLF party. During the preprocessing stages of this study, 175,922 comments were extracted and 172,942 were used after normalization. The data is collected via the Face pager tool as depicted below.

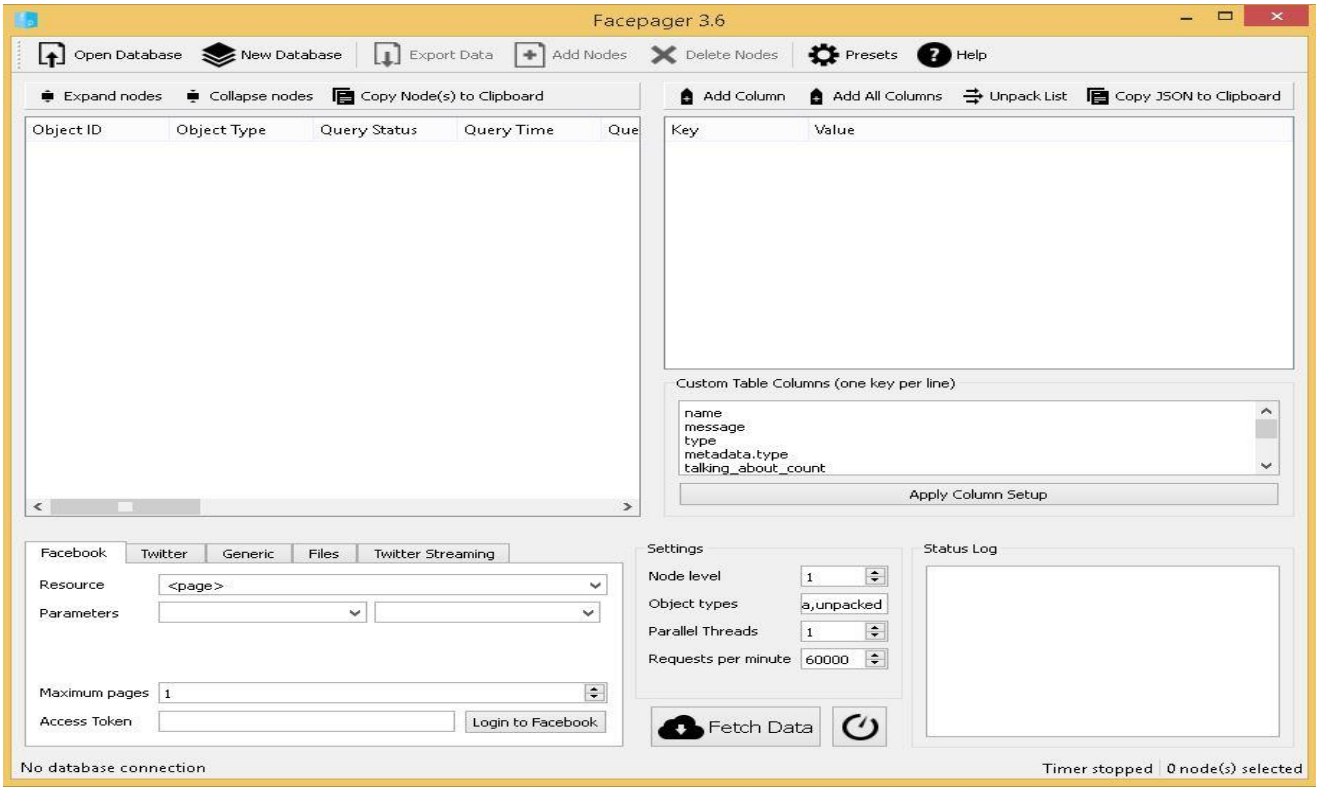


Figure 4. Face pager main page

4.2.1. Defining our Label

The process of labeling and adding target variables to training data so that a deep learning model knows what prediction and classification it is supposed to make is known as data annotation. This is a stage in the process of preparing data for supervised learning. "In favor" and "against" were the labels used for the experiments in this investigation. One indicated support for the offered position, while zero indicated opposition. In this experiment, remarks that are "against" the specified target are denoted as "Against," while comments that are "in favor" of the target are denoted as "in favor."

4.2.2. Manual Labeling

The labeling of the remarks for experimental purposes is the focus of the experiment. Using the annotation standards covered in Chapter Three, users manually classify all 172,942 comments into predetermined groups that are either in favor of or against the proposal. This stage is important since we use supervised learning methods, which require the system to learn from predefined labels first, followed by determining the relationship between the labels X (the comments) and Y (the posture). We were able to verify if the stance detection model correctly categorized the comments based on their labels thanks to this experiment.

4.2.3. Label Balancing

We will use one data set for this analysis, which has 172,942 comments with 137,929 labeled as in favor of and 135,013 labeled as against the TPLF Party target. Regardless of how different outcomes are distributed inside the target feature, accurate predictions are given greater weight in the Dataset[35]. This will contribute to our stance detection model's high forecast accuracy of unseen data. The entire gathered dataset is provided below, along with its post-preprocessing state.

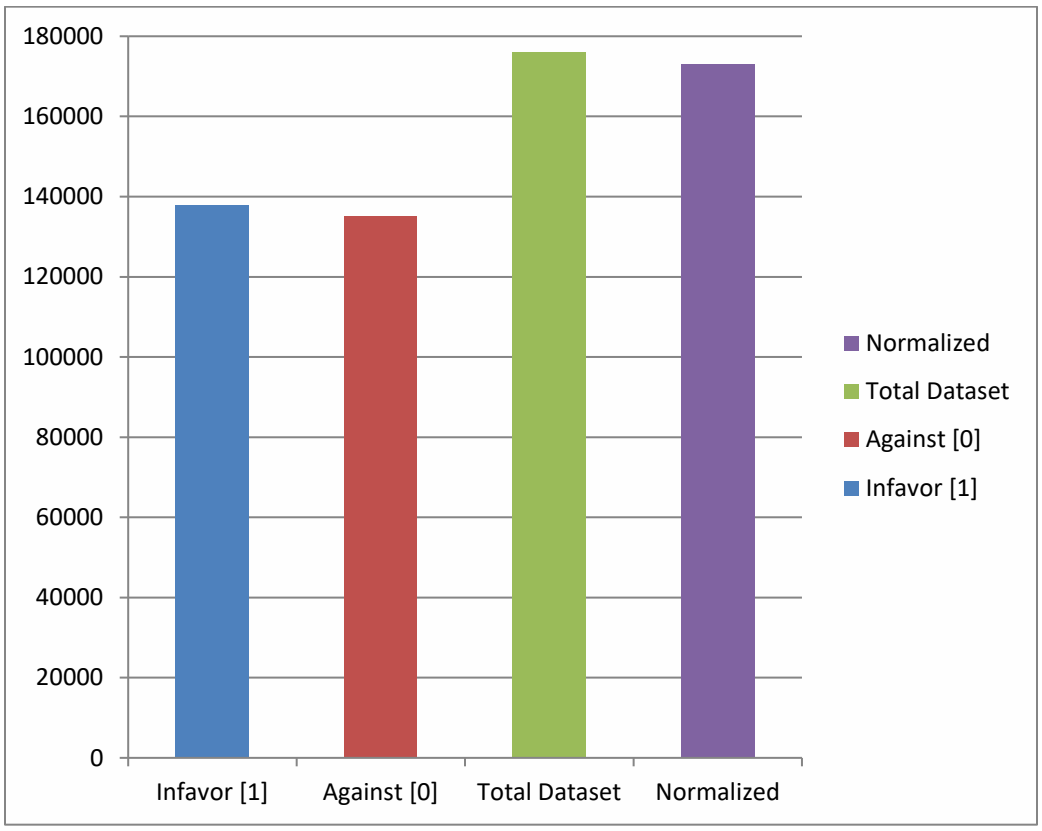


Figure 5. Distribution of the Dataset

4.3.Proposed Model Architecture

The proposed model architecture is designed to process and analyze the Facebook Comments Dataset through a comprehensive pipeline that integrates data preprocessing, feature extraction, and deep learning techniques. The architecture is structured to handle the complexities of textual data, ensuring accurate analysis and classification of comments. Below is a detailed description of the components that constitute this architecture.

The process begins with Dataset Acquisition and Annotation, where the initial "Facebook Comments Dataset" is collected. This dataset is then annotated to create a Dataset Annotation, which provides additional context or labels for the comments. This step is crucial for ensuring that the data is properly structured and ready for subsequent analysis. Following this, the Dataset Preprocessing phase is carried out, which involves three key sub-tasks. First, the comments data undergoes Cleaning, where noise and irrelevant information are removed to prepare the data for further analysis. Next, Tokenization is performed, breaking down the comments into individual words or meaningful units. Finally, the tokenized data is subjected to Normalization, which may involve converting words to a common form to ensure consistency and improve the quality of the analysis.

Once the data is preprocessed, it undergoes Morphological Analysis, where the structure and composition of the words in the comments are examined. This step helps in understanding the

underlying patterns and relationships within the text. Following this, Feature Extraction is performed using the Word2Vec model. This technique captures the semantic and syntactic relationships between words, transforming the text into a numerical format that can be processed by machine learning algorithms.

The extracted features are then fed into Deep Learning Models, specifically LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) architectures. These models are particularly well-suited for processing sequential data like text, as they can capture long-term dependencies and contextual information. The output of these models is a classification of the comments into two categories: "In-Favor" and "Against," representing the different opinions expressed in the comments.

To evaluate the performance of the trained deep learning models, a separate Testing Dataset is used. This dataset allows for the assessment of the model's accuracy and generalization capabilities, ensuring that it performs well on unseen data. The figure below illustrates the architecture of the proposed model, providing a visual representation of the workflow and the interplay between its various components. This architecture is designed to deliver robust and reliable analysis of the Facebook Comments Dataset, leveraging advanced techniques to achieve accurate and meaningful results.

