



**JIMMA UNIVERSITY**

**JIMMA INSTITUTE OF TECHNOLOGY**

**SCHOOL OF POST GRADUATE STUDIES**

**FACULTY OF ELECTRICAL AND COMPUTER ENGINEERING**

**OFFENSIVE AND HATE SPEECH DETECTION FOR  
AMHARIC LANGUAGE ON SOCIAL MEDIA USING  
DEEP LEARNING ALGORITHM**

By

**Miftah Adem**

**A Thesis Submitted to School of Graduate Studies of Jimma University in Partial  
Fulfillment of the Requirements for the Degree of Master of Science in Computer  
Engineering**

**June 2021**

**Jimma, Ethiopia**

**JIMMA UNIVERSITY**

**JIMMA INSTITUTE OF TECHNOLOGY**

**SCHOOL OF POST GRADUATE STUDIES**

**FACULTY OF ELECTRICAL AND COMPUTER ENGINEERING**

**OFFENSIVE AND HATE SPEECH DETECTION FOR  
AMHARIC LANGUAGE ON SOCIAL MEDIA USING  
DEEP LEARNING ALGORITHM**

**By**

**Miftah Adem**

**Advisor: Dr. Kinde Anlay**

**Co-Advisor: Mr. Fetulhak Abdurahman**

**Submission Date: June 2021**

# Declaration

I declare that this thesis entitled as **Offensive And Hate Speech Detection For Amharic Language On Social Media Using Deep Learning Algorithm** is my original work and has not been presented for a master's degree in this or any other universities, and all materials and resources used have been fully cited.



**Student Name:** Miftah Adem **Signature:**  **Date:** June 18, 2021

This research has been presented for review with my approval as university advisers.

**Main advisor:** Dr.Kinde Anlay **Signature:**  **Date:** December 29, 2020

**Co-advisor:** Mr. Fetulhak Abdurhman **Signature:** \_\_\_\_\_ **Date:** \_\_\_\_\_

This Thesis has been Approved by

| <b>Board of Examiners</b>                                      | <b>Signature</b>   | <b>Date</b>   |
|--|--|---------------|
| <b>External Examiner:</b> <u>Dr.Michael Melese</u>             |  | June 16, 2021 |
| <b>Internal Examiner:</b> <u>Workineh Tesema (Asst. Prof.)</u> |  | June 18, 2021 |
| <b>Chair Person:</b> <u>Mr. Fetulhak Abdurhman</u>             | _____  | _____         |

## *Acknowledgement*

First and foremost, praise and gratitude to the Almighty Allah for His blessings throughout my research work to complete the research successfully. I want to express my appreciation for **Dr. Kinde Anlay**, my advisor, for his support, guidance, understanding and inspiration during this study. He taught me how research generated and how to tackle the issue by spending his precious time. Additionally, I would like to acknowledge my co-advisor, **Mr Fetulhak Abdurrahman**, for providing me with helpful guidance and advice.

Lastly, to all my co-workers, thank you for your empathy and support in my many moments of crisis; your friendship makes my life a wonderful experience. I can't mention all the names here, but you're still on my mind. I want to thank my parents for their love and support, without which I would never have had so many opportunities.

# Contents

|   |       |
|---|-------|
| Declaration . . . . .   | i     |
| Acknowledgements . . . . .  | ii    |
| List of Figures . . . . .   | xiv   |
| List of Tables . . . . .  | xvi   |
| List of Abbreviations . . . . .   | xvii  |
| Abstract . . . . .  | xviii |
| 1 Introduction . . . . .  | 1     |
| 1.1 Background . . . . .  | 1     |
| 1.2 Research Motivation . . . . .   | 2     |
| 1.3 Statement of the problem . . . . .  | 4     |
| 1.4 Objective of the study . . . . .  | 5     |
| 1.4.1 General Objective . . . . .   | 5     |
| 1.4.2 Specific Objectives . . . . .   | 5     |
| 1.5 Scope and Limitation of the Study . . . . .   | 6     |
| 1.6 Significance of the Study . . . . .   | 6     |
| 1.7 Thesis Organization . . . . .   | 7     |
| 2 Theoretical Background and Related Work . . . . .   | 8     |
| 2.1 Introduction . . . . .  | 8     |
| 2.2 Definition of hate speech and offensive language . . . . .  | 8     |
| 2.2.1 Definition of hate speech . . . . .   | 8     |
| 2.2.2 Definition of offensive language . . . . .  | 10    |
| 2.3 Hate Speech on Social Media . . . . .   | 10    |
| 2.4 Automatic Hate Speech and offensive language Detection approaches in Social<br>Networks . . . . . | 11    |
| 2.4.1 Features representation for hate speech detection . . . . .                                     | 11    |
| 2.4.1.1 Dictionaries and Lexicons . . . . .   | 11    |
| 2.4.1.2 Bag-of-words (BOW) and N-grams . . . . .  | 12    |

---

|         |   |    |
|---------|---|----|
| 2.4.1.3 | Latent Dirichlet Allocation (LDA)   | 12 |
| 2.4.1.4 | Word embedding  | 12 |
| 2.4.2   | Machine learning for hate speech detection  | 15 |
| 2.4.2.1 | Supervised learning   | 16 |
| 2.4.2.2 | Semi-supervised learning  | 17 |
| 2.4.2.3 | Unsupervised learning   | 18 |
| 2.4.3   | Deep learning for hate speech detection   | 18 |
| 2.4.3.1 | Convolutional Neural Network  | 19 |
| 2.4.3.2 | Recurrent Neural Network  | 20 |
| 2.4.3.3 | Gated Recurrent neural network  | 21 |
| 2.4.4   | Combination of deep learning and machine learning classifier                      | 22 |
| 2.5     | Amharic language  | 23 |
| 2.5.1   | Amharic Word Classes  | 23 |
| 2.5.2   | Amharic Character Representation  | 25 |
| 2.5.3   | The Amharic Punctuation   | 25 |
| 2.5.4   | Challenges in Amharic Language  | 25 |
| 2.6     | Related Work  | 27 |
| 2.6.1   | Hate speech and offensive language detection for local languages                  | 27 |
| 2.6.2   | Hate speech and offensive language detection for foreign language                 | 29 |
| 3       | Proposed Dataset for Hate speech and Offensive Language Detection in Amharic Text | 33 |
| 3.1     | Introduction  | 33 |
| 3.2     | Dataset Generation Process  | 33 |
| 3.2.1   | Data Collection   | 34 |
| 3.2.2   | Data preparation  | 36 |
| 3.2.2.1 | Data cleaning   | 36 |
| 3.2.2.2 | Normalization   | 37 |
| 3.2.3   | Dataset Annotation  | 38 |
| 3.2.3.1 | Annotation Guidelines   | 39 |
| 3.2.3.2 | Manual annotation process   | 40 |

---

|          |  |           |
|----------|--|-----------|
| 3.2.4    | Dataset Characteristics . . . . .  | 40        |
| <b>4</b> | <b>The proposed model for Amharic hate speech and Offensive Language Detection</b> | <b>42</b> |
| 4.1      | Introduction . . . . .   | 42        |
| 4.2      | The proposed model . . . . .   | 42        |
| 4.2.1    | Hyperparamters . . . . .   | 48        |
| 4.3      | language and libraries . . . . .   | 51        |
| 4.4      | Evaluation metric . . . . .  | 52        |
| <b>5</b> | <b>Experimental Result, Discussion and Evaluation</b>                              | <b>55</b> |
| 5.1      | Hyperparameter Tuning . . . . .  | 56        |
| 5.2      | First Scenario Experiment Result . . . . .   | 56        |
| 5.2.1    | Effects of word embedding . . . . .  | 56        |
| 5.2.2    | Model comparison . . . . .   | 58        |
| 5.3      | Second Scenario Experiment Result . . . . .  | 60        |
| <b>6</b> | <b>Conclusion and Recommendation</b>   | <b>66</b> |
| 6.1      | Conclusion . . . . .   | 66        |
| 6.2      | Recommendation . . . . .   | 68        |
|          | References . . . . .   | 69        |

# List of Figures

|            |  |    |
|------------|--|----|
| Figure 2.1 | Skip-gram model (Adapted from, [22]) . . . . .   | 14 |
| Figure 2.2 | Continuous Bag of Words (CBOW) model (Adapted from, [22]) . . . . .  | 14 |
| Figure 2.3 | Model architecture of fastText for a sentence with N-gram features $x_1, \dots, x_N$ .<br>(Adapted from, [23]) . . . . . | 15 |
| Figure 2.4 | SVM linear separation (Adapted from, [27]) . . . . .   | 17 |
| Figure 2.5 | Random Forest (Adapted from, [en.wikipedia.org/wiki/Random forest ])   | 17 |
| Figure 2.6 | Convolutional Neural Networks for Sentence Classification, Y. Kim (2014)<br>(Adapted from, [35]) . . . . .               | 19 |
| Figure 2.7 | recurrent neural network and the unfolding in time of its forward com-<br>putation (Adapted from, [36]) . . . . .        | 21 |
| Figure 2.8 | Gated Recurrent Units (GRU) Cell (Adapted from, [38]) . . . . .  | 22 |
| Figure 3.1 | Collected data by source . . . . .   | 36 |
| Figure 4.1 | A Combined CNN-GRU-Softmax Model architecture for Amharic hate<br>speech and offensive language detection . . . . .      | 43 |
| Figure 4.2 | A Combined CNN-GRU-Svm/Rf Model architecture for Amharic hate<br>speech and offensive language detection . . . . .       | 44 |
| Figure 4.3 | Number of words description . . . . .  | 45 |
| Figure 4.4 | Maximum number of words . . . . .  | 46 |
| Figure 4.5 | Result of data covered by Maximum number of words . . . . .  | 46 |
| Figure 4.6 | A k-fold cross-validation approach [61] . . . . .  | 53 |
| Figure 5.1 | Comparison of classifier performance . . . . .   | 58 |
| Figure 5.2 | confusion matrix for best model . . . . .  | 59 |



---

|            |                                      |    |
|------------|--------------------------------------|----|
| Figure 5.3 | Comparison of classifier performance | 62 |
| Figure 5.4 | confusion matrix for best model      | 63 |
| Figure 5.5 | Comparison of classifier performance | 64 |
| Figure 5.6 | confusion matrix for best model      | 65 |

# List of Tables

|           |  |    |
|-----------|--|----|
| Table 2.1 | Amharic characters example . . . . .                                   | 25 |
| Table 2.2 | Amharic Characters with The Same Sound . . . . .                       | 27 |
| Table 3.1 | Normalization of character . . . . .                                   | 38 |
| Table 3.2 | Dataset Characteristics . . . . .                                      | 41 |
| Table 4.1 | Different values hyperparameters used during the grid search . . . . . | 49 |
| Table 5.1 | Over view of best hyper-parameters . . . . .                           | 56 |
| Table 5.2 | Classification performances using different embedding models . . . . . | 57 |
| Table 5.3 | Dataset distribution for three classes . . . . .                       | 61 |
| Table 5.4 | Dataset distribution for two classes . . . . .                         | 61 |

# List of Abbreviations

|             |                                     |
|-------------|-------------------------------------|
| <b>ANN</b>  | <b>Artificial Neural Network</b>    |
| <b>CBOW</b> | <b>Continuous Bag Of Word</b>       |
| <b>CNN</b>  | <b>Convolutional Neural Network</b> |
| <b>GRU</b>  | <b>Gated Recurrent Network</b>      |
| <b>RNN</b>  | <b>Recurrent Neural Network</b>     |
| <b>RELU</b> | <b>Rectified Linear Unit</b>        |
| <b>NLP</b>  | <b>Natural Language Processing</b>  |
| <b>SGD</b>  | <b>Stochastic Gradient Descent</b>  |
| <b>F1</b>   | <b>F1-Score</b>                     |
| <b>TP</b>   | <b>True Positives</b>               |
| <b>TN</b>   | <b>True Negatives</b>               |
| <b>FN</b>   | <b>False Negatives</b>              |
| <b>FP</b>   | <b>False Positives</b>              |

# *Abstract*

At this time, the number of social media users is increasing rapidly worldwide and in Ethiopia. So the use of social media becomes an essential tool for communication, increase tremendously in recent years. But this advancement also opens doors for trolls who poison these social media by their offensive and hate speech toward others. As a solution to this problem, this research proposed offensive and hate speech detection for Amharic text using a deep learning model. An offensive and hate speech data were collected from the Facebook and YouTube public page and manually labeled into hate speech, including their targets. Offensive language and not hate speech classes. The final dataset consists of 10,125 posts and comments. In recent times, Deep learning models such as Convolutional Neural Networks and Recurrent Neural Networks have been applied to offensive and hate speech detection with impressive results. The Convolutional neural networks are good at extract local information but cannot better express context information. Recurrent Neural Networks, on the other hand, can extract context dependencies and have a good classification effect, but training takes a long time. In this research, we used a combined CNN-RNN structure to use the strength of both CNN and CNN. The convolution layer will extract local features, and the GRU layer will use the sequence of those features to learn about the input. The feature maps extracted and learned by CNN and GRU are passed to SoftMax and machine learning classifiers such as SVM and RF classifier to generate the final classification. We used word2vec and Fasttext word embeddings with Cbow and skip-gram model architecture to represent words as vectors. The Best results obtained from Fasttext (Skipgram) and CNN-GRU-SVM model with an accuracy of 95.56%, the precision of 95.33%, recall of 95.44%, and F1 a score of 95.37% to classify comments and posts into religious hate speech, ethnic hate speech, offensive language, and not hate speech. However, the models lead to misclassifying offensive language as not hate speech class. Generally, replacing the SoftMax layer with an SVM classifier achieves good performance for offensive and hate speech detection including, the target of hate speech for the Amharic language.

***Key Words: Hate speech, Offensive language, word2vec, Fasttext, Deep learning, Amharic text***

# CHAPTER 1

## Introduction

### 1.1 Background

At this time, the number of social media users is increasing rapidly across the globe. Facebook, as the market leader, in April 2019, had 2.271 billion monthly active users, and YouTube is the second market leader, and it had 1.9 billion monthly active users<sup>1</sup>. In Ethiopia, the number of social media users has also increased, with Facebook, YouTube, and Twitter being the most well-known social media sites in terms of user numbers<sup>2</sup>. The above result shows that social media is an important communication platform for internet users to share thoughts, knowledge, get pieces of information, and give their ideas for something in terms of posting or commenting. Unfortunately, hate speech and offensive language can disseminate easily and quickly and leads to conflict between peoples [1, 2].

Although hate speech is protected under the provisions on freedom of speech in some countries, such as the United States, Canada, France, the United Kingdom, and Germany, laws prevent it from promoting violence or social disorder. Social networking sites services such as Facebook and Twitter have been criticized for not having done enough to restrict the use of their services

---

<sup>1</sup>Most used social media platform. Retrieved August 31, 2019, from <https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/>

<sup>2</sup>Social media STATS ETHIOPIA. (n.d.). Retrieved August 31, 2019, from <https://gs.statcounter.com/social-media-stats/all/ethiopia>

to harm people belonging to a specific race, minority, etc. However, they announced that they would seek to fight against racism and xenophobia. However, existing methods, such as Facebook and Twitter, have so far been to address the problem with manual effort, relying on users to submit offensive comments<sup>3</sup>. Not only does this require a great deal of action on the part of human annotators, but there is also a risk of discrimination on the grounds of subjective judgment. Besides, a non-automated task performed by human annotators would have a significant impact on response time, as a computer-based solution can achieve this task much faster than a human.

In essence, the Amharic languages are one of Ethiopia's widely spoken languages and working languages. The language is written left-to-right and has a unique script that lacks capitalization and 275 characters, primarily consonant-vowel pairs. It is the second-largest Semitic language after Arabic and spoken by the population as the first or second language [1, 3]. Amharic is still a language for which very few computational linguistic resources have built, and very little has been done to make useful higher-level Internet or computer-based applications. There are very few studies on identifying hate speech in the Amharic language; only four previous works [1, 2, 4, 5] on this issue have carried out.

This research creates a new dataset of posts and comments from the Facebook public page and comments from the YouTube public page. We also proposed a hybrid CNN-GRU-SoftMax, CNN-GRU-SVM, and CNN-GRU-RF, models and multiple word embeddings methods for offensive and hate speech detection, including the target of hate speech for the Amharic language.

## 1.2 Research Motivation

This study's motivation comes first from the increasing impact and popularity of the topic of hate speech. Hate speech is forbidden by law, characterized by known hate crimes, and therefore each country has its rules approved against such crimes. Also, the European Union Commission

---

<sup>3</sup>Press A. (2016, February 26). Zuckerberg in Germany: No room for hate speech on Facebook. Accessed September 01, 2019, <https://www.dailymail.co.uk/wires/ap/article-3465562/Zuckerberg-no-place-hate-speech-Facebook.html>

has established several strategies to reduce hate speech and has instructed Facebook, Twitter, YouTube and Microsoft to eliminate any hate speech from their platforms in less than 24 hours<sup>4</sup>. Ethiopia has also recently passed a law. The purpose of the law is to restrict individuals from speaking in a manner that encourages violence, promotes hatred and discriminates against a person or a group. The legislation prevents the spread of hate speech by broadcasting, publishing or using text, image, audio or video on social media<sup>5</sup>. The proposed rule is not automated, Which makes automatic hate speech detection method is required to tackle the problem of hate speech.

Secondly, the scientific study of hate speech and offensive language is recent from a computer science perspective. There are no tools that remove hate speech and offensive language automatically, and there is no sufficient amount of dataset about hate speech and offensive language in low resourced language. Social media sites have different approaches for removing content that violates their terms and conditions. They often depend on their users to report disturbing content, which is then reviewed by moderators who decide if the content should be removed or not. This manual approach scales poorly and quickly becomes infeasible as the amount of user-generated data grows larger every day. Although systems that can automatically detect unintended content solve the scalability problem, it is crucial that online communities preserve freedom of speech. The availability of shared resources, such as guidelines, annotated datasets, and algorithms, is a critical step towards the automated identification of hate speech and offensive language.

For this reason, this research has been motivated to examine offensive and hate speech identification for Amharic language using deep learning algorithms. The detection of online hate speech and abusive language is critical to address actual hate crime. It also contributes to developing a better detection system and reduces hate speech and offensive language datasets for future studies.

---

<sup>4</sup>Alex Hern, "Facebook, YouTube, Twitter and Microsoft Sign EU Hate Speech Code," May 31, 2016, <https://www.theguardian.com/technology/2016/may/31/facebook-youtube-Twitter-microsoft-eu-hate-speech-code>

<sup>5</sup>Admin, **ፌዴራል ነጋሪት ጋዜጣ**, accessed August 4, 2019, <https://webcache.googleusercontent.com/search?HATE-SPEECH-AND-DISINFORMATION-PREVENTION-AND-SUPPRESSION-PROCLAMATION.pdf>

### 1.3 Statement of the problem

The data spread in social media can be positive or negative, and hate speech and offensive language are toxic or destructive information disseminating on social media. The sharing of hate speech and offensive language on social media carried by a single person, activists, or organizations, and the targets can be a single person or groups. Spreading hate speech is a destructive action that can have some harmful effects, such as discrimination, social conflict, and even human genocide [6]. One of the most terrible genocides caused by the act of spreading hate speech was the 1994 Tutsi ethnic genocide in Rwanda [7].

