



Jimma University
Jimma Institute of Technology
Faculty of Computing and Informatics

**Hate Speech Detection Using Deep Recurrent Neural
Networks for Amharic Text**

Bereket Girma

A Thesis Submitted to the School of Graduate Studies of Jimma University
in Partial Fulfillment of Masters of Science in Information Technology

January, 2021

Jimma, Ethiopia

Jimma University

Jimma Institute of Technology


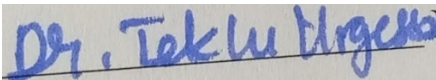

Faculty of Computing and Informatics

HATE SPEECH DETECTION USING DEEP RECURRENT NEURAL NETWORKS FOR AMHARIC TEXT

By: Bereket Girma

This is to certify that the thesis prepared by **Bereket Girma**, entitled **Hate Speech Detection Using Deep Recurrent Neural Networks for Amharic Text**, submitted in partial fulfillment of the requirements for the Degree of Master of Science in *Information Technology* complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Approved by board of Examining Committee:

	Name	Signature
Faculty Dean:	_____	_____
Advisor:	Dr. Girum Ketema	
External Examiner:		
Internal Examiner:	_____	_____
Chair Person:	_____	_____

Jimma, Ethiopia

January, 2021

Declaration

This thesis is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with proper citation of sources. I, the undersigned, declare that this thesis has not been presented for a degree in any other university.

Declared By:-

Name:- Bereket Girma

Signature: _____

Date: January, 2021

Confirmed By Advisor:

Name:- Dr. Girum Ketema

Signature: _____

Date: January, 2021

Acknowledgments

First and foremost, I would like to thank God for giving me his strength and ability to undertake this research study. Without his countless blessing, everything would not have been possible.

I would like to express my deep and sincere gratitude to my research advisor, Dr. Girum Ketema for his valuable time, guidance, constructive ideas, comments which enabled me to gain good research experience.

I would also like to thank Mr. Dessalew Yohannes (MSc.) for his valuable time, constructive ideas, and comments.

Finally, I must express my very profound gratitude to my parents, brothers, sisters, and friends for providing me with unfailing support and continuous encouragement throughout my years of study, through the process of researching and writing this thesis. This accomplishment would not have been possible without them Thank you.

Bereket Girma

Abstract

The exponential growth of social media such as Twitter and Facebook have revolutionized communication and content publishing but are also increasingly exploited for the propagation of hate speech and the organization of hate-based activities. Hate speech is a serious and growing problem in Ethiopia, both online and offline. It has a big contribution to the growing ethnic tensions and conflicts in Ethiopia that have created more than 1.4 million new internally displaced people in the first half of 2018 alone and it is serving as a fuel for the continued ethnic crimes in Ethiopia.

Previous works on Amharic hate speech detection chose to ignore the context in which the social media comments appeared and the sub-word information that would have improved the detection of hate speech in social media platforms where the users are careless about the spelling errors of their comment. This paper, employs a deep recurrent neural networks to capture the context of the social media comment and FastText word embedding for capturing the sub-word information. The proposed approach aims at investigating the importance of sub-word and context information for Amharic hate speech detection in social media platforms. The author treated the post-text, previous comment, and post metadata information as a context for predicting the hate-ness of a target comment in social media posts.

Our experiments show that using a feature that can capture sub-word information like FastText improved the accuracy of Amharic hate speech detection from 81.58% to 84.78% than using the word2vec feature. Additionally, that incorporating context information improves the accuracy of hate speech detection system from 81.73% to 85.87% and F-Score from 82.83% to 86.45% than using just the target comments.

Key-words: *Amharic hate speech detection, Hate Speech, Deep learning, Amharic posts and comments, context information for hate speech*

Contents

Abstract	iv
List of Tables.....	ix
List of Figures	x
List of Abbreviations.....	xi
1. Introduction	1
1.1. Background	1
1.2. Statement of the Problem	1
1.3. Objectives.....	3
1.3.1. General Objective	3
1.3.2. Specific Objectives	3
1.4. Methodology	4
1.5. Scope	4
1.6. Significance of the Study	5
1.7. Thesis Organization.....	5
2. Literature Review	7
2.1. Introduction	7
2.2. Traditional Machine Learning.....	7
2.3. Deep Learning	10
2.4. Hate Speech Detection for Amharic Language.....	13
3. Hate Speech Detection System	15
3.1. Overview	15
3.2. Definition of Hate Speech.....	15
3.3 Hate Speech Detection Approaches	16

3.3.1.	Keyword Approaches	17
3.3.2.	Machine Learning Approaches.....	17
3.3.3.	Content preprocessing and Feature Selection.....	17
3.3.3.1.	Bag of Words (BOW)	18
3.3.3.2.	N-grams.....	18
3.3.3.3.	TF-IDF	18
3.3.3.4.	Part of Speech.....	19
3.3.3.5.	Word Embeddings.....	19
3.3.4.	Hate Speech Detection Classification Algorithms	19
3.3.4.1.	Naïve Bayes.....	19
3.3.4.2.	Support Vector Machine (SVM).....	19
3.3.4.3.	Logistic Regression.....	20
3.3.5.	Deep Learning	20
3.3.5.1.	Convolutional Neural Network Ensemble	20
3.3.5.2.	C – GRU.....	20
3.3.5.2.	Long Short-term Memory	21
3.3.	Challenges of Hate Speech Detection System	21
4.	Design and Implementation	23
4.1.	Overview	23
4.2.	Preprocessing	24
4.2.1.	Cleaning	25
4.2.2.	Normalization	25
4.2.3.	Tokenization	27
4.3.	Word Vector.....	28

4.4.	Features	29
4.4.1.	Post Text Feature	29
4.4.2.	Previous Comment Feature.....	30
4.4.3.	Target Comment Feature	31
4.4.4.	Post metadata Feature	32
4.5.	Deep Learning Model.....	33
4.5.1.	BiLSTM (Bidirectional Long Short-Term Memory).....	33
4.5.2.	Attention Mechanism.....	33
4.5.3.	Batch Normalization	34
4.5.4.	Concatenation Layer	34
4.5.5.	Classification Layer	34
5.	Experiments.....	35
5.1.	Data Collection.....	35
5.2.	Data Annotation	35
5.3.	Dataset Description	37
5.4.	Development Tools	40
5.5.	Evaluation Metrics	41
5.6.	Experimental Setup	43
5.7.	Experimental Scenarios.....	45
5.8.	Results	47
5.8.1.	Importance of Sub-word information for Amharic Hate speech detection system	47
5.8.2.	Importance of Context Information for Amharic Hate speech detection	51
5.9.	Threats to Validity.....	55
6.	Conclusion and Future Works.....	56

6.1. Conclusions	56
6.2. Contributions.....	57
6.3. Future Works.....	57
References.....	59
Appendix I – List of Amharic Normalized words	64
Appendi II – List of Amharic normalized characters	65
Appendix III- Some of Amharic short words	66

List of Tables

Table 4.1 comments before and after the cleaning process 25

Table 4.2 E.g. Comments before and after cleaning and tokenization 28

Table 4.3 Example post texts before and after feature extraction..... 30

Table 4.4 Example previous comments before and after feature extraction..... 31

Table 4.5 Example target comments before and after feature extraction 32

Table 4.6 Example post metadata feature 32

Table 5.1 Example Annotated Comments 37

Table 5.2 Information about dataset used to train FastText and Word2vec embedding 38

Table 5.3 Dataset Information for a binary classifier 38

Table 5.4 dataset information used to train the four class classifier..... 39

Table 5.5 Specifications of azure compute instance used for training FastText 44

Table 5.6 Specifications of colab machine used for training..... 44

Table 5.7 word similarity test results with word2vec word vector..... 47

Table 5.8 word similarity test with FastText word vector 48

Table 5.9 Comparison of FastText embedding (which can capture sub-word information) feature and word2vec embedding feature for Amharic Hate speech detection system 49

Table 5.10 Performance of BLSTM with and without Attention and with and without Context Information 51

Table 5.11 Results of classifying comments into four categories (Political-hate, Ethnic-hate, Religious-hate and Neutral) 54

List of Figures

Figure 3.1 The CNN+GRU network architecture Source: [42] 21

Figure 3.2 LSTM Architecture Source: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/> 21

Figure 4.1 Bi-Directional LSTM Architecture 33

Figure 5.1 Dataset distribution when using only two classes (i.e., hate and neutral) 39

Figure 5.2 dataset distribution when using four classes (i.e., political-hate, ethnic-hate, religious-hate, or neutral) 40

Figure 5.3 Comparison of different fasttext and word2vec feature for Amharic hate speech detection 49

Figure 5.4 performance of BLSTM with and without attention and with and without context information 52

Figure 5.5 Summary of experiments to classify comments into four classes (Political-hate, Ethnic-hate, Religious-hate, Neutral) 54

List of Abbreviations

The following are some abbreviations used in this Thesis document

BI-LSTM	Bidirectional Long Short-Term Memory
CBOW	Continuous Bag of Words
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSV	Comma Separated Value
DNN	Deep Neural Network
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
NB	Naïve Bayes
NLTK	Natural Language Toolkit
POS	Part of Speech
RNN	Recurrent Neural Network
RF	Random Forest
SG	Skip-Gram
SVM	Support Vector Machine
TF-IDF	Term Frequency- Inverse Document Frequency

1. Introduction

1.1. Background

According to Hate Speech and Disinformation Prevention and Suppression Proclamation [1], Hate speech means a speech that promotes hatred, discrimination, or attack against a person or an identifiable group, based on ethnicity, religion, race, gender, or disability. On one hand, on the internet, social networks, in particular, people are more likely to adopt aggressive behavior because of the anonymity provided by these environments [2]. On the other hand, people have an increased willingness to express their opinions online, thus contributing to the propagation of hate speech as well. Since this type of prejudiced communication can be extremely harmful to society, governments, and social network platforms, they can benefit from detection and prevention tools.

Since mid-2018, Ethiopia has experienced serious communal violence that may have been provoked or exacerbated by online speech that fomented ethnic tension and violence, according to the report of Human Rights Watch [3]. Hate and dangerous speech is a serious and growing problem in Ethiopia, both online and offline. It has contributed to the growing ethnic tensions and conflicts in Ethiopia that have created more than 1.4 million new internally displaced people in the first half of 2018 alone and it is serving as a fuel for the continued ethnic tensions and hate crimes in Ethiopia [4]. The House of Peoples' Representatives approved Hate Speech and Disinformation Prevention and Suppression Proclamation on 13 February 2020 to prevent and suppress the dissemination of hate speech and disinformation. But because of the anonymity provided by the social media platforms it is difficult to eliminate online hate speech by conventional law enforcement.

1.2. Statement of the Problem

The exponential growth of social media such as Twitter and Facebook have revolutionized communication and content publishing but are also increasingly exploited for the propagation of hate speech and the organization of hate-based activities. The anonymity and mobility afforded by such media have made the breeding and spread of hate speech, eventually leading to hate crime effortless in a virtual landscape beyond the realms of traditional law enforcement [5]. Social media

companies such as Twitter, Facebook, and YouTube have been combating this issue and it has been estimated that hundreds of millions of euros are invested every year on counter-measures including manpower and developing some automated tools.

However, those automated tools are highly language-dependent and the tools are only developed for a few European languages, hate speech remains a problem mainly for under-resourced languages. Even though there are more than 21.14 million internet users and more than 6.4 million social media users in Ethiopia in 2020 [6], only a single research has been conducted for Amharic Hate speech detection [5] and there are still some uncovered concepts.

The previous work [5] has employed Naïve Bayes algorithm for the classification. Naïve Bayes assumes that features are independent to each other which means in our case it assumes the meanings of the words in a given comment are independent to each other. But in real world scenarios the meaning of the words are very dependent to each other, the meaning of the sentence is decided by the droplets of meaning that each word adds to the sentence not just by one word. The first word in the comment might decide the meaning of the fourth or fifth word in the comment. The previous work inherits this limitations of its approach. In recent years, deep recurrent neural network has shown itself that it can capture sequential information, the relationship among words in a sentence and has scored good result in many text classification tasks [7]. This research will employ Bi-Directional Long Short-Term Memory with some other layers for classification to overcome the limitations of the previous work.

This study will investigate the importance of features that can capture sub-word information like FastText feature for improving Amharic hate speech detection systems especially, in social media platforms where the user doesn't seem to care about the misspelling of the comments. We will pre-train a FastText embedding model and use it as a feature for the deep learning classifier to overcome the misspelled writing of the comments as it can capture character-level information.

Additionally, this work will study the importance of contexts for improving the performance of Amharic hate speech detection systems in social media posts. To show the effect of contexts in the Amharic hate speech detection system we will treat the post's text message, the previous comment, and the post's metadata as a context for classifying the target comment in social media posts.

The previous work [5] classifies a particular comment as a Hate or Neutral (Not-hate), but it might be more useful if we can further classify hate speech as political, religious, or ethnic. For instance, if a particular type of hate speech (political, religious, or ethnic) becomes dominant, governments and other concerning bodies can align their policies towards combating that type of hate speech because the policies and strategies that fit one type of hate speech might not fit well for the other type of the hate speech.

This research paper will in particular aim to answer the following research questions (RQ).

RQ1. Can sub-word information be used to improve the performance of the Amharic hate speech detection system in social media platforms where the user doesn't care about the misspelling of the comments?

RQ2. Can the context of the target comment to be classified be used to improve the performance of the Amharic hate speech detection system?

Finally, even though hate speech has been a very pressing problem for Ethiopia, still there is no public dataset available for the study. This research will exploit the Graph API of Facebook to collect Amharic comments from Facebook and then annotate and prepare it for building the model.

1.3. Objectives

1.3.1. General Objective

The general objective of this thesis is to investigate the importance of sub-word information in social media platforms where the user doesn't seem to care about the misspelling of the comments and the importance of the context for Amharic hate speech detection.

1.3.2. Specific Objectives

Specifically, this study focuses on:

- To review in-depth previous works that have been done on Amharic Hate Speech Detection and other related works
- To collect hate speech comments spread around social media by exploiting facebook's Graph API

- To Annotate collected social media comments as political-hate, ethnic-hate, religious-hate or neutral
- To preprocess the collected dataset
- To generate Word2Vec and FastText feature vector to represent semantic and syntactic features of the language and to investigate the importance of sub-word information
- To build the model using Keras API with TensorFlow backend, a tensor processing framework
- To evaluate the performance of the model using Precision, Recall, F-Score, Accuracy
- To Interpret and discuss the result of the study

1.4. Methodology

The methodology followed to address the problems mentioned above includes the following tasks.

Literature review: is done on the researches studied previously on Amharic hate speech Detection Systems and other related papers to understand the necessary concepts for the study.

Data collection: the comments on different social media and news pages will be collected via a web crawler.

Data Annotation: - the collected social media comments will be annotated or labeled as one of four classes political-hate, religious-hate, ethnic-hate, or neutral.

Preprocessing: includes cleaning, normalizing, and tokenizing the corpus.

Model building: is implemented the model using Keras with TensorFlow backend, using Bi-directional Long Short-Term Memory with some other layers.

Model evaluating: is done to measure the performance of the developed models using precision, recall, F-measure, and accuracy

1.5. Scope

Nowadays, Hate Speech has been spread across many communication channels like social media, broadcasting media, printing media, and others. In recent years, hate speech has been spreading

rapidly over social media. This study focuses on hate speech spread over online social media and news pages.