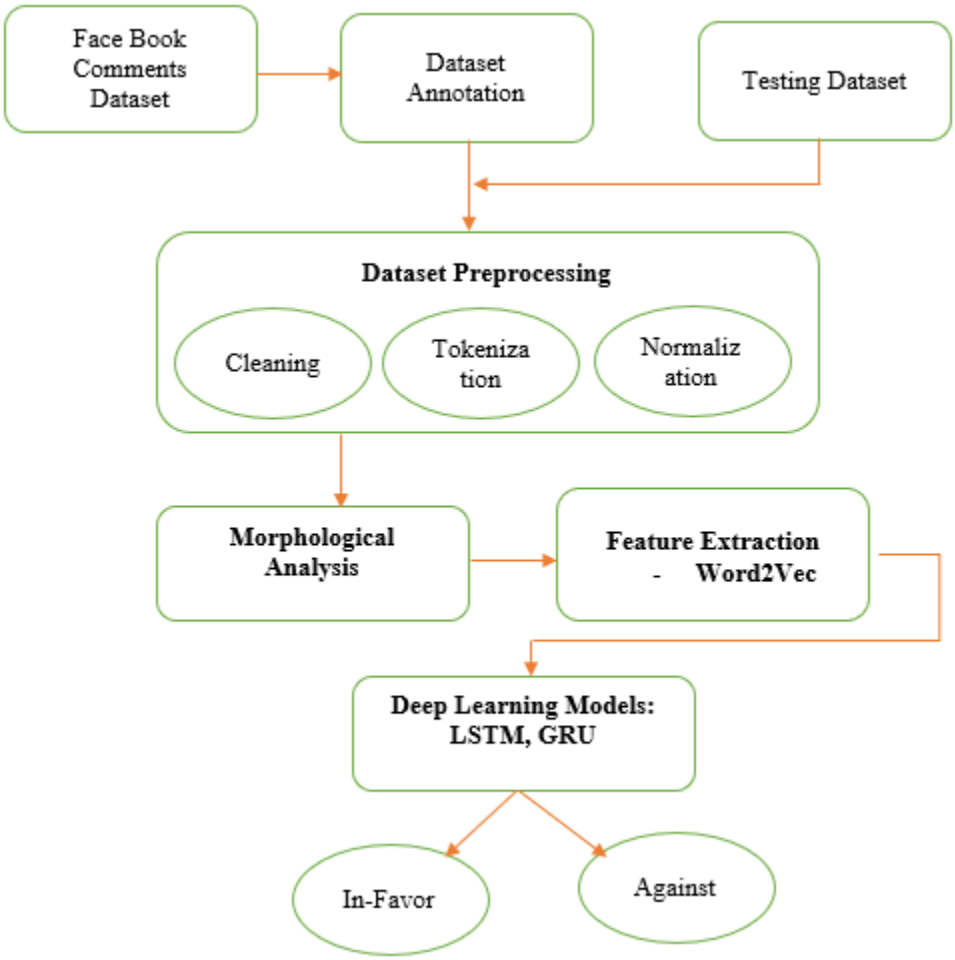


Figure 6. proposed Model Architecture

4.4. Proposed System Data Preprocessing Implementation

Data preprocessing comes next after data preparation. Data preprocessing aims to facilitate the computer's comprehension of natural language. To create the suggested system, various data preprocessing techniques have been employed, as covered in the methodology section. In this section, how we put the methodologies used to create the suggested system into practice has tried to demonstrate.

4.4.1. Tokenization Implementation

Tokenization is the process of splitting sentences into a list of words. The NLTK library contains the word tokenize function that is used for performing tokenization. By using this module, tokenization has been employed for the proposed system.

4.4.2. Data Cleaning and Normalization Algorithms

Eliminating unnecessary symbols or characters from the data is a process known as text cleaning. To increase the accuracy of our model's prediction, any punctuation, Ge`ez based numbers, symbols, or characters have been removed. The symbols were cleared using the following algorithms to put the suggested system into practice.

4.4.2.1. Data Cleaning Algorithm

INPUT: Uncleaned data

OUTPUT: cleaned dataset

START:

1: Read the dataset

2: WHILE (it is not the end of the file):

IF data include punctuations [',', '.', '"', ':', ')', '(', '-', '!', '?', '፡','፣', '፤'] THEN

Eliminate punctuations

```
def remove_punc_and_special_chars(text):
    normalized_text
=re.sub('[\!\@\#\$\%\^\«\»\&\*\(\)\…\[\]\{}\፡\፤\?\
    return normalized_text
```

IF data include number and English characters [፩ ፪ ፫ ፬ ፭ ፮ ፯ ፰ ፱ ፲ ፳ ፴ ፵ ፶ ፷ ፸ ፹ ፺ ፻ ፼ ፽ ፾

A B C . . . a b c . . . 0 1 2 …] THEN

Eliminate Number

```
def remove_ascii_and_numbers(text_input):
      rm_num_and_ascii=re.sub('[A-Za-z0-9]','',text_input)
      return re.sub('[\'\u1369-
u137C\']+','',rm_num_and_ascii)
```

IF data include symbols [

….] THEN

Eliminate Symbols

```
from clean-text import clean
clean(text, no_emoji=true)
```

Return cleaned data

3. STOP

4.4.2.2. Data Normalization Algorithm

INPUT: Unnormalized data
OUTPUT: Normalized data
START:
1: Read the dataset
2: WHILE (it is not the end of the file):
IF data includes [ጎሐ]' THEN
replace with 'ሀ'
IF text includes [ሑኍ] THEN
replace with 'ሁ'
IF text includes [ኺሒ]
THEN replace with 'ሂ'
IF text includes[ኈሔ] THEN
replace with 'ሄ'
IF text includes [ሕኀ] THEN
replace with 'ህ'
IF text includes [ኖሖ] THEN
replace with 'ሆ'
IF text includes [ሠ ሡ ሢ ሣ ሤ ሥ ሦ] THEN
replace with corresponding [ሰ ሱ ሲ ሳ ሴ ስ]
IF text includes [ዐ ዒ ዓ ዔ ዕ ዖ ኣ] THEN
replace with corresponding [አ ኢ ኣ ኤ እ ኦ ኣ]
IF text includes [ጸ ጹ ጺ ጻ ጼ ጽ ጾ] THEN
replace with corresponding [ፀ ፁ ፂ ፃ ፄ ፅ ፆ]
Return normalized text
3: **END**
	def normalize_char_level_missmatch (input_token):
	cha1=re.sub('[ሃኀኃሐኻ]','ሀ',input_token)
	cha2=re.sub('[ሑኁኹ]','ሁ',cha1)
	cha3=re.sub('[ኺኼዂ]','ሂ',cha2)
	cha4=re.sub('[ኈኼዄ]','ሄ',cha3)
	cha5=re.sub('[ሕኅ]','ህ',cha4)
	cha6=re.sub('[ኖሖኾ]','ሆ',cha5)
	cha7=re.sub('[ሠ]','ሰ',cha6)
	cha8=re.sub('[ሡ]','ሱ',cha7)
	cha9=re.sub('[ሢ]','ሲ',cha8)
	cha10=re.sub('[ሣ]','ሳ',cha9)
	cha11=re.sub('[ሤ]','ሴ',cha10)
	cha12=re.sub('[ሥ]','ስ',cha11)
	cha13=re.sub('[ሦ]','ሶ',cha12)
	cha14=re.sub('[ዓኣዐ]','አ',cha13)
	cha15=re.sub('[ዑ]','ኡ',cha14)
	cha16=re.sub('[ዒ]','ኢ',cha15)
	return cha16

4.4.3. Stop Word Removal Algorithm

INPUT: all list of words in the dataset

OUTPUT: list of words does not have a stop word

Variable: Stopwords [], RemovedStopWord []

1: Read the dataset

2: WHILE (list of words in the dataset):

IF the list of words in the dataset is not in Stopwords []: THEN

Append the word in RemovedStopWord []

Return RemovedStopWord []

3: Halt

4.5. Proposed Feature Extraction Implementation

When using deep learning or a machine learning algorithm, input text data should be transformed into a numeric vector. the word2vec technique was employed for this purpose, as covered in the methodology sections. The word2vec techniques were used for the proposed model's development.

4.6. Tigrigna Word2Vec Generation

This supports the model ability to extract contextual words for a successful understanding of the comments. For confined corpus, the Tigrigna word vector plays a big role in understanding the opinion of the commenters to detect and classify their intension. A custom-made Tigrigna Word2Vec is created here, with the majority of the sentences focused on political opinion and derived from a variety of websites, books, and manuals. Finally, gathered 135,000 sentences from which we constructed a word vector. The posterior distribution of words, which is a polynomial distribution, is obtained by applying a SoftMax of a log-linear classification model.

$$p(w_j \mid w_1 = y_j = \frac{\exp(u_j)}{\sum_{j=1}^{u} \exp(u_j)} \qquad \text{Equation (7)}$$

u_j is a score for each word in the vocabulary, and y_j is the output of the j-the unit in the output layer. It is the loss function (E). where $j * c$ is the value of the vocabulary word that belongs in the c-th output context. We use this to train the custom Tigrigna Word2Vec corpus.