According to [8], Ethiopia is one of the African countries currently experiencing problems due to hatred spread by social media. The level of hate speech continues to rise, ethnic tensions in countries keep rising both inside and outside the country. Controlling hate speech content manually in social media like Facebook and YouTube is problematic because it utilizes much time and requires much human power. Many studies have been done, on detecting hate speech and offensive language in English, German, Arabic, and other languages, but only a few studies have been done on hate speech detection in Amharic[1, 2, 4, 5]. The main reasons are the Amharic language's morphological richness and complexity, spelling variation, character redundancy, and lack of texts available on the web. The first three study [1, 2, 4] used a binary class dataset to classify posts or comments into hate speech or not. Considering hate speech as a binary problem leads to a non-hate expression that may contain offensive language to classified as hate speech [9]. The last study [5] researched hate speech and offensive language detection, but according to [6], hate speech has a specific target, category, and level. Hate speeches can belong to a particular category, such as ethnicity, religion, race, sexual orientation, etc.

This study built an Amharic post and comments dataset and utilized a deep learning algorithm with different word embedding for offensive language and hate speech detection, including detecting the target of hate speech. This study addressed the problem by answering the following questions:



- How the deep learning models separate offensive language and hate speech, including their targets from normal language?
- Which features extraction methods and classification layers are better for Amharic text offensive language and hate speech detection including, the target of hate speech?

## **1.4 Objective of the study**

### **1.4.1 General Objective**

Our research's main objective is to develop a deep learning algorithms that can detect hate speech, including the target of hate speech and offensive language for the Amharic language on social media.

### **1.4.2 Specific Objectives**

To achieve the general objective, we set the following specific objectives.

- To review related works in other languages with a different approach to have a conceptual understanding of the state-of-the-art for Amharic text offensive and hate speech detection.
- To develop an offensive and hate speech dataset for the Amharic language.
- To design a deep learning model for Amharic text offensive and hate speech detection.
- To compare the accuracy of different word embedding types and classification layers.

## **1.5 Scope and Limitation of the Study**

This study will only focus on hate speech and offensive language detection of comments and posts expressed only in text and written in the Amharic language on social media, such as Facebook and YouTube. The use of memes, audios, and videos within comments not considered.

## **1.6 Significance of the Study**

Although hate and offensive speech is not a new thing; it dominated by social media usages such as Facebook, YouTube and Twitter, the problem is becoming more prominent and visible than ever before. More and more people are now exposed to hate crimes, and social media platforms are acting as an engine for these malicious activities. Thus, the impact of the accurate detection of such hate and offensive speech is immense, and many areas of application will be profoundly affected. Some of the implications given below:

- Hate speech is certainly a common phenomenon on the Internet [10] and can cause serious harm to individuals in extreme cases. Thus, the importance of identifying and managing hate speech is evident from the apparent association between hate speech and actual hate crimes.
- The classification of posts or comments on social media platforms into hate speech and offensive language could be used as a data source by legal organizations to address this problem.
- Such systems could be adapted in other areas, such as police investigation, gender-based violence, counter-terrorism or cyberbullying prevention, thus enhancing the knowledge and capabilities of such organizations to identify potential criminal content.
- Prediction and advanced warning systems could allow us to take early action against the possible impact of hate crimes caused by the original hate speech.

- Automatic detection and removal of harmful content from the social media platform is a direct application, ensuring moderate users' integrity.

## 1.7 Thesis Organization

This study organized into six chapters in the following ways:-

**Chapter one**, Discussed above, it includes the introduction of the study, motivation, and statement of problems, the research questions answered in the proposed solutions, the scope and limitation, the objective of the study, and the significance of the study.

**Chapter two**, Presents the definition of hate and offensive language. It also presents hate speech on social media and Ethiopia and methodologies used in hate speech detection such as feature extraction, machine learning and deep learning classifiers. It discussed an overview of the Amharic language. Finally, it discussed the related work.

**Chapter three**. This chapter explains the detailed process for our dataset generation. We present the data sources, data collection, and preprocessing techniques. We described the annotation followed by annotation guidelines we proposed to ensure our proposed corpus' quality. We finished this chapter by exploring the different characteristics of the proposed dataset.

**Chapter four**. This chapter explains the proposed hybrid CNN-GRU deep learning model for Amharic hate speech and offensive language detection. After that, we describe the languages, libraries and the experimental setup. We also presented the parameters for the proposed CNN-GRU models. After that, we explain various evaluation metrics used to measure the performance of our model.

**Chapter five**. In this chapter, we present our experimental results and discussion for our proposed hybrid deep learning.

**Chapter six**. This chapter presents the conclusions and some suggestion for future work.

## CHAPTER 2

### Theoretical Background and Related Work

#### 2.1 Introduction

This chapter summarises the work done so far on the detection of hate speech and offensive language. We address the topic systematically, providing theoretical and practical aspects and giving an overview of the most recent approaches. In section 2.2, we define Hate Speech and offensive language. We processed the discussion of hate speech on social media in section 2.3, and 2.4 automatic hate speech and offensive language detection approach are presented. In section 2.5, Characteristics of the Amharic language described. Finally, the previous research on automatic hate speech detection in Amharic and English showed in section 2.6.

#### 2.2 Definition of hate speech and offensive language

##### 2.2.1 Definition of hate speech

Different organizations and scholars have defined hate speech based on different perspectives over the year. Some of the most influential definitions of hate speech presented below:

**Code of conduct, between E.U. and companies**, defines hate speech as "All conduct which

publicly incites violence or hatred directed toward a group of persons or a member of a group characterized by reference to race, colour, religion, descent or national or ethnic origin”<sup>1</sup>.

**ILGA-Europe 2010**, defined hate Speech is a public expression that spreads, incites, promotes or justifies hatred, discrimination or hostility towards a particular group. They add to a global climate of intolerance which, in turn, causes attacks toward these organizations more likely<sup>2</sup>.

**Ethiopia law**, defined Hate speech means speech that intentionally encourages hatred, discrimination or attack on a person or a discernible identity group based on ethnicity, religion, race, gender or disability;<sup>3</sup>.

**Facebook**, defined hate speech as ”Content that targets people based on their real or perceived race, ethnicity, national origin, religion, sex, gender or gender identity, sexual orientation, disability or disease are not permitted. However, we allow for clear attempts at humour or sarcasm that might otherwise mention a possible threat or attack. It includes Content that many people may find to be in bad taste (e.g. jokes, stand-up comedy, popular song lyrics, etc.)”<sup>4</sup>.

**Twitter**, defined hate speech as ”Hateful conduct: You may not encourage violence against or directly attack or threaten others based on race, ethnicity, religious affiliation, age, national origin, sexual orientation, gender, gender identity, disability. We also do not permit accounts whose main purpose is to cause harm to others based on these categories.”<sup>5</sup>.

**YouTube**, define hate speech as ”We support free speech and try to protect your right to express unpopular opinions, but we do not tolerate hate speech. Hate speech leads to Content that encourages violence or hatred against individuals or groups based on characteristics such as race or religion, ethnic origin, disability, gender, age, veteran status and sexual orientation/gender

---

<sup>1</sup>Alex Hern ”Facebook YouTube Twitter and Microsoft Sign E.U. Hate Speech Code,” May 31, 2016, <https://www.theguardian.com/technology/2016/may/31/facebook-youtube-twitter-microsoft-eu-hate-speech-code>.

<sup>2</sup>”Hate Crime & Hate Speech,” accessed September 2, 2019, <https://www.ilga-europe.org/what-we-do/our-advocacy-work/hate-crime-hate-speech>.

<sup>3</sup>Admin, **ፌዴራል ነጋሪት ጋዜጣ**, accessed August 4, 2020, <https://webcache.googleusercontent.com/search?HATE-SPEECH-AND-DISINFORMATION-PREVENTION-AND-SUPPRESSION-PROCLAMATION.pdf>

<sup>4</sup>”Community Standards,” Facebook, accessed September 14, 2019, <https://www.facebook.com/community-standards/hate-speech>

<sup>5</sup>”Hateful Conduct Policy,” Twitter (Twitter), accessed September 8, 2019, <https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy>.

identity. There is a distinct boundary between what is and what is not considered hate speech. For example, it's generally okay to criticize a nation-state, but not all right to make hateful, hateful comments concerning a group of people solely based on their ethnicity.<sup>6</sup>.

### 2.2.2 Definition of offensive language

Offensive speech is defined as a language or expression that offends and negatively characterizes individuals or groups. This kind of speech causes someone to feel upset, upset, insulated, angry, hurt, and disgusted. The separation between hate speech and offensive language often based on understated semantic distinction[9]. Offensive speech occurs when there is a low degree of hate speech characterization. The speech may contain a target but does not directly incite violence, attack or diminish based on target group characters. People often use offensive language to point in the debate, engage in a heated conversation, and condemn another violent act by others. Offensive speech contains sarcastic, mockery, and joke that offends considered offensive if spoken in a language with high levels of harmful, abusive, dirty words or phrases in both the oral and the text.

## 2.3 Hate Speech on Social Media

The Internet is one of the most extraordinary discoveries of humankind, which has brought together people from every race, religion, and nationality. Social media places such as Twitter and Facebook have connected billions of people and allowed them to share their ideas and opinions instantly. That said, there are several negative consequences, such as online harassment, trolling, cyber-bullying, and hate speech [8]. A social media nature makes a perfect place for peoples to create and share Content or participate online. These online platforms often exploited and misused to spread Content that can attack or diminish, which incites violence or hate against groups or individuals. Because the anonymity and mobility of social media platform features for

---

<sup>6</sup>"Hate Speech Policy - YouTube Help," Google (Google), accessed September 2, 2019, <https://support.google.com/youtube/answer/2801939?hl=en>.

security and privacy enable their users to hide their real identity behind the screen and express or spread hateful Content than they might otherwise.

## **2.4 Automatic Hate Speech and offensive language Detection approaches in Social Networks**

One of the main applications of social media mining is the automatic detection of events and behaviours that involve identifying people's behaviour in real-world events by monitoring their interactions with each other. Researchers can use these explosive data to obtain substantial insights[11]. This task depends primarily on text mining approaches, such as NLP and machine learning algorithms. Researchers have explored many automatic detection tasks on Twitter, such as anti-social behaviour detection, spam detection, natural disasters (e.g. earthquakes), trends, and public opinion events. To accomplish this study, several features and common patterns need to identify. Machine learning algorithms used to perform a classification task to obtain a targeted result from the data.

### **2.4.1 Features representation for hate speech detection**

To perform an automatic detection task, such as hate speech detection, the corpus' general features need to specify to allow the classification algorithms to complete the job. Some of these approaches presented.

#### **2.4.1.1 Dictionaries and Lexicons**

This feature usually used in unsupervised machine learning scenarios. The authors in [12] addressed the detection of profane words by taking advantage of corpora and lexical resources. They used several features and a general-purpose lexical resource to build their lexicon. Usually, lexicon-based approaches are not competitive with other components used in supervised

methods. They are domain-independent, and [13] also used the dictionary as a primary feature aggregating opinions and setting rates for subjective words.

#### 2.4.1.2 Bag-of-words (BOW) and N-grams

It can be considered a feature of the word co-occurrence. The vectorization method is carried on tokenized words in the corpus by assigning weight for each term according to its frequency in the tweet. The vectorization process performed using some statistical models ( e.g. TF-IDF weight). After that, the list of words together is called BOW, which will present as weight vectors[14]. N-gram representation means a sequence of adjacent N-words. Waseem and Hovy[15] analyzed the impact of using several features in conjunction with character N-gram to detect hate speech. They reported that using character n-gram representation is a great way to detect hate speech. BOW limited by its need to be accompanied by other features to improve the performance, but, from the other side, it is computationally expensive[16]. For N-grams, careful selection of the value of N required to avoid a high degree of distance between related words[17].

#### 2.4.1.3 Latent Dirichlet Allocation (LDA)

It's a method of probabilistic concept modelling. It mainly used to estimate latent concepts in the data set, and these latent concepts will be used as features instead of words. However, LDA is appropriate for unsupervised and semi-supervised machine learning settings. The authors[18]argued that BOW did not work well to detect abusive text on Twitter. Instead, they include highly expressive topical features and other lexicon features using the LDA model. This approach can be an alternative to supervised methods.

#### 2.4.1.4 Word embedding

Word embeddings refer to various techniques that map words or phrases to dense vector representations that enable the computation of semantic similarities of words. Word embeddings were first made accessible by [19] and primarily used in natural language processing tasks.



Words described approximately in an N-dimensional space that is appropriate to encode the semantics of each specific language. Each dimension encodes meaning in the world, such as tense (past, present, future), count (singular, plural), and gender (masculine, feminine, neutral). New techniques for building word embedding have based on neural network language models [20], which considers the adjacent words of the word as a context for the meaning of the word. Cosine similarity is a standard measure of vector similarity. Another way of presenting words is with a fixed vector length that can capture context and semantics, such as Word2Vec, Glove, fastText and Elmo. Word2Vec, Glove and fastText are the three most common models. These models based on the fact that words that occur and used in the same context tend to be semantically similar to others and have a similar meaning. These pre-trained word embedding techniques used to initialize the neural network's first layer.

**Word2Vec:**- Several techniques used to obtain these vectors with such properties, and almost all of them rely on unsupervised methods. One of the most popular models is Word2Vec[21], which learns representation using a neural network architecture to solve a predictive task. Such a neural network has a simple architecture consisting of an input layer, a hidden layer, and an output layer. There are two variations in this structure: Skip-gram and Continuous Word Bag (CBOW) [21], [22].

Skip-gram model topology shown in Figure 2.1, the aim Skip-gram is to predict the context of a word (words before and after that word) given that word. The input layer consists of the one-hot word encoded,  $V$  being the vocabulary size. The hidden layer has  $N$  neurons with  $N < V$  and the output layer is a one-hot encoded context word, with  $C$  as the context window size. Finally, the weights learned between the input and the hidden layer give the word vector the size  $N$ . The CBOW model inverts the topology of the skip-gram model. Figure 2.2, the aim of which is to predict a word in its context. In this case, the input layer consists of one-hot encoded context words. The output layer is the word indicated in the one-hot encoded format as well. After training the network, the word vectors are given the weight matrix between the hidden and output layers.

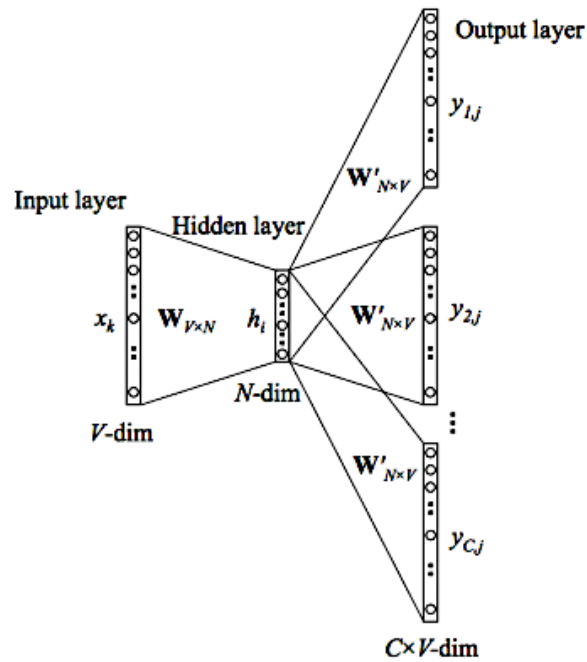


Figure 2.1: Skip-gram model (Adapted from, [22])

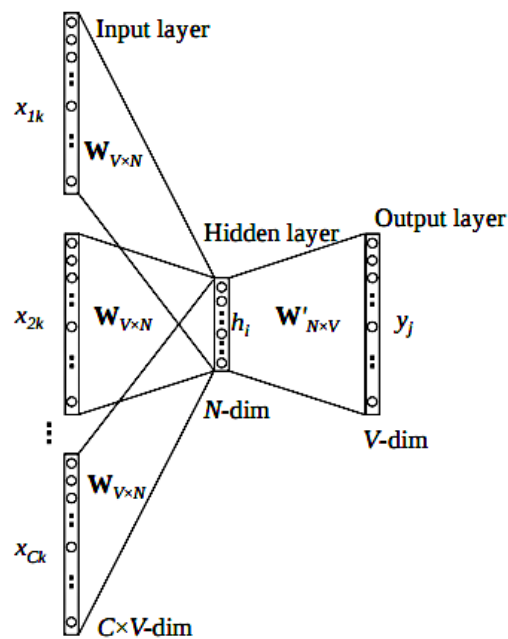
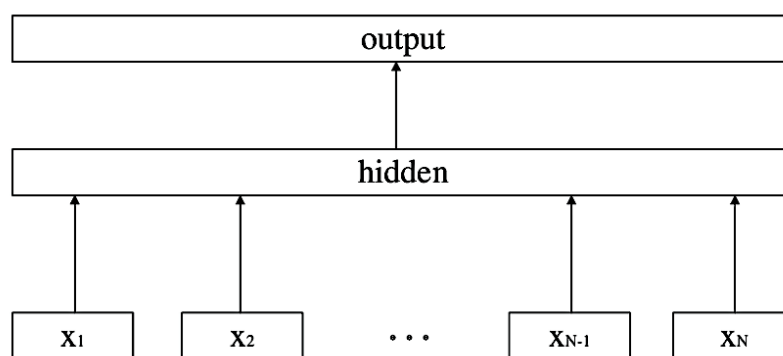


Figure 2.2: Continuous Bag of Words (CBOW) model (Adapted from, [22])

Although the training process depends on a neural network-based, supervised prediction model, the actual training results represent words rather than the neural network prediction model. Because of this idea, word embedding training is unsupervised and used in a variety of textual corpus without labelled datasets.

**FastText**:- FastText is the extension of the Word2Vec Continuous Skip-Gram Model. Use sub-word level embeddings to learn the word representations. Previous methods ignore the morphology of words and can not handle out-of-vocabulary words. It brings language with a large vocabulary and many rare words at a disadvantage. Some languages have a lot of different inflected forms without morphology. It is not easy to learn a good word representation for these words. FastText improves vector representation and considers morphology with subword units (character level information). The words divided into a bag of n-gram characters. Each word represented as a bag of characters n-grams, plus special boundary symbols < and > at the beginning and end of the word, plus the word itself in the set of its n-grams. A word represented by the sum of its character n-grams [23], [24]. fastText model topology is shown in Figure 2.3, which is a bag of words classifier with a hidden linear layer. Word vectors for all words in the document averaged into one document vector representation. The fasttext model has three parts: the inputs (sequence of words, a piece of text or a sentence), the hidden layer (Softmax function to compute the probability distribution) and the output (the probability that the word sequence belongs to a certain category).



**Figure 2.3:** Model architecture of fastText for a sentence with N-gram features  $x_1, \dots, x_N$ .  
(Adapted from, [23])

## 2.4.2 Machine learning for hate speech detection

After preparing the text to work with a machine, the classification algorithms used to perform the detection task. In terms of classifiers, machine learning approaches categorized as supervised, semi-supervised and unsupervised approaches.