Even though Hate Speech can have any types of speech forms such as verbal, textual, graphical, or other, this study focuses on only Amharic Textual hate speeches, spread over social media and news pages. Besides classifying speech as Hate or Neutral this thesis also further classifies hate as a political, ethnic, or religious hate speech.

1.6. Significance of the Study

The beneficiaries of this study are the government, different ethnic groups, social media platform providers, and all Ethiopians that are suffering from hate speech spread across social media and news page.

- The government will be benefited from this system, by using this system it can detect hate speeches spread across public social media and news pages to strengthen the law enforcement on hate speech thereby meeting the goal of reducing hate speech.
- Different ethnic groups in Ethiopia will be benefited from this system. As hate speech encourages crimes against other ethnic groups in different areas the group that has small numbers feels insecure and this system will help to reduce such types of crimes thereby increasing the security of ethnic groups that small number in different parts of Ethiopia.
- As the law obligates any online platform provider not to encourage crime on their platforms, they can easily deploy filtering mechanisms of hate speech from their platform.

1.7. Thesis Organization

The rest of this document is organized as follows. Chapter Two discusses previous research works on both resource-rich and Amharic language and state of the art approaches for hate speech detection.

Chapter Three discusses a general overview of Amharic language, challenges in developing hate speech detection for the Amharic Language, and how LSTM and word2vec, fasttext algorithms work. Our proposed approach for sentiment analysis is elaborated in Chapter Four. The experimental setups, procedures, evaluation metrics, results, and discussion are discussed in

Chapter Five. Finally, Chapter Six discusses the conclusion and future research direction on Amharic hate speech detection

2. Literature Review

2.1. Introduction

Due to the massive rise of user-generated web content, in particular on social media networks, the amount of hate speech is also steadily increasing. Over the past years, interest in online hate speech detection and particularly the automatization of this task has continuously grown, along with the societal impact of the phenomenon. Natural language processing focusing specifically on this phenomenon is required since basic word filters do not provide a sufficient remedy on what is considered a hate speech message might be influenced by aspects such as the domain of an utterance, its discourse context, as well as context consisting of co-occurring media objects (e.g., images, videos, audio).

Nowadays, due to the societal impact of hate speech, various researches have been made using different techniques of Traditional Machine Learning (i.e., Random Forest, SVM, Naïve Bayes, etc.) and Deep Learning Techniques such as Convolutional Neural Networks and Recurrent Neural Networks. Most of the previous works use traditional machine learning classifiers, such as logistic regression [8, 9, 10] and support vector machines [11], or ensemble classifiers of such traditional methods [2]. Some studies experiment with deep learning on this task, especially after the major advancements of the last years. Due to a large amount of related research concerning these tasks, we only analyze works that are most relevant to ours in terms of domain and methodology.

2.2. Traditional Machine Learning

Machine Learning techniques have shown themselves to be useful in various areas of Natural language Processing fields like text classification, sentiment analysis, etc. hate speech is also no exception, scholars have tried to detect offensive social media contents by applying different machine learning techniques. A large number of studies have been conducted to detect hate speech on social media however, it is difficult to cover all the researches. Therefore, we have compiled the most important researches as the following.

In 2016 Waseem and Hovy [9], proposed predictive Features for Hate Speech Detection on Twitter. The authors provided a list of criteria, founded in a critical race theory, used them to

annotate the tweets as hate or neutral, and made a 16k annotated dataset publicly available which is extracted from Twitter. The authors collected the dataset in two months and annotated 16,914 tweets out of 136,052 retrieved tweets, 3,382 sexist, 1,972 racist, and 11,559 for neither. Besides, the tweets the authors also collected metadata information like geographic location and gender. The authors used character n-gram and word n-gram as a feature and Logistic Regression as a classification algorithm. The researchers used 10-fold cross-validation to test the performance of the model and they found that the character n-gram feature outperforms the word n-gram by at least 5 points of F1 score. The authors also show that including location or word length meta information for training the model has no positive effect on the performance of the model in F1 score. Finally, the authors obtained their best result using character n-gram up to a length of 4 with gender meta information as a feature and logistic regression as a classification algorithm scoring 73.93 F1 scores.

In 2015 [12] proposed Hate Speech Detection with comment Embeddings. The authors proposed a two-step method for hate speech detection. Firstly, to construct paragraph2vec embedding model jointly for comments and words and learn their distributed representations in a vector space using a continuous bag of words (CBOW). Secondly, use the embedding to train logistic regression binary classifier. The authors used a dataset that comprises 56,280 hate speech comments and 895,450 clean comments. The researchers used vector dimensionality, $D = 200$, and length of context for word sequence, $C = 5$ while training paragraph2vec model for five iterations. After building the final model using logistic regression classifier the researchers tested it with 5-fold cross-validation and reported that their paragraph2vec comment embedding was able to capture the semantic meaning of words. Finally, the researchers showed that their final model developed with paragraph2vec comment embedding features having 0.8007 AUC (Area Under the Curve) has outperformed the preceding works which used BOW with *tf* and *tf-idf* feature and scored 0.7889 and 0.6933 AUC respectively.

In 2016 [13] proposed a dictionary-based approach to detect racism in Dutch social media. The authors defined racism as “all negative utterances, negative generalization, and insulting concerning ethnicity, nationality, religion, and culture”. After collecting corpus from two site’s social media comments of 100 most recent posts they retrieved 5759 comments and labeled them

as racist, non-racist and invalid (i.e. comments that are not in Dutch or it has only picture). The authors used the kappa score to calculate the inter-annotator agreement. After annotating the dataset, the authors manually extracted the terms pertained to racism from the training data to create the dictionary and then, expanded the dictionary automatically using word2vec. The authors used Support Vector Machine classifier and their best performing model obtained a 0.46 F-score for the racist class. The researchers found that the automatic expansion of the dictionary didn't help to improve the performance of the model and hypothesize that the words added by the word2vec model were irrelevant for the task of discriminating between racist and neutral text. The authors also mentioned that the downside of using a dictionary to detect racism is that they do not add context into consideration. Finally [14] as the data is collected from two sites that are skewed towards racism words are mostly used in their racist context which makes the model unable to distinguish between insult and racist comments with the same central word to address this author has promised to add more general social media texts in the future.

In 2017 [10] studied automated hate speech detection and the problem of offensive language and after using crowd-sourced hate speech lexicon to collect 85.4 million tweets containing hate speech keywords and took randomly 25k tweets and they labeled them as hate speech, offensive only and neither. The authors defined hate speech as a language that is used to express hatred towards a targeted group or is intended to derogatory, humiliate, or to insult members of the group. Each tweet was coded by three or more people and take the majority class as the label of the tweet.

As they took stricter criteria of hate speech Only 5% of the tweets were labeled hate speech and most of the tweets were labeled as offensive language. The authors' used the n-gram feature weighted by *tf-idf* and POS tag n-gram to capture the syntactic structure. The researchers also used a sentiment lexicon designed for social media to assign sentiment scores to each tweet and the number of characters, words, syllables in each tweet as a feature.

The author employed linear regression with L2 regularization as a classifier and they used a one-versus-rest framework in which a separate classifier is trained for each class and the class label with the highest predicted probability will be taken as the label for the tweet. The researchers obtained their best performing model with an overall precision of 0.91 and F1 score of 0.90 and mentioned the model performs well at prevalent forms of hate speech particularly anti-black racism

and homophobia but, less reliable at detecting hate speech that occurs infrequently. The authors also noticed that peoples tend to see sexist comments as offensive language and racist and homophobia as hate speech more. Finally, the authors suggested future works should focus on looking more closely at the social contexts and conversations in which hate speech occurs.

2.3. Deep Learning

In 2018 [15] proposed a predictive embedding for hate speech detection with the minimal tuning of parameters, fewer weight parameters, minimal preprocessing, and no meta information. The authors used the definition of hate speech from [10] work and three previously prepared datasets by [9], [10] and [16]. The researchers used 300-dimensional pre-trained Glove Common crawl word embeddings and transformed each embedding by applying 300-dimensional 1-layer Multi-Layer Perceptron with ReLu activation and then they employed max-pooling operation on the resulting word embeddings which are then denoted as m . The authors found the transformation of the embedding helps to better handle unseen or rare tokens and the max-pooling layer to capture salient word features from the input. They also average word embeddings to capture the overall meaning of a sentence and denoted as a . After concatenating a and m to form a document representation d and feed the representation into a 50 node 2-layer Multi-Layer Perceptron followed by ReLu Activation, the representation is passed to a fully connected SoftMax layer whose output is the probability distribution over labels. After padding each input to 50 words the authors trained RMSProp with a learning rate of 0.001, batch size of 512, and added a dropout with a drop rate of 0.1 on the final layer to reduce overfitting. The researchers used logistic regression classifier and tested their model with 10-fold cross-validation and obtained 0.86, 0.92, and 0.71 F1 scores from their best performing model on the three datasets used. The authors also reported that they compared their model with four preceding works on the same dataset and their model outperformed by as much as 12 F1 points. The authors concluded that the previous works that used deep learning models may rely on word embeddings for the bulk predictive power, not on the addition of sequence-based meta information. Finally, the authors planned to investigate character-based representations, using character CNNs and highway layers along with word embeddings to allow robust representations for sparse words such as hashtags.

In 2017 [17] proposed Deep Learning Approach for detecting hate speeches in a tweet and defined hate speech as an abusive speech targeting individuals (cyberbullying a politician, a celebrity, a group) or particular groups (a country, LGBT, a religion, gender, an organization, etc.). The authors used 16k annotated tweets and pre-trained Glove word embeddings and tried to classify a tweet as racist, sexist, or neither by experimenting with multiple classifiers such as Logistic Regression, Random Forest, SVMs, GBDT, and DNNs.

The authors explored state-of-the-art semantic embeddings like *tf-idf*, *BOW*, and char n-grams over Glove and task-specific embeddings learned using Fast Text, CNN, and LSTM with either Random embedding or Glove embedding initialization. While training the deep learning architectures, besides learning the weights the network also learned task-specific word embeddings tuned towards hate speech labels. Using these embeddings as a feature the researchers experimented with different classifiers such as SVM and GBDT. The authors used Adam optimizer for CNN and LSTM with a batch size of 128 and RMSProp optimizer with 64 batch size for FastText while training the model and 10-fold cross-validation for testing.

The researchers obtained their best model “LSTM + Random embedding + GBDT” where tweets are initialized with random embeddings, LSTM was trained with Adam optimizer, and then, learned embeddings were used to train a GBDT classifier and scored 0.930 F1 points. The authors found that the initialization with random embedding is slightly better than Glove embedding and also noted that similar words obtained using task-specific learned embeddings clearly show the hatred towards the target words which is in general not visible at all in similar words obtained using Glove. Finally, the authors promised to explore the importance of user network features for the task.

In 2017 [18] carried out a Comparative Study of CNN and RNN for Natural Language Processing (NLP) to get the insight of which architecture fits more for most of NLP tasks from three Deep Learning Architectures such as CNN, LSTM, and GRU. Authors trained those three architectures, on seven different NLP tasks such as

- Sentiment Classification on [19] dataset
- Relation classification on SemEval 2010 task 8 dataset [20]
- Textual Entailment on Stanford Language Inference dataset [21]

- Answer Selection on WikiQA [22]
- Question Relation Match on WebQSP dataset [23]
- Path Query Answering on Path query dataset [24]
- Part of Speech Tagging on Wall Street Journal (WSJ) data

While investigating the performance of three architectures the researchers always

- Trained these models from scratch and there was no additional information provided for the models (e.g. no pre-trained word embeddings)
- While CNN consists of convolution and max-pooling layer, for GRU and LSTM the input was provided from left to right and the last hidden state was taken as the final representation of the input
- Searched for the optimal parameters of each task and each model separately

Having this experiment setup, the authors reported that for the Sequence order and Context dependency tasks the results are as expected RNN models perform better but, for Text Classification and Sentiment Matching tasks the result was unexpected. As CNN's are considered good for extracting local and position-invariant information they were expected to perform better but, outperformed by RNNs in e.g., Sentiment Analysis which mostly depends on a few local regions. The researchers found that

- GRU performs better when the sentiment is determined by the entire sentence or long-range dependency rather than local key phrases
- As GRU models the whole sentence, sometimes this feature makes it hard for key phrase to play the main role in the representation of the sentence thereby missing the important feature and making the wrong generalization when the sentiment is determined by few key phrases
- GRU and CNN are comparable when the sentence length is short e.g. <10 tokens, then GRU performs better when the sentence length becomes longer

Additionally, the hidden size and batch size can make Deep learning model performance vary dramatically. Finally, the researchers concluded that RNN performs well robustly in a broad range of tasks, except when the task is essentially a key-phrase recognition task.

In 2018 [14] have studied Improving Hate Speech Detection using Deep Learning Ensembles by averaging the summation of soft-max results of 10 different models and taking the class with the highest average as the winning class. The authors used two datasets from previous works “Abusive Speech” of [9] and Sentiment Analysis of [25] and Keras with TensorFlow backend and switched Theano backend due to weight initialization reproducibility. The researchers used CNN with 50 token window size and padded with zero vectors in case the length of the tweet is less than 50 tokens, word embedding with 50 tokens by 400 dimensions, and n-grams as a feature for the classifiers. For the CNN the authors constructed the model by feeding the output of the convolution layer to the global max-pooling layer and then, to a hidden layer that has 250 units with a 0.2 dropout rate only for the SemEval dataset. After the hidden layer with 0.2 dropout rate, there is one hidden layer with ReLu activation and then, the output layer that has three nodes with sigmoid activation. The researchers used Glorot Uniform distribution for weight initialization and Adam optimizer with binary cross-entropy loss function for learning the weights. Having the above experiment setup, the authors obtained a 2.95% improvement of the ensemble model over individual sub-models and found that the result gets better when the batch size decreases and epochs increase. The authors also noted that the weight initialization method has an impact on the performance of the model and seeding the initialization with some value is good for reproducibility. Finally, the authors reported that their ensemble method performs better than individual models 98% of the time by testing with 10-fold cross-validation.

2.4. Hate Speech Detection for Amharic Language

A limited number of studies have been conducted on Hate speech detection in the Amharic language. This study is also to improve the work done by [5] and to add some additional features.

The author worked on the application of apache spark in hate speech detection to reduce the challenges. Authors developed an apache spark-based model to classify Amharic Facebook posts and comments into hate and not hate. Authors employed Random forest and Naïve Bayes for learning and Word2Vec and *tf-idf* for feature selection. Tested by 10-fold cross-validation, the model based on word2vec embedding performed best with 79.83% accuracy.

The author collected a dataset from social media posts and comments particularly from Facebook and annotated it. This research used 6,120 Amharic posts and comments, 4,882 for training, and 1,238 for testing.

3. Hate Speech Detection System

3.1. Overview

In recent years, we have seen exponential growth in the number of people using online forums and social networks. Every 60 seconds, there are 510,000 comments generated on Facebook [26] and around 350,000 tweets generated on Twitter [27]. The people interacting on these forums or social networks come from different cultures and educational backgrounds. sometimes, the difference in opinions leads to verbal assaults. Besides, people also have an increased willingness to express their opinions online [28], thus contributing to the propagation of hate speech as well. Moreover, unchecked freedom of speech over the web and the mask of anonymity that the internet provides incite people to use racist slurs or derogatory terms. This issue has shown to be increasingly important in the last decade and detecting or removing such content manually from the web is a tedious task. So, there is a need of devising an automated model that can detect such toxic content on the web which we call the Hate Speech Detection System.