4.6.1. Procedures For Creating Tigrigna Word2Vec Model

Preparing data:

Table 4. Tigrigna Comments Stance Dataset

Comment ID	Comment Text (Tigrigna)	Stance Label
1	"ህወሓት ንደቂ ህዝቢ ኤርትራ ኣዝዩ ጽቡቕ እዩ።"	In favor
2	"ህወሓት ኣብ ልዕሊ ህዝቢ ትግራይ ሕማቅ ዝገበረ እዩ።"	Against
3	"ብሰንኪ ህወሓት ዝተወለዐ ኩናት ህዝቢ ትግራይ ኣዝዩ ተንዲኡ እዩ።"	Against
4	"ህወሓት ንህዝቢ ትግራይ ኣይሕግዝን እዩ።"	Against
5	"ህወሓት ንህዝቢ ትግራይ ኣዝዩ ጽቡቕ እዩ።"	In favor
6	"እቲ ውድብ ኣብ ትልሚ ፖለቲካዊ ትሕዝትኡ ምንም ዓይነት ሰላም ናይምፍጣር ዛእባ ዘለዎ ኣይኮነን።"	Against
7	"ከምቲ ኣብ ስትራተጂኣ ዝኣወጀቶ ግብራዊ ሰላም ንምርግጋጽ ዓብይ ወፈያ ገይራ እያ።"	In favor
8	"ህወሓት ንህዝቢ ትግራይ ኣዝዩ ጽቡቕ እዩ።"	In favor
9	"ህወሓት ኣብ ልዕሊ ህዝቢ ትግራይ ሕማቅ ዝገበረ እዩ።"	Against
10	"ብሰንኪ ህወሓት ዝተወለዐ ኩናት ህዝቢ ትግራይ ኣዝዩ ተንዲኡ እዩ።"	Against

4.6.1.1. Steps to Prepare the Dataset for Word2Vec

Data Collection:

The first step in preparing a dataset for Word2Vec is data collection. The goal is to gather a large and diverse corpus of Tigrigna text data from various sources. These sources can include social media platforms like Facebook, where users frequently engage in discussions in Tigrigna. Additionally, news articles from Tigrigna news websites, public forums, and even digitized books or literature can provide valuable text data. It is important to ensure that the data is publicly available and complies with ethical guidelines. Collecting data from diverse domains, such as politics, culture, and education, will help improve the model's ability to generalize across different contexts. The larger and more varied the dataset, the better the Word2Vec model will perform in capturing the semantic relationships between words.

Data Cleaning:

Objective:

- **Remove noise and irrelevant content from the dataset.**

Steps:

- **Remove Special Characters**: Eliminate emojis, symbols, and non-Tigrigna characters.
    - Example: Remove " ", "#", "@username", etc.

- **Normalize Script**: Ensure all text uses the Ge'ez script (e.g., convert Latin script to Ge'ez if necessary). The Tigrigna script used for this study are referenced in appendix A.
- **Remove Punctuation**: Strip out punctuation marks like periods, commas, and question marks. the punctuation marks used for this study are referenced in appendix F.
    - Example: Convert "ሰላም! እንታይ ኣሎ?" to "ሰላም እንታይ ኣሎ".
- **Remove Stop Words**: Eliminate common words that do not contribute to semantic meaning. The Tigrigna script used for this study are referenced in appendix A. used for this study are reference in appendix B.
    - Example: Remove words like "ን", "ናብ", "ናይ", "ኣነ", "ውሽጢ".
- **Handle Numbers**: Convert or remove numerical values. The numerical used for this study are referenced in appendix E.
    - Example: Replace "10 ዓመት" with "ዓሰርተ ዓመት".
- **Lowercase Conversion**: Convert all text to lowercase to ensure consistency.
    - Example: Convert "ህወሓት" to "ህወሓት" (Tigrigna is case-insensitive, but this step ensures uniformity).

Tokenization:

Objective:

Split the text into individual words or tokens.

Steps:

- **Sentence Splitting**: Divide the text into sentences using punctuation marks like periods or question marks.
    - Example: Split "ሰላም! እንታይ ኣሎ? ደሓን መጻእኩም." into ["ሰላም", "እንታይ ኣሎ", "ደሓን መጻእኩም"].
- **Word Splitting**: Split each sentence into individual words.
    - Example: Split "እንታይ ኣሎ" into ["እንታይ", "ኣሎ"].

Dataset Formatting:

Objective:

Prepare the dataset in a format suitable for Word2Vec training.

Steps:

- **Create a List of Sentences**: Format the dataset as a list of sentences, where each sentence is a list of tokens.

Example:

```
corpus = [
    ["ህወሓት", "ንደቂ", "ህዝቢ", "ኤርትራ", "ኣዝዩ", "ጽቡቕ", "እዩ"],
    ["ህወሓት", "ኣብ", "ልዕሊ", "ህዝቢ", "ትግራይ", "ሕማቅ", "ዝገበረ", "እዩ"],
    ["ብሰንኪ", "ህወሓት", "ዝተወለዐ", "ኩናት", "ህዝቢ", "ትግራይ", "ኣዝዩ", "ተጎዲኡ", "እዩ"]
]
```

- **Remove Rare Words**: Optionally, filter out words that appear fewer than a specified number of times (e.g., min_count=5).
- **Shuffle the Dataset**: Randomize the order of sentences to improve training efficiency.

Word2Vec Training:

- **Objective**: Train a Word2Vec model using the formatted dataset.
- **Steps**:
    1. **Install Gensim**: Use the Python library **Gensim** to train the Word2Vec model.

```
from gensim.models import Word2Vec

# Sample corpus
corpus = [ ["ህወሓት", "ንደቂ", "ህዝቢ", "ኤርትራ", "ኣዝዩ", "ጽቡቕ", "እዩ"],
   ["ህወሓት", "ኣብ", "ልዕሊ", "ህዝቢ", "ትግራይ", "ሕማቅ", "ዝገበረ", "እዩ"],
   ["ብሰንኪ", "ህወሓት", "ዝተወለዐ", "ኩናት", "ህዝቢ", "ትግራይ", "ኣዝዩ", "ተጎዲኡ", "እዩ"]]
# Train Word2Vec model
model = Word2Vec(
   sentences=corpus,          # Formatted dataset
   vector_size=100,           # Dimensionality of word vectors
   window=5,                  # Context window size
   min_count=1,               # Ignore words with frequency < min_count
   workers=4        # Number of CPU cores
)
# Save the model
```

```
model.save("tigrigna_word2vec.model")
```

Used the formatted corpus to train a Word2Vec model using libraries Gensim. The following is a Python code snippet for training a Word2Vec model using a sample corpus.

```
from gensim.models import Word2Vec
# Sample corpus
corpus = [ ["ህወሓት", "ንደቂ", "ህዝቢ", "ኤርትራ", "ኣዝዩ", "ጽቡቕ", "እዩ"],
   ["ህወሓት", "ኣብ", "ልዕሊ", "ህዝቢ", "ትግራይ", "ሕማቅ", "ዝገበረ", "እዩ"],
   ["ብሰንኪ", "ህወሓት", "ዝተወለዐ", "ኩናት", "ህዝቢ", "ትግራይ", "ኣዝዩ", "ተጎዲኡ", "እዩ"]]
# Train Word2Vec model
model = Word2Vec(sentences=corpus, vector_size=100, window=3, min_count=1, workers=4)
# Save the model
model.save("tigrigna_word2vec.model")

# Example: Find similar words
similar_words = model.wv.most_similar("ህወሓት")
print(similar_words)
```

Evaluate the Model:

- Check word similarities to ensure the model captures semantic relationships.

```
similar_words = model.wv.most_similar("ህወሓት")
print(similar_words)
```

4.7. Deep Learning Classifiers

Deep learning classifiers refer to machine learning models that use deep neural networks for the task of classification. Feature extraction is one of the most crucial NLP procedures to follow to better understand the context of the content we are dealing with. The original text needs to be cleaned up and then transformed into its features to be used for modeling. Comments cannot be computed directly; instead, they must be transformed into numerical data, like a vector space model. This transformation operation is often known as feature extraction of data from documents. Feature extraction, sometimes referred to as text representation, text extraction, or text vectorization, might affect the machine learning model because of how they express data in numerical form. The BoW and Skip-gram feature extraction techniques are applied with Word2vec. We explored supervised learning algorithms in Chapter two and attempted to compare the outcomes of these experiments by utilizing assessment measures for each model.

4.8. Structure of a Recurrent Neural Network

When using a Recurrent Neural Network (RNN) for stance text classification, the structure typically involves the following components:

Word Embedding Layer: Stance text classification involves processing text data, as we discussed in the last chapter, and the first step is often to represent words as dense vectors. A word embedding layer is used to convert words into continuous vector representations, capturing their semantic meaning and relationships. This is then accomplished using word embedding techniques called Word2Vec for this study.

RNN Layer: The core component of the network is the RNN layer, which processes the sequential nature of text data. It maintains an internal state that is updated at each time RNN variants used for the text classification are Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) as discussed earlier. These variants address the vanishing gradient problem and allow the network to capture long-term dependencies in the input text.

Stance Classification Layer: After processing the input text through the RNN layers, the final step is to make predictions about the stance of the text. This is typically achieved using a dense layer followed by a SoftMax activation function. The dense layer maps the RNN output to a vector of suitable dimensions for the target stance classes called in favor and against. The SoftMax function then produces a probability distribution over the possible stance classes, indicating the model's confidence for each class.

Each network's applicable network has the same fundamental structure, as seen in Figure 7. The tokenized texts were delivered to the buried layer (input messages). At this point, the tokenize stage is executed, converting each word (w1, w2, w3, etc.) into a series of vectors (x1, x2, x3, etc.), which each uniquely defines the word's frequency and its state.

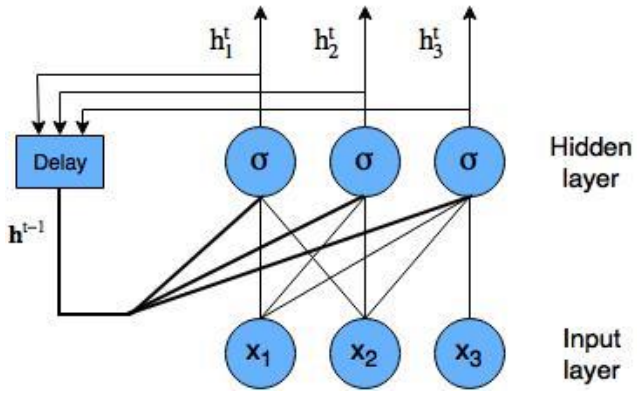


Figure 7. RNN network nodes diagram

Then, as shown in Figure 8, an embedding layer transforms those sequences of integers once more into a sequence of real vectors of size 128.