### 2.4.2.1 Supervised learning.

This approach is a domain-dependent approach since it relies on the manual labelling of a large volume of text. Labelling is time-consuming and effort-consuming but is more efficient for domain-dependent events. Most of the approaches used to detect hate speech tasks are supervised methods. For example, Burnap and Williams [25] used several supervised classifiers to see the hate speech on Twitter, the results of which showed that all classifiers performed the same. Still, the different settings of features changed the accuracy of the model. Consequently, a classifier's choice depends on the characteristics extracted from the corpus. There are various machine learning algorithms used for hate speech detection. We found SVM and R.F. algorithms showed better results than L.R., NB, KNN, D.T., AdaBoost for hate speech and offensive language detection [26].

**Support vector machine:-** Support Vector Machines (SVM) is a compelling machine learning method initially implemented by [27]. The key concept behind the SVMs is to find a separating hyperplane with the most significant margin in the learning process. SVMs are described by the point vectors closest to the separating hyperplane [28]. The best hyperplane is defined as the most significant margin between the two groups. The SVMs can distinguish only two classes. LinearSVC overcomes this downside and automatically uses a one-on-one approach by default. It can also define this directly by setting the multiclass parameter to ovr (one-vs-the-rest). An example of linear separation can be seen in Fig. 2.4.

**Random Forest:-** Random Forest [29] is a supervised learning ensemble system for classification and regression problems. As its name creates a forest of decision trees to boost the overall outcome. The random forest has the same hyper-parameters as the decision tree and corrects the over-fitting decision tree's main problem. Random forest adds extra randomness to the mode while increasing. Searches for the best features in a random subset of features instead of searching for essential nodes when separating. Figure 2.5 illustrates the workings of the random forest.

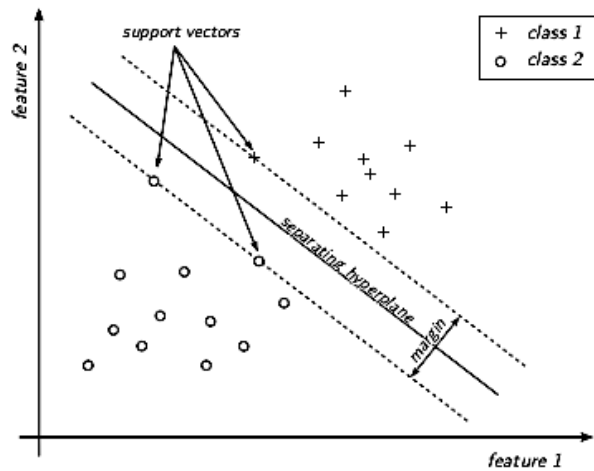


Figure 2.4: SVM linear separation (Adapted from, [27])

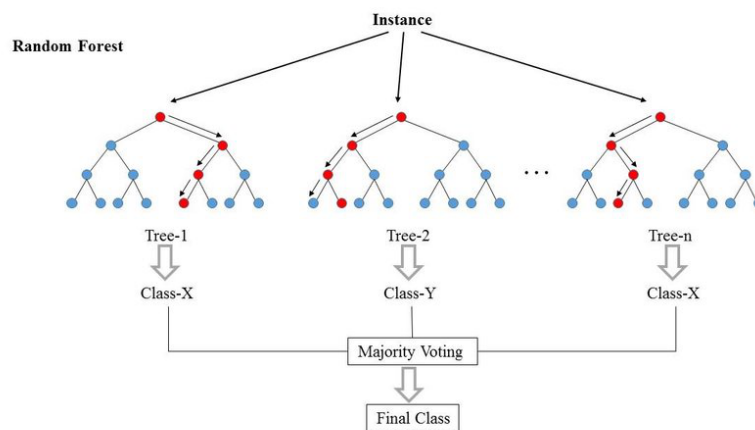


Figure 2.5: Random Forest (Adapted from, [en.wikipedia.org/wiki/Random forest ])

### 2.4.2.2 Semi-supervised learning.

In this approach, algorithms trained using both labelled and unlabeled text data. The use of labelled data in conjunction with unlabeled text data can effectively improve performance, as can be seen in the Hua et al. [30] model. They argued that unsupervised learning has limited ability to handle small-scale events. On the contrary, supervised learning can effectively capture small-scale events, but the need to manually label the data set reduces the model's scalability. To achieve the right balance between these two situations, the authors suggested a semi-supervised approach. Also, Xiang et al. [18] replaced the costly manual annotation with an automatically generated feature.

### 2.4.2.3 Unsupervised learning.

It is a domain-independent approach and is capable of handling content diversity while maintaining scalability[18]. It does not rely on human labour to label a large-volume training set; instead, it dynamically extracts key domain-related terms. Gitari et al. [13] used a bootstrapping approach to construct their lexicon by starting with a tiny seed of hate verb and then iteratively expanding it. Their model's best results obtained when they incorporated semantic hatred and features based on them.

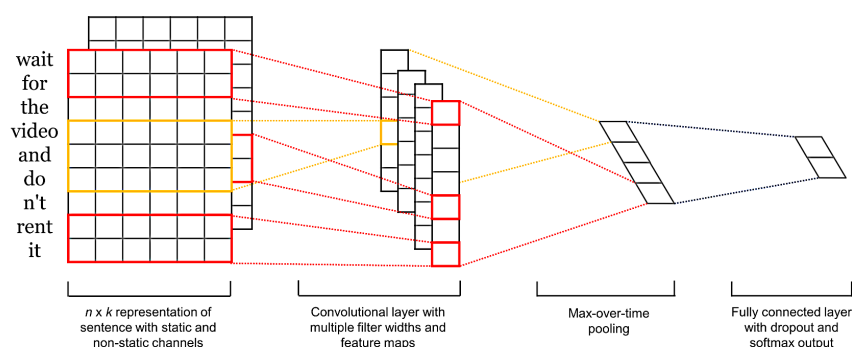
### 2.4.3 Deep learning for hate speech detection

Deep learning, commonly mentioned as hierarchical learning or deep structured learning, is a set of machine learning algorithms that attempt to learn a layered input model widely referred to as neural networks. It allows computational models made up of multiple layers of processing to understand data representations with various abstraction levels. Deep learning algorithms can find complex structures from large amounts of information using backpropagation algorithms to update their internal parameters. Researchers in this area, like [31], are trying to create models that learn representation from extensive unlabeled data. Various deep neural network architectures, such as deep faith neural networks, convolutional deep neural networks, and recurrent neural networks, have been used in many areas and have shown excellent results. These areas include computer speech recognition, natural language processing, and bioinformatics [32].

There are many types of deep neural networks and deep neural network architectures. This study briefly discusses networks used in our research, a specifically convolutional neural network with a combination of one of the variants recurrent neural network known as a gated recurrent neural network. Better performance (97.40% accuracy) observed when used in English with word embeddings as features [33], and they tested with Amharic.

### 2.4.3.1 Convolutional Neural Network

Convolutional neural networks (CNNs) initially designed to identify features in two-dimensional image data. CNN's first inspired by Hubel and Wiesel (1968) biological research on neurobiological image processing in the cat cortex. They have adapted to perform image recognition in machine learning and labelling scenes, facial recognition, and speech recognition. [34] have shown that CNN uses NLP tasks such as part-of-speech tagging, chunking, entity recognition, and semantic role tagging. Most recently, CNNs have been used in the classification of text [35] shown in Figure 2.6. In the classification of the sentence, the input is the sentence  $s$  and the output is the class  $y \in \{0, 1\}$ , which represents two different classes for the sentence  $s$ . Instead of image pixels, an input to CNN for Natural language processing tasks is a matrix of sentences or documents. CNN first processes the sentence through an embedding layer that transforms words into word embedding, called word vectors. Each line of the matrix correlates with a specific token, usually a word. The matrix dimension used by CNNs, in this case, is sentence length times the word embedding dimension. Each input and output layers, two hidden layers: the convolutional layer and the pooling layer, form a convolutional neural network.



**Figure 2.6:** Convolutional Neural Networks for Sentence Classification, Y. Kim (2014) (Adapted from, [35])

**Convolutional Layer:-** the convolutional layer transfers embeddings through convolutional layers to extract important and relevant  $n$ -gram features from the input sentence to create latent semantic information of the sentence. A convolution operation involves a filter  $W \in \mathbb{R}^{hk}$ ,

which applies to a window of  $h$  words to produce a feature. The window operations generate a new feature. A filter  $c_i$  is generated from a window of words  $x_{i:i+h-1}$  by

$$c_i = f(W.x_{i:i+h-1} + b) \quad (2.1)$$

The filter applies to each and every different window of word in the sentence  $\{x_{1:h} + x_{2:h+1} \dots, x_{n-h+1:n}\}$  to create a feature map

$$\mathbf{c} = [c_1, c_2, \dots, c_{n-h+1}] \quad (2.2)$$

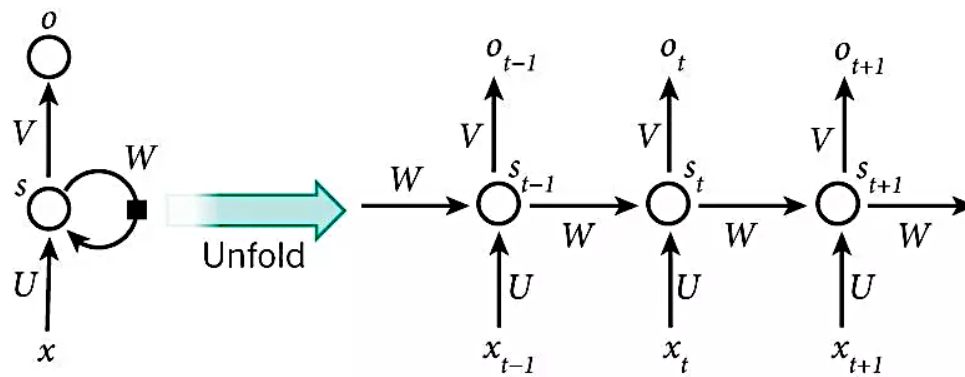
**Pooling Layer:**-The pooling layer, also referred to as the subsampling layer, reduces the spatial size of the representation by applying an operation such as max, sum, average, or L2-norm. Two main reasons motivate pooling: to convert a variable-size input into a fixed size output matrix that is typically needed for classification and reduce output dimensionality while keeping the most relevant information on the input phrase. The feature map  $\mathbf{c}$ , a max-over-time pooling operation, takes the maximum value  $\hat{c} = \max\{\mathbf{c}\}$  as the value corresponding to this specific feature, and This captures an essential part (the one with the highest value) for each feature map. The pooling layer has an additional advantage when working with natural languages in that it automatically standardizes variable sentence lengths.

### 2.4.3.2 Recurrent Neural Network

Convolutional neural networks are not capturing long-range dependencies. However, CNN can detect multiple adjacent word patterns, such as bigram, trigram, and fourgram. Recurrent neural networks (RNNs) proposed to overcome this limitation. The idea behind recurring neural networks designed to work with sequences. Recurrent neural networks have a memory of what has calculated so far and use it for current output computing. Theoretically, RNN uses information in an arbitrarily long sequence, but in practice, it is limited to a few time steps[36]. A typical recurrent neural network is shown in Figure 2.7.

Unlike a traditional deep neural network that uses different parameters for each layer, the RNN shares the same parameters ( $U, V, W$  above) throughout all steps. Where  $U$  is the input of the





**Figure 2.7:** recurrent neural network and the unfolding in time of its forward computation (Adapted from, [36])

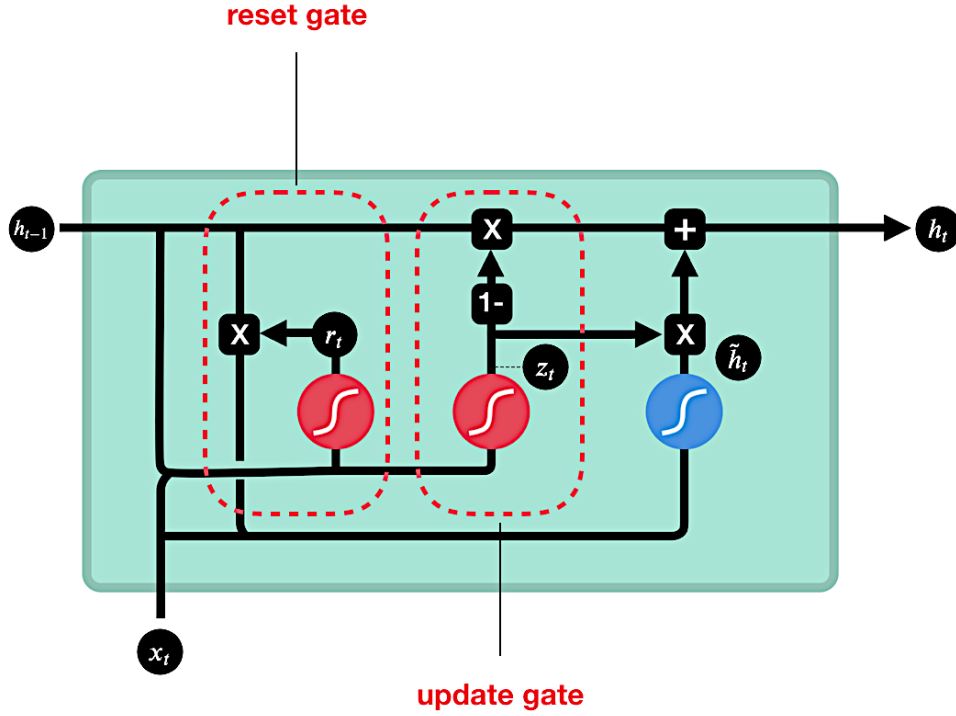
current sequence,  $W$  is the same node in the previous epoch's data, which is also used as input, and  $V$  is the non-regular output. It reflects that it performs the same task at each stage, only with different inputs. It significantly reduces the total number of parameters that the network needs to learn [36]. Where  $x_t$  is the input at time step  $t$ ,  $s_t$  is the hidden state at time  $t$ , and it is the network memory that calculated based on the previous hidden state and the input at the current step  $s_t = f(ux_t + U_r h_{t-1} + b_r)$ . Function  $f$  is usually not a linear function, such as tanh or ReLu. It is time  $t$  output [36]. The recurrent neural network has shown great success in many NLP tasks. The most commonly used type of RNNs is gated recurrent and short term memory recurrent neural networks (GRU and LSTM), which are much better at capturing long-term dependencies than a typical recurrent neural network. We used a gated recurrent neural network in this study because of its good with a small dataset [33].

### 2.4.3.3 Gated Recurrent neural network

Gatted recurrent neural network [37] is another gating structure and the newer generation of Recurrent Neural networks. It has two gates, a reset gate ( $rt$ ) and update gate ( $zt$ ) with no internal memory ( $ct$ ), see Figure 2.8.

The equations below show as the equation of the GRU

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$



**Figure 2.8:** Gated Recurrent Units (GRU) Cell (Adapted from, [38])

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$

$$\tilde{h}_t = \tanh(W_h x_t + U_h (r_t \circ h_{t-1}) + b_h)$$

$$h_t = (1 - z_t) \circ \tilde{h}_t + z_t \circ h_{t-1} \quad (2.3)$$

The reset gate ( $r_t$ ) decides whether the previous hidden state ignored. When the reset gate is near 0, the hidden state ignores the previous hidden state and takes only the current input. The ( $z_t$ ) update gate decides whether the hidden state updated with a new hidden state,  $\tilde{h}$ , which defines how much of the last memory should be stored. [38]. %subsectionCombination of Convolution Neural Networks and Recurrent Neural

#### 2.4.4 Combination of deep learning and machine learning classifier

In recent times, classifiers based on word embeddings and deep learning have performed well in hate speech and offensive detection tasks. A deep learning algorithm is good at learning invariant characteristics and capturing long-term relationships; nevertheless, deep learning only

utilizes fully linked softmax layers as the classification layer. The fully connected layer of a deep learning algorithm can not efficiently classify non-linearly distributed data [37]. Machine learning classifiers are good at constructing decision surfaces from well-behaved feature vectors but cannot learn complicated invariants or capture long-term relationships [39]. In this thesis, we replaced the softmax layer of the deep learning algorithm with a machine learning algorithms such as support vector machine and random forest machine learning classifier.

## 2.5 Amharic language

Amharic is the national language of Ethiopia's Federal Government. Amharic is a Semitic language that is spoken as a first or second language in many parts of Ethiopia. It is the second common spoken Semitic language globally (after Arabic) and closely linked to Tigrinya. It is the second-largest language in Ethiopia after Afan Oromo, the Cushitic language and perhaps one of the five most prominent languages on the African continent. Despite the comparatively large amount of speakers, Amharic is still a language for which significantly less computational linguistic resources have developed [3, 39].

### 2.5.1 Amharic Word Classes

According to [40], Amharic words are classified into five basic classes based on the morphology and position of the word in an Amharic sentence. These five categories are **ስም** (noun), **ግስ** (verb), **ቅፅሌ** (adjective), **ተውሳክ ግስ** (Adverb) and **መስተዋድድ** (preposition). **Noun**: are words used to name or identify any category of things, people, animals, places, or ideas, or a specific one of these entities. If a word can be pluralized by adding the suffix / (“owch”) **አች/ ዎች** and used to nominate something, such as a person or an animal, it is classified as a noun. It is the subject of a sentence. Pronouns, which were previously considered an independent category by linguistics professionals, have recently been categorized as nouns by considering the unique feature of the language in comparison to others rather than adopting other language structures.

The following are some Amharic pronouns: ይህ , ያ , እሱ, እሷ, እኔ , አንተ, አንች; quantitative specifiers, which include አንድ, አንዳንድ, ብዙ, ጥቂት, በጣም; and possession specifiers, which include የእኔ , የአንተ, የእሱ.

**Verb:** any word that can be placed at the end of a sentence and accept suffixes such as /ህ/, /ሁ/, /ሽ/, and so on to indicate subject markers. A verb is one that is used to indicate masculine, feminine, or plurality. The verb expresses the completion of an action and is used to end a sentence. For example, in the sentence “የእየብ እህት ድጋሚ ወለደች” the word “ወለደች” is a verb because it appears at the end of the sentence and closes the meaning of the sentence. The first order of the seven Amharic writing symbols is used by the majority of verb words. System of writing in Amharic Verbs can use the "-እየ" prefix morpheme, and their last symbol can be changed to the Amharic seven order writing system. Finally, this modified verbs may use the "-አሌ" suffix morpheme. Verbs, like nouns and adjectives, are derived from Verbal Roots by affixing the vowel "አ", Verbal Stems by affixing morphemes, and compound verbs by affixing compound verbs. Words made up of stems and verbs.

**Adjectives:** are words that describe nouns or pronouns to denote a thing's quality; that is, it specifies how distinct a thing is from something else. It will appear before a noun to qualify it with size, kind, or behavior. Example ቀላል፣ ትንሽ፣ ለብቻ፣ ተመሳሳይ፣ መጨረሻ.

**Adverb:** a word or phrase that changes the meaning of an adjective, verb, or other adverb by expressing manner, place, time, or degree. Some adverbs in Amharic include ሁሌጊዜ፣ ለምን፣ ማን፣ ምናልባት፣ ምክንያቱም፣ ስለዚህ፣ ተቀድሞ፣ በመካከሌ፣ ይልቅ፣ ደህና፣ ደግሞ፣ የተገለበጠ፣ እንደሁም etc.