Detecting abusive language is often more difficult than one expects for a variety of reasons. the noisiness of the data in conjunction with a need for world knowledge not only makes this a challenging task to automate but also potentially a difficult task for people as well. For Instance, the intentional obfuscation of words and phrases to evade manual or automatic checking often makes detection difficult. Obfuscations such as *ni9 9er*, *whoopiuglyniggerratgolberg*, and *JOOZ* make it impossible for simple keyword spotting metrics to be successful, especially as there are many permutations to a source word or phrase. Conversely, the use of keyword spotting could lead to false positives. Besides, deciding if a portion of text contains hate speech is not simple, even for humans. Hate speech is a complex phenomenon, intrinsically associated with relationships between groups, and also relying upon language nuances. This is notorious in the low agreement found between annotators in the process of building new collections. Therefore, it is crucial to clearly define hate speech to make the task of its automatic identification easier.

3.2. Definition of Hate Speech

The definition of hate speech is neither universally accepted nor are individual facets of the definition fully agreed upon. Ross, et al. believe that a clear definition of hate speech can help the study of detecting hate speech by making annotating hate speech an easier task, and thus, making

the annotations more reliable [29]. However, the line between hate speech and appropriate free expression is blurry, making some wary to give hate speech a precise definition. For instance, the American Bar Association does not give an official definition but instead asserts that speech that contributes to a criminal act can be punished as part of a hate crime [30]. The leading definitions of hate speech from varying sources, as well as some aspects of the definitions that make the detection of hate speech difficult have been summarized as follows.

1. Twitter: “Hateful conduct: You may not promote violence against or directly attack or threaten other people based on race, ethnicity, national origin, sexual orientation, gender, gender identity, religious affiliation, age, disability, or serious disease.” [31]
2. Facebook: “We define hate speech as a direct attack on people based on what we call protected characteristics race, ethnicity, national origin, religious affiliation, sexual orientation, caste, sex, gender, gender identity, and serious disease or disability. We define attack as violent or dehumanizing speech, harmful stereotypes, statements of inferiority, or calls for exclusion or segregation.” [32]
3. YouTube: “Hate speech refers to the content promoting violence or hatred against individuals or groups based on any of the following attributes: Age, Caste, Disability, Ethnicity, Gender Identity, and Expression, Nationality, Race, Immigration Status, Religion, Sex/Gender, Sexual Orientation, Victims of a major violent event and their kin, Veteran Status.” [33]
4. Scholars: Language which attacks or demeans a group based on race, ethnic origin, religion, disability, gender, age, disability, or sexual orientation/gender identity [34].
5. Hate Speech and Disinformation Prevention and Suppression Proclamation in Ethiopia: “Hate speech” means speech that promotes hatred, discrimination, or attack against a person or an identifiable group, based on ethnicity, religion, race, gender, or disability

3.3 Hate Speech Detection Approaches

As the number of conflicts encouraged by hate speeches spread in social media increases, hate speech gained a lot of attention from governments and scholars. Different scholars have tried their best to detect hate speech on social media using different approaches such as Keyword-based approaches and Machine Learning methods.

3.3.1. Keyword Approaches

A basic approach for identifying hate speech is using a keyword-based approach. By using an ontology or dictionary, text that contains potentially hateful keywords are identified. For instance, [35] maintains a database of derogatory terms for many groups across 95 languages. Such well-maintained resources are valuable, as terminology changes over time. However, as we observed in our study of the definitions of hate speech, simply using a hateful slur is not necessarily enough to constitute hate speech.

Keyword-based approaches are fast and straightforward to understand. However, they have severe limitations. Detecting only racial slurs would result in a highly precise system but with low recall where precision is the percentage of relevant from the set detected and recall is the percent of relevant from within the global population. In other words, a system that relies chiefly on keywords would not identify hateful content that does not use these terms. In contrast, including terms that could but are not always hateful (e.g., “trash”, “swine”, etc.) would create too many false alarms, increasing recall at the expense of precision.

Furthermore, keyword-based approaches cannot identify hate speech that does not have any hateful keywords (e.g., figurative or nuanced language). Slang such as “build that wall” literally means constructing a physical barrier (wall). However, with the political context, some interpret this as a condemnation of some immigrants to the United States.

3.3.2. Machine Learning Approaches

Machine learning models take samples of labeled text to produce a classifier that can detect hate speech based on labels annotated by content reviewers. Various models were proposed and proved successful in the past.

3.3.3. Content preprocessing and Feature Selection

To identify or classify user-generated content, text features indicating hate must be extracted. Obvious features are individual words or phrases (n-grams, i.e., a sequence of n consecutive words). To improve the matching of features, words can be stemmed to obtain only the root removing morphological differences.

3.3.3.1. Bag of Words (BOW)

The bag-of-words assumption is commonly used in text categorization. Under this assumption, a post is represented simply as a set of words or n-grams without any order. This assumption certainly omits an important aspect of languages but proved powerful in numerous tasks. In this setting, there are various ways to assign weights to the terms that may be more important, such as TF-IDF [36]

3.3.3.2. N-grams

N-grams are one of the most used techniques in hate speech automatic detection and related tasks [2], [10], [34] and [9]. The most common N-grams approach consists of combining sequential words into lists with size N. In this case, the goal is to enumerate all the expressions of size N and count all occurrences. This allows improving classifiers' performance because it incorporates at some degree the context of each word. Instead of using words, it is also possible to use N-grams with characters or syllables. This approach is not so susceptible to spelling variations as for when words are used. Character N-gram features proved to be more predictive than token N-gram features, for the specific problem of abusive language detection [7].

However, using N-grams also have disadvantages. One disadvantage is that related words can have a high distance in a sentence [2] and a solution for this problem, such as increasing the N value, slows down the processing speed. Also, studies point out that higher N values (5) perform better than lower values (unigrams and trigrams). In a survey [37], researchers report that N-grams features are often reported to be highly predictive in the problem of hate speech automatic detection, but perform better when combined with others.

3.3.3.3. TF-IDF

TF-IDF. The TF-IDF (term frequency-inverse document frequency) was also used in this kind of classification problem [38] and [5]. TF-IDF is a measure of the importance of a word in a document within a corpus and increases in proportion to the number of times that a word appears in the document. However, it is distinct from a bag of words, or N-grams, because the frequency of the term is off-setted by the frequency of the word in the corpus, which compensates for the fact that some words appear more frequently in general (e.g., stop words).

3.3.3.4. Part of Speech

Part-of-speech (POS) approaches to make it possible to improve the importance of the context and detect the role of the word in the context of a sentence. These approaches consist of detecting the category of the word, for instance, personal pronoun (PRP), Verb non-3rd person singular present form (VBP), Adjectives (JJ), Determiners (DT), Verb base forms (VB). Part-of-speech has also been used in hate speech detection problems [39]. With these features, it was possible to identify frequent bigram pairs, namely PRP_VBP, JJ_DT, and VB_PRP, which would map as “you are” [38]. It was also used to detect sentences such as “send them home,” “get them out,” or “should be hung”. However, POS proved to confuse the class identification, when used as features.

3.3.3.5. Word Embeddings

Word Embeddings. Some authors [12] used a paragraph2vec approach to classify language on user comments as abusive or clean and also to predict the central word in the message [12]. FastText is also being used [17]. A problem that is referred to hate speech detection is that sentences must be classified and not words [37]. Averaging the vectors of all words in a sentence can be a solution, however, this method has limited effectiveness [34]. Alternatively, other authors propose comment embeddings to solve this problem [12].

3.3.4. Hate Speech Detection Classification Algorithms

3.3.4.1. Naïve Bayes

Naïve Bayes is a classification technique based on Bayes’ Theorem with an assumption of independence among predictors. Naïve Bayes models label probabilities directly with the assumption that the features do not interact with one another. [40] used it to detect racist tweets against blacks

3.3.4.2 Support Vector Machine (SVM)

Support Vector Machine is a supervised machine learning algorithm that can be used for both classification or regression tasks. However, it is mostly used in classification problems. [10] proposed a state-of-the-art feature-based classification model that incorporates distributional TF-IDF features, part-of-speech tags, and other linguistic features using support vector machines. The incorporation of these linguistic features helps identify hate speech by distinguishing between

different usages of the terms but still suffers from some subtleties, such as when typical offensive terms are used in a positive sense.

3.3.4.3. Logistic Regression

Logistic regression is a supervised Machine learning classification algorithm used to predict the probability of a target class. [12] proposed to use paragraph2vec for joint modeling of comments and words, where they learn their distributed representations in joint space using the continuous BOW (CBOW) neural language model. Then, they fed the resulting text embedding to a logistic regression classifier to classify comments into hateful and clean. The proposed approach addressed high-dimensionality and sparsity that impact the current state-of-the-art, resulting in highly efficient and effective hate speech detectors.

3.3.5. Deep Learning

3.3.5.1. Convolutional Neural Network Ensemble

[14] proposed a CNN ensemble approach, which combines the decisions of ten convolutional neural networks with different weight initializations. Their network structure is similar to the one proposed by [41], with convolutions of length 3 pooled over the entire document length. The results of each model are combined by averaging the scores.

3.3.5.2. C – GRU

C-GRU, a Convolution-GRU Based Deep Neural Network proposed by [42] combines convolutional neural networks (CNN) and gated recurrent networks (GRU) to detect hate speech on Twitter. They conduct several evaluations on publicly available Twitter datasets demonstrating their ability to capture word sequence and order in a short text. Note, in the HatebaseTwitter [10] dataset, they treat both Hate and Offensive as Hate resulting in a binary label instead of its original multi-class label. In our evaluation, we use the original multi-class labels where different model evaluation results are expected.

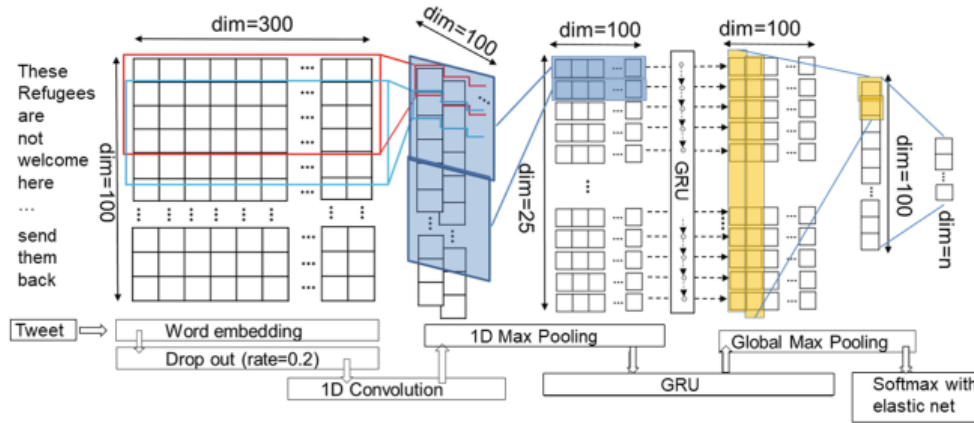


Figure 3.1 The CNN+GRU network architecture Source: [42]

3.3.5.2. Long Short-term Memory

The LSTM unit was initially proposed by [43]. LSTM units can propagate an important feature that came early in the input sequence over a long distance, thus capturing potential long-distance dependencies. [44] used LSTM to classify Facebook comments into *Hate* or *Not Hate*.

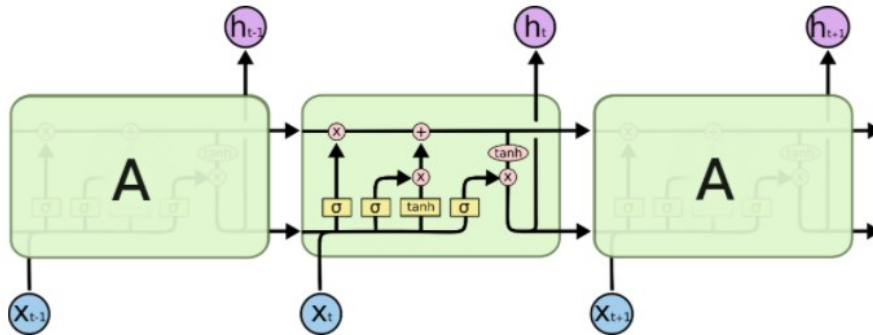


Figure 3.2 LSTM Architecture Source: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

3.3. Challenges of Hate Speech Detection System

Detecting hate speech is a challenging task, however. First, there are disagreements in how hate speech should be defined. This means that some content can be considered hate speech to some and not to others, based on their respective definitions. We show the competing definitions of hate speech in the section ‘different definitions of hate speech’. Competing definitions provide challenges for evaluation of hate speech detection systems; existing datasets differ in their

definition of hate speech, leading to datasets that are not only from different sources but also capture different information. This can make it difficult to directly access which aspects of hate speech to identify.

Despite differences, some recent approaches found promising results for detecting hate speech in textual content [10], [45] and [14]. The proposed solutions employ machine learning techniques to classify text as hate speech. One limitation of these approaches is that the decisions they make can be opaque and difficult for humans to interpret why the decision was made. This is a practical concern because systems that automatically censor a person's speech likely need a manual appeal process.