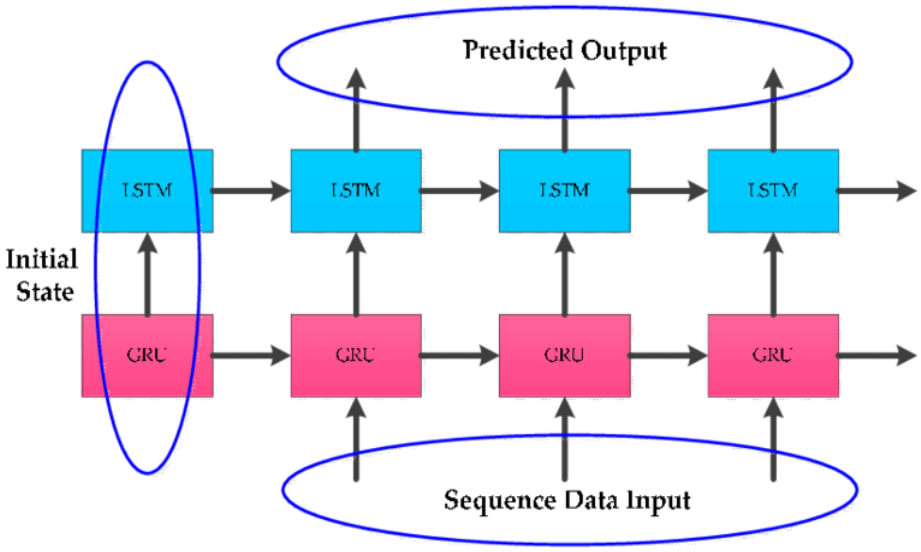


Figure 8. LSTM and GRU embedding inputs

After that, these integer sequences are passed to the RNN substructure. Lastly, using the usual SoftMax function, S (z), the RNN's output is directed via a SoftMax layer.

$$S(z)_j = \frac{e^{z^t w_j}}{\sum_{i=1}^{m} e^{z^t w_j}} \quad \text{..... Equation (8)}$$

where m is the output size of the SoftMax layer or the number of classes, and w is the weight vector for all connections between the SoftMax layer and the preceding layer. The SoftMax function's intriguing characteristic is that

$$\sum_{j=1}^{m} S(z)_{(j)} = 1 \quad \text{.....Equation (9)}$$

And therefore,

$$(z)(j) \approx (y = j|z) \quad \text{..... Equation (10)}$$

The intent classifier is supposed to output a vector of probability along with its prediction; thus, this is helpful in this situation. The categorical cross-entropy loss is the minimized loss function during training and is provided as:

$$L(\theta) = -\frac{1}{m}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij} \log(\dot{y}_{i,j}(\theta)) \quad \text{..... Equation (11)}$$

$y_{i,j}$ 0,1 is 1 if the class of sample i is j and 0 otherwise, and $y_{i,j}$ [0,1] is the projected probability that sample i has label j. The accuracy of the model's predictions is another statistic that is created during training. To train the neural networks, the Adam optimizer was used because it has been proven to give good results in the field of deep learning. It's also computationally efficient, which

comes in handy here because there are so many samples to optimize.

4.9. Selection and Implementation of Network Structures

Network structures, which are different RNN versions by applying network variants, are shown in this section. In general, network selection for sequence modeling is required to determine which loss, training duration, accuracy, and were measured during training. Based on the findings of[60], two networks were tested to find the best fit. In this study, the accuracy of the two networks was compared throughout the training period. All the models were built with the optimal network hyperparameters and then trained with the prepared data set to get the optimal value from each variant.

4.9.1. Results of Single LSTM Network

The single-layer LSTM network structure is the first network structure. Figure 9 depicts the structure of a single LSTM cell neural network. To generate appropriate mathematical functions throughout the network cell structure, the basic method of neural networks in a sequence of data is to take the data, change it to the kind of words in the time stamp, and then modify those input sequences of words to the embedded method.

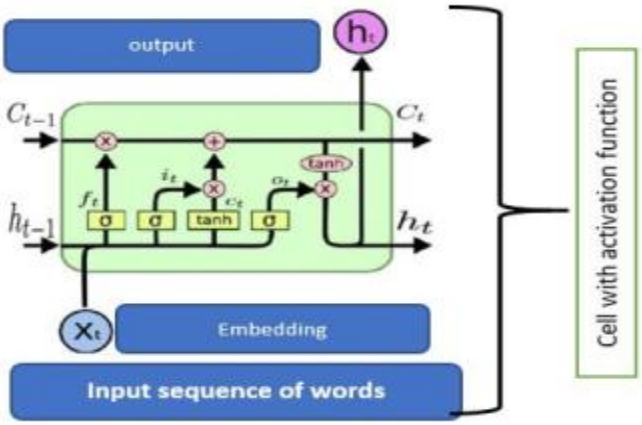


Figure 9. Single LSTM, cell workflow from accepting input to output

Figure 10 shows how the network accuracy develops as the number of training and validation epochs increases. The loss and accuracy of the network display significant variation during training, which makes its properties inconsistent. However, the network learns almost entirely to properly identify the whole training set on the validation set, with a final measured accuracy of 79.87%.

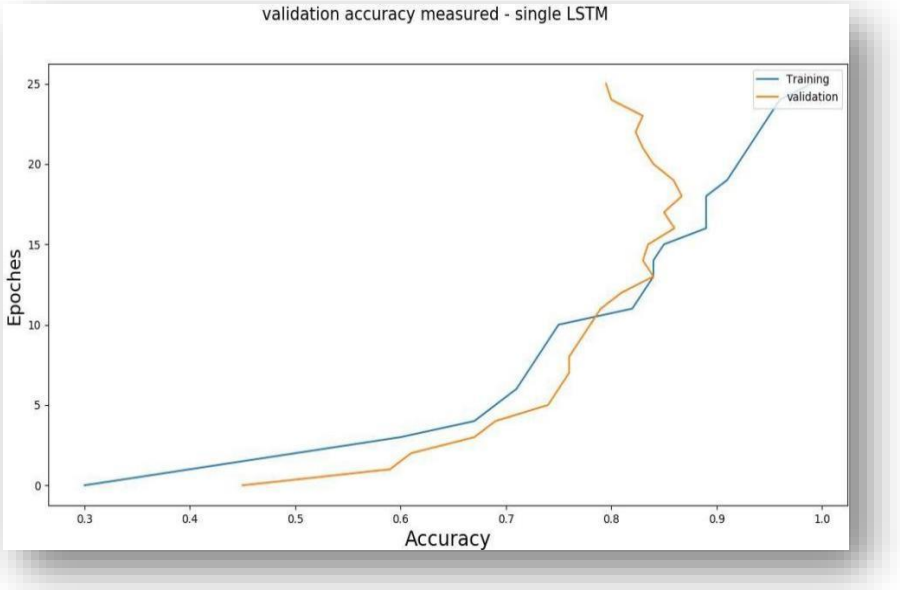


Figure 10. Single LSTM- Network accuracy

4.9.2. Results of Single GRU Network

GRU-Gated Recurrent Unit is the other recurrent neural network technique. The goal of including GRUs in our experiment is to look at the similarities and differences between employing LSTM and GRU layers in terms of network consideration. Figure 11 depicts the design of one GRU network. In this network training, a single GRU layer layout yields results that are remarkably similar to those of a single LSTM layer, as shown in Figure 12. The network achieves roughly the same end accuracy of 79.05%, as seen in the graph while continuing to display the same unpredictable behavior throughout training.

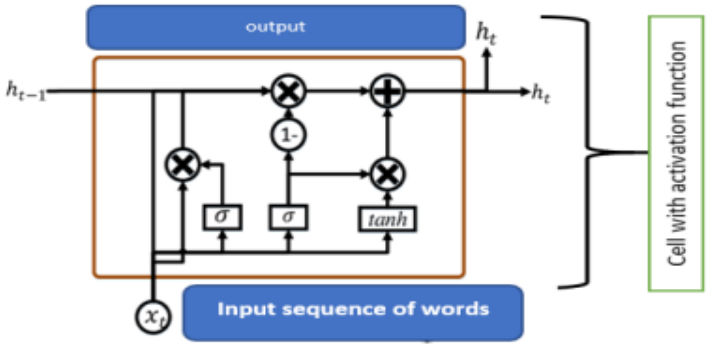


Figure 11. Single-cell GRU network structure

The gated recurrent unit lowers training time in half while keeping almost identical performance.

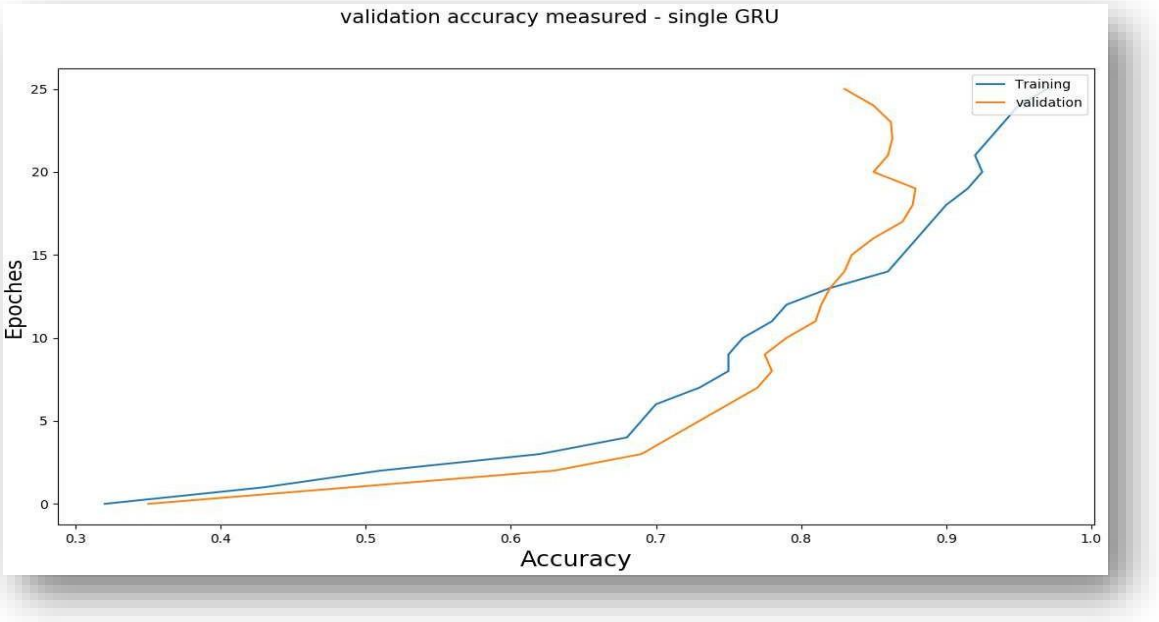


Figure 12. Single-cell GRU network Accuracy

4.9.3. Results of Transposed or Inverted GRU Network

Comparatively, the LSTM performs poorly, as seen in section 4.9.1, and neither inverting LSTM nor transposing GRU by altering the input sequence can improve training accuracy more than GRU. By modifying the structure (moving places) of the data acquisition, the architecture demonstrates a smaller difference from the one-layer GRU architecture. Since then, transposing the input order has proven to be a simple trial that might potentially alter the performance of certain networks, particularly in machine translation and its subfields[60].