**Prepositions** are word classes that express temporal and spatial relations (ውስጥ፣ ከታች፣ በፊት) or mark various semantic roles such as ( ለ እና ከ).

## 2.5.2 Amharic Character Representation

Amharic uses Geez characters; the characters trace back to the 4th century A.D. The Geez script's first forms included only consonants, while the subsequent variants of the characters represent phoneme pairs of consonant-vowel. Like Geez, Amharic writing utilizes characters formed by a consonant-vowel combination. In Amharic, seven vowels used, each in seven distinct forms that reflect the seven vowel sounds they are ከ ፣ ኡ ፣ ኢ ፣ ኣ ፣ ኤ ፣ ኦ ፣ ኧ ፣ ከ. There are 33 primary characters with seven forms representing a consonant and a vowel simultaneously, which gives the Amharic text said in the syllable. The initial order is the primary form, and there are 33 main forms with six sources for each giving 231 characters [3]. Table 2.1 adapted from [3] and shows an example of the Amharic alphabet.

**Table 2.1:** Amharic characters example

| 1st order | 2nd order | 3rd order | 4th order | 5th order | 6th order | 7th order |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| ሀ         | ሁ         | ሂ         | ሃ         | ሄ         | ህ         | ሆ         |
| Hä        | Hu        | Hi        | Ha        | He        | H         | Ho        |
| ለ         | ሉ         | ሊ         | ላ         | ሌ         | ል         | ሎ         |
| La        | Lu        | Li        | La        | Le        | L         | Lo        |
| መ         | ሙ         | ሚ         | ማ         | ሜ         | ም         | ሞ         |
| Ma        | Mu        | Mi        | Ma        | Me        | m         | Mo        |

## 2.5.3 The Amharic Punctuation

There are approximately ten punctuation marks in the Amharic language, but only a handful of them are found in a computer system. Furthermore, the majority of them are sentence separator tags. Punctuation mark such as ፡ (hulet neteb)/ (word separator or space), ፥ (Arat Neteb)/ (full stop (period)), ፣ (Netela Serez)/(comma), and ፤ (Dereb Serez)/(semicolon).

## 2.5.4 Challenges in Amharic Language

The main difficulty in the Amharic language are divided into three categories:

**Different ways of writing the same word:** It is usual to see different people write the same word in various ways. This spelling variation occurs often when the word is written with the nearest or corresponding character without translation, a process known as transliteration. For example, the words Ethiopia can be written in two ways: ኢትዮጵያ and ኢትዮጲያ, with the characters in the fifth position. These spelling variations should be considered when developing any Amharic word embedding.

**Amharic word formation:** Amharic is one of the most inflected languages, as it is possible to generate several words from a single Amharic term. For instance, from the stem word of ፈለግ one can generate ፈለገ፣ አሰፈለገ፣ ፈለገኝ፣ ለመፈለግ፣ በመፈለግ, and so on. This demonstrates the breadth of the Amharic vocabulary. Amharic also has numerous compound words formed by joining words with or without modification, such as አንድላይ፣ ቤተክርስቲያን፣ ህገመንግስት፣ ቤተመንግስት, and so on. Most NLP models learn such representations by ignoring word morphology and assigning a unique vector to each word. This is the most significant limitation, particularly for languages with large vocabularies and many rare words, such as Amharic. As a result, there is a need to create a precise word representation mechanism that is aware of word morphology.

**Character Redundancy :** Amharic adopted the entire Geez alphabet, including all seven orders of the 26 Geez symbols, without determining whether all 26 characters have meaning in the Amharic writing system. It then added some additional symbols to represent some other sounds that were not represented by the symbols of the Geez writing system. The Character redundancy in the Amharic FIDEL was caused by the unsystematic borrowing from Geez. Only about 231 of the 275 characters are necessary to represent Amharic [41], i.e., by using only one character from a group of characters with the same sound. The inflectional and derivational nature of Amharic makes morphological analysis extremely difficult. Many characters in the Amharic language are homophones, meaning they are pronounced the same but have different symbols. Without providing any benefits, this increases the number of features that will be extracted. For example, ሀይለኛ and ሐይለኛ, there is no clear rule about whether or not to use ሀ or ሐ, and people use such characters for semantically similar words interchangeably. In the table below

examples of the Amharic character repetition where more than one symbol is used for a given sound are and adapted from [5].

**Table 2.2:** Amharic Characters with The Same Sound

| Character | Other forms of characters | Other     |
|-----------|---------------------------|-----------|
| ሀ (he)    | ሐ and ኀ                   | ሃ ፣ ሐ ፣ ኃ |
| ሠ (se)    | ሰ                         |           |
| አ (a)     | ዐ                         | አ ፣ ኅ     |
| ጸ (tse)   | ፀ                         |           |

This variation in Amharic language spellings would unnecessarily increase the number of words that make up a sentence, potentially reducing the classifiers' efficiency and accuracy. As a result, word variants (spelling differences) caused by inconsistent use of redundant characters in Amharic sentence processing should be normalized. The different forms of a character that have the same sound are changed to one common form during the pre-processing activity of Amharic documents for the research.

## 2.6 Related Work

This section includes a review of literature on Amharic languages hate speech and offensive language detection, and foreign languages hate speech and foul language detection researches.

### 2.6.1 Hate speech and offensive language detection for local languages

We found only the following research for hate speech detection for the Amharic language. The first research was done by Mossie and Wang [1], and they created a dataset of 6,120 Facebook posts and comment to classify speech as "hate" and "not hate" using word2vec and TF-IDF for feature extraction and Random forest and Naive Bayes machine learning algorithms for classification. They achieved 79.83% accuracy by using word2vec feature selection and the Naive Bayes classifier. The authors conclude that the result promises to calculate a large amount

of data for the social network. The limitation of this study was they consider hate speech as a binary issue and did not consider the target of hate speech.

Mossie did the second research, and Wang [2] developed a model for hate speech detection and vulnerable community identification for Amharic texts on Facebook. They extended and modified the dataset from previous work [1]. They used Word2Vec word embedding controlled by TF-IDF, TF-IDF alone, and word n-grams as feature extraction. The classical GBT and R.F. algorithms and deep learning algorithms included RNN-LSTM and RNN-GRU model for classification. They achieved an accuracy of 92.56% using Word2Vec embedding and RNN-GRU. They consider hate speech as a binary issue and did not consider the target of hate speech.

The third research was done by [5], and they created a dataset of 5000 from Facebook posts and comments to classify speech as "offensive speech", "hate speech", and "neither", and they converted the three-class dataset to two class datasets by considering all offensive language as hate speech. They used word unigram, bigram, trigram, combined n-grams, TF-IDF, combined n-grams weighted by TF-IDF and word2vec for feature extraction and SVM, NB, and R.F. machine learning algorithms for classification. They achieved 53%F1-score by using word2vec feature selection and SVM classifier for three-class type, and they also achieved 73% F1-score by using word2vec feature selection and SVM classifier for three-class classification. The limitation of this study was they did not consider the target of hate speech.

The last research was done by [4], and they created a dataset of 30000 from Facebook posts and comments to classify speech as "hates speech" and "free" using word2vec for feature extraction and LSTM and GRU deep learning algorithms for classification. They achieved 97.9% accuracy by using word2vec feature selection and classifier LSTM. The authors conclude that the result promises to calculate a large amount of data for the social network. The limitation of this study was they consider hate speech as a binary issue and did not consider the target of hate speech.

All the previous research considers hate speech as a binary issue, and they did not detect the target of hate speech. Considering hate speech as a binary problem leads to a non-hate expression that may contain offensive language to be miss-classified as hate speech. If we combined



hate speech with other types of speech like offensive speech, then miss classification is solved. Hate speech also has a target such as an ethnicity, religion, sexual orientation, etc. So the target of hate speech needed to be detected and helps the authority to prioritize which hate speech targets must be addressed immediately. However, based on our literature study, there has been no research on offensive language and hate speech detection including, the detection of hate speech targets conducted simultaneously for the Amharic language.

## 2.6.2 Hate speech and offensive language detection for foreign language

As mentioned, the field of hate speech detection has attracted a great deal of interest over the last decade, and several approaches to hate speech detection have introduced. However, hate speech detection is still considered to be one of the most challenging machine learning problems and is far from being a problem that has resolved. Robata et al.[42] mention some of the challenges in hate speech, e.g. that abusive language with time is developing new slurs and clever ways to avoid being detected. As a result, longitudinal research was conducted over the years to see how trained models react over time. The model used different features, such as n-grams, word embeddings and other linguistic and syntactic features. The experiments carried out by [42] focus on features rather than model selection and how well the models perform with different parts. According to the authors, data collected from two domains, finance and news, and, according to the authors, contained some noise. All the features combined have yielded the most efficient model; however, looking at the individual parts, the n-gram characters have performed best. Waseem and Hovy [15] also found that their model was performing well with character n-grams as features. They experimented with extra-linguistic characteristics such as gender and location, although such demographic characteristics brought little improved performance to the performance of the model.

Over the last few years, the transfer of knowledge from word embeddings to be used as inputs to neural networks has been a common technique used by field researchers. Gambäck and Sikdar [43] experimented with character n-grams combined with word 2vec embeddings. Four CNNs trained, one character based on 4-grams, one with word vectors from Word2vec, one with

randomly generated word vectors, and finally one with a combination of characters n-grams and word embeddings from Word2vec. All models have trained on Waseem [44] data. With an F1-score of 0.783, the most efficient neural network was a model with knowledge transferred from Word2vec word embeddings. Adding the character n-grams to this system enhanced the precision but lowered the recall to a lower F-score.

Badjatiya et al. [45] experimented with a variety of traditional machine learning models and neural networks, including LSTMs. Besides, word embeddings from GloVe have used to increase overall performance possibly. Their best-performing model was the LSTM neural network with random word vectors where the output of the neural network used as input to the Gradient Boosted Decision Tree (GBDT). This system generated an F1 score of 93%, which increased the state-of-the-art by approximately 18 F1 points.

Pitsilis et al. [46] had similar findings with their RNN ensemble, although without the use of word embeddings. Instead, they used word-based frequency to convert tweets to vectors. This approach is valuable because it is independent of the message's language. They fed standard vectorized word unigrams to multiple LSTM networks and used two classification aggregation mechanisms, Voting and Confidence. The majority of voting is the preferred mechanism, but the most trusted classifier's preference is if all classifiers are not in agreement. The approach showed the potential of the ensemble methods for detecting hate speech and exceeded the previous state-of-the-art with an F1-score of 0.932.

Park and Fung [47] developed a hybrid model that attempted to capture features from two input levels. Their system includes two CNNs, one of which is character-based, and the other one is word-based. These two components joined at a deeper level of the network where the classification carried out. Park and Fung created multiple datasets by merging Waseem and Waseem and Hovy samples. Their objective was to explore a two-step classification technique where the first phase was to classify the data set as either abusive or normal. Since the model organized as offensive, the second step would further collect the data set as either "Sexist" or "Racist." The two-step approach was compared with a one-step multiclass classification, although the results are very similar. However, the CNN hybrid has outperformed the L.R. and SVM models in the

one-step multiclass category. Park and Fung have shown the potential of a two-step classification and will continue to explore this approach. They suggest training the two classifiers with separate datasets, e.g. a large general one for the first classifier and a more task-specific dataset for the second classifier.

Meyer and Gambäck [48] proposed an optimized architecture that would combine components with CNNs and LSTMs into one model. One element of the system used character n-grams as input, while another part used word embeddings as inputs. The output of each component has been joined to obtain the final classification. Meyer and Gambäck used the Waseem and Hovy dataset and achieved higher results than the previous comparable methods. They argue that different convergence rates in the system's components may have decreased performance and suggest using dynamic convolutions as future work.

Most of the research discussed throughout this section uses the widely known dataset from Waseem and Hovy or the slightly updated Waseem dataset. Another dataset commonly used in recent studies is that of Davidson et al. They differentiate hateful language from offensive and normal language, making the task more difficult than separating hurtful language from just normal speech. Zhang et al. [33] used this dataset, together with others, to evaluate the performance of their model, which consists of a combination of LSTM and CNN. However, the offensive class merged with the normal level. Therefore the final dataset contained only two categories: hateful and normal language. The system works by feeding word embeddings from Word2vec to CNN to generate input vectors for GRU cells' LSTM network that perform the final classification. The model outperformed the state-of-the-art of 6 out of 7 datasets and shows that deep neural networks effectively detect hate speech.

Founta et al. [49] used the dataset from Davidson et al. but decided not to combine the offensive samples with the normal ones, thus took the problem of separating hateful and offensive language. Their system comprises two networks, one RNN based on text input and one feed-forward ANN with metadata input. The two networks run parallel to the concatenation layer and finally the classification layer. Founta et al. have tried many configurations of these two networks with different features and training strategies. The best-performing model achieved

an F 1-score of 0.89, slightly just below the baseline L.R. model by Davidson et al. Unfortunately, the paper did not provide metrics for each class, which would be essential to consider the model categorized accurate hate speech samples. Kshirsagar et al. [50] created a model that outperforms the baseline of Davidson et al. The method, named Transformed Word Embedding Model (TWEM), consists of pre-trained word embeddings as inputs to multiple MLP layers. Simple network architecture enables a few parameters and minimal pre-processing features compared with previous methods. The model achieved an overall F 1-score of 0.924, with a baseline of 0.900. However, the increase in F-score is only due to better results in the "Normal" and "Offensive" classes, which performs worse in the "Hate" class with an F 1-score of 0.49, which is two F-score points far below the baseline. This is in line with [51], who argue that their research's key finding is the visible difficulty of differentiating hateful language from offensive language. They implemented a few supervised machine learning algorithms and more advanced classification algorithms and tested them on the Davidson et al. dataset (2017). Classifiers have experimented with a variety of features, including n-grams and cluster-based word representations. The RBF kernel SVM meta-classifier achieved the best accuracy with an F 1 score of 79.8 per cent. The problematic class to categorize was the "Hate" class, often confused with the "Offensive" category.

## **CHAPTER 3**

# **Proposed Dataset for Hate speech and Offensive Language Detection in Amharic Text**

### **3.1 Introduction**

This chapter explains generating our dataset and the steps we have taken to ensure our proposed dataset's quality. This chapter has two parts. The first part describes developing the dataset, representing the sources, collecting data from those sources, the preprocessing steps we conducted on the data, and the annotation process, including detailed annotation guidelines to ensure dataset quality. We also describe the agreement between the annotators. In the last part, we conclude this chapter by briefly explaining the characteristics of the dataset generated. The resulting dataset used to develop and evaluate the hate speech and offensive language detection system of the Amharic text.

### **3.2 Dataset Generation Process**

This section explains the dataset generation process for our proposed Amharic hate speech and offensive language detection. The barrier to detecting offensive and hate speech detection, including the target of hate speech for the Amharic text is the unavailability of publicly available

datasets. So we build a new dataset for offensive and hate speech detection, including their targets for Amharic text. Creating a dataset for hate speech and offensive language consists of three phases: data collection, data processing, and data annotation. A recent dataset for Amharic hate speech was published by [4] after we created our dataset. This dataset contains two labels, Hate and Free. We did not use this dataset because the dataset is not appropriate for our task, and there is a limitation of time and budget to convert the dataset into our study.

### 3.2.1 Data Collection

The first step of dataset creation starts with data collection. The most popular social networks that Ethiopia peoples use are Facebook Twitter, and YouTube <sup>1</sup>. We did not use Twitter for data collection because Twitter restricts tweets to 140 characters, and you can write a book on Facebook. Twitter is less about making social connections, and People can use Twitter to keep track of important topics, people, and conversations that are relevant or interesting to them. It's a much more distant link. People use Facebook to connect with friends, family members, and others with whom they want to stay in touch. Due to this reason, we use these social networks for data collection.

**Data collection from Facebook:-** To complete the first steps of dataset creation, we need to collect data from Facebook, and Facebook is the first platform; we included in this research for data collection. Facebook provides a service called the Facebook Graph API<sup>2</sup> for development or research purposes. It is illegal to scrap Facebook with any third party software or APIs. The Facebook Graph API makes it easier for developers to do what they need for development purposes. Facebook page information is public information, and the definition of privacy is relaxed here, so there is no room for violating the confidentiality of any random person. We can obtain some raw data from any page from the creation of the pages. We can get posts, post

<sup>1</sup>Most used social media platform. Accessed August 19, 2019, from <https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/>

<sup>2</sup>Graph API: Developer Documentation. (n.d.). Accessed September 01, 2019, from <https://developers.facebook.com/docs/graph-api>,

and comments, total reactions, total comment counts per post, comments on the post, etc. We collected the data based on different criteria, and those criteria are detailed below.

The first criteria were to select Facebook pages with more than 100 000 followers and likes, making more active public pages. The second criteria were the language of posts and comments. We have decided to collect posts and comments written in the Amharic language. The third criterion was to select the Facebook page that posts on politics, religious, and ethnic every daily. We scraped posts and comments from the following Facebook pages.

**News Media and Broadcasting Pages:-** Dire Tube, Fana Broadcasting, VOA Amharic, EBC, ESAT.

**Bloggers and Journalists' Pages:-** Yegna Tube – የኛ ተዩብ, Haro Tube – ሐሮ ተዩብ:- Getu Temesgen, Ethio Daily ኢትዮዴይሊ.

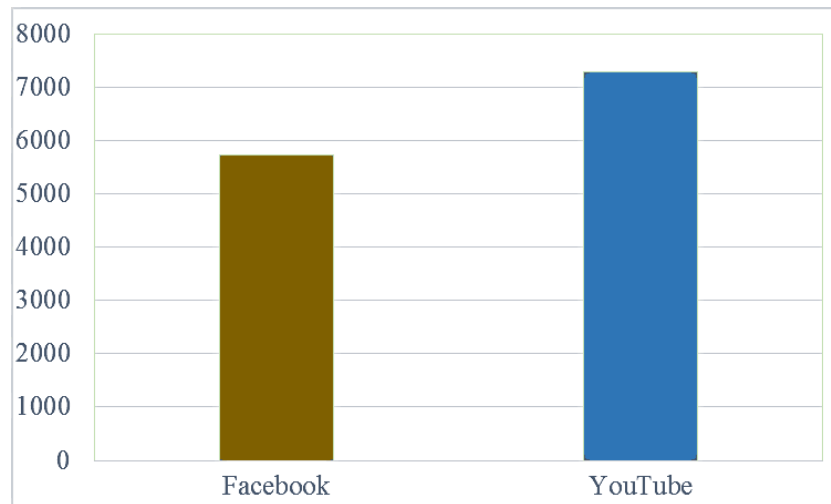
**Religious Media and Religious Group Pages:-** አርቶዶክስ ተዋህዶ ለዘለአለም ትኑር, ማኅበረ ቅዱሳን ዋናው ማዕከል, ቢቢኤን የናንተው ድምፅ:- Ahmedin Jebel official, Memehar Dr Zebene Lemma, Ustaz Abubeker Ahmed, Marsil TV.

**Political Party and Politician Pages:-** Patriotic Ginbot 7, EPRDF Official, የአማራ ብሔራዊ ንቅናቄ NMA, Amhara Democratic Party /ADP/ CC.