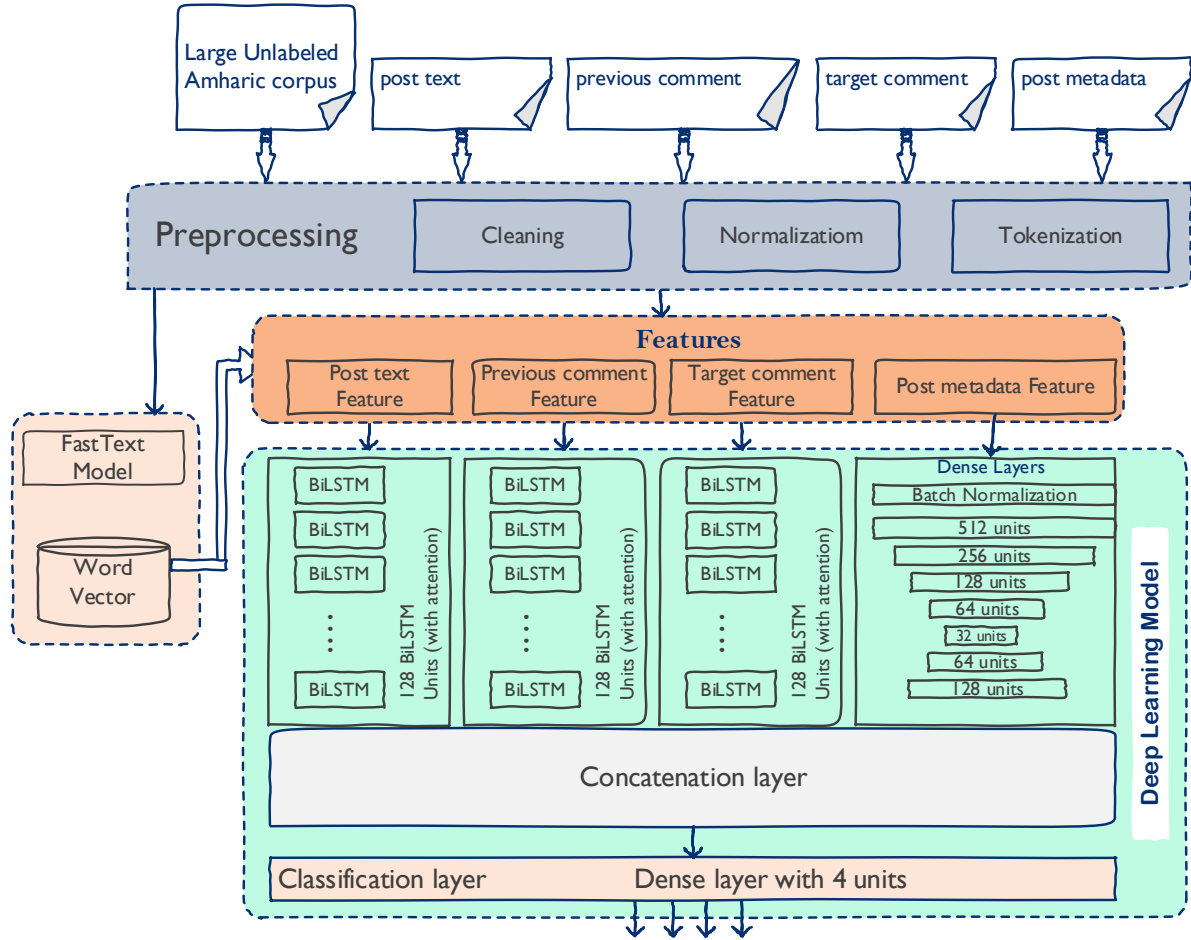
However, there are practical challenges that remain among all systems. For instance, armed with the knowledge that the platforms they use are trying to silence them, those seeking to spread hateful content actively try to find ways to circumvent measures put in place. Besides, the scholars and the companies are mostly focusing on the English language while hate speech is becoming the problem of many other languages. There are even no public datasets available for under-resourced languages such as Amharic, Afaan Oromo, Tigrigna, and many other African languages to try different approaches for researchers. Finally, we conclude the challenges of the hate speech detection system with the following five points.

- Automatic hate speech detection is technically difficult
- Without societal context, systems cannot generalize sufficiently.
- There are no public datasets available for under-resourced languages such as Amharic, Afaan Oromo, Tigrigna, and many African languages to try different approaches to it
- Some approaches achieve reasonable performance
- Specific challenges remain among all solutions

4. Design and Implementation

4.1. Overview

In this chapter, we will discuss the overall design of the proposed hate speech detection system using Deep Neural Networks. Firstly, the proposed system has the preprocessing component that cleans, normalizes, and tokenizes the annotated data coming as an input. And then the preprocessed textual data will be fed into the embedding layer that has 200 output dimensions. A large unlabeled cleaned Amharic corpus has been used to train the FastText and word2vec model for the word embeddings. Since the system has four inputs namely post's text message, previous comment, target comment, and post's metadata, the feature of first three inputs (textual inputs) will be extracted from the pre-trained fasttext word model and pass into three different paths each of them having a BiLSTM layer that has 64 units with attention mechanism, the later one (post's metadata) will go through batch normalization layer and passes to the series of dense layers. Next, the concatenation layer merges those four different paths and yields one vector with 512 units. Finally, the classification layer containing 4 units will take the outputs of the concatenation layer as an input and produces the probability of the sample belonging to the respective classes (Not Hate, Ethnic Hate, Religious Hate, Political Hate classes).



4.2. Preprocessing

In this section, the raw data is converted into meaningful data using the process called text preprocessing. The need for preprocessing is to remove raw data that contain unwanted punctuation marks, stop words, numerical value, and special character, and to replace one alphabet in different representations, etc. These values can affect the performance as well as the correctness of the model. So, before applying any text modeling or featuring (before training the model) data preparation procedures (like changing text corpus into suitable format) must be done before performing word embedding based on this thesis work. Pre-processing is involved in preparing the input text corpus into a format that is suitable for the word2vec model generation. The preprocessing stage consists of a variety of processes, those are cleaning, normalization, word segmentation (tokenization).

4.2.1. Cleaning

Cleaning refers to the task of removing unwanted (for this study) parts of the text and includes the following tasks:

- Removing non-Amharic characters
- Removing comments and posts that only had images and videos (not having a text) in their content
- Removing emojis and replacing them with white space or if the sentence is of all emoji content removing the whole text
- Replacing all of the URLs in the text to '<ድረገጽ>' token and numbers to '<ቁጥር>' token
- Replacing all of the punctuation marks with a whitespace character except for the question mark '?' which is replaced with '<ጥያቄምልክት>' token, as it might sometimes decide whether a specific text is a hate or not

Table 4.1 shows example comments texts before and after the cleaning has been applied

E.g. Comments before cleaning	E.g. Comments after cleaning
አንተ ይህን ሁሉ የጦር ስትራቴጂ ፕላን ከየት አገኘህ? ከቀላቢው???? ቀላቢውን እንዴት አገኘህ?? ?? በሰልክ???? ሰልኩን ከየት አገኘህ??? 🙄 🙄 🙄 🙄 ልቦልክህ አንድ ፖስት ነው የቀረህ።	አንተ ይህን ሁሉ የጦር ስትራቴጂ ፕላን ከየት አገኘህ <ጥያቄምልክት> ከቀላቢው <ጥያቄምልክት> <ድግግሞሽ> ቀላቢውን እንዴት አገኘህ <ጥያቄምልክት> <ድግግሞሽ> በሰልክ <ጥያቄምልክት> <ድግግሞሽ> ሰልኩን ከየት አገኘህ <ጥያቄምልክት> <ድግግሞሽ> <ድግግሞሽ> ልቦልክህ አንድ ፖስት ነው የቀረህ
#Ethiopian_DJ የኢትዮጵያ ሙዚቃ's_post # ወሽት ቀለቡ 🙄 🙄	የኢትዮጵያ ሙዚቃ ቁጥር ወሽት ቀለቡ <ድግግሞሽ>
AbrHam Unitd Junta ማለት ወታደራዊ መንግስት ነው እንግሊዘኛ ቃል ነው	ማለት ወታደራዊ መንግስት ነው እንግሊዘኛ ቃል ነው

Table 4.1 comments before and after the cleaning process

4.2.2. Normalization

Normalization is the process of transforming text into a single canonical form that it might not have had before. Normalizing text before storing or processing it allows for separation of concerns since the input is guaranteed to be consistent before operations are performed on it. Text normalization requires being aware of what type of text is to be normalized and how it is to be processed afterward; there is no all-purpose normalization procedure.

In the Amharic writing system, there are characters with the same pronunciation but different symbols which are called homophones. The letters such as አ, ኣ, ዐ and ዓ; ሠ and ሰ; ሀ, ኀ, ሂ, ኸ, ኡ, ኃ and ኣ, ጸ and ፀ are examples of characters with the same meaning and pronunciation but a different symbol. For example, if we take the word “Habtamu” ሀብታሙ it could have different forms like ኣብታሙ, ኀብታሙ, ኡብታሙ, ኸብታሙ, and ኃብታሙ. Therefore, all the above different forms must be normalized into ሀብታሙ by changing the first character of a word. Therefore, these characters should be normalized. the same word is written in various forms [46]. For example, the word ‘ሰምቶአል’ (‘he hears’) can be written in Amharic as ሰምቶአል, ሰምቷል, ሰምትዋል.

While (! end of text)

From a document read a character

If the character is one of ኀ ኁ ኂ ኃ ኄ ኅ and their orders

replace them with ሀ and its orders

If the character is ሠ and its orders

replace it with ሰ and its orders

If the character is one ዐ and its orders

replace it with አ and its orders

If the character is ፀ and its orders

replace it with ጸ and its orders

If the character is ቆ

replace it with ቆ

If the character is ዉ

replace it with ው

4.2.3. Tokenization

In tokenization (splitting data into a small chunk of words) the whole corpus is split into word level. Tokenization takes the input text supplied from a user and split it into a sequence of tokens, which is the process of breaking a stream of text down into words. And finally, it gives the list of words that are used in the next phase of preprocessing. Due to the model that we use we may also split into character level. To generate the word2vec model, the word is an input for the word2vec algorithm which will be extracted from the input text corpus.

In most Latin language white spaces and other punctuation marks (like question mark {?}) are used as the main approximation of word-to-word delimiter (boundary markers between sequences of words). Like the other languages, the Amharic language also has its own punctuation marks which separate texts or sentences into a stream of words. Amharic punctuation marks including ‘Hulet netb’ or colon (:), ‘arat netb’ (::), ‘netela serez’ (፣), ‘drib sereze’ (፤), ‘question mark’ (?) and exclamation mark ‘! Or ፤’ are used as sentence delimiter or as white space most of the time. In this work, we take all the punctuation marks mentioned above except question mark and arat netb (::) as word delimiter, question mark and arat netb (::) as sentence delimiter when splitting the text corpus into sentences. If question mark ‘?’ or any Amharic character appear consecutively more than two (not including two) times it will be replaced with one instance of itself and <ድግግሞሽ> token (e.g., እንግዲህም እንዲሁ አትበሉ will be converted to እንግዲህ <ድግግሞሽ> አትበሉ) so that ‘እንግዲህም’, ‘እንግዲህ’, ‘እንግዲህ’ can be considered the same as their meaning tends to be the same.

E.g. Comments before cleaning and tokenization	E.g. Comments after cleaning tokenization
<p>አንተ ይህን ሁሉ የጦር ሰትራቴጂ ፕላን ከየት አገኘህ? ከቀላቢው???? ቀላቢውን እንዴት አገኘህ?? ?? በሰልክ???? ሰልኩን ከየት አገኘህ??? 🤖🤖 🤖🤖 ልቦልክህ አንድ ፖስት ነው የቀረህ።</p>	<p>['አንተ', 'ይህን', 'ሁሉ', 'የጦር', 'ሰትራቴጂ', 'ፕላን', 'ከየት', 'አገኘህ', '<ጥያቄምልክት>', 'ከቀላቢው', '<ጥያቄምልክት>', '<ድግግሞሽ>', 'ቀላቢውን', 'እንዴት', 'አገኘህ', '<ጥያቄምልክት>', '<ድግግሞሽ>', 'በሰልክ', '<ጥያቄምልክት>', '<ድግግሞሽ>', 'ሰልኩን', 'ከየት', 'አገኘህ', '<ጥያቄምልክት>', '<ድግግሞሽ>', '<ድግግሞሽ>', '<ድግግሞሽ>', 'ልቦልክህ', 'አንድ', 'ፖስት', 'ነው', 'የቀረህ']</p>

Biniam Berhe ጥጥር ከቤትህ አይወጣ እድሜህ በሀዘን ይለቅ	['ጥጥ', 'ከቤትህ', 'አይወጣ', 'እድሜህ', 'በሀዘን', 'ይለቅ']
--	---

Table 4.2 E.g. Comments before and after cleaning and tokenization

4.3. Word Vector

After preprocessing tasks, the unlabeled free-text document is transformed into numeric values (dense vector representation of words that capture something about their meaning). The main reason is to make a machine learning model to understand and process the natural language. Word embedding captured the contextually related meaning of a word in a document, syntactic, and semantic similarity, and relation with other words.

We pre-trained an embedding model called FastText for learning word vectors and used it when training the whole model. The main goal of the FastText embeddings is to take into account the internal structure of words while learning word representations; this is especially useful for morphologically rich languages like Amharic, where otherwise the representations for different morphological forms of words would be learned independently.

Unlike word2vec, FastText treats each word as composed of character n-grams. So, the vector for a word is made of the sum of this character n-grams. For example, the word vector “ዘገግ” is a sum of the vectors of the n-grams “<ዘገ”, “ዘገግ”, “ገግ>”. With this manifest, it benefits to generate better word embeddings for rare words. Moreover, FastText predict vector for out of vocabulary words from its character n-grams even if the word doesn't appear in the training corpus. In contrast, both Word2vec and Glove leave unseen words as out-of-vocabulary words. Treating character n-gram manifests FastText the following advantage over Word2Vec:

- Generate better word embeddings for rare words (even if words are rare their character n-grams are still shared with other words - hence the embeddings can still be good). This is simply because, in word2vec a rare word (e.g., 10 occurrences) has fewer neighbors to be pulled by, in comparison to a word that occurs 100 times whereas FastText has more neighbor context words and hence is pulled more often resulting in better word vectors.
- Out of vocabulary words - they can construct the vector for a word from its character n-grams even if the word doesn't appear in a training corpus. Both Word2vec and Glove can't.

4.4. Features

After preprocessing, as the text inputs cannot be directly fed into our deep learning model we need to extract the feature of three textual inputs, namely post text, previous comment, and target comment out of four total inputs. We have pre trained a fasttext model as discussed in previous section for extracting the features of post text, previous comment, and target comment and converting the text inputs to the vector of floating numbers so that it can be processed easily with our deep learning model. We have four features post text, previous comment, target comment and post metadata as an input for the deep learning model.

4.4.1. Post Text Feature

Post text is the textual content of the post under which the target comment to be classified is appeared. In this paper, post text is used as one of the contexts for the target comment. As it is a textual input, we extracted its feature from a pre-trained fasttext model and converted it to a vector of floating numbers then it can be processed by our deep learning model. When sometimes the content of the post does not contain textual context (i.e, if its only of picture or video or other type of content not textual) we inital this input as a zero vector. Table 4.3 shows the example post text before and after feature extraction

Example post texts before feature extraction	Example post texts after feature extraction
['የኢትዮጵያ', 'አየር', 'ጎይል', 'በመቀሌ', 'ከተማ', 'በሚገኙ' 'ሰትራቴጂ', 'ወታደራዊ', 'አላማዎች', 'ላይ', 'ድብደባ', 'ፈጸመ']	[[-0.01688833, -0.10193669, 0.11383979, ..., -0.03813029, -0.03946952, 0.06956015], [0.03863554, -0.06258125, 0.09216724, ..., 0.1133863, 0.09389522, 0.04503796], [0.01530407, -0.05318451, 0.00350013, ..., -0.04649082, 0.00741332, 0.07647355], ..., [0.05968927, -0.08716951, 0.02466055, ..., -0.00496814, -0.00592556, 0.07515685], [-0.01499257, -0.04478383, -0.06302188, ..., 0.04351642, -0.06570013, 0.13125806], [-0.0187029, 0.00140089, -0.01370498, ..., 0.00086999, 0.00275725, -0.00566212]]

<p>['ፋክት', 'ቸኩሩ', 'ቸክ', 'እየተደረገ', 'ኑው']</p>	<p>[[0.0301745 , -0.10538826, 0.07537206, ..., -0.09162361, 0.17084228, 0.10610545], [0.02430764, -0.05517985, -0.0299606 , ..., 0.01730761, 0.07411034, 0.04633911], [0.11449689, -0.14896883, -0.00769278, ..., 0.0604276 , 0.09465981, 0.07105343], [0.1131053 , 0.09474211, -0.0104038 , ..., -0.04400101, -0.06200308, 0.1405531], [-0.00782073, 0.00720706, -0.0007492 , ..., 0.02544684, -0.04007503, 0.10075814]]</p>
---	--

Table 4.3 Example post texts before and after feature extraction

4.4.2. Previous Comment Feature

Previous comment is the comment appeared before the target comment to be classified when the comments are rendered in their chronological order. In this study we used previous comment as one of the target comment contexts. As it's predecessor, it is also a textual content we extracted its feature from a pre-trained fasttext model and converted it to the vector of floating numbers so that it can easily fed into our deep learning model. Sometimes a previous comment might not be a textual comment at this time we will use the closest previous comment that has textual comment. Additionally, when the target comment to be classified is the first comment for the post it will not have any previous comments then we will initialize it as a zero vector. Table 4.4 shows the example previous comment before and after feature extraction.