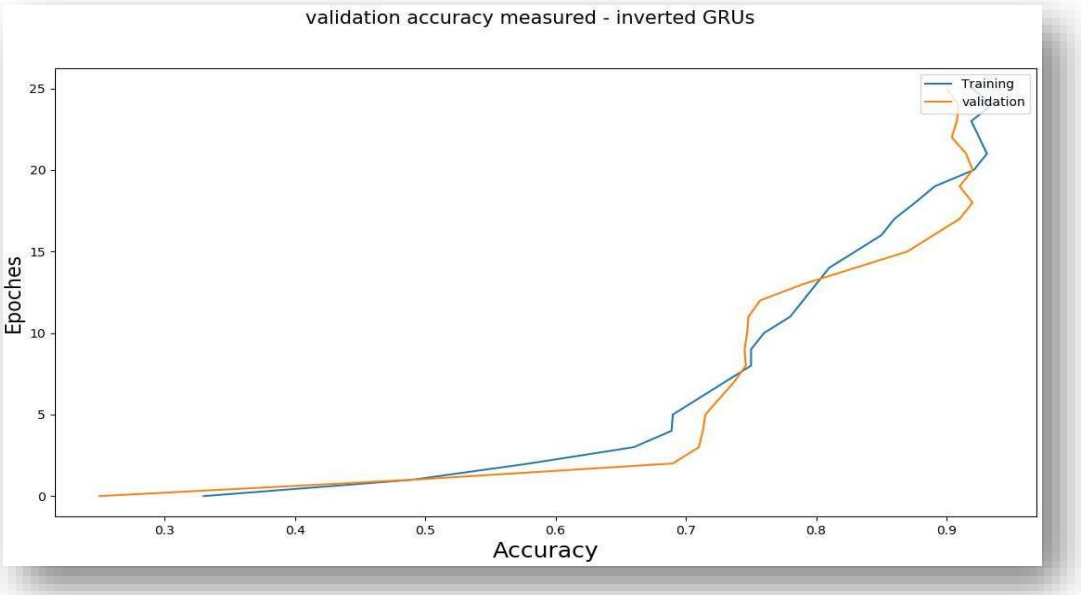


Figure 13. Transposed GRU performance

Figure 13 depicts the progression of its accuracy. Due to decreased jitter for loss and accuracy, this structure is noticeably more stable during training than the prior ones. Despite having the input sequence reversed, the training process only took 1434 minutes and resulted 81.17% higher accuracy score.

4.9.4. Summary of the Network Variants

Starting with section 4.9.1, the three networks are investigated. Table 5 summarizes the findings while taking loss and accuracy into account.

Table 5. Summarizing the network variants

Network Architecture	Validation loss (lower is chosen)	Validation accuracy (higher is chosen)
Single LSTM	0.347	79.87%
Single GRU	0.423	79.05%
Transposed GRU	0.345	81.17%

Viewing the properties of each network based on its visualization is critical to choosing the ideal network architecture. According to the validation loss and accuracy validation, a transposed GRU is preferred.

The effects of a growing number of epochs are now thoroughly explored and stated. For optimal epoch selection, the neural network is trained for 80 epochs to ensure that the most favorable training progress does not lie in the preceding experiment of 25 epochs. The first 30 epochs are employed. because, after that, the minor learning rate, loss, and accuracy are explored.

CHAPTER FIVE

5. FINDINGS AND DISCUSSION

This chapter describes the model's overall outcomes based on quantitative performance measures as well as its performance in a realistic context. The findings were used to evaluate research questions.

5.1. Testing Result of the Model

5.1.1. Using Stratified Cross-Validation

Different data processing is done while the system model is implemented, as stated in the previous chapter three. In the last section, the overall accuracy of the training model is displayed. Using cross-validation, as discussed in the methodology section, is applied, and a stratified cross-validation type was selected for validation after partitioning the data into training, validation, and testing sets, Table 6 displays the accuracy captured during training and testing in five (5-fold) different data distributions.

Table 6. Data distribution description with its performance.

Data Distribution Based on Training, Validation, and Testing)	Model Training Accuracy	Model Testing Accuracy	Description
(50%,5%,45%)	85.63%	57.22%	Overfitting and increasing the validation set can solve this problem.
(60%,10%,30%)	74.94%	66.31%	Still, over-fitting is there.
(60%,20%,20%)	73.05%	65.29%	Accuracy diminution
(70%,10%,20%)	81.88%	81.17%	Achieves better than others
(80%,10%,10%)	89.65%	73.77%	The model generalizes the training, and high over-fitting is recorded.

When the 172,942 data is split into 70%, 10%, and 20%, which is 1,21,059, 17,294, and 34,588 data for training, validation, and testing respectively, the performance is better than other data distributions. Because the documented difference between modeling, training, accuracy, and testing accuracy is smaller in this training distribution than in others, there is less over-fitting.

By incorporating a large amount of data, the model's performance can be improved. Tigrigna word vectors can help this model with data for better classification. As a result, it necessitates the use of very advanced Tigrigna Word2Vec, which has a vast amount of Tigrigna knowledge. Even though the model's accuracy is less than 85%, the data annotation is done by hand, and it is not too open to everyone's opinion stance because the way data is presented might affect the model's performance.

The table below describes the accuracy of each network variant as it yields better accuracy.

Table 7. Summary of each network under stratified 5-fold cross-validation

LSTM			**GRU**		**Transposed GRU**	
Stratified CV	Accuracy	Lose	Accuracy	Lose	Accuracy	Lose
5 – Fold	78.53%	0.170	79.05%	0.178	81.17%	0.208

5.1.2. Using the F-1 Score

The other validation technique is the F-1 score, the f-1 score (i.e., good for unbalanced data distribution) is then employed by evaluating the distribution of the test data for intentions. The F-1 score of each model is described in Table 8.

Table 8. Summary of each network under F-1 score evaluation

	LSTM	**GRU**	**Transposed GRU**
F-1 score	0.7580	0.7856	0.8822

For the proposed model, we get the same f- 1 score regardless of the data provided for different classes. The equation below describes the confusion matrix of the classification model with its F-1 score value. the value of F-1 score is calculated as follows.

Precision = True Positive / (True positive + False Positive)

The average precision of the model = 0.7995

Recall = True Positive / (True Positive + False Negative)

Average Recall of the model = 0.9841

F-1 score = 2 x (precision x recall) / (precision + recall)

F-1 score = 2 x (0. 7995x 0. 9841) / (0. 7995+ 0. 9841)

F-1 score = 2 (0.78679) / (1.7836)

F-1 score = 1.57358 / 1. 7836

F-1 score = 0.8822……Equation (12)

5.1.3. Using the Two Types of Word2vec Model

The word2vec model has two approaches: the Skip-gram approach and the Continuous bag of words. The Skip-gram method predicts its neighbors using the current words, whereas CBOW predicts the target word using its neighbors. Table 9 compares these two approaches when applied to proposed deep learning models.

Table 9. Comparison of the 2 word2vec models

	LSTM		GRU		Transposed GRU	
Word2vec type	Accuracy	Lose	Accuracy	Lose	Accuracy	Lose
Skip gram	78.53%	0.151	79.05%	0.191	79.86%	0.221
CBOW	77.69%	0.1685	79.01%	0.188	79.84%	0.219

The Skip-gram model outperforms the CBOW among all proposed deep-learning models, as shown in the table above. Aside from performance, the skip-gram model is recommended because it better captures the semantic relationship between words to categorize the opinion, whereas the CBOW model only considers morphologically related words placed in nearby space.

To sum up, The initial dataset was divided into three parts: training, testing, and validation; we used 80% of our dataset for training, 10% for testing (evaluation), and 10% for validation. According to the architecture of the proposed system, feature extraction was performed before the training phase in which learning was used. As a result, we created distinct training and testing and validating datasets for data extracted using word2vec with Bag of words, skip-gram.

We train the models with those features to predict the class of 20% of the dataset based on 80% of the training dataset, and we observe the model's accuracy by comparing the prediction to the actual class of 20% of the test dataset.

5.2. Proposed Model Comparisons with Existing Models

This section compares the performance of the proposed Transposed GRU model with existing models used in stance detection, particularly for low-resource languages like Tigrigna. The table below summarizes the recorded accuracy performance and descriptions of various models, including the proposed model.

Table 10. Comparison with the existing models

Model Name	**Recorded Accuracy Performance**	**Description**
SVM [1]	82%	Used TF-IDF for feature extraction on Prosperity Party Facebook posts. Limited to traditional machine learning techniques.
CNN + BI-LSTM [48]	86%(Target Identification), 65%	Applied deep learning models to web-scraped social media data. Did not explore advanced word embeddings or mixed

	(Stance Classification), 50% (Sentiment Classification)	stance tweets.
LSTM (Proposed Model)	79.87%	Single-layer LSTM model with Skip-gram feature extraction. Demonstrated moderate accuracy but struggled with stability during training.
GRU (Proposed Model)	79.05%	Single-layer GRU model with Skip-gram feature extraction. Similar performance to LSTM but with faster training times.
Transposed GRU (Proposed Model)	**82%**	Transposed GRU model with Skip-gram feature extraction. Achieved the highest accuracy and F1-score (0.8822) among the proposed models.