**Government Office Official Pages:-** FDRE Communication Affairs Office Ethiopia.

**Data collection from YouTube :-** YouTube is the second platform; we included in this research for data collection. In Ethiopia, people are actively interested in politics, films, drama, sports and faith. We scraped the comments posted on a video in the following three categories: News, Religion, and Politics. Because our primary focus is on collecting comments related to hate speech and offensive language written in Amharic text. We have selected videos that are most likely to be discussed and viewed in the Amharic language. We used keywords to find related YouTube videos, and we scraped comments with the YouTube Comment Scraper. In this platform, we downloaded the comments and replies from YouTube video by using YouTube

comment-scraper<sup>3</sup>. Figure 3.1 represents all the datasets that we collected from Facebook and Youtube public pages.



**Figure 3.1:** Collected data by source

### 3.2.2 Data preparation

Our raw data consist of around 13003 posts, and comments scraped from more than 100 videos and 150 posts and comments in the categories mentioned above. We collected our data for four months. Because we did not have any way to sample data based on language, so our raw data consist of comments in Amharic text, and Amharic text written in Latin Script. so we have used the following operation for data preprocessing.

#### 3.2.2.1 Data cleaning

The comments and posts that we scrapped from YouTube and Facebook contain many unwanted elements such as unique character, punctuation's, symbol, and emojis. We are preparing to use preprocessing approaches to design social media texts for training. In this stage, we follow

<sup>3</sup>Klostermann, P. (n.d.). YouTube Comment Scraper. Accessed September 01, 2019, from <http://ytcomments.klostermann.ca/>



the method used by [1]with some modification.The following tasks are performed to clean the collected data.

- Removing all Amharic Punctuation and Geez numbers ( ፡, ።, ፣, ፤, ፥, ፦, ፧, ፨, ፩, ፪, ፫, ፬, ፭, ፮, ፯, ፰, ፱, ፲, ፳, ፴, ፵, ፶, ፷, ፸, ፹, ፺, ፻) in posts and comments.
- Removing all number (0,1,2,3,4,5,6,7,8,9) in posts and comments.
- Removing special characters, emoji and symbols (\$ % & ) in posts and comments.
- If a posts and comments contains extra white space then we simply trim the posts and comments.
- Eliminating symbols in posts and comments like hashtags, URLs, links, and emojis.

---

**Algorithm 1:** Data cleaning algorithm

---

1. *Read the posts and comments in the dataset;*
    - If the posts and comments contains HTML and URL then remove it*
    - If the posts and comments contains pecial characters then remove it*
    - If the posts and comments contains symbol then remove it*
    - If a posts and comments contains Amharic punction and geez number then removeit*
    - If posts and comments contain English word and number then remove it*
    - If a posts and comments contains emoji then remove it*
    - If a posts and comments contains extra white space then remove it*
  2. *Return clean posts and comments;*
- 

### 3.2.2.2 Normalization

Normalization used to solve the redundancy problem of Amharic language, the same sound character with different character form . It is a process of changing words into a single form by performing character replacement with a similar sound to one common form of character. The change of the character into one common representation does not cause a meaning difference, but it decreases the chance of getting the same feature with different characters, which leads to duplication of a feature.

**Table 3.1:** Normalization of character

| Amharic characters that have the same sound and function             | Normalized to  |
|--|--|
| <b>ሐ, ሐ</b> | <b>ሀ, ሀ</b> |
| <b>ሠ, ሠ</b> | <b>ሰ, ሰ</b> |
| <b>ሰ, ሰ</b> | <b>አ, አ, አ, አ, አ, አ, አ, አ, አ, አ, አ, አ, አ, አ, አ, አ, አ, አ, አ, አ</b>    |
| <b>ጸ, ጸ</b> | <b>ፀ, ፀ</b> |

**Algorithm 2:** Normalization Algorithm

1. Read the posts and comments in the dataset;
  - If the posts and comments contain characters **ሐ**, **ሐ** or any order then Changed to **ሀ**
  - If the posts and comments contain characters **ሠ** or any order then Changed to **ሰ**
  - If the posts and comments contain characters **ጸ** or any order then Changed to **ፀ**
  - If the posts and comments contain characters **ሰ** or any order then Changed to **አ**
2. Return clean posts and comments;

After cleaning unwanted elements from the collected data, our data's size is approximately 11,625 Amharic posts and comments. We did use stop word removal in the preprocessing stage because it changes the meaning of posts and comments.

### 3.2.3 Dataset Annotation

The comments and posts that were collected during the data collection and data preparation phase were unlabelled. Within a supervised learning algorithm to classify comments and posts effectively into different classes, the comments and posts first should be appropriately labeled. There are two techniques for Data annotation for hate speech manually and crowdsourcing. The manual annotation performed better than crowdsourcing annotation [52]. Due to this reason, we applied a manual annotation mechanism for our tasks to classify posts and comments into hate speech, including the target of hate speech and offensive language.

### 3.2.3.1 Annotation Guidelines

To clarify the annotation process, the following annotation guidelines provided to the annotator to annotate posts and comments as ethnicity hate speech, religious hate speech, Disability hate speech, Gender hate speech, offensive language and not hate speech reliably. Annotation guidelines provided here are based [9, 53] and the laws that were prepared by the Ethiopian government<sup>4</sup> and by interviewing law experts to interpret the law. Here is the annotation provided to the annotator.

**Religion hate speech:-** If posts and comments attacks or diminishes, that incites violence or hate against groups, based on their religion such as Islam, Ethiopian orthodox, Christian, Catholic, protestant, Waaqeffanna or a religious organization, or a particular creed.

**Race/ethnicity hate speech:-** If posts and comments attacks or diminishes that incites violence or hate against individual or groups, based on Ethiopian ethnic groups such as Amhara, Oromo, Tigre, Agew, Sidama, Afar. etc. and based on physical characteristics such as face shape, height, skin colour, and others.

**Disability hate speech:-** If posts and comments attacks or diminishes, that incites violence or hate against individual or groups, based on physical deficiencies or Disability (e.g. the shape of the face, eye, autism, idiot, blind, deaf, etc.), whether it is insulting anyone or a group with terms specific to a disability.

**Gender hate speech :-** If posts and comments attacks or diminishes, that incites violence or hate against individual or groups, based on insulting someone or a group using terms that insult their Gender. This category includes any violence to a particular gender or any devaluation based on the Gender of the individual.

**Offensive language:-** If posts or comments contains a language or speech that embarrasses and negatively characterizes individuals or groups of people. Still, it does not include any hate speech target listed above.

---

<sup>4</sup>Admin, ፌዴራል ነጋሪት ጋዜጣ, accessed June 4, 2020, <https://webcache.googleusercontent.com/search?HATE-SPEECH-AND-DISINFORMATION-PREVENTION-AND-SUPPRESSION-PROCLAMATION.pdf>

**Not hate speech:**- If posts or comments do not contain any hate speech and offensives language described in both hate and offensives speech section.

### 3.2.3.2 Manual annotation process

The limiting factor for the development of a large dataset is human annotators' availability. We used three annotators, and we carefully selected them based on different backgrounds, such as religion, ethnicity, gender, and educational backgrounds. All annotators are not linguistic experts, but they can write, read, speak and listen to the Amharic language fluently. Before the annotation starts, we ensured that the annotators understood the definition of hate speech prepared by the Ethiopian government and the meaning of the offensive language used in this study. The annotators were asked not to allow their personal beliefs that influence their judgment to reduce bias. The annotators, called person A, person B and person C, given the same set of 11,625 unique posts and comments to annotate together with instructions defining what comments and posts belong in each category. The purpose of letting them annotate the same amount of posts and comments was to evaluate their annotation agreement. The final label in this phase decided using majority voting. Since we use three annotators, each post and comments label must agree with three annotators. If there is no agreement between the annotators in giving the label, then the posts and comments are deleted. There were 1500 posts and comments deleted from the annotation results because there was no agreement in hate speech categories and offensive language labels. From this phase, we get 10,125 posts and comments. According to [54], this 100% of the annotation agreement shows that the annotation result has an adequate level of understanding and used for a research experiment.

### 3.2.4 Dataset Characteristics

Our final dataset contains 10,125 unique posts and comments, where we have 2062 Ethnic hate speech, 1875 Religious hate speech, 2300 offensive language, and 3888 Not hate speech posts and comments. We did not get any posts and comments about the type of Disability and Gender

hate speech specifically. In our proposed dataset, the posts and comments contain Disability and Gender hate speech accompanied by Religious and Ethnic hate speech. The dataset contains 39948 unique words or tokens. The minimum number of words is 2, the average number of words is about 65, and the maximum number of words is 7322. Table 3.2 shows the distribution of our proposed dataset.

**Table 3.2:** Dataset Characteristics

| Classes               | Number of posts and comments for each class | Total number of comments and posts | Total number of tokens | Maximum length | Minimum length | Average length |
|-----------------------|---|------------------------------------|------------------------|----------------|----------------|----------------|
| Religious hate speech | 2062  | 10,125                             | 39948                  | 7322           | 2              | 65             |
| Ethnic hate speech    | 1875  |                                    |                        |                |                |                |
| Offensive language    | 2300  |                                    |                        |                |                |                |
| Not hate speech       | 3888  |                                    |                        |                |                |                |

## **CHAPTER 4**

# **The proposed model for Amharic hate speech and Offensive Language Detection**

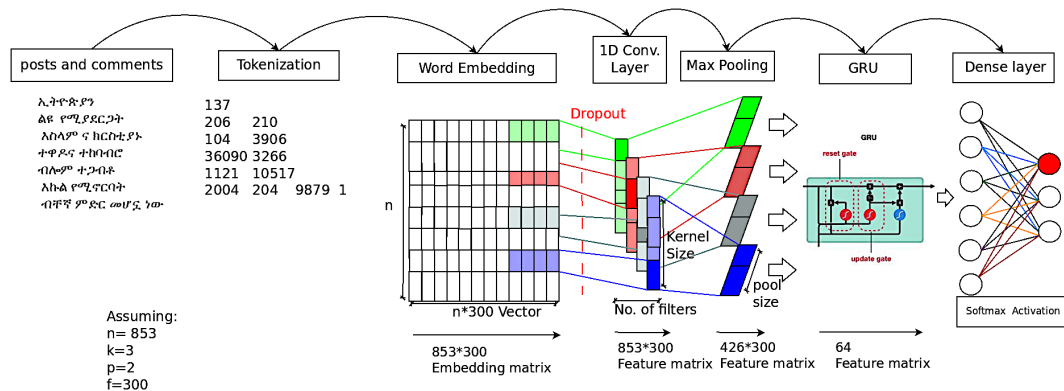
### **4.1 Introduction**

This chapter will describe our proposed models to classify Amharic hate speech, including their targets, offensive language, and not hate speech. In section 4.2, We start by explaining the proposed models. Section 4.3 describes hyperparameters for the proposed models. The programming languages and libraries and the experimental setup described in section 4.4. In section 4.5, we present the metrics used to measure our model's performance.

### **4.2 The proposed model**

In this study, we used the combination of both architectures, CNN-RNN, for our task. Because CNN is good for extract features and struggles to capture the long-term dependency of a text, on the other hand, RNN is good to learn sequential structure, but it needs much time. The combination of CNN-GRU integrates the strength of both convolutional and recurrent neural network architectures. We propose a single CNN-GRU-softmax, CNN-GRU-SVM, and CNN-GRU-Rf model for offensive and hate speech, including their targets, and not hate speech for Amharic

text. The CNN-GRU-softmax architecture has improved performance over individual CNN or GRU in sentiment analysis and hates speech detection models [33, 55]. Replacing the softmax layer of the deep learning algorithm with a machine learning classifier has shown an improvement in performance over CNN in sentiment analysis [56, 57]. The idea behind the integration of CNN and GRU is that the convolution layer will extract local features. The GRU layer will use the sequence of those features to learn about the input and softmax, SVM, and Rf classifiers used to generate the final classification. Figures 4.1 and 4.2 show the model’s architectural design that we used for this study. The model uses word embeddings as input to the convolutional layer. The convolution layer’s output is fed into the max-pooling layer to condense into smaller dimensions to reduce the training parameter and extract the necessary features. Then, the max-pooling layer’s output is provided into the GRU, followed by the softmax, SVM, and Rf layer that determines whether posts and comments belong to hate speech, including their targets or offensive language and not hate speech. So our proposed model has seven layers: tokenization, embedding layer, dropout layer, convolutional layer, pooling layer, specifically max Pooling, GRU layer, and output layer. The detail of each layer explained below.



**Figure 4.1:** A Combined CNN-GRU-Softmax Model architecture for Amharic hate speech and offensive language detection

**Tokenization:-** Figure 4.1 shows the model takes Amharic datasets (posts and comments) as input. For example, if the posts or comments are something like **ኢትዮጵያን ልዩ የሚያደርጋት እስላም ና ክርስቲያኑ ተዋዶና ተከባብሮ ብሎም ተጋብቶ እኩል የሚኖርባት ብቸኛ ምድር መሆኗ ነው**, our proposed model to produce a Not hate speech for it, but a neural network cannot work on a

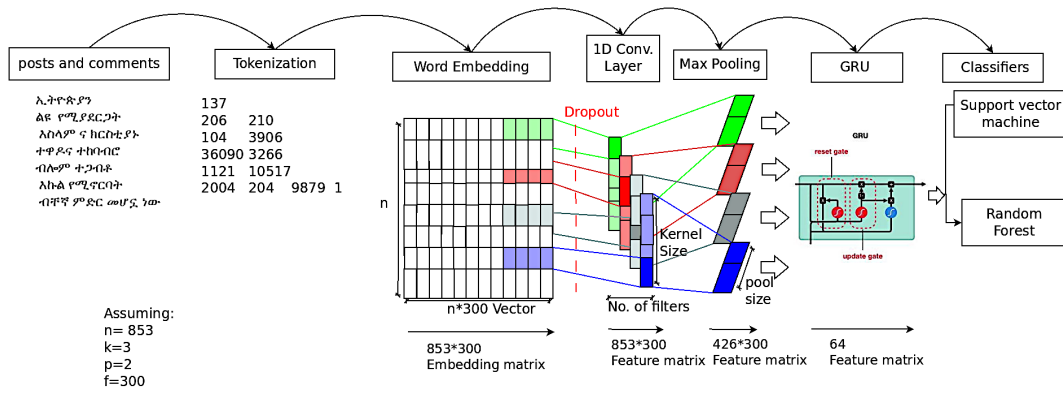


Figure 4.2: A Combined CNN-GRU-Svm/Rf Model architecture for Amharic hate speech and of-fensive language detection

raw text, Neural networks can only learn to find patterns in numerical data before we feed posts or comments in a neural network as input, we have to transform each word into a numerical value. This method is often described as word encoding or tokenization. A standard encoding method is as follows.

1. We record each of the unique words that appear in our dataset and these as the vocabulary of our model.
2. We encode each vocabulary word as a unique integer, called a token. These tokens are often assigned based on the frequency of occurrence of a word in the dataset. The word that looks most commonly during the dataset will hold the associated token: 1. For example, if the most common word were 'ነው', it would have the associated token value of 1. The next most frequent word will be tokenized as 2, and that method continues.
3. In code, this word-token association is represented in a dictionary that maps each unique word to their token, integer value: So the word አ.ት.ዮ.ያን has an integer ID, or token 137, ልዩ ለዩ የሚያደርጋት has a token 206, የሚያደርጋት has a token 201, እስላም has a token 104, ክርስቲያኑ has a token 3906, ተዋዶና has a token 3609, ተከባብሮ has a token 3266, ብሎም has a token 1121, ተጋብቶ has a token 10517, እኩል has a token 592, የሚኖርባት has a token 6205, ብቸኛ has a token 2004, ምድር has a token 204, መሆኗ has a token 9879 and ነው has a token 1. So now we have converted the raw text into a list of integers called tokens. Eventually, after assigning those tokens to unique words, we can then tokenize the whole dataset. There are frequently so many words in



a distributed corpus that these tokens will vary from the value 1 to 39948, then we can call the 'texts to sequences' approach to get a sequential representation of every sentence. The 'texts to sequences' of ኢትዮጵያን ልዩ የሚያደርጋት እስላም ና ክርስቲያኑ ተዋዶና ተከባብሮ ብሎም ተጋብቶ እኩል የሚኖርባት ብቸኛ ምድር መሆኗ ነው, will be [131 206 6140 210 104 3906 36090 3266 1121 10517 592 6205 2004 204 9879 1]. We also converted the rest of the posts and comments into 'texts to sequences'. A neural network also requires posts, and comments must have the same size. Not all the posts and comments have the same length. In other words, naturally, some of the posts and comments are longer or shorter. We need to have the same size inputs; this is where the padding is necessary. First, we need to count all the words or the number of tokens in each of these input sequences. The minimum number of words is 2; the average number of words in a row is about 65, and the maximum number of words is more than 7322, as shown in figure 4.3.

```
print('Maximum number of words')
print(np.max(total_num_tokens))
```

```
Maximum number number of words
7322
```

```
print('Minimum number number of words')
print(np.min(total_num_tokens))
```

```
Minimum number number of words
2
```

```
print('Average number of words')
print(np.mean(total_num_tokens))
```

```
Average number of words
64.88839506172839
```

**Figure 4.3:** Number of words description

It will hurt our memory very much if we have all the short sentences with 7322 words, and there's a vast difference between the average and the maximum. Again we would be wasting a lot of memory if we just padded all the sentences in the dataset to all have 7322 tokens. We will compromise where we will pad all sequences and truncate the ones that are too long to have 853 words. The way we calculated this was like this, we took the average number of words in all the sequences in the dataset, and we added two standard deviations and shown in figure 4.4.

```
max_num_tokens=np.mean(total_num_tokens)+4*np.std(total_num_tokens)
max_num_tokens=int(max_num_tokens)
print('Maximum number of words')
print(max_num_tokens)
Maximum number of words
853
```

**Figure 4.4:** Maximum number of words

What do we get out of this is? We cover about 99% of the text in the dataset, so only about 1% are longer than 853 words and shown in figure 4.5.

```
np.sum (total_num_tokens < max_num_tokens) / len (total_num_tokens)
0.9913086419753087
```

**Figure 4.5:** Result of data covered by Maximum number of words

**Word Embedding** :- The word embedding layer takes inputs from the tokenization layer converts each token or integers of each post and comments into real value vectors. In this work, we used word2vec and Fasttext with (Cbow) and (Skip-gram) word embedding models to create feature vectors, and Each word in the text is taken and converted into a vector preserving the semantics relation between the words. We have used our unlabeled hate speech and offensive language dataset to create the word vectors.

The window size and word embedding dimension parameter affect word vectors' quality. The window size parameter specifies the distance between the predicted and the current word in a sentence. For a given term, the context window size determines the number of words before and after to be included as context words. According to [21], a value of window size 10 for skip-gram training algorithm and a value of 5 for a continuous bag of words algorithm have shown the best results. So we used 5 and 10 window size for a continuous bag of words and skip-gram, respectively. The embedding dimension specifies the dimension were feature vector for each term in posts and comments. According to [58], vector representation's performance enhances with increased vector size until the size reaches 300 dimensions. After 300 dimensions, the word vectors' excellent performance decreases again because too complicated representations can give rise to over-fitting without introducing more accuracy to word representations. This research used a fixed size of 300 for the vector dimension.