Example previous comment before feature extraction	Example previous comment after feature extraction
<p>['አንተማ', 'የተረገምክ', 'ከ', 'ሰላም', 'የማይናፍቅህ', 'ከ']</p>	<p>[[0.03990273, -0.07371604, -0.10285497, ..., -0.07464771, 0.00270337, 0.06109005], [0.12051129, -0.1216344 , -0.10263249, ..., 0.00141009, 0.10495174, 0.07511191], [0.00820494, -0.16150917, -0.02411012, ..., -0.02723845, 0.01078039, 0.06324317], [-0.09427443, -0.21395724, 0.01081028, ..., 0.1599592 , 0.04821133, 0.01920544], [0.12175129, -0.05962006, 0.01087631, ..., 0.00321711, 0.0603162 , 0.11337058], [0.00820494, -0.16150917, -0.02411012, ..., -0.02723845, 0.01078039, 0.06324317]]</p>

['ተዋቸው', 'ቢክህ', 'እነሱ', 'ነቸው', 'ህግ', 'ወጣቶ']	[[0.03758936, -0.1389556 , 0.02797763, ..., -0.11060007, -0.01397057, -0.03809834], [0.03188203, -0.27332392, -0.06280826, ..., -0.03525108, 0.10561734, 0.05279372], [0.0491369 , -0.05686475, -0.05430181, ..., -0.01900719, 0.01601776, 0.12606584], [-0.02172587, 0.01422888, 0.08728638, ..., -0.01800859, -0.00214459, -0.07578452], [0.08334207, -0.04098514, 0.05902518, ..., -0.09020817, 0.02093207, 0.02848827], [-0.04550269, -0.04713039, 0.03843137, ..., 0.03661114, 0.09235124, 0.06321894]]
---	---

Table 4.4 Example previous comments before and after feature extraction

4.4.3. Target Comment Feature

Target comment is the comment to be classified as political-hate, ethnic-hate, religious-hate, or neutral. All other input to the system post text, previous comment, and post metadata are the contexts for this target comment. As this target comment is also a textual content, we will extract its features and convert it to vector of floating point numbers from pre-trained fasttext model so that it can be fed to our deep learning model. This paper deals with only target comments that have Amharic textual contents. If the the target comment doesn't have Amharic textual content then we just ignore it, its beyond this paper's scope as specified in *Section 1.5* of this document. Table 4.5 shows example target comments before and after feature extraction

Example target comment before feature extraction	Example target comment before feature extraction
['አሸንድዩ', 'ልናኩብር', 'የጋራ', 'እሴታችንን', 'ልናኩብር', 'ነው', 'የመጣነው']	[[0.0521326 , 0.00661807, 0.030799 , ..., 0.01576342, -0.04340664, 0.12275533], [0.04971191, -0.04853543, 0.03322709, ..., -0.05081153, 0.03526784, 0.0703284], [0.0393754 , -0.03524695, -0.01839374, ..., 0.09654225, 0.02457598, 0.11819324], ..., [0.04971191, -0.04853543, 0.03322709, ..., -0.05081153, 0.03526784, 0.0703284], [-0.00782073, 0.00720706, -0.0007492 , ..., 0.02544684, -0.04007503, 0.10075814], [0.06807542, -0.11717074, -0.08765751, ..., -0.13367927, -0.00049983, 0.04867809]]

['አገግ', 'ማለት', 'ሰው', 'በላ', 'ጭራቅ', 'አገር', 'አውዳሚ', 'የህዝብ', 'ጠላት', 'አላማ', 'የሌለው', 'የመርዘ', 'እብዶች', 'እና', 'የዝንጀሮዎች', 'ሰብሰብ', 'ነው']	[[-0.07011815, -0.11815817, -0.03134687, ..., -0.01263579, 0.07781696, 0.0425341], [-0.0652432, -0.04814807, -0.00052552, ..., 0.00921389, -0.10986543, 0.05914893], [0.14092883, -0.06662863, -0.09305759, ..., -0.1210481, -0.07146692, 0.0599811], ..., [0.19248529, 0.06116455, -0.00918964, ..., 0.06929129, -0.00690382, 0.07674801], [-0.02763664, -0.01019792, -0.10143664, ..., 0.03129777, 0.01006327, -0.03077951], [0.06691434, 0.00890634, -0.04646131, ..., -0.0018158, 0.00813293, 0.06711975]]
--	--

Table 4.5 Example target comments before and after feature extraction

4.4.4. Post metadata Feature

Post metadata is the statistical information of the post including number of

- post_reactions
- post_no_of_comments
- post_share
- likes
- love
- wow
- haha
- sad
- angry

As post metadata is numerical and tabular data having nine parameter it can be directly fed to the batch normalization layer and then to the dense layer of the deep learning model as the fourth input to our system.

post_reactions	post_no_of_comments	post_share	likes	love	wow	haha	sad	angry
710.0	43.0	0	539.0	3.0	1.0	165.0	1.0	1.0
575.0	158.0	0	426.0	1.0	5.0	136.0	4.0	3.0

Table 4.6 Example post metadata feature

4.5. Deep Learning Model

4.5.1. BiLSTM (Bidirectional Long Short-Term Memory)

Bidirectional recurrent neural networks (RNN) are just putting two independent RNNs together. This structure allows the networks to have both backward and forward information about the sequence at every time step.

Using bidirectional will run your inputs in two ways, one from past to future and one from future to past and what differs this approach from unidirectional is that in the LSTM that runs backward you preserve information from the future, and using the two hidden states combined you are able in any point in time to preserve information from both past and future. We have used BiLSTM layers with an attention mechanism, each of them having 128 LSTM units for learning the post's message, previous comment, and the target comment.

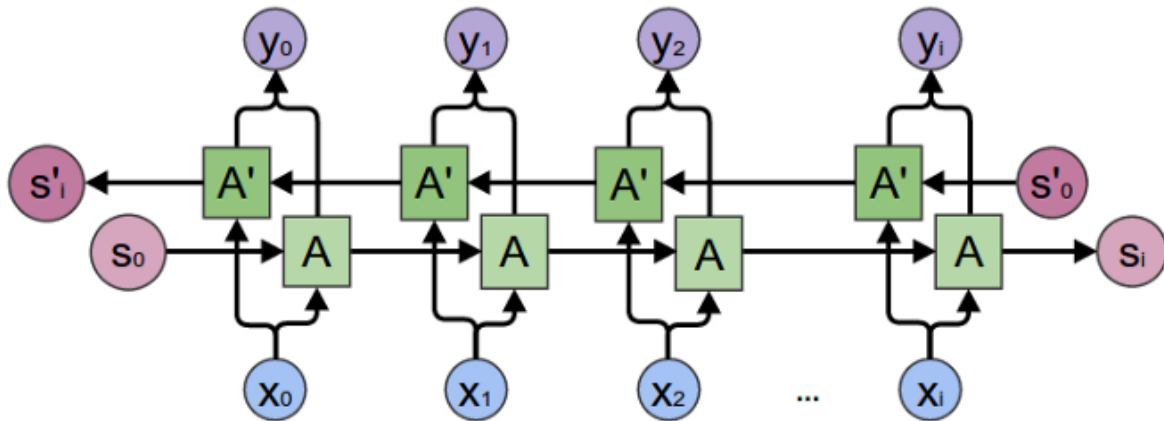


Figure 4.1 Bi-Directional LSTM Architecture

4.5.2. Attention Mechanism

A neural network is considered to be an effort to mimic human brain actions in a simplified manner. Attention Mechanism is also an attempt to implement the same action of selectively concentrating on a few relevant things while ignoring others in deep neural networks. A critical and apparent disadvantage of recurrent neural networks with longer sequence lengths is the

incapability of the system to remember longer sequences. Often it has forgotten the earlier parts of the sequence once it has processed the entire sequence. The attention mechanism was born to resolve this problem. Like the posts, previous comments, and target comment's text are going to be often lengthy we used an attention mechanism with LSTM cells to solve the problem of not remembering longer sequences in the LSTM.

4.5.3. Batch Normalization

Neural network layers work best when the input data have zero mean and unit variance, as it enables faster learning and higher overall accuracy. Batch normalization reduces the amount by which the hidden unit values shift around (covariance shift). Batch normalization, or the batch norm for short, is proposed as a technique to help coordinate the update of multiple layers in the model. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks. Batch normalization provides an elegant way of re-parametrizing almost any deep network. The reparameterization significantly reduces the problem of coordinating updates across many layers. We used batch-normalization before sending the post's metadata to the dense layers with 512 units.

4.5.4. Concatenation Layer

The concatenation layer takes input from activations of four different paths coming from the post's message path, previous comment's path, target comment's path, and the posts metadata path respectively. It will then join those four paths each of them having 128 activations to a single vector by concatenating them and passes it to the classification layer having dense layers of 4 neurons, equal to the number of classes.

4.5.5. Classification Layer

The classification layer takes an input from the concatenation layer and produces the probability of the sample that it belongs to one of predefined four classes. We used a classification layer that has four neurons which is equal to our number of classes with SoftMax activation, to normalize the outputs between 0 and 1. At this stage, the output of the neuron is the probability of the sample that it belongs to the respective classes.

5. Experiments

5.1. Data Collection

Aiming at classifying the hate level across Facebook for Amharic language users, we have built a corpus of comments and posts retrieved from Ethiopian Facebook public pages. These pages typically post discussions spanning across a variety of political, ethnic, and religious topics. By doing so, we could capture both casual conversations and politically, ethnically, or religiously hatred posts and comments.

Authors have employed a versatile Facebook crawler, which exploits the Graph API to retrieve the content of the comments from Facebook posts using Facepager. Facebook is selected to collect data from social media for the following reasons. Facebook is the most important platform for reaching out to online audiences, and especially the youth. Comparative studies have shown how in countries with limited Internet penetration, like Ethiopia, Facebook has become almost a synonym for the Internet, a platform through which users access information, services, and participate in online communications.

The above dataset was prepared for training the classifier model and an other large unlabeled dataset is also used for training the fasttext word model which is prepared from the previous work [47] having 686,573,429 tokens and we also collected 11,352,658 tokens. We merged those two dataset together getting 697,926,087 of total tokens and trained fasttext model for word vector generation.

5.2. Data Annotation

We defined hate speech as “a speech that promotes hatred, discrimination or attack against a person or an identifiable group based on ethnicity, religion or political ideas”, which is partly derived from [48]. After defining hate speech, we prepared a list of criteria partly driven from [9] to better standardize the annotation procedure and to make clear all hate and not hate speech-related category concepts. A comment or post belongs to one of the hate types (political hate, religious hate, or ethnic hate) if it fulfills 1, 2, and at least one of the listed criteria under three, of the following lists.

- 1) It targets an individual or group of individuals
- 2) It targets individuals or groups based on their religion, ethnicity, or political ideas
- 3) Promotes hatred, discrimination, or attack
 - a) Uses hate or slur verbally
 - b) It promotes violence
 - c) Seeks to silence individuals because of their religion, ethnicity, or political ideas
 - d) Uses straw man argument to promote hatred against individuals or groups based on their religion, ethnicity, or political ideas
 - e) Negatively stereotypes individuals or groups based on their religion, ethnicity, or political ideas
 - f) Promotes, but not directly use, hate speech or violent crime

Then we identified three native Amharic speakers and BSc. Graduates of Law department trained them how to identify hate speeches by following the above list of criteria that helps to identify a given text as political hate, ethnic hate, religious hate, or not-hate (neutral), and the annotators also discussed and practiced before they started annotation.

In addition to the annotation rules the Kappa decision agreement based on Cohen's kappa statistic which is an estimate of the population coefficient between $0 \leq \kappa \leq 1$ [49] is also used. Kappa agreement < 0 less than chance agreement, 0.01-0.20 Slight agreement, 0.21- 0.40 Fair agreement, 0.41-0.60 Moderate agreement, 0.61-0.80 Substantial agreement, 0.81-0.99, almost perfect agreement. Not everyone would agree which one is a “good” agreement but as the commonly cited scale is kappa value of 0.57 is in the “moderate” agreement range for a better agreement. Remember that perfect agreement would equate to a kappa of 1, and the chance agreement would equate to 0 All of the three annotators annotated the same 1,346 comments and they achieved a substantial kappa score of 0.79. We think the training and discussion helped a lot in achieving a high kappa score among annotators. For those comments which annotators disagreed on, we label them as they have been labeled by two of the annotators. Then the rest of the comments were labeled by only one annotator.

5.3. Dataset Description

Comment	Label
Đuňiýa ěthö tǝvĕ ቦቅጧቃ ጋላ ጦርነት ሲባል ቀዘናችሁ	Ethnic Hate
ነፍጠኛ በተገኘበት ግደሉ	Ethnic Hate
Raya Love ሀሀሀሀሀ እዩትማ በጃዋር ፎቶ TPLF hahaha ገና እናጠፋችሁለን ጥንብ ጧላ ሉ	Ethnic Hate
Ahmed Muhammed Yasin ጥምብ እሰላም አሀያ ነክ የገማክ	Religious Hate
ኢዜማ ነባራዊ ሁኔታ የማይገባው የሆዳሞች ስብስብ ነው።	Political Hate
አነግ ማለት ሰው በላ ጭራቅጅአገር አውዳሚ ፤ የሀዘብ ጠላት ፤ አላማ የሌለው የመርዝ እብዶች እና የዘንጀሮዎች ስብስብ ነው	Political Hate
ጊዜው አደለም አሁን ይሄን ማድረግ።	Not Hate
<i>Table 5.1 Example Annotated Comments</i>	
አሸንዶቼ ልናከብር የጋራ እሴታችንን ልናከብር ነው የመጣኑ	Not Hate

We created two datasets one for training FastText and word2vec embedding models, to be used in the classifier and another for training the classifiers. Table 5.1 shows information about the first dataset used to train the FastText model, table 5.2 shows the dataset used to train the binary

classifier in which the three classes of hate (i.e., political, religious, and ethnic hate) are treated as hate in general and Table 5.3 shows information about the second dataset used to train the four class classifier.