Summary of Comparisons

Traditional Machine Learning Models: The SVM model used by Surafel Tadesse[1] achieved an accuracy of 82% using TF-IDF for feature extraction. However, this model was limited to traditional machine learning techniques and did not explore deep learning approaches or advanced feature extraction methods. In contrast, the proposed Transposed GRU model achieved similar accuracy but leveraged deep learning and advanced word embeddings, making it more robust for complex linguistic tasks.

Deep Learning Models: Bewketu Molla[48] applied CNN and BI-LSTM models to social media data, achieving high accuracy (86%) for target identification but lower performance for stance (65%) and sentiment classification (50%). This model did not explore alternative word embeddings or mixed stance tweets, which limited its effectiveness. The proposed Transposed GRU model outperformed this approach in stance classification, achieving 82% accuracy and demonstrating better stability and generalization.

Proposed LSTM and GRU Models: The single-layer LSTM and GRU models proposed in this study achieved moderate accuracy (79.87% and 79.05%, respectively). While these models performed well, they exhibited instability during training and struggled with long-term dependencies in the text. The Transposed GRU model addressed these limitations by reversing the input sequence, resulting in improved stability and higher accuracy (82%).

Transposed GRU Model: The Transposed GRU model emerged as the best-performing model in this study, achieving an accuracy of 82% and an F1-score of 0.8822. This model outperformed both traditional machine learning models and other deep learning approaches by effectively capturing the semantic and contextual nuances of Tigrigna text. Its ability to handle long-term

dependencies and its stability during training make it a strong candidate for stance detection in low-resource languages.

5.3. Research question discussion

RQ1: What method can be developed to create annotated Tigrigna text datasets that effectively capture diverse political stances?

The study successfully established a systematic methodology for creating a representative and annotated Tigrigna text dataset. This dataset was collected from the TPLF Facebook page using the Facepager tool, which facilitated the extraction of publicly available comments from 2021 to 2023. The comments were manually categorized into two groups: "In Favor" and "Against," based on the expressed stance toward the TPLF party. To ensure consistency and reliability, the annotation process adhered to clear guidelines. Additionally, the dataset underwent preprocessing steps, including cleaning, tokenization, normalization, and morphological analysis, to make it suitable for deep learning models. This methodology not only filled the gap in annotated Tigrigna datasets but also laid a foundation for future research in low-resource languages.

RQ2: Which feature extraction techniques best fits the linguistic and cultural characteristics of Tigrigna for accurate stance detection

The study explored two feature extraction techniques: Bag of Words (BOW) and Skip-gram from Word2Vec. The results indicated that the Skip-gram model outperformed the BOW approach in capturing the semantic relationships between words, which is crucial for understanding the context and nuances of Tigrigna text. The Skip-gram model was particularly effective in capturing the morphological richness of Tigrigna, which is essential for accurate stance detection. This finding aligns with the linguistic characteristics of Tigrigna, where word order and context play a significant role in conveying meaning. The study concluded that Skip-gram is a more suitable feature extraction technique for Tigrigna text, especially when combined with deep learning models like GRU and LSTM.

RQ3. Which classification algorithms achieve high accuracy in political stance detection for Tigrigna text?

The study evaluated three deep learning models: LSTM, GRU, and Transposed GRU. The results showed that the Transposed GRU model achieved the highest accuracy of 82% and an F1-score of 0.8822 when combined with the Skip-gram feature extraction technique. This model outperformed the standard LSTM and GRU models, which achieved accuracies of 79.87% and 79.05%, respectively. The Transposed GRU model demonstrated better stability during training and was more effective in capturing long-term dependencies in the text, which is critical for stance detection. The study concluded that the Transposed GRU model, combined with Skip-gram, is the

most effective classification algorithm for political stance detection in Tigrigna text.

RQ4. How do the performance and robustness of the proposed models compare to existing research for low-resource languages?

The performance of the proposed models was compared to existing research in stance detection, particularly for low-resource languages like Tigrigna. The study found that the proposed Transposed GRU model achieved competitive results, with an accuracy of 82% and an F1-score of 0.8822. This performance is notable given the challenges of working with a low-resource language like Tigrigna, which lacks large-scale annotated datasets and advanced NLP tools. The study also highlighted the importance of feature extraction techniques, as the Skip-gram model significantly improved the performance of the deep learning models. These findings suggest that the proposed methodology and models are robust and can serve as a benchmark for future research in low-resource languages.

RQ5. What are the most effective evaluation metrics for assessing Tigrigna stance detection models in practical applications?

The study employed several evaluation metrics, including accuracy, precision, recall, and F1-score, to assess the effectiveness of the proposed models. The F1-score was particularly useful for evaluating the model's performance on imbalanced datasets, as it provides a balance between precision and recall. The study also used stratified cross-validation to ensure the generalizability of the results. The results showed that the Transposed GRU model achieved the highest F1-score of 0.8822, indicating its effectiveness in classifying political stances in Tigrigna text. These metrics were hold reliable for assessing the performance of stance detection models in practical applications, especially in low-resource language settings.

CHAPTER SIX

6. CONCLUSION AND RECOMMENDATION

6.1. Conclusion

A thorough grasp of social media stance detection is offered by this thesis. It begins by reviewing the literature on stance detection on social media and giving a summary of the methods that are currently in use to handle stance modeling. In analytical research measuring public opinion on social media, particularly on political and social themes, stance detection plays a crucial role. To prepare our dataset for feature extraction, we used BOW and Skip-gram for word2vec, two of the various feature extraction techniques. The output of this step is an input to three different classification algorithms: LSTM and GRU variants. In this study, we attempted to outline some preliminary techniques utilized to obtain comments from social media comments toward the TPLF party. We first construct the model using the training dataset's vector representation, and then we use the test set to assess the model. With an accuracy score of 82%, the Skip Gram feature extraction approach, in conjunction with the Transposed GRU model, produced a better categorization of political attitudes in this investigation.

In this work, we found that obtaining excellent classification accuracy in deep learning algorithms depends on the feature extraction strategy and data preparation. The scikit classification report and several accuracy test functions are used to assess the models' overall performance. In summary, we can say that the experiment yielded positive results, but the research may be more interesting if it made use of deep learning, a variety of feature extraction techniques, and more datasets to boost accuracy. The implications of this research extend to both political analysis and the NLP community. For political analysts, the findings can inform strategies for monitoring public sentiment and understanding voter behavior, thereby enhancing engagement and communication efforts. For the NLP community, the development of an annotated Tigrigna dataset and effective deep learning models contributes to the advancement of low-resource language processing, paving the way for further research and applications in similar linguistic contexts. While the experiment yielded promising results, integrating diverse deep learning approaches, additional feature extraction techniques, and larger datasets could further enhance accuracy and robustness, ultimately benefiting both political discourse analysis and NLP advancements.

6.2. Contribution of the study

Following the completion of this study, we made the following contributions to feature researchers and to anybody who wants to know the Tigrigna stance classification.

- We have compiled a dataset of approximately 172,942 Tigrigna sentences.
- A comprehensive Tigrigna stance dataset was created, annotated, and categorized into "In Favor" and "Against" stances.
- A Tigrigna Word2Vec model was developed to aid future researchers in NLP tasks.
- The effectiveness of deep learning methodologies for Tigrigna stance categorization was validated, offering a framework for subsequent studies.

6.3. Recommendation

As future work the researchers recommended the following issues to be addressed in future work.

- A smaller dataset was used to train our model; however, more targets might be added to expand the work to a bigger dataset. Therefore, by applying deep learning algorithms to generate more accurate models, the study may be improved.
- The research may be further enhanced by including Latin Tigrigna alphabets, as our model was trained using Ge'ez alphabets.
- Our classification algorithm only predicts one target attitude; however, by recognizing many target stances in a single comment, the work may be extended.
- To address the lack of labeled data for each target in the stance identification issue, transfer learning techniques are used to increase overall stance detection performance and enrich the representation of targets in the dataset.
- Investigating other word embedding methods, good performance may be attained with BERT, Glove, and fast Text word embedding feature extraction.

Future studies can utilize idiomatic expressions and emojis to assess people's opinions in addition to their position since they can provide important insights into the real meaning of the statement.

REFERENCES

[1] S. Tadesse, “POLITICAL STANCE DETECTION ON AMHARIC TEXT USING MACHINE LEARNING,” St. Mary’s University, 2023.

[2] A. Tazeb, “The Framing of Political Uprising on Social Media: The Case of Amhara and Oromia Regional States,” 2017.

[3] D. Küçük and F. Can, “Stance detection: Concepts, approaches, resources, and outstanding issues,” in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2673–2676.

[4] D. Küçük and F. Can, “Stance detection: A survey,” *ACM Comput. Surv.*, vol. 53, no. 1, pp. 1–37, 2020.

[5] A. AlDayel and W. Magdy, “Stance detection on social media: State of the art and trends,” *Inf. Process. \& Manag.*, vol. 58, no. 4, p. 102597, 2021.

[6] J. Ebrahimi, D. Dou, and D. Lowd, “A joint sentiment-target-stance model for stance classification in tweets,” in *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers*, 2016, pp. 2656–2665.

[7] C. Liu *et al.*, “Iucl at semeval-2016 task 6: An ensemble model for stance detection in twitter,” in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 2016, pp. 394–400.

[8] H. Elfardy and M. Diab, “Cu-gwu perspective at semeval-2016 task 6: Ideological stance detection in informal text,” in *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, 2016, pp. 434–439.

[9] N. Alturayeif, H. Luqman, and M. Ahmed, “A systematic review of machine learning techniques for stance detection and its applications,” *Neural Comput. Appl.*, vol. 35, no. 7, pp. 5113–5144, 2023.

[10] A. Brenner, “The prolonging Abyssinian Crisis,” 2024.

[11] S. W. Andemariam, “The Story of the Translation of the Bible into Təgre”.

[12] E. Ullendorff, *Ethiopia and the Bible: The Schweich Lectures 1967*. OUP Oxford, 1968.

[13] K. C. Gandar Dower, “The first to be freed: the record of British Military administration in Eritrea and Somalia, 1941-1943,” *(No Title)*, 1944.