After the training process completed, each word with its corresponding features vector is saved as an output model used for our neural network model as a lookup table. After that, The embedding layer passes an input feature space with a shape of  $853 \times 300$  to a dropout layer, where 853 is the average length of posts and comments, and 300 is the embedding dimension.

**Convolutional layer:-** The dropout layer's output feeds into a 1D convolutional layer with an  $f$ -number of filters or feature maps and  $k$  kernel size or sliding window. Assuming a kernel size or sliding window and feature maps of the convolutional layer are  $3 \times 300$ . When the convolutional "kernel" slides over posts and comments, it sees three words (tri-gram features) from the embedding vector at a time. It computes an element-wise product of each word's weights, multiplied by the weights assigned to the convolutional filter until all the input matrix values are covered. As the filter moves on, which capture the syntactic and semantic features and stores the local information needed to classify the posts and comments class in the embedding matrix and judges whether each feature matches the relevant label or not. Finally, a feature map generated on which activation function is applied to remove non-linearity. We used the value padding the same as the original input length equals to output length. The convolutional layer passed  $853 \times 300$  convolved feature representation to a 1D max-pooling.

**Max pooling layer:-** The max-pooling layer used to minimize the dimension of the feature map produced by the convolutional layer by aggregating the information. So, the max-pooling applied to every posts and comment in a dataset. We used max-pooling to get the required feature of posts and comments by selecting the maximum value. In other words, the convolutional layer looks at the context and extracts the main features, and the pooling layer plays a role in selecting the most prominent features. Assuming the best model had a pool size of 2, a max-pooling layer passes  $426 \times 300$  features matrix to the GRU layer, and the output 1D max-pooling layer can be considered an extracted feature.

**GRU layer:-** The extracted features from 1D max-pooling layer then feed into the GRU layer. The GRU layer will use the sequence of those features to learn about the input. It will treat each of the 426 feature dimensions as time steps of the recurrence and output 64 hidden units per time step. The GRU layer passed 64 features matrix to a softmax layer.

**Softmax layer:-** Finally, a softmax layer takes the output GRU layer as input to predict probability distribution over all possible four classes. The softmax layer outputs, the value, which is the probability value, are generated for each class.

The second model (Figure 4.2) is very similar to the first model but uses machine learning classifiers instead of a softmax classifier. It is made by extracting the intermediate output of the last hidden layer of the GRU. The extracted Result considered as meaningful features which temporal GRUs has captured. Then we feed them into machine learning algorithms such as support vector machine and random forest machine learning classifier.

### 4.2.1 Hyperparamters

All learning algorithms, even simple ones, require hyper-parameters to be determined by the practitioner. Neural networks share many hyper-parameters like activation function, input word vector representation, learning rate, optimization, and dropout. In addition to all shared parameters for the neural network, each particular type of neural network still has more hyperparameters. To optimize each hyperparameter, one can see either reason about sensible values for these parameters based on the learning task or complete a full grid-search on the model. A grid-search is a step-by-step optimization of hyperparameters and is expensive for learning lessons with many input data. Since this thesis is only dealing with text data, we performed a full grid-search on the model. Our proposed model's structure has been modified for the following most frequently adjusted hyperparameters and network architecture parameters using A grid-search. Table 4.1 shows the different hyperparameters that tested in the grid-search optimization.

**optimizer:-** The optimizer is responsible for minimizing the objective role of the neural network. The widely chosen optimizer is stochastic gradient descent (SGD), which has proven to be an efficient and reliable method of optimization for a large number of published machine learning systems. However, the SGD can be very sensitive to the selection of the learning rate. Choosing too high a rate will cause the system to diverge in terms of objective operation, and choosing too low a rate may result in a slow learning process. SGD also has difficulty navigating the ravines

**Table 4.1:** Different values hyperparameters used during the grid search

| Hyperparameter      | Values ranges                                   |
|---------------------|---|
| Optimizer           | SGD,RMSprop,Adam,Adadelata,Adagrad,Adamax,Nadam |
| Learning rate       | 0.1,0.01,0.001,0.0001,0.00001                   |
| Kernel size         | 3,5,4,7,9                                       |
| Activation function | Relu,tanh,elu,selu                              |
| Pool size           | 2,4,6,8   |
| GRU units           | 8,16,32,64,128,256,512                          |
| Dropout             | 0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.9                 |
| Batch size          | 8,16,32,64,128,256,512                          |
| Number of epochs    | 10,20,30,40,50,60,70,80,90,100                  |

and saddle points. Other gradient-based optimization algorithms have suggested removing the shortcoming of SGD. Namely Adagrad [59], Adadelata [60], RMSProp [61], Adam [62], Nadam [63], an Adam version containing Nesterov momentum and Adamax [62], an alternative form of Adam.

**Learning rate:-** the Learning rate is one of the critical hyperparameters that maintains the speed at which the model parameters need to be updated to deduce a satisfactory output neural network. The process of obtaining optimum value can become an entangled effort in many practical problems. According to [35] The choice of a reasonable learning rate can turn out to be a difference between a model that is unable to learn from the data and a model that delivers outstanding results. On the other hand, a neural net with a small learning rate examines data slowly, requiring a significant amount of time to converge. On the other side, excessively high learning rates lead to weight dispersion and fluctuation around the minimum. The learning rate increases the training stage's potential to achieve a global minimum, eliminating the chance of being confined at a local minimum that could give rise to incorrect outputs. To distract both extreme positions, the range of learning rate values usually tested is used to observe the impact of learning on cost function by using a validation dataset. The learning rate, which results in a lower loss of the cross-validated set maintained for the experiment's present state.

**Activation function:-** Activation is a function that gives non-linearity to the information going through the neural network. Networks will compose data abstractions via these functions [35].

The activation functions evaluated for this work are the Rectext Linear Unit (ReLU), Sigmoid, Hyperbolic Tangent (Tanh) and Regular.

**Kernel size:-** Filters size are parts of CNN and defined as n-grams, i.e. n-line filters that use data convolution operations, and the size of the filters is related to the number of neural net weights [35]. The larger the filter, the higher the number of values to be trained.

**Pooling:-** Pooling is an operation used to simplify the information given at the output of a function map. The knowledge can be transformed by an average of values the map or by the maximum amount of the graph [35].

**Dropout rate:-** dropout is a basic co-adaptation technique used during training to prevent neural networks from overfitting by co-adaptation [64]. This technique compared to a random sampling of function weights in a network, i.e. an unexpected decrease in some of the network's weight connections to prevent over-dependence on a single feature.

**Batch size:-** The batch size is a hyperparameter that describes the number of units to work through before renewing the internal model parameters that means a batch as a for-loop was iterating over one or more pieces and making predictions at the end of the batch, the forecasts compared to the expected output variables. An error calculated, and from this error, the update algorithm used to improve the model <sup>1</sup>.

**Epochs:-** The number of epochs is a hyperparameter that determines the number of times the learning algorithm can operate through the entire training dataset. One epoch means each sample in the training dataset has had the chance to update the model's internal parameters. The epoch consists of one or more batches; we can think of a for-loop over the number of iterations where each loop continues over the training dataset. In this for-loop, there is another nested for-loop that iterates across each batch of variables, where one batch has the number of observations defined in the batch size <sup>2</sup>.

---

<sup>1</sup>Brownlee J. (2019, October 25). Dissimilarity Between a Batch and an Epoch in a Neural Network. Accessed June 06, 2020, from <https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/>

<sup>2</sup>Brownlee, J. (2019, October 25). Dissimilarity Between a Batch and an Epoch in a Neural Network. Accessed June 06, 2020, from <https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/>

### 4.3 language and libraries

Python is a general-purpose programming language commonly used in this study. It defined as an object-oriented, structured, high-level programming language that used too many problems and situations. Python has a standard library for processing areas such as string processing, Internet protocols, and interface operating systems. However, Python also supports a wide variety of third-party extensions<sup>3</sup>. The main Python modules or plugins used in this study include Scrapy, IPython, Jupyter Notebook, Pandas, Matplotlib, Scikit-learn, Tensorflow, Keras and Gensim. Such libraries and what they used for briefly discussed below.

**IPython and Jupyter Notebook:-** IPython is an interactive python shell with a Jupyter Notebook kernel that enables interactive code development in the browser<sup>4</sup>. It is commonly used in data science and data processing, mathematical modelling, machine learning, numerical simulation and visualization. Throughout this study, Jupyter Notebook used for data processing, machine learning, analysis and visualization .

**Scikit-learn:-** Scikit-learn is a Python open-source machine learning, data mining, and data analysis library<sup>5</sup>. The library includes, among other items, a logistic regression model used as a basis for this study.

**panads:-** Pandas, is an open-source library for manipulation of data and analysis<sup>6</sup>. Convenient handling of data structures and high performance. Pandas have used throughout this study for the processing and processing of data and data analysis.

**Matplotlib:-** Matplotlib is a Python module devoted to plotting figures, graphs and diagrams<sup>7</sup>. It used for visualization in combination with Jupyter Notebook.

<sup>3</sup>”Welcome to Python.org,” Python.org, accessed June 3, 2020, <https://www.python.org/>.

<sup>4</sup>”Jupyter and the Future of IPython¶,” IPython, accessed June 3, 2020, <https://ipython.org/>.

<sup>5</sup>”Learn” scikit, accessed June 3, 2020, <https://scikit-learn.org/stable/>

<sup>6</sup>”Pandas,” pandas, accessed June 3, 2020, <https://pandas.pydata.org/>

<sup>7</sup>”(Tutorial) MATPLOTLIB Tutorial: PYTHON Plotting” DataCamp Community accessed June 3, 2020 <https://www.datacamp.com/community/tutorials/matplotlib-tutorial-python>

**Gensim:-** Gensim is a library for the unsupervised semantic modelling of plain text. The library specializes in processing raw, unstructured digital text efficiently <sup>8</sup>. Building and training a Word2vec model based on the text of the posts and comments achieved through the Gensim Library.

**TensorFlow:-** TensorFlow is an open-source library for artificial intelligence and numerical computing which was created by Google <sup>9</sup>. The library is particularly well suited for deep learning, where the scalable architecture enables computation to apply to one or more CPUs or GPUs. Tensorflow has been used as a backend program to facilitate deep learning in this study.

**Keras:-** Keras is a high-level API that can operate with TensorFlow and other deep learning libraries<sup>10</sup>. The main focus is to facilitate efficient prototyping and experimentation. The Keras API used to build the hybrid CNN-GRU model with several architectures analyzed in this study.

**Experimental setup:-** Experiments conducted on a personal computer equipped with Intel® Core™ i5-6200U CPU @ 2.30GHz × 4 with two physical and four logical cores, 8 GB RAM with 1 TB hard disk. The operating system is Ubuntu 18.04, 64 bit.

## 4.4 Evaluation metric

There are many supervised learning algorithms methods to train and measure the accuracy of the model. The data set is split into three groups: training, validation and testing. The first is the set of data that the model trains and learns, while the second is used to provide an unbiased assessment of the fit model on the training data set while tuning the model hyper-parameters. After training, the test set is used to validate the model with data that has not been before. In some instances, the cost of setting aside a large portion of the data set as required by the holdout approach is too high. In these cases, a re-sampling based technique used, such as cross-validation as a solution instead. The k-fold validation method is used for our project [65]. The

---

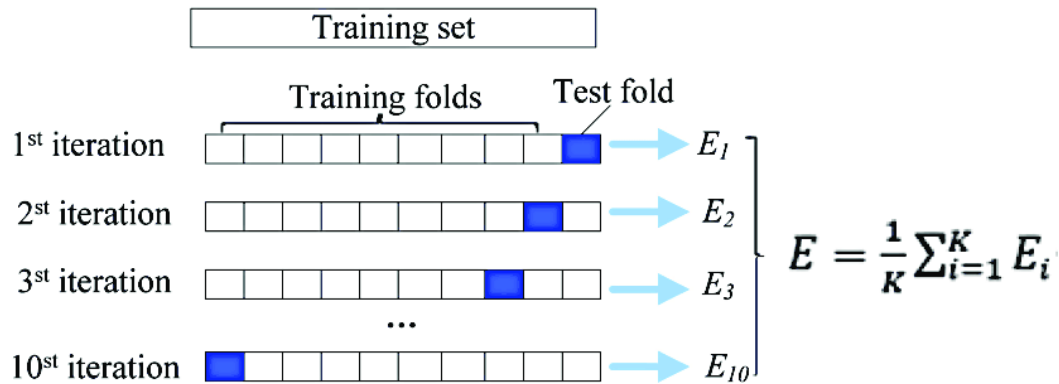
<sup>8</sup>”Gensim: Topic Modelling for Humans,” Radim AehÅ Aek: Machine learning consulting, accessed June 3, 2020, <https://radimrehurek.com/gensim/>.

<sup>9</sup>”TensorFlow White Papers” TensorFlow, accessed June 3, 2020, <https://www.tensorflow.org/about/bib>.

<sup>10</sup>Keras Team, ”Simple. Flexible. Powerful.,” Keras, accessed June 3, 2020, <https://keras.io/>.



complete training set is split into  $k$  folds (subsets) in this approach. A combination of  $k - 1$  folds used to train the model over  $k$  iterations, while the remaining fold used for validation purposes. An average score used to measure the generalization error. Figure 4.6 describes the cross-validation technique used in this study.



**Figure 4.6:** A  $k$ -fold cross-validation approach [61]

To perform the project model measurements,  $k = 10$  and four standard performance metrics are determined: Accuracy, Precision, Recall, F1 score and confusion matrix. These four measures are used in deep learning, artificial intelligence, sentiment analysis, and hate speech detection. Before discussing the criteria, it is crucial to introduce certain concepts that will use later. These concepts are part of the vocabulary used to define a classification [66].

**True positive:-** When evaluating the results of classification, an actual positive rate measures the number of instances that have classified in a category and belong to that category.

**True negative:-** The real negative rate describes the number of cases that have not organized in a class, and that efficiently does not belong to that class.

**False positive:-** The false-positive rate indicates the number of instances that have been classified in a category but do not belong to that class.

**False negative:-** The false-negative rate indicates the number of instances that have not classified in the category but that do belong to that class.

**Accuracy:-** Accuracy is the simplest metric used in the classification process. It measures the number of correctly identified instances [66]. Although intuitive, accuracy is not enough to

determine the quality of the classifier.

$$Accuracy = \frac{TruePositives + TrueNegatives}{TruePositives + FalsePositives + FalseNegatives + TrueNegatives} \quad (4.1)$$

**Precision:-** Precision is a metric used to calculate the Precision of positive predictions. It is defined by the number of true positive, divided by the number of true positive plus the number of false-positive [66]. Precision is an important determinant when creating a model in which the FP performance is very significant.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (4.2)$$

**Recall:-** Recall is a metric used to measure the ratio of positive instances to the correct classification. It represented by the number of true positive, divided by the number of true positive plus the number of false negatives [66]. The Recall is an important metric when creating a model where the FN rate is very significant.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (4.3)$$

**F1 Score:-** There is a way to combine accuracy and Recall with one metric The metric is called the F1 Score. The F1 Score is the harmonic means of Precision and Recall. The harmonic mean differs from the traditional standard because it gives greater weight to the low results, and therefore, the F1 Score will only be high if the Recall and accuracy are both high [66].

$$F_1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4.4)$$

**Confusion Matrix:-** The confusion matrix is a method used for evaluating the performance of a classification algorithm. The number of correct and incorrect forecast are given by count values and decomposed by a group. It gives intuition not only into the errors produced by the classifier but also, most importantly, into the types of errors which have occurred [67].

## CHAPTER 5

### Experimental Result, Discussion and Evaluation

This chapter examines the results of our models described in the previous chapters. Section 5.1 presents the optimization of the parameters used in this study. In this research, we are conducting two experiment scenarios. In section, 5.2 the result of the first experiment scenario is presented. The first experiment scenario aims to show the effect of word embedding and classification layers to identify hate speech, including the target, offensive language, and not hate speech. These experiments answer RQ1 and RQ2:

RQ1: How the deep learning models separate offensive language and hate speech, including their targets from normal language?

RQ2: Which features extraction methods and classification layers are better for Amharic text offensive language and hate speech detection including, the target of hate speech?

Section 5.3 shows the result of the second experiment scenario. The experiment aimed to show our proposed model for offensive and hate speech detection without including the target of hate speech and binary classification.

## 5.1 Hyperparameter Tuning

The hyperparameters explained in Chapter 4 have been optimized using a grid search. The best parameters after the grid search shown in Appendix A. For each step of the grid search, all the hyperparameters fixed apart from the one that is optimized. The best hyperparameters summarized in table 5.1 below.

**Table 5.1:** Over view of best hyper-parameters

| <b>Hyper-parameters</b> | <b>values</b> |
|-------------------------|---------------|
| optimizer               | Adamax        |
| Learning rate           | 0.01          |
| Kernel size             | 3             |
| Activation function     | relu          |
| Pool size               | 2             |
| GRU units               | 64            |
| Droupout rate           | 0.3           |
| Batch size              | 16            |
| Number of epochs        | 10            |

## 5.2 First Scenario Experiment Result

In this research, the first experiment scenario did to know the best word embedding models, and classification models to identify hate speech, including the target, offensive language and not hate speech.

### 5.2.1 Effects of word embedding

Our proposed models' classification performance is CNN-GRU-SoftMax, CNN-GRU-SVM, and CNN-GRU-RF examined using word2vec and Fasttext word representations models with a continuous bag of words and skip-gram model architecture. In total, four different models were trained, namely word2vec-Skip-Gram (word2vecSkipGram), word2vec-Continous Bag of

Words (word2vecCBOW), FastText-Continuous Bag of Words (FastTextCBOW) and FastText-Skip-Gram ( FastTextSkipGram), respectively and the result shown in table 5.2.