Total Number of Sentences	Total number of words	Unique words after preprocessing	Mean length of words in the sentences	maximum length of words in the sentence
6,488,049	96,338,402	2,476,060	22	64

Table 5.2 Information about dataset used to train FastText and Word2vec embedding

Hate	Neutral	Total
6,091	11,280	18181

Table 5.3 Dataset Information for a binary classifier

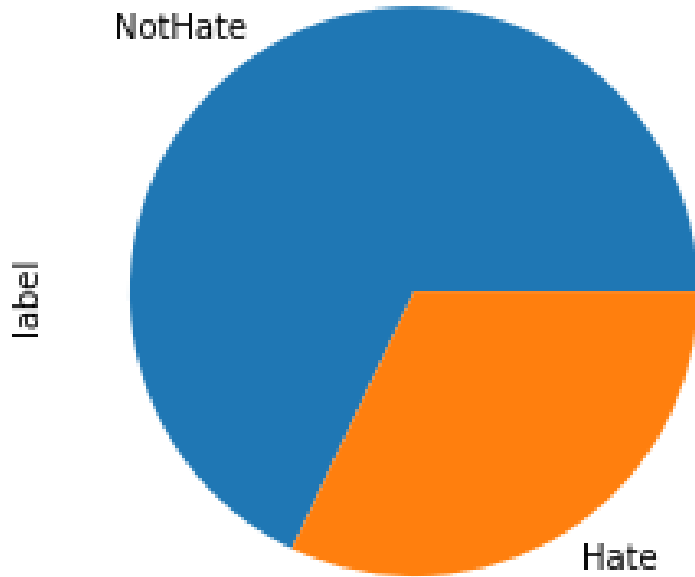


Figure 5.1 Dataset distribution when using only two classes (i.e., hate and neutral)

Ethnic Hate	Political Hate	Religious Hate	Neutral	Total
2,614	3,263	1,024	11,280	18181

Table 5.4 dataset information used to train the four class classifier

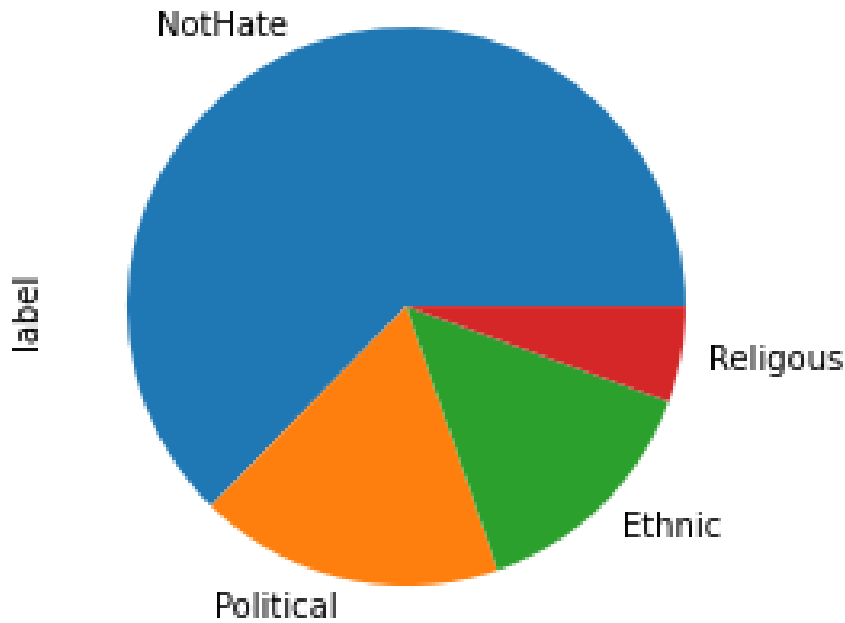


Figure 5.2 dataset distribution when using four classes (i.e., political-hate, ethnic-hate, religious-hate, or neutral)

5.4. Development Tools

Different programming languages and tools were used in the progress of the study and they are discussed as follows:

Python: Python is a high-level programming language that is processed by a python interpreter to produce results. Many features makes python the best choice of language for machine learning and artificial intelligence.

- Python is a high-level language that is easy to learn, read, and maintain
- Short development time in comparison to other programming languages like Java, C++, and Ruby.
- There are plenty of libraries in python that makes our task easier, for example, NumPy is a library for python that can solve scientific computation easily.
- It is interactive (has a terminal for debugging and testing) and portable (runs on a variety of hardware platforms).

Pandas: Pandas is an open-source library that provides high-performance, easy-to-use data structure, and data analysis tools for python programming language. Pandas is used to read CSV files and perform different operations on the CSV files.

Keras: is a library for developing deep neural networks in python that can run using TensorFlow or Theano as a back-end to preprocess data, to create, evaluate, optimize, fit, and test a model. It is now also an official high-level API for TensorFlow and shipped with the TensorFlow platform.

TensorFlow: TensorFlow is an end-to-end open-source platform that has flexible tools and resources for machine learning. Many states of the art NLP applications are developed using TensorFlow as a backend. Popular organizations like Google, Intel, and others also use TensorFlow to develop systems.

Scikit-Learn: Scikit-learn is a library for python machine learning library. Scikit-learn contains simple and efficient tools for data mining and data analysis algorithms for both supervised and unsupervised problems. Scikit-learn is used to evaluate deep neural classifiers by calculating a confusion matrix which is a table that is often used to describe the performance of a classification model.

5.5. Evaluation Metrics

A common approach for evaluating machine learning models is through the comparison of predicted outputs of the model with that of labeled data by humans. Depending on the similarity between the two, there are many evaluation metrics such as accuracy, precision, recall, and F score. The most commonly used performance measures in text classification and hate speech detections are precision, recall, accuracy, and F score. These measures are based on

TP (True Positive): The number of samples that are classified correctly into their respective classes.

FP (False Positive): represents the number of samples classified into some classes when they are actually not members of that class.

FN (False Negative): represents the number of samples that are not classified as a member of their corresponding class when they actually are members of that class

TN (True Negative): represents the number of samples classified as they are not part of the class when they are actually are not members of that class.

Precision (P): is the ratio of the number of samples predicted as a particular desired class, true positives (TP) to a total number of samples predicted as that class.

$$P = \frac{TP}{TP+FP} \quad 5.1$$

Recall(R): is the number of true positives divided by the total number of samples that are known to belong to that class

$$R = \frac{TP}{TP+FN} \quad 5.2$$

F1-score: is the weighted average of Precision and Recall. Therefore, this score takes both false positives and negatives into account

$$F1\text{-score} = \frac{2 * Recall * Precision}{Recall + Precision} \quad 5.3$$

F1 score is a better and well-known measure to use if we need a balance between precision and Recall. F1 score is usually more useful than accuracy, especially if we have an uneven class distribution.

Accuracy: The other performance measure of a model is accuracy. Accuracy measures how much accurately the model learns to classify the data

$$A = \frac{TP + TN}{TP + TN + FP + FN} \quad 5.4$$

The overall precision, recall and F score of a classifier is calculated by using a weighted average called a macro-average.

Macro-average values are used to know how the system performs overall across the sets of data.

$$\text{Macro-average-Recall} = \frac{R1+R2+R3+R4}{4} \quad 5.5$$

$$\text{Macro-average-Precision} = \frac{P1+P2+P3+P4}{4} \quad 5.6$$

R1, R2, R3, and R4 are recall values of the four classes Not-Hate, Ethnic hate, Religious hate, and Political hate respectively. P1, P2, P3, and P4 are also the precision values of the four classes respectively. The macro-average method can be used when we want to know how the system performs overall across the sets of data. In this paper, we use macro-average precision, recall, and F score as a weighted average

5.6. Experimental Setup

Two cloud computing services are used to carry out the experiments. The first one is the Microsoft Azure cloud service which is used for pre-training two FastText word embedding models with 200 and 300 dimensions on the raw-Amharic text of around 700 million tokens. The second service was Google's colab laboratory service (Colab) used to train some of the experiments. The hardware and software specifications of services are presented below.

Processor	Intel(R) Xeon(R) 6 Core CPU E5-2690 v3 @ 2.60GHz
Size name	STANDARD_NC6
Memory	56 GiB RAM
Operating System	Linux

Table 5.5 Specifications of azure compute instance used for training FastText

Processor	Intel(R) Xeon(R) 2 Core CPU @ 2.30GHz
Memory	12 GiB
Operating System	Linux

Table 5.6 Specifications of colab machine used for training

FastText embedding: we pre-trained the FastText embedding model and used it as a feature for training the model. We trained the FastText model with genism python’s library with the following configuration, window = 10, emb_dim = 200, learning_rate = 0.001 with SGD estimation, 20 epochs, and the skip-gram flavor of word models.

Bi-Direction LSTM: In addition to hardware and software specifications, we need to configure the LSTM network for developing a hate speech classification model. In the experiment, the LSTM network is configured with the following parameters: one LSTM layer with 64 neurons, 100 epoch, 32 batch_size, 'adam' optimizer, 23 seed, 80% train size, and 20% test size.

Dropout: Dropout helps to prevent overfitting by randomly dropping some of the units and their connection from the neural network during training [50]. In this paper, the optimal choice of dropout is between 0.3 and 0.5. In order to choose an appropriate epoch number, this paper has used the training versus validation accuracy plot of the model. To identify the occurrence of overfitting from a plot the following conditions will occur.

1. When both training and validation loss simultaneously decreases. However, at some point in time validation loss increases while training loss decreases.

2. When both training and validation accuracy starts increasing and at some point when validation accuracy increases over training accuracy.

Early Stopping Configuration: - Early stopping is one of the mechanisms used to reduce the occurrence of overfitting [51] [50]. Early stopping is a mechanism to stop training as soon as the performance of a model stop improving. Early stopping parameters are configured as follows: patience=5, monitor=val_loss, mode=min, and baseline=0.3.

5.7. Experimental Scenarios

To evaluate our hypothesis different experiments are conducted. Generally, these experiments can be grouped into two. The first group of experiments is aimed at evaluating that sub-word information can be used to improve the performance of Amharic hate speech detection systems in morphologically rich languages like Amharic and answering the first research question **RQ1:** Could sub-word information be used to improve the performance of hate speech detection in morphologically rich languages like Amharic and in social media platforms where most of the text comments are misspelled?

The second group of experiments is to evaluate whether the context information can help to improve the performance of Amharic hate speech detection systems and to answer the second research question, **RQ2:** Could the contexts be used to improve the performance of Amharic Hate speech detection system in social media posts?

To answer RQ1 [Sub-word information for hate speech detection], the initial step was to generate the neural word embeddings from large unlabeled Amharic raw text. As we are trying to detect hate speeches in Amharic texts and hate words occur relatively less frequently than neutral words, we used the skip-gram model of Word2vec and FastText as it can capture less frequent words than the CBOW (Continuous Bag of Words), model. A vector with a fixed length of 200 dimensions is generated for each word in the corpus.

The next step was an experiment to check the quality of word vectors before they are used as an input for the model. To evaluate the quality, we used a simple similarity test between words. For this purpose, the top ten closest words by using cosine distance, are retrieved for a given target word. The target words are representative instances of hate words and not-hate(neutral) words.

The next step was the actual usage of word vectors as a feature for Amharic hate speech classification. We pre-trained FastText word embedding, which computes character n-grams of large Amharic raw text before the training starts therefore, it can capture sub-word (character) information. We also pre-trained the Word2vec embedding model which treats words as an atomic unit for modeling the language, therefore it might not capture sub-word (character) information. We used the pre-trained FastText and Word2vec embedding model as a feature to evaluate the importance of sub-word (character) information in Amharic hate speech detection.

To answer RQ2 [Context information for Amharic hate speech detection], we used recurrent deep neural networks, BLSTM with and without attention mechanism, and FastText first and then Word2vec as a feature. Training data prepared for previous experiments with and without context information is used to evaluate the importance of context information for hate speech detection. We used the text of the commented post, the previous comment (a comment before the target comment), and the post's metadata as a context for the target comment. If the target comment is a reply to another comment then we take the replied-to comment (the parent comment) as a post for the target comment, as this comment is a comment for the parent comment not for the original post. If the target comment is the first comment which means it doesn't have a previous comment, we pad the previous comment vector with zero.

After passing those three inputs (i.e., target comment, previous comments and post text) through parallel BLSTM layers, we concatenate and pass them into the dense layer that classifies them into four classes. By changing the recurrent layer neuron cells (BLSTM) our neural net is trained with 100 iterations for each case. For evaluation, we used a train test split of 80% for training and 20% for the test. binary cross-entropy objective function with 'adam' optimizer is used for the training process.

5.8. Results

5.8.1. Importance of Sub-word information for Amharic Hate speech detection system

Word	Ten closest words from word2vec word vector
እናት	እናት፣ እናትም, እናቱ, እናት፤, አባት, እህት, አክሰት, እናትን, እናቷ, ሴት
ወያኔ	ህወሀት, ህወ-ሀት, ሻቢያ, ኢህአዴግ, ኢህአዲግ, ወያኔም, ህወሀትኢህአዴግ, ሻቢያ, የወያኔ, አገዛዙ
ጋላ	ኩሽ, ነፍጠኛ, አሮሞ, አማራ, ቅማንት, ቤጃ, አጋዚያን, ሻንቅላ, ኢልማን, አማራና
አማራ	አሮሞ, አማራው, ትግሬ, ቅማንት, አማራ፣, የአማራ, አሮሞው, ጎንደራ, የአማራው, ነፍጠኛ
መናፍቅ	ኑፋቄ, ጴጌ, መናፍቃን, መነኩሴ, ፕሮቴስታንታዊ, አርዮስ, አውጣኪ, ንስጥርስ, ይወገዝ, የመናፍቃን
ነፍጠኛ	ትምክህተኛ, አማራ, ትግሬ, አሮሞ, ጋላ, ነፍጠኞች, ሸንታም, ጎንደራ, ባንዳ, አማራው
አሮሞ	አማራ, ትግሬ, አሮሞ፣, ቅማንት, አማራ፣, አሮሞም, አሮሞው, አሮሞን, ነፍጠኛ, አማራና

Table 5.7 word similarity test results with word2vec word vector

Word	Ten closest Words from fasttext word vector
እናት	እናትልጅዋ, ኹሉእናት, እናትናት, እናት, የወለደቻትና, ልጅን, እናትም, እናትልጅ, እናትና, እናትአባት
ወያኔ	ህወሀት, ወያኔንህወሀት, ወያኔህወሀትኢህአዴግ, የወያኔ, ሩልወያኔ, የወያኔህወሀትኢህአዴግ, ህወሀትኢህአዴግጭቆና, ህወሀትኢህአዴግ, ወያኔወያኔ, ወያኔህወሀት
ጋላ	ጋላጋላ, ጋላመሪ, ጋላኮ, ጋላነወ, ማዳጋስካር, ጋላላ, ጋላኛ, ጋላና, ጋላኛ, ጋላሶ
አማራ	ቅማንትከአማራ, የአማራ, አሮሞናአማራ, አሮሚያናአማራ, አሮሞአማራ, ሄአማራ, ብሄረአማራ, ሀአማራ, ትግሬዎችከትግሬይ, አገውወአማራ
መናፍቅ	መናፍቅት, መናፍቅወደ, መናፍቅቆች, መናፍቅና, መናፍቅም, መናፍት, በመናፍቅ, የመናፍቅ, መናፍቃ, አተመናፍቅ
ነፍጠኛ	ነፍጠኛኛ, ነፍጠኛነ, ነፍጠኛኮ, ነፍጠኛና, አዎነፍጠኛ, ነፍጠኛየ, ነፍጠኛነኛ, ነፍጠ, ነፍጠኛኝ, ነፍጠኛስ

አሮጥ	አሮጥአሮጥ, አሮጥዶ, አሮጥሆ, አሮጥጠል, አሮጥዌቹ, አሮጥዎ, አሮጥብሄር, አሮጥእና, አሮጥሲዳግ, አሮጥና
-----	---

Table 5.8 word similarity test with FastText word vector

Table 5.7 and Table 5.8 shows that word vectors can capture word relations based on their semantic meaning. We took two target words from hate and neutral words to show the ability of word vectors in capturing the semantic meaning of words and overcoming the problem of misspelled words being treated as different words even though they have the same intention. Closest words to a given hate word are mostly hate words and the closest words to a given neutral word are mostly neutral words. From this result we can see that automatically generated features are capable of capturing contextual word relations from a given text but, the closest words in the word2vec vector are, the word that are more or less correctly spelled and it was unable to capture misspelled words even if they mean the same thing as the given word. On the other hand, FastText embedding seems to capture the closest words even though they are spelled incorrectly.