[14] SBS, “ʾEs Bi ʾEs Təgrəññа.” [Online]. Available: https://www.sbs.com.au/language/tigrinya

[15] A. A. Hailu, A. T. Hayleslassie, D. W. Gebresilasie, R. E. Haile, T. T. Ghebremedhin, and Y. K. Tedla, “Tigrinya OCR: Applying CRNN for Text Recognition,” in *International Conference on Neural Information Processing*, 2023, pp. 456–467.

[16] A. Rahman, *English-Tigrigna Dictionary*. Simon Wallenberg Press, 2007.

[17] J. H. Martin and D. Jurafsky, *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*, vol. 23. Pearson/Prentice

Hall Upper Saddle River, 2009.

[18] A. AlDayel, “Stance characterization and detection on social media,” 2021.

[19] B. Schiller, J. Daxenberger, and I. Gurevych, “Stance detection benchmark: How robust is your stance detection?,” *KI-Künstliche Intelligenz*, pp. 1–13, 2021.

[20] W.-F. Chen and L.-W. Ku, “Utcnn: a deep learning model of stance classificationon on social media text,” *arXiv Prepr. arXiv1611.03599*, 2016.

[21] M. Lai, V. Patti, G. Ruffo, and P. Rosso, “Stance evolution and twitter interactions in an italian political debate,” in *Natural Language Processing and Information Systems: 23rd International Conference on Applications of Natural Language to Information Systems, NLDB 2018, Paris, France, June 13-15, 2018, Proceedings 23*, 2018, pp. 15–27.

[22] M. Taulé, F. M. R. Pardo, M. A. Mart\’\i, and P. Rosso, “Overview of the task on multimodal stance detection in tweets on catalan\# 1oct referendum.,” in *IberEval@ SEPLN*, 2018, pp. 149–166.

[23] E. Tromp and M. Pechenizkiy, “Senticorr: Multilingual sentiment analysis of personal correspondence,” in *2011 IEEE 11th International Conference on Data Mining Workshops*, 2011, pp. 1247–1250.

[24] M. Tutek *et al.*, “Takelab at semeval-2016 task 6: Stance classification in tweets using a genetic algorithm based ensemble,” in *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, 2016, pp. 464–468.

[25] G. Zarrella and A. Marsh, “Mitre at semeval-2016 task 6: Transfer learning for stance detection,” *arXiv Prepr. arXiv1606.03784*, 2016.

[26] K. Dey, R. Shrivastava, and S. Kaushik, “Topical stance detection for Twitter: A two-phase LSTM model using attention,” in *Advances in Information Retrieval: 40th European Conference on IR Research, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings 40*, 2018, pp. 529–536.

[27] S. Zhou, J. Lin, L. Tan, and X. Liu, “Condensed convolution neural network by attention over self-attention for stance detection in twitter,” in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8.

[28] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” *arXiv Prepr. arXiv1508.04025*, 2015.

[29] V. Ashish, “Attention is all you need,” *Adv. Neural Inf. Process. Syst.*, vol. 30, p. I, 2017.

[30] A. Vaswani *et al.*, “Attention is all you need,” *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[31] T. Wang, K. Lu, K. P. Chow, and Q. Zhu, “COVID-19 sensing: negative sentiment analysis on social media in China via BERT model,” *Ieee Access*, vol. 8, pp. 138162–138169, 2020.

[32] D. Sridhar, L. Getoor, and M. Walker, “Collective stance classification of posts in online debate forums,” in *Proceedings of the Joint Workshop on Social Dynamics and Personal Attributes in Social Media*, 2014, pp. 109–117.

[33] I. El Alaoui, Y. Gahi, R. Messoussi, Y. Chaabi, A. Todoskoff, and A. Kobi, “A novel adaptable

approach for sentiment analysis on big social data," *J. Big Data*, vol. 5, no. 1, pp. 1–18, 2018.

[34] I. Gupta and N. Joshi, "Enhanced twitter sentiment analysis using hybrid approach and by accounting local contextual semantic," *J. Intell. Syst.*, vol. 29, no. 1, pp. 1611–1625, 2019.

[35] R. N. Waykole and A. D. Thakare, "A review of feature extraction methods for text classification," *Int. J. Adv. Eng. Res. Dev*, vol. 5, no. 04, pp. 351–354, 2018.

[36] J. Okell, *A reference grammar of colloquial Burmese*. Routledge, 1969.

[37] M. Baroni, G. Dinu, and G. Kruszewski, "Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, pp. 238–247.

[38] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv Prepr. arXiv1301.3781*, 2013.

[39] J. Borge-Holthoefer, W. Magdy, K. Darwish, and I. Weber, "Content and network dynamics behind Egyptian political polarization on Twitter," in *Proceedings of the 18th ACM conference on computer supported cooperative work \& social computing*, 2015, pp. 700–711.

[40] S. Baird, D. Sibley, and Y. Pan, "Talos targets disinformation with fake news challenge victory," *Fake News Chall.*, 2017.

[41] D. Effrosynidis, S. Symeonidis, and A. Arampatzis, "A comparison of pre-processing techniques for twitter sentiment analysis," in *Research and Advanced Technology for Digital Libraries: 21st International Conference on Theory and Practice of Digital Libraries, TPDL 2017, Thessaloniki, Greece, September 18-21, 2017, Proceedings 21*, 2017, pp. 394–406.

[42] P. Nakov, A. Ritter, S. Rosenthal, F. Sebastiani, and V. Stoyanov, "SemEval-2016 Task 4: Sentiment Analysis in Twitter," pp. 1–18, Accessed: Apr. 01, 2025. [Online]. Available: https://github.com/aritter/twitter

[43] J. Parts, "A small 6-chromatic two-distance graph in the plane," *arXiv Prepr. arXiv2010.12656*, 2020.

[44] S. Kiritchenko, X. Zhu, and S. M. Mohammad, "Sentiment analysis of short informal texts," *J. Artif. Intell. Res.*, vol. 50, pp. 723–762, 2014.

[45] A. Zadeh, S. Poria, E. Cambria, and T. Morency, "Tensor-based sentiment classification of movie reviews," *J. Inf. Sci.*, vol. 43, no. 3, pp. 407–417, 2017.

[46] L. Barbosa and J. Feng, "Robust sentiment detection on twitter from biased and noisy data," in *Coling 2010: Posters*, 2010, pp. 36–44.

[47] W. Wang, S. H. Kim, and J. Oh, "Sentiment analysis of microblogging data with human factors for disaster management," *IEEE Trans. Comput. Soc. Syst.*, vol. 4, no. 4, pp. 1010–1022, 2017.

[48] B. Molla, "Target identification, stance classification, and sentiment analysis on social media data using CNN and BI-LSTM," 2021.

[49] I. Augenstein, T. Rocktäschel, J. Vlachos, and K. Bontcheva, "SemEval-2016 Task 4: Sentiment

analysis in Twittero Title," in *SemEval-2016 Task 4: Sentiment analysis in Twitter*, 2016, pp. 1–18.

[50] F. Yonnas, "Development of stemming algorithm for tigrigna text," *Dep. Inf. Sci. Addis Ababa, Ethiop.*, 2011.

[51] M. Gasser, "Semitic morphological analysis and generation using finite state transducers with feature structures," in *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, 2009, pp. 309–317.

[52] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Comput. networks ISDN Syst.*, vol. 30, no. 1–7, pp. 107–117, 1998.

[53] M. Gasser, "HornMorpho: a system for morphological processing of Amharic, Oromo, and Tigrinya," in *Conference on Human Language Technology for Development, Alexandria, Egypt*, 2011, pp. 94–99.

[54] S. Narkhede, "Understanding Confusion Matrix Towards Data Science. Towards Data Science," 2018.

[55] A. Sharma, "Confusion matrix in machine learning," *Geeksforgeeks. org. https//www. geeksforgeeks. org/confusion-matrixmachine-learning/(accesed August 5, 2021)*, 2018.

[56] T. B. Arnold, "kerasR: R interface to the keras deep learning library.," *J. Open Source Softw.*, vol. 2, no. 14, p. 296, 2017.

[57] S. Pattanayak and S. Pattanayak, "Introduction to deep-learning concepts and TensorFlow," *Pro Deep Learn. with TensorFlow A Math. Approach to Adv. Artif. Intell. Python*, pp. 89–152, 2017.

[58] J. Bernard, "Python data analysis with pandas," in *Python Recipes Handbook: A Problem-Solution Approach*, Springer, 2016, pp. 37–48.

[59] D. Paper, "Scikit-Learn Regression Tuning," in *Hands-on Scikit-Learn for Machine Learning Applications: Data Science Fundamentals with Python*, Springer, 2019, pp. 189–213.

[60] I. Sutskever, O. Vinyals, and Q. V Le, "Sequence to sequence learning with neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 27, 2014.