**Table 5.2:** Classification performances using different embedding models

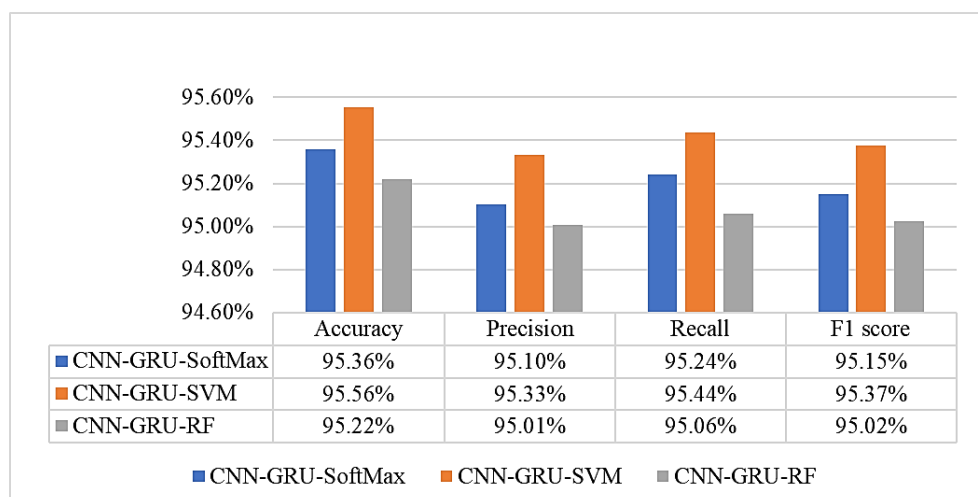
| Models          | Word embedding       | Accuracy | Precision | Recall | F1 score |
|-----------------|----------------------|----------|-----------|--------|----------|
| CNN-GRU-SoftMax | Word2vec (Cbow)      | 87.70%   | 87.69%    | 86.97% | 87.25%   |
|                 | Word2vec (Skip-gram) | 90.08%   | 90.01%    | 89.35% | 89.63%   |
|                 | Fasttext (Cbow)      | 93.30%   | 93.09%    | 93.02% | 93.01%   |
|                 | Fasttext (Skip-gram) | 95.36%   | 95.10%    | 95.24% | 95.15%   |
| CNN-GRU-SVM     | Word2vec (Cbow)      | 87.92%   | 87.58%    | 87.30% | 87.41%   |
|                 | Word2vec (Skip-gram) | 90.39%   | 90.13%    | 89.79% | 89.94%   |
|                 | Fasttext (Cbow)      | 93.64%   | 93.63%    | 93.24% | 93.41%   |
|                 | Fasttext (Skip-gram) | 95.56%   | 95.33%    | 95.44% | 95.37%   |
| CNN-GRU-RF      | Word2vec (Cbow)      | 86.89%   | 86.56%    | 86.06% | 86.27%   |
|                 | Word2vec (Skip-gram) | 89.76%   | 89.37%    | 89.22% | 89.28%   |
|                 | Fasttext (Cbow)      | 93.12%   | 92.94%    | 92.82% | 92.86%   |
|                 | Fasttext (Skip-gram) | 95.22%   | 95.01%    | 95.06% | 95.02%   |

Table 5.2 show the test performance of comparisons of Word2vec ( CBOW ), Word2vec (Skip-gram ), FastText (CBOW) and FastText (Skip-gram ). According to the results, FastText (Skip-gram and CBOW) based methods achieved superior performance Word2vec ( CBOW ) and Word2vec (Skip-gram ) for all three models CNN-GRU-SoftMax, CNN-GRU-SVM and CNN-GRU-RF models. This mainly because FastText considers subword information when generating word vectors. On the other hand, the FastText Skip-gram model achieved superior performance with an accuracy of 95.36%, the precision of 95.10%, recall of 95.24% and F1 score of 95.15%, which outperformed FastText ( CBOW) with up to + 2.06% accuracy, + 2.01% precision, 2.22% recall and + 2.14% F1 improvements on CNN-GRU-softmax respectively. it also achieved superior performance with an accuracy of 95.56%, the precision of 95.33%, recall of 95.44% and F1 score of 95.37%, which outperformed FastText ( CBOW) with up to + 1.92% accuracy, + 1.7% precision, 12.2% recall and + 1.96% F1 improvements on CNN-GRU-SVM and also achieved superior performance with an accuracy of 95.22%, the precision of 95.01%, recall of 95.06% and F1 score of 95.02%, which outperformed FastText ( CBOW) with up to + 2.1% accuracy, + 2.07% precision, 2.24% recall and + 2.16% F1 improvements on CNN-GRU-Rf respectively. The results are consistent with [24], FastText with skipgram model can produce

high-quality vector representations utilizing semantic and syntactical data from texts. The Skip-gram model works by predicting the given context from one term, whereas the CBOW model uses the context word to predict the target word; additionally, it can cover out of vocabulary words.

## 5.2.2 Model comparison

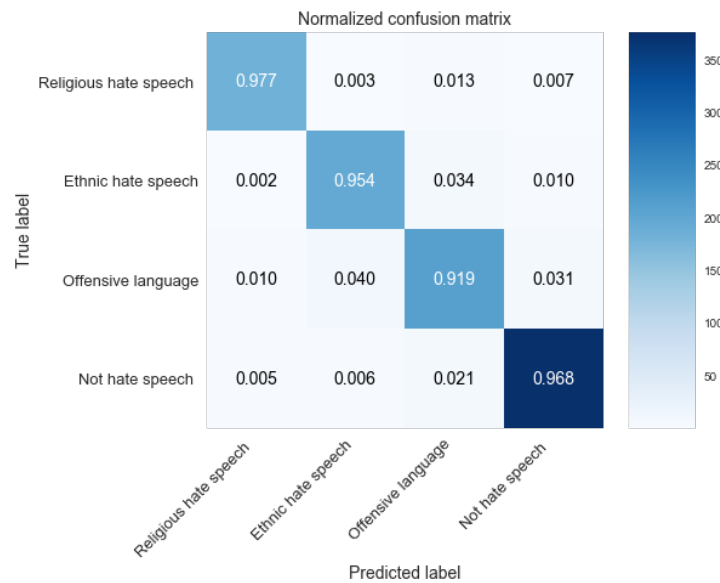
After identifying the best word embedding, we compared the following three models to demonstrate their performance. The first model is CNN-GRU-SoftMax; The feature maps learned by CNN and GRU passed to the Softmax classifier. The second model is CNN-GRU-SVM, and the third model is CNN-GRU-RF. The second and the third models combine deep learning with machine learning by extracting the intermediate output from the end hidden layer of the GRU. The extracted result of the GRU layer passed to support vector machine, and random forest machine learning classifiers to classify posts and comments into hate speech including the target of hate speech, offensive language and not hate speech and the result shown in figure 5.1.



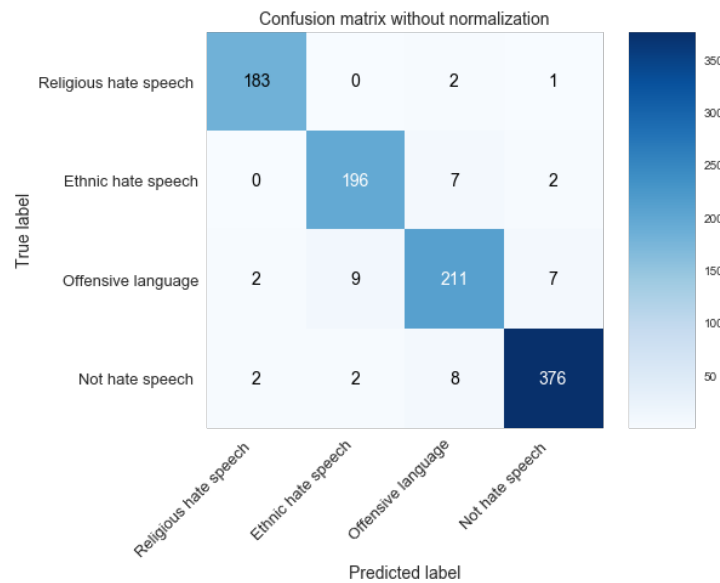
**Figure 5.1:** Comparison of classifier performance

Figure 5.1 shows the comparison of CNN-GRU-softmax, CNN-GRU-SVM and CNN-GRU-Rf models with the best word embedding that is Fasttext (skip-gram). CNN-GRU-SVM model achieved superior performance with an accuracy of 95.56%, the precision of 95.33%, recall of 95.44% and F1 score of 95.37%, which outperformed CNN-GRU-softmax with up to + 0.2%

accuracy, + 0.23% precision, 0.2% recall and + 0.22% F1 improvements respectively. It also outperformed CNN-GRU-softmax with up to + 0.34% accuracy, + 0.32% precision, 0.38% recall and + 0.35% F1 improvements respectively. This result clearly shows that The model combines the advantage of CNN-GRU and SVM classifiers in the classification of posts and comments into hate speech, including their targets, offensive language and not hate speech.



(a) Normalized confusion matrix



(b) confusion matrix, without normalization

Figure 5.2: confusion matrix for best model

Figure 5.2 shows our best model’s confusion matrix, Fasttext (Skip-gram)-CNN-GRU-SVM with normalization and without normalization. From figure 5.2 (a) and 5.2 (b), we can analyze

that 97.7% of the religious hate speech class were correctly classified and 2.3% wrongly classified as other classes. With 186 posts and comments out of 183 posts and comments predicted correctly and misclassified two posts and comments as offensive language and one posts and comments as not hate speech. For ethnic hate speech class, our model correctly classified 95.4% as ethnic hate speech and misclassified 4.6% to other classes. That means 196 types correctly classified as ethnic hate speech class and misclassified seven comment and posts as religious hate speech and two posts and comments to note hate speech class. Our model correctly classified 91.9% as offensive language and 8.1% as other classes for an offensive language class. That means out of 229 offensive posts and comments, it classified correctly 211 as offensive language class and misclassified two comments and posts as religious hate speech, nine comments and posts as ethnic hate speech and seven posts and comments as note hate speech class. For the type of not hate speech, the model predicted 96.8% correctly, and 3.2 % wrongly predicted to other classes. That means 388 posts and comments out of 376 predicted correctly and miss classified two posts and comments as religious hate speech class, two posts and comments as ethnic hate speech class, eight posts and comments as offensive language.

The results presented in Table 5.2 and figure 5.1 answer RQ1: How the deep learning models separate offensive language and hate speech, including their targets from normal language? The architecture and experiments presented in Chapter 4 and Chapter 5, respectively, were designed to answer this research question. The results presented in Table 5.2 and figure 5.1 also answer RQ2: Which features extraction methods and classification layers are better for Amharic text offensive language and hate speech detection including, the target of hate speech? Fasttext with the skip-gram model is better than other word embedding models for feature extraction, and SVM classification performs better than softmax and Rf classifier to generate the final output.

### **5.3 Second Scenario Experiment Result**

This experiment aimed to show that our proposed model is not only helpful in identifying hate speech with their targets target, offensive language, and not hate speech. We ran the three



models with Fasttext (Skip-gram) for vector generation that was effective in the first experiment scenario on two other datasets. A description of the dataset presented in table 5.3 and 5.4. The first dataset has three class hate speech, offensive language and not hate speech, by merging the religious and ethnic hate speech class into hate speech classes, as shown in table 5.3. The second dataset has two categories hate speech and not hate speech by merging all religious and ethnic hate speech class into hate speech classes and by removing offensive language, as shown in table 5.4.

**Table 5.3:** Dataset distribution for three classes

| Class              | Number of posts and comments |
|--------------------|------------------------------|
| Hate speech        | 3977                         |
| Offensive language | 2300                         |
| Not hate speech    | 3888                         |

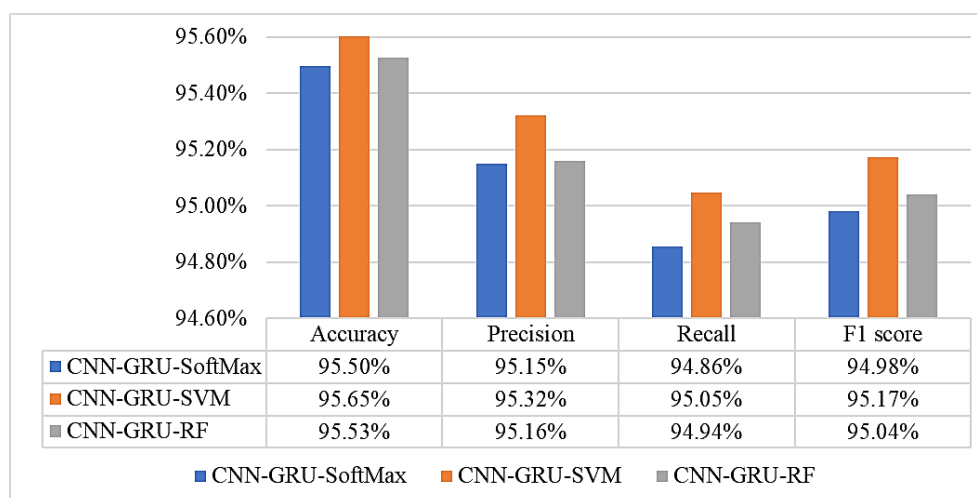
**Table 5.4:** Dataset distribution for two classes

| Class           | Number of posts and comments |
|-----------------|------------------------------|
| Hate speech     | 3977                         |
| Not hate speech | 3888                         |

### **Experiment 1: Detection of hate speech, offensive language and not hate speech**

We applied Fasttext with skip-gram word embedding and the same valuable hyperparameter in the first experiment scenario in this experiment. For this experiment, we have three models: CNN-GRU-SoftMax, CNN-GRU-SVM and CNN-GRU-RF, to classify given comments into hate speech, offensive language and not hate speech and the result presented in figure 5.3.

From figure 5.3, We can see the proposed CNN-GRU-SVM model was attaining higher performance across the entire set of evaluation metrics with an accuracy of 95.65%, the precision of 95.32%, recall. 95.05%, and F-score of 95.17% to classify posts and comments into hate speech, offensive language and not hate speech. The proposed CNN-GRU-SVM model outperformed CNN-GRU-softmax with up to + 0.15% accuracy,+ 0.17% precision,+ 0.16% recall,+ 0.19% F1 score improvements. it also improve the performance of CNN-GRU-Rf with up to + 0.12% accuracy,+ 0.16% precision,+ 0.11% recall and 0.13% F1 score.

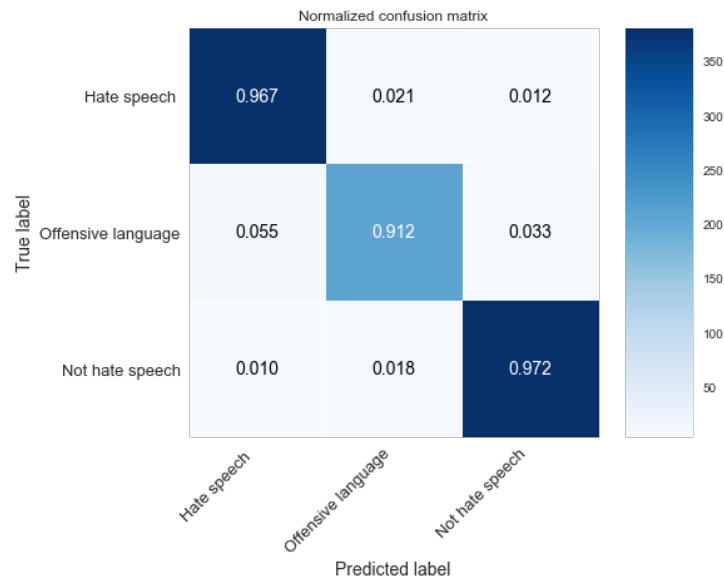


**Figure 5.3:** Comparison of classifier performance

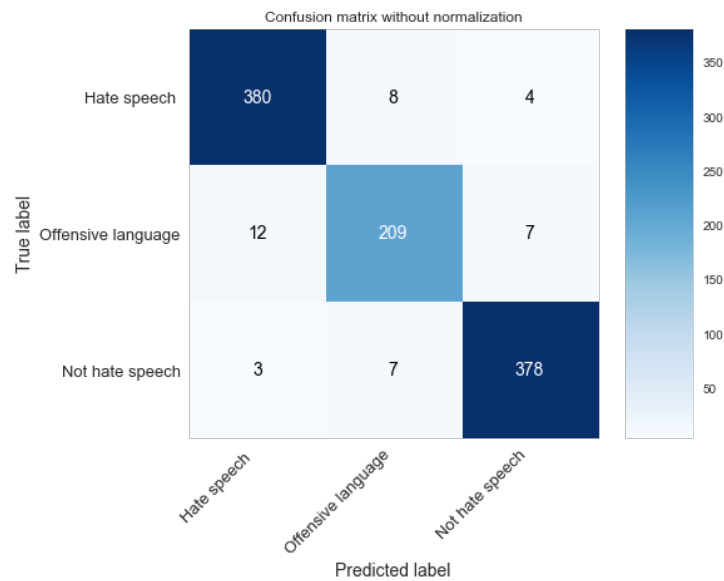
Figure 5.4 shows the confusion matrix of the best model that is Fasttext (Skip-gram)-CNN-GRU-SVM, and From figure 5.4 (a) and 5.4 (b), we can analyze that 96.7% the hate speech class were correctly classified and 3.3% wrongly classified as other classes. With 380 posts and comments out of 392 posts and comments predicted correctly and misclassified eight posts and comments as offensive language and four posts and comments as not hate speech. For offensive language class, our model correctly classified 91.2% as offensive language and misclassified 5.5% to hate speech class and 3.3% as not hate speech. That means 209 levels were correctly classified as an offensive language class, misclassified twelve comments and posts as hate speech, and seven comments and posts as not hate speech. For not hate speech class, our model correctly classified 97.2% as offensive language and 1% as hate speech and 1.8% as an offensive class. That means out of 388 not hate speech posts, and comments classified correctly 378 as not hate speech class and misclassified three comments and posts as hate speech and seven comments and posts as an offensive language class.

### Experiment 2: Detection of hate speech and not hate speech

In this experiment, we classify posts and comments into hate speech and not hate speech. We used the same word embedding and hyperparameter effective in the first experiment scenario. We have three different models for this experiment, such as CNN-GRU-softmax, CNN-GRU-SVM, and CNN-GRU-Rf, and the performance result shown in figure 5.5. From figure 5.5, We can see the proposed CNN-GRU-SVM model was attaining higher performance across the



(a) Normalized confusion matrix

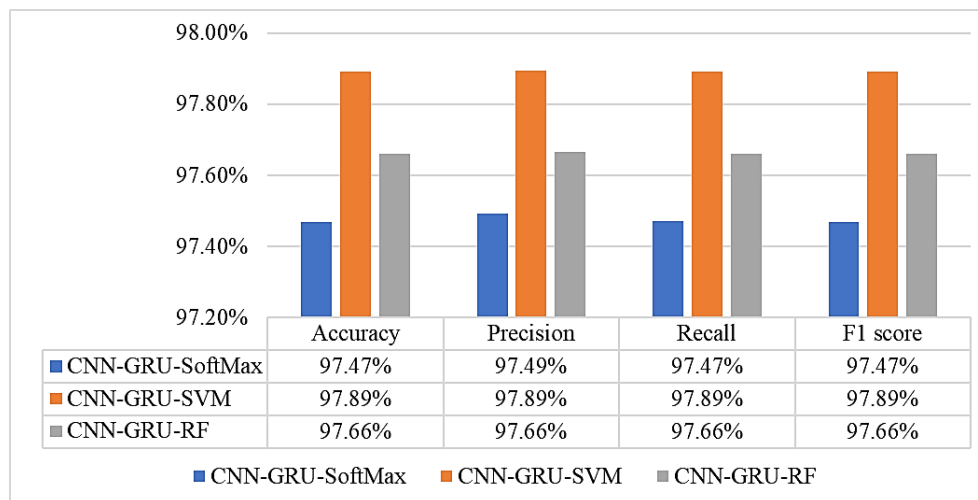


(b) confusion matrix, without normalization

Figure 5.4: confusion matrix for best model

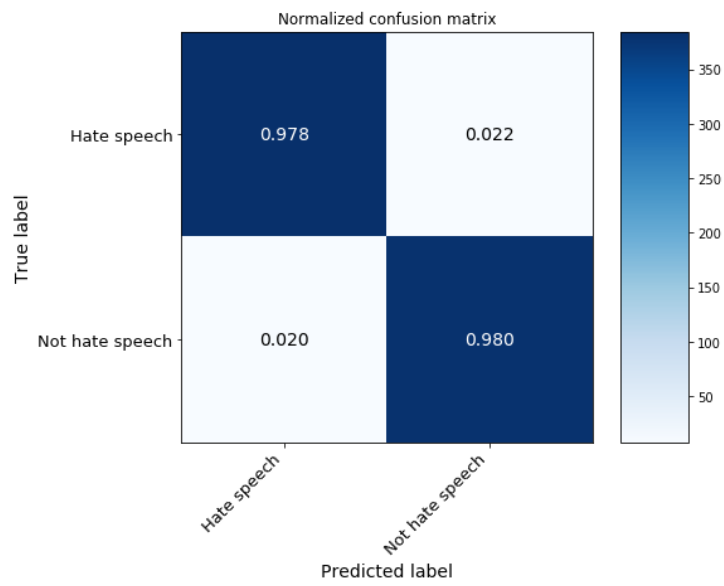
entire set of evaluation metrics with an accuracy 97.89%, precision 97.89%, recall 97.89%, and F-score 97.89% to classify posts and comments into hate speech, not hate speech. The proposed CNN-GRU-SVM model outperformed CNN-GRU-softmax with up to + 0.42% accuracy,+ 0.4% precision,+ 0.42% recall,+0.42% F1 score improvements. it also improves the performance of CNN-GRU-Rf with up to + 0.23% accuracy,+ 0.23% precision,+ 0.23% recall and 0.23% F1 score.