The next step is to evaluate the importance of sub-word information for Amharic hate speech detection. For this purpose, we pre-trained FastText as it extracts character n-grams of the words, it can capture sub-word information than a word2vec embedding and Word2vec which takes a word as an atomic unit and may not capture sub-word information as FastText could. We used those pre-trained word embedding models as a feature to train our classifier which is a deep learning model.

Performance Measures	80% training and 20% test			
	With FastText embedding		With Word2vec embedding	
	Hate	Not Hate	Hate	Not Hate
Precision	86.47	86.85	84.82	83.37
Recall	85.36	84.65	82.65	82.49

F-Score	85.91	85.73	83.72	82.92
Macro-Precision	86.66		84.09	
Macro-Recall	85		82.47	
Macro-F-Score	85.82		83.32	
Accuracy	84.78		81.58	

Table 5.9 Comparison of FastText embedding (which can capture sub-word information) feature and word2vec embedding feature for Amharic Hate speech detection system

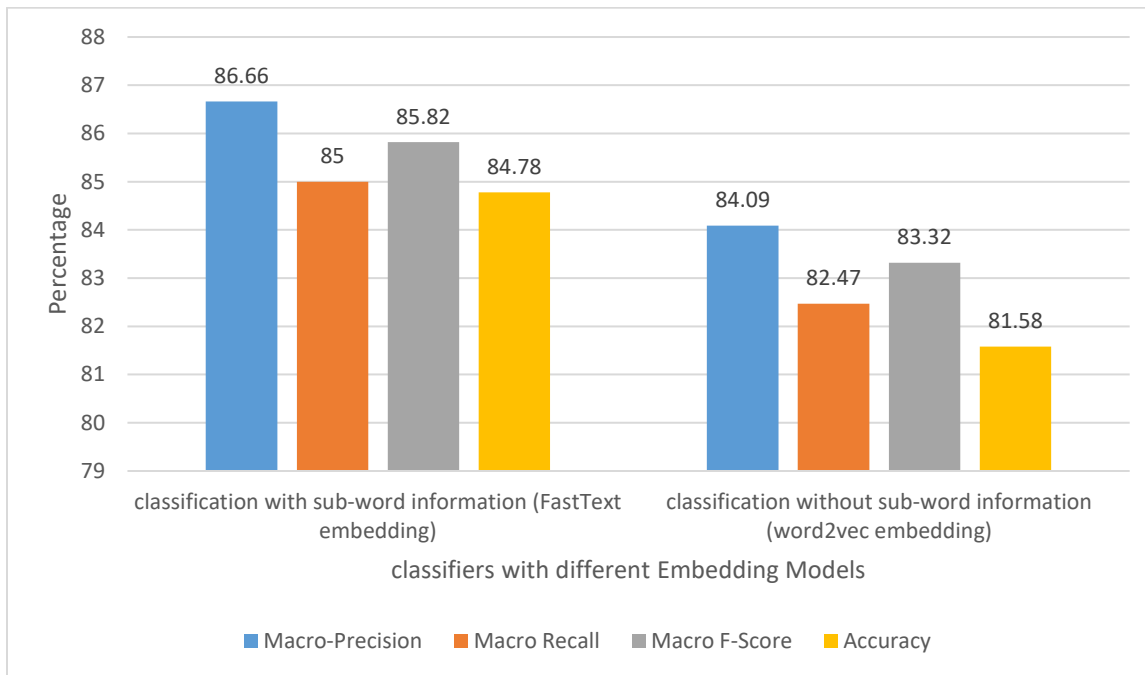


Figure 5.3 Comparison of different fasttext and word2vec feature for Amharic hate speech detection

As we can see from Table 5.9 and Figure 5.3 sub-word information in the language model than taking a word as an atomic unit for Amharic hate speech detection can improve the accuracy from 81.58% to 84.78 and F-score from 83.32% to 85.82%. we think three things might be a reason for the improvement using FastText embedding that also captures sub-word information than word2vec embedding. Firstly, word2vec embedding faces out-of-vocabulary problems if the word were not in the training corpus of the word2vec, in which case we initialized the word’s vector to zero. For example, even if the word ‘ሃጻጠኛ’ is in the vocabulary and in case the user typed it as

'ነፍጢኛ', 'ነፍጠኛ', 'ነፍጥኛ' or 'ነፍጥኛ' those words will be out-of-vocabulary if they did not present by their own in the training corpus. Second, as our dataset is collected from public social media comments, we found most of the users doesn't care for the correctness of the spelling while writing comments and this made the path difficult for word2vec as it needs to see every misspelled representation of the word to keep it in the word-vectors. On the other hand, even though the words are represented differently as they share many characters FastText can obtain the word-vector for them, the vectors are also much closer and this would be useful as the user is also intended to say the same thing.

Third, word2vec also needs to see every word-forms in the training corpus to obtain the word-vector, which is much difficult for morphologically rich and under-resourced languages like Amharic. The user uses different forms of the word for showing their hate also e.g., ጋላፊ, ጋልፎ, ጋላ or ጋልኛ those words will have their own vector representation in word2vec only if they occur on their own in the training corpus otherwise, they will be an out-of-vocabulary word in which case we initialize them as a zero vector. When we use the FastText feature all of them will have related vector representation as they share characters. We believe those are the three reasons that improved the Amharic hate speech detection when using the FastText feature than word2vec.

The results obtained from experiments conducted to answer RQ1 demonstrated that taking sub-word (character-level) information as a feature can improve the Amharic Hate speech detection system. Word vector features that can capture sub-word information like FastText can be used to improve the performance of hate speech detection systems in a morphologically rich language like Amharic. Furthermore, for languages that have limited resources out-of-vocabulary-words might be a problem with taking a word as an atomic unit of a language, as word2vec does. On the other hand, taking sub-word (character) information into account as FastText does, even if the word is not in the vocabulary we won't face the out-of-vocabulary problems as long as that word has at least two characters shared with another word in the vocabulary.

5.8.2. Importance of Context Information for Amharic Hate speech detection

Classifiers		80% train and 20% test			
		Performance measure			
		Precision	Recall	F-Score	Accuracy
BLSTM without, context and attention	Hate	83.26	82.41	82.83	81.73
	Not-Hate	82.19	82.63	82.4	
	Macro-Average	82.72	82.52	82.61	
BLSTM with attention but, without context	Hate	83.88	82.76	83.31	82.71
	Not Hate	84.02	81.38	82.67	
	Macro-Average	83.95	82.07	82.99	
BLSTM with Context but, not attention	Hate	85.65	84.76	85.2	84.37
	Not Hate	85.93	84.89	85.4	
	Macro-Average	85.79	84.82	85.3	
BLSTM with attention + Context Information	Hate	86.94	85.98	86.45	85.87
	Not Hate	86.77	85.86	86.31	
	Macro-Average	86.85	85.92	86.38	

Table 5.10 Performance of BLSTM with and without Attention and with and without Context Information

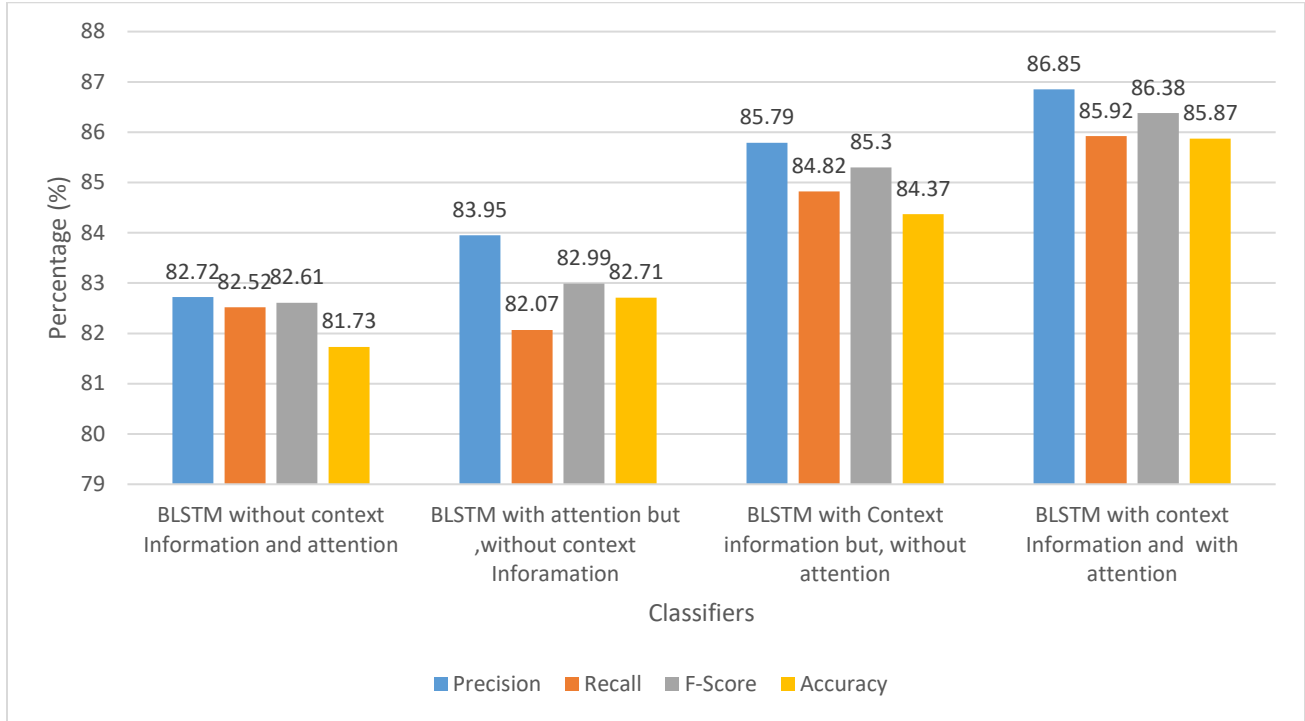


Figure 5.4 performance of BLSTM with and without attention and with and without context information

Table 5.10 and Figure 5.4 demonstrates that the Amharic hate speech detection system improves from the accuracy of 81.73% to 84.37% and F-score 82.61% to 85.3% when using context information. Furthermore, Amharic hate speech detection improves its accuracy from 82.71% to 85.87% and F-score from 82.99% to 86.38% when incorporating attention mechanism with deep learning model.

We believe the performance improvement of the classifier when adding context information is reasonable. Sometimes, even human needs the context of the comment very much to label the comments. For example, the comment ‘ጤፍፋ ድርሰ ጭፈራ እንጂ ምን ታውቃላቸ’ is clearly not any type of hate on its own or if it is commented on the post that says ‘የጃኖ ባንዶች ኮንሰርት መሰከረም 26 ዩኒቨርሲቲ’ but, if it appears to be a comment for the post that says ‘ቀ/አ ዝናሽ ታያቸው የፕሮቴስታንት መዝሙር ለቀቁ’ it will clearly be Religious-hate as it negatively stereotypes someone’s religion.

The model also increased its performance when using the Attention mechanism with context information. The main advantage of recurrent neural networks in general and specifically LSTMs is that they can learn sequence information but, if the sequence is getting longer and longer, they

will start forgetting the earlier parts of the sequence after processing the entire sequences. For example, sometimes comments can have a clear hate word at the start of the comment followed by a long neutral or positive sentence at this time, the LSTMs will start to forget the presence of hate word at the start and classify it as neutral. The attention mechanism comes to the rescue here to solve this problem by selectively concentrating on a few relevant parts of the sentence while ignoring others. we believe this helped the model to improve its performance when adding attention to the LSTM layer.

We also made exploratory experiments to classify comments into political-hate, ethnic-hate, religious-hate or neutral. Table 5.11 shows the results of the experiment

Classifiers		80% train and 20% test			
		Performance measure			
		Precision	Recall	F-Score	Accuracy
BLSTM without, context and attention	Political Hate	83.26	82.41	82.83	81.03
	Ethnic Hate	82.37	81.64	82	
	Religious Hate	79.43	77.48	78.44	
	Not-Hate	82.19	82.63	82.4	
	Macro-Average	81.81	81.04	81.41	
BLSTM with attention but, without context	Political Hate	83.88	82.76	83.31	82.31
	Ethnic Hate	83.02	82.2	82.6	
	Religious Hate	79.83	78.92	79.37	
	Not Hate	84.02	81.38	82.67	
	Macro-Average	82.68	81.31	81.98	
BLSTM with Context but, not attention	Political Hate	85.65	84.76	85.2	83.37
	Ethnic Hate	84.67	83.92	84.29	
	Religious Hate	78.85	77.63	78.23	
	Not Hate	85.93	84.89	85.4	
	Macro-Average	83.77	82.8	83.28	

BLSTM with attention + Context Information	Political Hate	86.94	85.98	86.45	84.57
	Ethnic Hate	85.87	85.43	85.64	
	Religious Hate	79.87	78.94	79.40	
	Not Hate	86.77	85.86	86.31	
	Macro-Average	84.86	84.05	84.45	

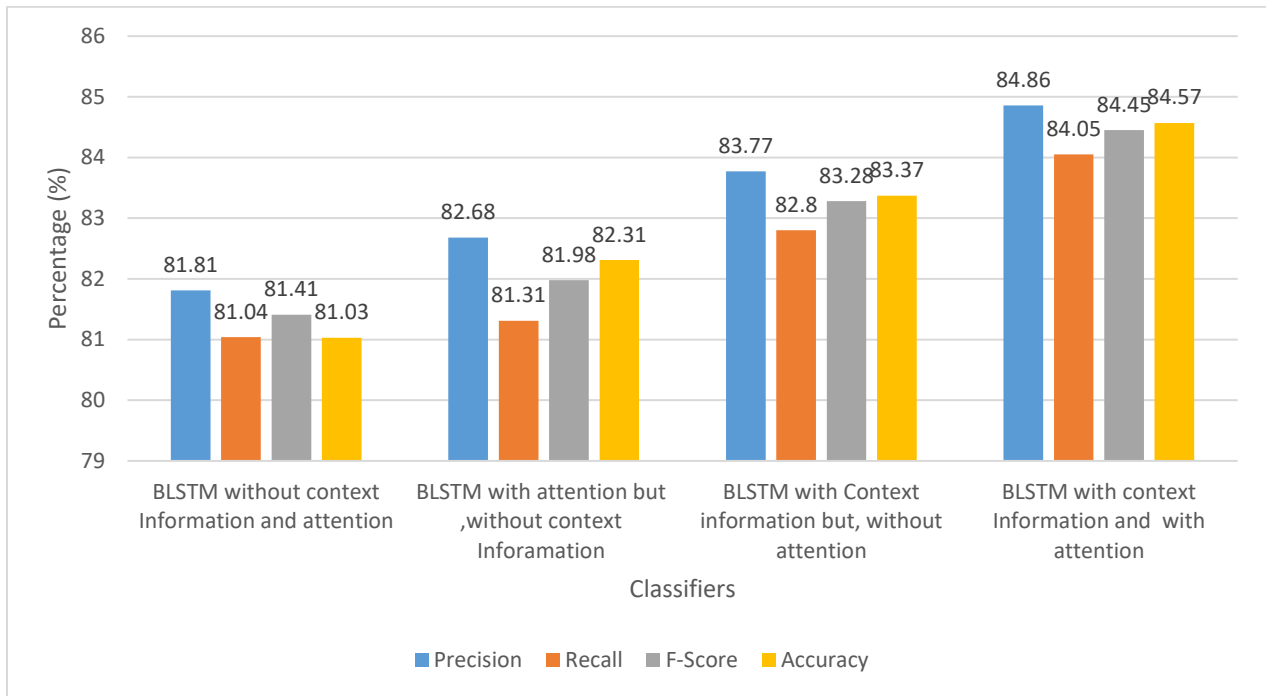


Figure 5.5 Summary of experiments to classify comments into four classes (Political-hate, Ethnic-hate, Religious-hate, Neutral)

We also conducted exploratory experiments to see the effect of the context in which the social media comment appeared when classifying into three specialized hate classes (Political-hate, Ethnic-hate, and Religious-hate) and neutral (not-hate). We experimented with four models first BiLSTM model without context information and without Attention mechanism, second BiLSTM model with attention mechanism but without context information, third BiLSTM model with

Table 5.11 Results of classifying comments into four categories (Political-hate, Ethnic-hate, Religious-hate and Neutral)

context information but without attention mechanism and finally BiLSTM model with context information and attention mechanism. Table 5.11 and Figure 5.5 shows Amharic hate speech detection system can improve when we get the context in which the social media comments appeared into consideration. The classifier accuracy improved from 81.03% to 83.37% when incorporating context information and the model further improved to 84.57% when we add an attention mechanism. The incorporation of context information improved the Amharic hate speech detection system when we classify comments into generic hate or not-hate binary classes. This improvement also continued when we classify the hate into three specialization classes political-hate, ethnic-hate, or religious-hate and neutral.