APPENDIX

APPENDIX A: The Tigrinya script

	ä [ɐ]	u [u]	i [i]	a [a]	e [e]	(ə) [ɨ]	o [o]
ha [h]	ሀ hä	ሁ hu	ሂ hi	ሃ ha	ሄ he	ህ h(ə)	ሆ ho
la [l]	ለ lä	ሉ lu	ሊ li	ላ la	ሌ le	ል l(ə)	ሎ lo
ḥa [ḥ]	ሐ ḥä	ሑ ḥu	ሒ ḥi	ሓ ḥa	ሔ ḥe	ሕ ḥ(ə)	ሖ ḥo
ma [m]	መ mä	ሙ mu	ሚ mi	ማ ma	ሜ me	ም m(ə)	ሞ mo
ra [r]	ረ rä	ሩ ru	ሪ ri	ራ ra	ሬ re	ር r(ə)	ሮ ro
sa [s]	ሰ sä	ሱ su	ሲ si	ሳ sa	ሴ se	ስ s(ə)	ሶ so
ša [ʃ]	ሸ šä	ሹ šu	ሺ ši	ሻ ša	ሼ še	ሽ š(ə)	ሾ šo
q'a [q']	ቀ q'ä	ቁ q'u	ቂ q'i	ቃ q'a	ቄ q'e	ቅ q'(ə)	ቆ q'o
q'wa [q'ʷ]	ቈ qw'ä		ቍ qw'i	ቋ qw'a	ቌ qw'e	ቍ qw'(ə)	
q'a [qʰ]	ቐ q'ä	ቑ q'u	ቒ q'i	ቓ q'a	ቔ q'e	ቕ q'(ə)	ቖ q'o
q'wa [qʰʷ]	ቘ qw'ä		ቝ qw'i	ቛ qw'a	ቜ qw'e	ቝ qw'(ə)	
ba [b]	በ bä	ቡ bu	ቢ bi	ባ ba	ቤ be	ብ b(ə)	ቦ bo
ta [t]	ተ tä	ቱ tu	ቲ ti	ታ ta	ቴ te	ት t(ə)	ቶ to
ča [ʧ]	ቸ čä	ቹču	ቺ či	ቻ ča	ቼ če	ች č(ə)	ቾ čo
na [n]	ነ nä	ኑ nu	ኒ ni	ና na	ኔ ne	ን n(ə)	ኖ no
ña [ɲ]	ኘ ñä	ኙ ñu	ኚ ñi	ኛ ña	ኜ ñe	ኝ ñ(ə)	ኞ ño
ʾa [ʔ]	አ ʾä	ኡ ʾu	ኢ ʾi	ኣ ʾa	ኤ ʾe	እ ʾ(ə)	ኦ ʾo
ka [k]	ከ kä	ኩ ku	ኪ ki	ካ ka	ኬ ke	ክ k(ə)	ኮ ko

	ä/e [ɐ]	u [u]	i [i]	a [a]	e [e]	(ə) [ɨ]	o [o]
kwa [kʷ]	ኰ kwä		ኵ kwi	ኳ kwa	ኴ kwe	ኵ kw(ə)	
xwa [xʷ]	ኸ xwä		ኹ xwi	ኻ xwa	ኼ xwe	ኽ xw(ə)	
wa [w]	ወ wä	ዉ wu	ዊ wi	ዋ wa	ዌ we	ው w(ə)	ዎ wo
ʿa [ʕ]	ዐ ʿä	ዑ ʿu	ዒ ʿi	ዓ ʿa	ዔ ʿe	ዕ ʿ(ə)	ዖ ʿo
za [z]	ዘ zä	ዙ zu	ዚ zi	ዛ za	ዜ ze	ዝ z(ə)	ዞ zo
ža [ʒ]	ዠ žä	ዡ žu	ዢ ži	ዣ ža	ዤ že	ዥ ž(ə)	ዦ žo
ya [j]	የ yä	ዩ yu	ዪ yi	ያ ya	ዬ ye	ይ y(ə)	ዮ yo
da [d]	ደ dä	ዱ du	ዲ di	ዳ da	ዴ de	ድ d(ə)	ዶ do
ǧa [ʤ]	ጀ ǧä	ጁ ǧu	ጂ ǧi	ጃ ǧa	ጄ ǧe	ጅ ǧ(ə)	ጆ ǧo
ga [g]	ገ gä	ጉ gu	ጊ gi	ጋ ga	ጌ ge	ግ g(ə)	ጎ go
gwa [gʷ]	ጐ gwä		ጕ gwi	ጓ gwa	ጔ gwe	ጕ gw(ə)	
ṭa [tʼ]	ጠ ṭä	ጡ ṭu	ጢ ṭi	ጣ ṭa	ጤ ṭe	ጥ ṭ(ə)	ጦ ṭo
č̣a [ʧʼ]	ጨ č̣ä	ጩ č̣u	ጪ č̣i	ጫ č̣a	ጬ č̣e	ጭ č̣(ə)	ጮ č̣o
p̣a [pʼ]	ጰ p̣ä	ጱ p̣u	ጲ p̣i	ጳ p̣a	ጴ p̣e	ጵ p̣(ə)	ጶ p̣o
ṣa [sʼ]	ጸ ṣä	ጹ ṣu	ጺ ṣi	ጻ ṣa	ጼ ṣe	ጽ ṣ(ə)	ጾ ṣo
fa [f]	ፈ fä	ፉ fu	ፊ fi	ፋ fa	ፌ fe	ፍ f(ə)	ፎ fo
pa [p]	ፐ pä	ፑ pu	ፒ pi	ፓ pa	ፔ pe	ፕ p(ə)	ፖ po
va [v]	ቨ vä	ቩ vu	ቪ vi	ቫ va	ቬ ve	ቭ v(ə)	ቮ vo

APPENDIX B: Tigrigna Stop Words

ኣብቲ	ኩሉ	በዙይ	የለዋን
ማለት	ኣነ	ናይዞም	ምስ
እዩ	ንስኻ	ኩሎም	ከማይ
ከም	ንስኺ	ኢሎም	የብሉን
ኑቲ	ዘለው	ዘለክን	ከማና
ኣብ	ንሱ	ናይቲ	ኩለን
ከማኻ	ንሳ	እንተድኣ	እያ
ከምዘለኺ	ንሕና	ናይ	ናይዘን
ከምዘለዎ	ንስኻትኩም	ዘለዎም	ናይቶም
እቲ	ንስኻትክን	ከምኡውን	ዝኾነ ኮይኑ
ወይ	ሲጋዕ	እዮም	ኣብዛ
ድማ	ብዛዕባ	ካብተን	ኣብዚኣ
ክሳብ	ከለዋ	ድኣ	ኣላ
መን	ነታ	ናይዞም	የብልናን
ኣበይ	እታ	ናይዘን	የብልካን
እምበር	ናብ	ናይቶም	ኣለ
በቶም	ዘለኒ	ናይተን	ከምተን
ነዚ	ግን	ኣብዚኣም	ኢሉ
ይኹን	ጥራሕ	እዙይ	ኢላ
ኣሎ	ካብቶም	ዘለዋ	ኢለን
እዚ	እዞም	የብለንን	ኣይኮኑን
ዝኾነ	እኳ	የብለይን	የብሎምን
ኸዓ	ምኳኖም	እንተዝኾና	ከምዚኣቶ ም
እዛ	ብኣኣም	እዚኣም	ከምዚኣተ ን
ኣይኮነን	በቲ	ዘለወን	ኣይኮነትን
እንታይ	ብቲ	ከምዘለኪ	ከምቶም
እታ	ክኾና	ስለዝኾነ	ዘለና

APPENDIX C: Data cleaning code snapshot

```
import re
import pandas as pd
from cleantext import clean

def remove_punc_and_special_chars(text):
    # Remove punctuation and special characters
    normalized_text = re.sub(r'[!@#$%^«»&*()…\[\]{}⁍⁞]', '', text)
    return normalized_text

def remove_ascii_and_numbers(text_input):
    # Remove ASCII characters and numbers
    rm_num_and_ascii = re.sub(r'[A-Za-z0-9]', '', text_input)
    return re.sub(r'[\u1369-\u137C]+', '', rm_num_and_ascii)

def clean_symbols(text):
    # Remove emojis and symbols using clean-text library
    return clean(text, no_emoji=True)

def clean_data(dataset):
    cleaned_data = []

    for index, row in dataset.iterrows():
        # Assuming the text to clean is in the first column
        text = row[0]

        # Step 1: Remove punctuation and special characters
        text = remove_punc_and_special_chars(text)

        # Step 2: Remove numbers and ASCII characters
        text = remove_ascii_and_numbers(text)

        # Step 3: Remove symbols and emojis
        text = clean_symbols(text)

        cleaned_data.append(text)

    return pd.DataFrame(cleaned_data, columns=['Cleaned Data'])

# Example usage
if __name__ == "__main__":
    # dataset Loading)
    dataset =
pd.read_csv(C:\Users\PC\Desktop\researches\nigsti\MyDataset.csv')

    cleaned_dataset = clean_data(dataset)
    cleaned_dataset.to_csv('cleaned_dataset.csv', index=False)
```

APPENDIX D: Data cleaning Algorithm

INPUT: Uncleaned data
OUTPUT: cleaned dataset
START:
1: Read the dataset

2: WHILE (it is not the end of the file):

IF data include punctuations [',', '.', '"', ':', ')', '(', '-', '!', '?', '፡','፣', '፤'] THEN

Eliminate punctuations

```
def remove_punc_and_special_chars(text):
    normalized_text
=re.sub('[\!\@\#\$\%\^\«\»\&\*\(\)\…\[\]\{}\፡\፤\?\
    return normalized_text
```

IF data include number and English characters [꣐ ꣑ ꣒ ꣓ ꣔ ꣕ ꣖ ꣗ ꣘ ꣙ ꣚ ꣛ ꣜ ꣝ ꣞ ꣟ ꣠ ꣡ ꣢ ꣣ ꣤ A B C . . . a b c . . . ๐ ๑ ๒ …] THEN

Eliminate Number

```
def remove_ascii_and_numbers(text_input):
      rm_num_and_ascii=re.sub('[A-Za-z0-9]','',text_input)
      return re.sub('[\'\u1369-u137C\']+','',rm_num_and_ascii)
```

IF data include symbols [

….] THEN

Eliminate Symbols

```
      from clean-text import clean
      clean(text, no_emoji=true)
```

Return cleaned data

3. STOP

APPENDIX E: Numerals

፩	፪	፫	፬	፭	፮	፯	፰	፱	፲
ሃደ	ኪሊ.ተ	ስለስተ	ዓርብተ	ሓሙ'ሽተ	ሸዱ'ሽተ	ሾብዓተ	ሾሞንተ	ተ'ሽዓተ	ዓስርተ
hade	kilite	seleste	arbate	hamuschte	chiduchte	cheviate	chomonte	tichiate	aserte
1	3	3	4	5	6	7	8	9	10

፳	፴	፵	፶	፷	፸	፹	፺	፻
isira	salasa	arbaah	hamssa	susa	sebiah	semaniah	tesiah	miiti
20	30	40	50	60	70	80	90	100

APPENDIX F: Tigrigna Punctuations

፣	።	፤	፥	፦	፧
comma	full stop / period	colon	semi-colon	preface colon	question mark (no longer used)