The best model’s confusion matrix is Fasttext with skip-gram-CNN-GRU-SVM, and the result

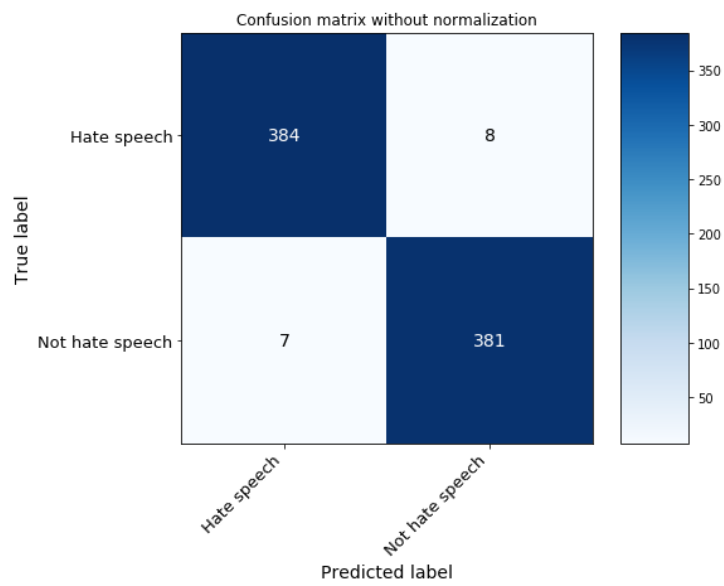


**Figure 5.5:** Comparison of classifier performance

shown in figure 5.6. From figure 5.6 (a) and 5.6(b), we can analyze that 97.8% of the hate speech class were correctly classified, and 2.2% are wrongly classified not to hate speech classes. With 384 posts and comments out of 392 posts and comments predicted correctly and misclassified eight posts and comments as hate speech class. For not hate speech class, the proposed model correctly classified 98% as not hate speech and miss-classified 2% to hate speech class. That means 381 levels correctly classified as not hate speech class and misclassified seven comment and posts as hate speech.



(a) Normalized confusion matrix



(b) confusion matrix without normalization

Figure 5.6: confusion matrix for best model

# CHAPTER 6

## Conclusion and Recommendation

This chapter will provide a conclusion to work carried out in this study. It also contains future works to show further researches that can be done in the future.

### 6.1 Conclusion

This research discussed hate speech and offensive language detection for the Amharic language. We used different approaches to execute the study successfully. It was essential to understand and define hate and offensive language on social media, explore existing techniques used to tackle the problem and understand the Amharic language, as discussed in chapter two. The different methods followed to implement and design models that can detect hate speech and offensive language. These methods include collecting post and comment from Facebook and YouTube to build the dataset, develop annotation guidelines, and preprocessing. The features extracted using word2vec (Cbow), word2vec(Skip-gram), Fasttext (Cbow) and Fasttext (Skip-gram) and CNN-GRU-SoftMax, CNN-GRU-SVM, and CNN-GRU-RF models for classification. Finally, performance models compared based on accuracy, precision, recall, F1 score and confusion matrix.

This study manually annotated the posts and comments into religious hate speech, ethnic hate speech, offensive language, and not hate speech. After building the dataset, we did two experiment scenarios. The first experiment scenarios were to identify hate speech, including hate speech (religious hate speech, ethnic hate speech), offensive language and not hate speech. Our experiment results show Fasttext (Skip-gram) word embedding for feature extraction and CNN-GRU-SVM model for classification reach an excellent performance. The proposed model achieves an accuracy of 95.56%, 95.33% precision, 95.44% recall and 95.53% F1 score to classify posts and comments into religious hate speech, ethnic hate speech, offensive language and not hate speech for Amharic text.

The second experiment scenarios were to show our approach not only useful for identifying religious hate speech, ethnic hate speech, offensive language and not hate speech. We ran the same parameters, Fasttext (Skip-gram) for feature extraction that was effective in the first experiment scenario and the three proposed models on two other datasets. The first dataset has three classes: hate speech, offensive language and not hate speech; by merging religious and ethnic hate speech into hate speech classes. The second one has two categories hate speech and not hate speech, by combining and religious and ethnic hate speech into hate speech classes and removing offensive language. The proposed model CNN-GRU-SVM with Fasttext( skip-gram) achieves 95.65% accuracy, 95.32% precision. 95.05% recall, and 95.17% F1 score for three-class classification(hate speech, offensive language and not hate speech) and also achieve 97.89% accuracy, 97.89% precision: 97.89% recall and 97.89% F1 score on two-class classification.

Lastly, to the best of our knowledge, the classification of posts and comments into religious hate speech, ethnic hate speech, offensive language, and not hate speech is the first experiment for the Amharic language.

## **6.2 Recommendation**

This thesis researched hate speech and offensive language detection for Amharic language using deep learning neural networks with word embedding techniques. It is better to see the difference in performance results using semisupervised, unsupervised transfer learning. Another perspective at future work is to apply other word embedding techniques such as Elmo word representation to initialize word vectors. There are several different methods for generating vector representation for one sentence or paragraph that can be experimented with.

We used a manual annotation method to annotate the dataset with three persons annotating the dataset. We recommend more people for annotation to improve the dataset quality and to avoid biased. We optimized the hyperparameter of our proposed model using a grid search. For each step of the grid search, all the hyperparameters are fixed apart from the optimized one. The grid search is expensive to optimize all parameter values. Hence, we recommend a future full hyperparameter search to avoid the interdependence of hyperparameters. The maximum performance of the model is achieved based on our dataset only.

In Ethiopia, Hate speech expressed on social media in more than one language. This study limited its scope to only hate speech defined in Amharic. It is necessary for a more comprehensive detection tool to build on the dataset by including posts and comments expressed by Amharic text written in Latin letters and other languages. Another perspective at future work is detecting other forms of hate speech and offensive content on social media such as images, memes, video and audio.



## References

- [1] Zewdie Mossie and Jenq-Haur Wang. Social network hate speech detection for amharic language. *Computer Science & Information Technology*, pages 41–55, 2018.
- [2] Zewdie Mossie and Jenq-Haur Wang. Vulnerable community identification using hate speech detection on social media. *Information Processing & Management*, 57(3):102087, 2020.
- [3] Worku Kelemework. Automatic amharic text news classification: Aneural networks approach. *Ethiopian Journal of Science and Technology*, 6(2):127–137, 2013.
- [4] Surafel Getachew Tesfaye and Kula Kakeba. Automated amharic hate speech posts and comments detection model using recurrent neural network. 2020.
- [5] Kenenisa Yonas. Hate speech detection for amharic language on social media using machine learning techniques, 2019.
- [6] Komisi Nasional Hak Asasi Manusia. Buku saku penanganan ujaran kebencian (hate speech). *Jakarta: KOMNAS HAM*, 2015.
- [7] Gregory H Stanton. The rwandan genocide: Why early warning failed. *Journal of African Conflicts and Peace Studies*, 1(2):3, 2009.
- [8] Binny Mathew, Ritam Dutt, Pawan Goyal, and Animesh Mukherjee. Spread of hate speech in online social media. In *Proceedings of the 10th ACM conference on web science*, pages 173–182, 2019.

- [9] Thomas Davidson, Dana Warmley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *Eleventh international aaai conference on web and social media*, 2017.
- [10] Heather Hensman Kettrey and Whitney Nicole Laster. Staking territory in the “world white web” an exploration of the roles of overt and color-blind racism in maintaining racial boundaries on a popular web site. *Social Currents*, 1(3):257–274, 2014.
- [11] Anuradha Goswami and Ajey Kumar. A survey of event detection techniques in online social networks. *Social Network Analysis and Mining*, 6(1):107, 2016.
- [12] Michael Wiegand, Josef Ruppenhofer, Anna Schmidt, and Clayton Greenberg. Inducing a lexicon of abusive words—a feature-based approach. 2018.
- [13] Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4):215–230, 2015.
- [14] K Soumya George and Shibily Joseph. Text classification by augmenting bag of words (bow) representation with co-occurrence feature. *IOSR J. Comput. Eng*, 16(1):34–38, 2014.
- [15] Zeerak Waseem and Dirk Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93, 2016.
- [16] Chih-Fong Tsai. Bag-of-words representation in image annotation: A review. *International Scholarly Research Notices*, 2012, 2012.
- [17] Pete Burnap and Matthew L Williams. Us and them: identifying cyber hate on twitter across multiple protected characteristics. *EPJ Data science*, 5(1):11, 2016.
- [18] Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In *Proceedings of*

- the 21st ACM international conference on Information and knowledge management*, pages 1980–1984, 2012.
- [19] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.
- [20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
- [21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [22] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.
- [23] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.
- [24] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.
- [25] Pete Burnap and Matthew L Williams. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242, 2015.
- [26] Sindhu Abro, Sarang Shaikh, Zahid Hussain Khand, Zafar Ali, Sajid Khan, and Ghulam Mujtaba. Automatic hate speech detection using machine learning: A comparative study. *International Journal of Advanced Computer Science and Applications*, 11(8), 2020. doi: 10.14569/IJACSA.2020.0110861. URL <http://dx.doi.org/10.14569/IJACSA.2020.0110861>.
- [27] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

- [28] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.
- [29] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [30] Ting Hua, Feng Chen, Liang Zhao, Chang-Tien Lu, and Naren Ramakrishnan. Sted: semi-supervised targeted-interest event detection in twitter. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1466–1469, 2013.
- [31] Vinayak Athavale, Shreenivas Bharadwaj, Monik Pamecha, Ameya Prabhu, and Manish Shrivastava. Towards deep learning in hindi ner: An approach to tackle the labelled data scarcity. *arXiv preprint arXiv:1610.09756*, 2016.
- [32] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
- [33] Ziqi Zhang, David Robinson, and Jonathan Tepper. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *European semantic web conference*, pages 745–760. Springer, 2018.
- [34] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.
- [35] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.
- [36] Denny Britz. Recurrent neural network tutorial, part 4 implementing a gru/lstm rnn with python and theano. URL <http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano>, 2015.
- [37] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using

- rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [38] Michael Nguyen. Illustrated guide to lstm's and gru's: A step by step explanation. *Towards data*, 2018.
- [39] Björn Gambäck and Utpal Kumar Sikdar. Named entity recognition for amharic using deep learning. In *2017 IST-Africa Week Conference (IST-Africa)*, pages 1–8. IEEE, 2017.
- [40] Baye Yimam. ባህሪ ባህሪ ባህሪ. *Modern Amharic Grammar*, rev. 2nd edn. Addis Ababa: Elleni Printers, 2008.
- [41] Yeshiwas Getachew Msc Abebe Alemu. Deep learning approach for amharic sentiment analysis university of gondar.
- [42] Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153, 2016.
- [43] Björn Gambäck and Utpal Kumar Sikdar. Using convolutional neural networks to classify hate-speech. In *Proceedings of the first workshop on abusive language online*, pages 85–90, 2017.
- [44] Zeerak Waseem. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In *Proceedings of the first workshop on NLP and computational social science*, pages 138–142, 2016.
- [45] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760, 2017.
- [46] Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*, 2018.

- [47] Ji Ho Park and Pascale Fung. One-step and two-step classification for abusive language detection on twitter. *arXiv preprint arXiv:1706.01206*, 2017.
- [48] Johannes Skjeggstad Meyer and Björn Gambäck. A platform agnostic dual-strand hate speech detector. In *ACL 2019 The Third Workshop on Abusive Language Online Proceedings of the Workshop*. Association for Computational Linguistics, 2019.
- [49] Antigoni Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. A unified deep learning architecture for abuse detection. In *Proceedings of the 10th ACM Conference on Web Science*, pages 105–114, 2019.
- [50] Rohan Kshirsagar, Tyus Cukuvac, Kathleen McKeown, and Susan McGregor. Predictive embeddings for hate speech detection on twitter. *arXiv preprint arXiv:1809.10644*, 2018.
- [51] Shervin Malmasi and Marcos Zampieri. Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30(2):187–202, 2018.
- [52] Hamdy Mubarak, Kareem Darwish, and Walid Magdy. Abusive language detection on arabic social media. In *Proceedings of the first workshop on abusive language online*, pages 52–56, 2017.
- [53] Paula Cristina Teixeira Fortuna. Automatic detection of hate speech in text: an overview of the topic and dataset annotation with hierarchical classes. 2017.
- [54] Mary L McHugh. Interrater reliability: the kappa statistic. *Biochemia medica: Biochemia medica*, 22(3):276–282, 2012.
- [55] Mathieu Cliche. Bb\_twtr at semeval-2017 task 4: Twitter sentiment analysis with cnns and lstms. *arXiv preprint arXiv:1704.06125*, 2017.
- [56] Yuhui Cao, Ruifeng Xu, and Tao Chen. Combining convolutional neural network and support vector machine for sentiment classification. In *Chinese national conference on social media processing*, pages 144–155. Springer, 2015.

- [57] Monalisha Ghosh and Goutam Sanyal. Exploring the effectiveness of convolutional neural network with ensemble technique. 2020.
- [58] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
- [59] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.
- [60] Matthew D Zeiler. Adadelata: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.
- [61] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8), 2012.
- [62] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [63] T Dozat. Incorporating nesterov momentum into adam technical report, 2015.
- [64] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.
- [65] Yoonsuh Jung. Multiple predicting k-fold cross-validation for model selection. *Journal of Nonparametric Statistics*, 30(1):197–215, 2018.
- [66] Aurélien Géron. Hands-on machine learning with scikit-learn and tensorflow: Concepts, Tools, and Techniques to build intelligent systems, 2017.
- [67] Stephen V Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89, 1997.

## Appendix A: Hyperparameter tuning

### Optimizer Optimization

```
Best: 0.896296 using {'optimizer': 'Adamax'}
0.719012 (0.069077) with: {'optimizer': 'SGD'}
0.892716 (0.002145) with: {'optimizer': 'RMSprop'}
0.893210 (0.004860) with: {'optimizer': 'Adagrad'}
0.895556 (0.003408) with: {'optimizer': 'Adadelta'}
0.890864 (0.000761) with: {'optimizer': 'Adam'}
0.896296 (0.004997) with: {'optimizer': 'Adamax'}
0.892593 (0.002978) with: {'optimizer': 'Nadam'}
```

### Learning rate Optimization

```
Best: 0.896667 using {'lr': 0.01}
0.881111 (0.007722) with: {'lr': 0.1}
0.896667 (0.002584) with: {'lr': 0.01}
0.894691 (0.003732) with: {'lr': 0.001}
0.840494 (0.006374) with: {'lr': 0.0001}
0.481481 (0.006293) with: {'lr': 1e-05}
```

### Kernel size Optimization

```
Best: 0.895309 using {'kernel_size': 3}
0.895309 (0.004926) with: {'kernel_size': 3}
0.887901 (0.007003) with: {'kernel_size': 4}
0.893580 (0.003757) with: {'kernel_size': 5}
0.892839 (0.004280) with: {'kernel_size': 7}
0.893210 (0.003805) with: {'kernel_size': 9}
```

### Activation function Optimization

```
Best: 0.897407 using {'activation': 'relu'}
0.897407 (0.003666) with: {'activation': 'relu'}
0.893210 (0.003544) with: {'activation': 'tanh'}
0.892469 (0.002463) with: {'activation': 'elu'}
0.891852 (0.002419) with: {'activation': 'selu'}
```



## Pool size Optimization

```
Best: 0.897037 using {'pool_size': 2}
0.897037 (0.005833) with: {'pool_size': 2}
0.781852 (0.002400) with: {'pool_size': 4}
0.773827 (0.005421) with: {'pool_size': 6}
0.779753 (0.002058) with: {'pool_size': 8}
```

## GRU units Optimization

```
Best: 0.896420 using {'GRU_units': 64}
0.892099 (0.009224) with: {'GRU_units': 8}
0.894321 (0.003853) with: {'GRU_units': 16}
0.890247 (0.002744) with: {'GRU_units': 32}
0.896420 (0.008555) with: {'GRU_units': 64}
0.893210 (0.005293) with: {'GRU_units': 128}
0.894198 (0.005206) with: {'GRU_units': 256}
0.887901 (0.007682) with: {'GRU_units': 512}
```

## Droupout rate Optimization

```
Best: 0.899383 using {'dropout_rate': 0.3}
0.893580 (0.000972) with: {'dropout_rate': 0.1}
0.897531 (0.003148) with: {'dropout_rate': 0.2}
0.899383 (0.003853) with: {'dropout_rate': 0.3}
0.897284 (0.001666) with: {'dropout_rate': 0.4}
0.897160 (0.003888) with: {'dropout_rate': 0.5}
0.898889 (0.000605) with: {'dropout_rate': 0.6}
0.889259 (0.005051) with: {'dropout_rate': 0.7}
0.882593 (0.002772) with: {'dropout_rate': 0.8}
0.864691 (0.006662) with: {'dropout_rate': 0.9}
```

## Batch size optimization

```
Best: 0.896173 using {'batch_size': 16}
0.895679 (0.005327) with: {'batch_size': 8}
0.896173 (0.004172) with: {'batch_size': 16}
0.892716 (0.002572) with: {'batch_size': 32}
0.892593 (0.004516) with: {'batch_size': 64}
0.893827 (0.003331) with: {'batch_size': 128}
0.887901 (0.001848) with: {'batch_size': 256}
0.883951 (0.004248) with: {'batch_size': 512}
```

## Epoch size optimization

```
Best: 0.896173 using {'batch_size': 16}
0.895679 (0.005327) with: {'batch_size': 8}
0.896173 (0.004172) with: {'batch_size': 16}
0.892716 (0.002572) with: {'batch_size': 32}
0.892593 (0.004516) with: {'batch_size': 64}
0.893827 (0.003331) with: {'batch_size': 128}
0.887901 (0.001848) with: {'batch_size': 256}
0.883951 (0.004248) with: {'batch_size': 512}
```