5.9. Threats to Validity

The dataset used in this study is lower than that of the other resource-rich languages such as English. The quality of the dataset is also not good and the same approach applied for a better dataset, both in terms of amount and quality, may bring better evaluation results. Different results may be observed on different input datasets.

Deep neural networks are prone to overfitting to reduce this unwanted characteristic, the author used two techniques (early stopping and a dropout range from 0.3 to 0.5). However, another combination of overfitting reduction techniques may result in a different result.

6. Conclusion and Future Works

6.1. Conclusions

The main objective of this paper is to investigate the importance of context information for the Amharic hate speech detection system and demonstrate the effect of using features that can capture sub-word (character level) information like FastText for Amharic hate speech detection.

The experiments in this paper demonstrated that the accuracy of the Amharic hate speech detection system improved from 81.58% to 84.78% when using a FastText feature that can capture sub-word (character level) information than word2vec feature. We believe the FastText feature improved the Amharic hate speech detection over word2vec feature because social media users mostly don't care about spelling errors when writing their comments and those misspelled words can be captured and kept close to the correct representation of the word with FastText, not word2vec. The word2vec embedding doesn't seem to capture the relationship between correctly spelled words and misspelled words, they are being treated as totally different entities even though they refer to the same thing. On the other hand, FastText can capture the relationship between correctly spelled words and misspelled words as it can capture sub-word (character level) information of words. This paper concludes that using a feature that can capture sub-word (character level) information like FastText can improve the Amharic hate speech detection systems.

The results on this paper also shown that incorporating context information in the Amharic hate speech detection system improved the accuracy from 81.73% to 85.87%. We have noticed that even human needs the context information to label most of the comments. We treat the post's text under which the comment appears, the previous comment, and the post's metadata as the context of the target comment. This paper concludes that the incorporation of context information in the Amharic hate speech detection system can improve the performance of the system.

6.2. Contributions

This research paper has the following contributions.

- Deep learning model is designed and developed for Amharic hate speech detection system that can take advantage of the context in which the social media comments appeared
- This paper showed the difference between Amharic hate speech detection system with context information in social media comments and without context information
- This study also showed the difference between Amharic hate speech detection system that uses a feature that can capture sub-word information like FastText and without sub-word information like word2vec
- This research paper also contributes for the future researches by collecting and labeling 18,181 Amharic social media comments.
- This study showed the importance of sub-word information for Amharic hate speech detection system especially, when we have a lot of misspelled words like most of Amharic social media comments.

6.3. Future Works

Developing an efficient fully-functional Amharic Hate speech detection system requires coordinated team efforts that comprise linguistic professionals, computer science professionals, and other people that can collect more comments from both public society and social media. Good coordination of these different professionals can result in a full functional sentiment analysis model.

This research paper identified the following directions as future work.

1. This research paper on hate speech detection is limited to comments written in the Amharic language and geez script **only**. However, there are users that write a comment by transliterating Amharic alphabets to English alphabets, we believe this can help to develop full model for Amharic hate speech detection
2. There are also comments written by combining the Amharic language with other languages like English, Tigrigna, and Afaan Oromo. For the next works adding these languages will help in developing a full model of hate speech detection system.

3. When conducting our experiments we have used a small amount of dataset of comments and also unbalanced for some of the classes like religious hate classes. We believe that adding more datasets and balancing the dataset per class could achieve better performance.

References

- [1] FDRE House of Peoples Representative, "Hate Speech and Disinformation Prevention and Suppression Proclamation," 2020.
- [2] B. Pete and W. L. Matthew, "Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making," *Policy & Internet* 7, p. 223–242, 2015.
- [3] M. Abdurazak, 2020.
- [4] H. Felix, "Tackling Hate Speech in Ethiopia," Dec 3, 2018.
- [5] Z. Mossie and J.-H. Wang, "Social Network Hate Speech Detection for Amharic Language," 2018.
- [6] K. Simon, "Digital 2020: Ethiopia," Data Reportal, 2020.
- [7] M. Yashar and T. Joel, "Do characters abuse more than words?," 2016.
- [8] G. Xiang, B. Fan, L. Wan, J. Hong and C. Rose, "Detecting offensive tweets via topical feature discovery over a large scale twitter corpus," in *21st ACM CIKM*, 2012.
- [9] Z. Waseem and D. Hovy, "Predictive Features for Hate Speech Detection on Twitter," in *Proceedings of NAACL-HLT*, San Diego, California, 2016.
- [10] T. Davidson, D. Warmusley, M. Macy and I. Weber, "Automated Hate Speech Detection and the Problem of Offensive Language," in *Proceedings of the Eleventh International AAAI Conference on Web and Social Media (ICWSM 2017)*, 2017.
- [11] W. Warner and J. Hirschberg, "Detecting hate speech on the world wide web," in *2nd Workshop on Language in Social Media*, 2012.

- [12] N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic and N. Bhamidipati, "Hate Speech Detection with Comment Embeddings," in *WWW 2015 Companion*, Florence, Italy, 2015.
- [13] S. Tulkens, L. Hilte, E. Lodewyckx, B. Verhoeven and W. Daelemans, "A Dictionary-based Approach to Racism Detection in Dutch Social Media," Antwerpen, Belgium, 2016.
- [14] S. Zimmerman, C. Fox and U. Kruschwitz, "Improving Hate Speech Detection with Deep Learning Ensembles," 2018.
- [15] R. Kshirsagar, T. Cukuvac, K. McKeown and S. McGregor, "Predictive Embeddings for Hate Speech Detection on Twitter," in *arXiv:1809.10644*, 2018.
- [16] J. Golbeck, Z. Ashktorab, R. O. Banjo, A. Berlinger, S. Bhagwan, C. Buntain, P. Chekalos, A. A. Geller, Q. Gergory, R. K. Gnanasekaran, R. R. Gunasekaran, K. M. Hoffman, J. Hottle, V. Jienjitler and K. Shivika, "A Large Human-Labeled Corpus for Online Harassment Research," in *WebSci'17*, Troy, NY, USA, 2017.
- [17] P. Badjatiya, S. Gupta, M. Gupta and V. Varma, "Deep Learning for Hate Speech Detection in Tweets," in *WWW 2017 Companion*, Perth, Australia, 2017.
- [18] Y. Wenpeng, K. Katharina, Y. Mo and S. Hinrich, "Comparative Study of CNN and RNN for Natural Language Processing," in *arXiv:1702.01923v1*, 2017.
- [19] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng and C. Potts, "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA, 2013.
- [20] I. Hendrickx, S. N. Kim, Z. Kozareva, P. Nakov, D. Ó. Séaghdha, S. Padó, M. Pennacchiotti, L. Romano and S. Stan, "SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations Between Pairs of Nominals," in *SemEval-2010*, 2019.

- [21] S. R. Bowman, G. Angeli, C. Potts and C. D. Manning, "A large annotated corpus for learning natural language inference," in *arXiv:1508.05326v1*, 2015.
- [22] Y. Yang, W.-t. Yih and C. Meek, "WIKIQA: A Challenge Dataset for Open-Domain Question Answering," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015.
- [23] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang and J. Suh, "The Value of Semantic Parse Labeling for Knowledge Base Question Answering," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany, 2016.
- [24] K. Guu, J. Miller and P. Liang, "Traversing Knowledge Graphs in Vector Space," in *arXiv:1506.01094v2*, 2015.
- [25] P. Nakov, S. Rosenthal, Z. Kozareva, V. Stoyanov, A. Ritter and T. Wilson, "Semeval-2013 task 2: Sentiment analysis in twitter," in *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval)*, 2013.
- [26] Zephoria.com, "The Top 20 Valuable Statistics - Updated May 2020," Zephoria, 2020.
- [27] InternetLiveStats.com, "Twitter Usage Statistics," InternetLiveStats.com, 2020.
- [28] M. Wendling, "The year that angry won the internet," 2015.
- [29] B. Ross, M. Rist, G. Carbonell, B. Cabrera, N. Kurowsky and M. Wojatzki, "Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis," in *The 3rd Workshop on Natural Language Processing for Computer-Mediated Communication*, 2016.
- [30] S. Wermiel, "The Ongoing Challenge to Define Free Speech," in *Human Rights Magazine*, 2018.
- [31] Twitter, "The Twitter Rules," 2017. [Online]. Available: <https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy>. [Accessed 20 June 2020].

- [32] Facebook, "Community Standards," 20 10 2020. [Online]. Available: https://www.facebook.com/communitystandards/recentupdates/hate_speech/.
- [33] YouTube, "YouTube Help," 2020. [Online]. Available: <https://support.google.com/youtube/answer/2801939?hl=en>. [Accessed 20 10 2020].
- [34] C. Nobata, A. Thomas, J. Tetreault, Y. Chang and Y. Mehdad, "Abusive Language Detection in Online User Content," in *WWW '16: Proceedings of the 25th International Conference on World Wide Web*, April, 2016.
- [35] Hatebase.org, "Hatebase," 26 10 2020. [Online]. Available: <https://hatebase.org>. [Accessed 26 10 2020].
- [36] G. Salton, C. Yang and A. Wong, "A Vector-Space Model for Automatic Indexing," 1975.
- [37] S. Anna and W. Michael, "A survey on hate speech detection using natural language processing," in *In Proceedings of the Workshop on Natural Language Processing for Social Media*, 2017.
- [38] D. Karthik, R. Roi and L. Henry, "Modeling the detection of textual cyberbullying," in *Soc. Mobile Web 11, 02 (2011)*, 2011.
- [39] E. Greevy and A. F. Smeaton, "Classifying racist texts using a support vector machine," in *27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*, 2004.
- [40] I. Kwok and Y. Wang, "Locate the hate: Detecting tweets against blacks," in *In Proceedings of the Association for the Advancement of Artificial Intelligence*, 2013.
- [41] Y. Kim, "Convolutional Neural Networks for Sentence Classification," in *EMNLP*, 2014.
- [42] Z. Zhang, D. Robinson and J. Tepper, "Detecting hate speech on twitter using a convolution-gru based deep neural Network," in *European Semantic Web Conference*, 2018.
- [43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," in *Neural computation*, 1997.

- [44] F. D. Vigna, A. Cimino, F. Dell'Orletta, M. Petrocchi and M. Tesconi, "Hate me, hate me not: Hate speech detection on Facebook," in *In Proceedings of the First Italian Conference on Cybersecurity (ITASEC17)*, Venice, Italy, 2017.
- [45] P. Fortuna and S. Nunes, "A Survey on Automatic Detection of Hate Speech in Text," in *ACM Comput Surv.*, 2018.
- [46] S. Zelalem, "Automatic classification of Amharic news items: The case of Ethiopian News Agency," in *MSc Thesis, Addis Ababa University*, Addis Ababa, Ethiopia, 2001.
- [47] E. Abebawu, T. Getenesh and A. Tewodros, "Learning Word and Sub-word Vectors for Amharic (Less Resourced Language)," *International Journal of Advanced Engineering Research and Science (IJAERS)*, pp. 358 - 366, August, 2020.
- [48] Hate Speech and Disinformation Prevention and Suppression Proclamation, "Federal Democratic Republic of Ethiopian House of People's Representative," Addis Ababa, 2019.
- [49] J. V. Anthony, "Understanding Inter observer Agreement: The Kappa Statistic," From the Robert Wood Johnson Clinical Scholars Program, University of North Carolina, 2005.
- [50] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research," 2014, pp. 1929-1958.
- [51] H. Jabbar and D. Khan, "Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study)," *Computer Science, Communication and Instrumentation Devices*, 2015.
- [52] YouTube, "YouTube Help," 2020. [Online]. Available: <https://support.google.com/youtube/answer/2801939?hl=en>. [Accessed 20 10 2020].

Appendix I – List of Amharic Normalized words

ጠዋት፣ ጠዋት፣ ጧት → ጠዋት

ኢሜይል፣ ኢሜይል፣ ኢሜል → ኢሜይል

ቴሌቪዥን፣ ቴሌቪዥን → ቴሌቪዥን

ትናንት፣ ትላንት → ትናንት

ደመወዝ፣ ደሞዝ፣ ደምዎዝ → ደመወዝ

ውሻ፣ ወሻ → ውሻ

ሰድሳ፣ ሰልሳ → ሰድሳ

ረቡዕ፣ እሮብ → ረቡዕ

መካከል፣ መሀከል → መካከል

ኢትዮጵያ፣ ኢትዮጵያ → ኢትዮጵያ

ጊዜ፣ ግዜ → ጊዜ

ይህ፣ ይህ → ይህ

ውጪ፣ ውጭ → ውጪ

Appendix III- Some of Amharic short words

ት/ቤት → ትምህርት ቤት

ት/ት → ትምህርት

ት/ክፍል → ትምህርት ክፍል

ሃ/አለቃ → ሃምሳለቃ

ሃ/ሰላሴ → ሃይለ ሰላሴ

ደ/ዘይት → ደብረ ዘይት

ደ/ታቦር → ደብረ ታቦር

መ/ር → መምህር

መ/ቤት → መሰሪያ ቤት

መ/አለቃ → መቶአለቃ

ክ/ከተማ → ክፍለ ከተማ

ክ/ሃገር → ክፍለ ሃገር

ወ/ር → ወታደር

ወ/ሮ → ወይዘሮ

ወ/ሪት → ወይዘሪት

ወ/ሰላሴ → ወልደ ሰላሴ