Hate Speech Detection in Sinhala and Romanized Sinhala Mixed with Emojis using Hybrid CNN-RNN Model

A thesis submitted for the Degree of Masters of Computer Science

H.K. Waidyarathne

University of Colombo School of Computing

2025

DECLARATION

Name of the student: H. K. Waidyarathne
Registration number: 2022/MCS/065
Name of the Degree Programme: Master of Computer Science (MCS)
Project/Thesis title: Hate Speech Detection in Sinhala and Romanized Sinhala Mixed with Emojis using Hybrid CNN-RNN Model

1. The project/thesis is my original work and has not been submitted previously for a degree at this or any other University/Institute. To the best of my knowledge, it does not contain any material published or written by another person, except as acknowledged in the text.
2. I understand what plagiarism is, the various types of plagiarism, how to avoid it, what my resources are, who can help me if I am unsure about a research or plagiarism issue,as well as what the consequences are at University of Colombo School of Computing (UCSC) for plagiarism.
3. I understand that ignorance is not an excuse for plagiarism and that I am responsible for clarifying, asking questions and utilizing all available resources in order to educate myself and prevent myself from plagiarizing.
4. I am also aware of the dangers of using online plagiarism checkers and sites that offer essays for sale. I understand that if I use these resources, I am solely responsible for the consequences of my actions.
5. I assure that any work I submit with my name on it will reflect my own ideas and effort. I will properly cite all material that is not my own.
6. I understand that there is no acceptable excuse for committing plagiarism and that doing so is a violation of the Student Code of Conduct.

Signature of the Student	**Date (DD/MM/YYYY)**
H. Kathyana	24/06/2025

Certified by Supervisor

This is to certify that this project/thesis is based on the work of the above-mentioned student under my/our supervision. The thesis has been prepared according to the format stipulated and is of an acceptable standard.

	Supervisor
Name	Dr. B.H.R. Pushpananda
Signature	Randil
Date	24/06/2025

I would like to dedicate this thesis to fostering a safer and more inclusive online environment in Sri Lanka, where everyone can express themselves freely without fear of hate speech.

ACKNOWLEDGMENTS

I would like to express my deep appreciation and gratitude to Dr. Randil Pushpananda, my supervisor, for his invaluable guidance and support throughout my thesis project. Dr. Randil's expertise and commitment to excellence have been instrumental in shaping my research and helping me achieve my academic goals.

I would also like to extend my heartfelt thanks to Dr. Kasun Karunanayaka, our project coordinator, for his constant motivation and guidance. Dr. Kasun's encouragement and leadership have been a driving force behind my success, and I am truly grateful for his unwavering support.

I would be remiss not to acknowledge the unwavering support and love of my husband, Sachintha Rathnayake, whose unwavering support and encouragement sustained me through the ups and downs of this challenging journey. Your patience, understanding, and unwavering belief in me have been a constant source of strength, and I am incredibly fortunate to have you in my life.

Finally, I would like to thank my parents for their unconditional love and support. Their guidance, encouragement, and sacrifices have been instrumental in helping me achieve my academic goals. I am grateful for their unwavering belief in me and their unwavering support throughout this journey.

Thank you all for your invaluable contributions to my thesis project and for being a part of my journey. Your support, guidance, and encouragement have been truly transformative, and I am grateful beyond words for your presence in my life.

ABSTRACT

The paper addresses the critical issue of the hate speech detection in online Sinhala and Romanized sinhala content associated with emojis. The widespread of online hateful content is particularly significant in low-resource languages such as Sinhala, where natural language processing tools and resources are limited. The study examined the development and evaluation of a novel deep learning approach to effectively identify the words of hatred in the Sinhala language and the Romanized Sinhala text, and complicated the inclusion of emojis, which are prevalent in the online communications of the Sinhala language.

In order to address this challenge, we propose a combined model of the Convolutional Neural Network - Bidirectional Long Short-Term Memory (CNN-BiLSTM) as our primary architecture. This model is designed to use the strengths of both CNNs to capture local n-gram patterns and BiLSTMs to understand the sequential context within the Sinhala text. CNN extracts local signs of hate speech, while BiLSTM captures long-term dependencies and contextual information in Sinhala comments. An Embedding layer, shared by both branches, learns dense vector representations for Sinhala words and emojis optimized for the task of detecting offensive speeches. In order to compare, we also implemented and evaluated independent CNN, BiLSTM, LSTM, Support Vector Machine (SVM), Random Forest, Logistic Regression, and Naive Bayes models.

The results show that our CNN-BiLSTM combined model is superior to the comparable models. The CNN-BiLSTM model achieved a high accuracy of 0.9251 and a balanced F1 score of 0.92 for the "Offensive" class, indicating the robustness of detecting hatred speech while minimizing both positive and negative false information. The analysis of the confusion matrix further revealed a balanced distribution of errors, highlighting the effectiveness of the model in the treatment of offensive and non-offensive Sinhala content. Although simple models such as SVM and Random Forest showed competitive accuracy, their precision and F1 points were generally lower, especially in the "Offensive" class. Neural network architectures, in particular CNN-BiLSTM and CNN, have consistently surpassed traditional machine learning methods.

Keywords: Hate Speech Detection, Sinhala Language, Low-Resource Language, CNN-BiLSTM, Deep Learning, Natural Language Processing, Romanized Sinhala, Emojis

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

BiLSTM Bidirectional Long Short-Term Memory.

CNN Convolutional Neural Network.

DL Deep Learning.

DT Decision Tree.

HS Hate Speech.

KNN K-Nearest Neighbors.

LR Logistic Regression.

LSTM Long Short-Term Memory.

ML Machine Learning.

NB Naïve Bayes.

NLP Natural Language Processing.

RF Random Forest.

RNN Recurrent Neural Network.

SVM Support Vector Machine.

CHAPTER 1

INTRODUCTION

1.1 Chapter Overview

This chapter provides the foundation for research by introducing the context of the hate speech in Sinhala and Romanized Sinhala (Singlish) mixed code with emojis. The chapter also presents the research objectives, scope and structure of the thesis.

1.2 Background to the Research

In the era of social computing, people are increasingly interacting with one another, particularly through social media platforms and online chat forums. Microblogging applications have allowed people to share their thoughts instantly and widely with a global audience (Jahan & Oussalah 2023). However, this openness also facilitates the dissemination of destructive and hostile information, driven by users' desire to dominate discussions, strengthen views, and sometimes even to pursue business incentives.

Hate speech (HS) is widely understood as any form of expression that provokes hostility or insults individuals or groups based on inherent characteristics such as race, color, national origin, gender, disability, religion or sexual orientation (Jahan & Oussalah 2023). Although the legal definitions of hatred are different in each nation, the common factor remains its potential to perpetuate discrimination and social division (Alkomah & Ma 2022).

Further classification of hate speech identifies various subcategories, including cyberbullying, racism, sexism, gender discrimination, radicalization, abusive language, offensive language, and religious hate speech (Jahan & Oussalah 2023). These categories are interrelated, forming a hierarchical structure depicted in Figure 1.1 where hate speech serves as the parent node, encompassing these subcategories. At the second level, cyberbullying and abusive/offensive language are distinguished as key branches. Further distinctions are made for sexism, racism, and radicalization, each representing specific dimensions of online toxicity.

With the fast development in technology, internet users have increased exponentially over the years (Tarwani et al. 2019). With the advancement of smartphones and more affordable data plans, the use of social media has become one of the main communication channels in both global and local contexts. It has been stated that the number of social media

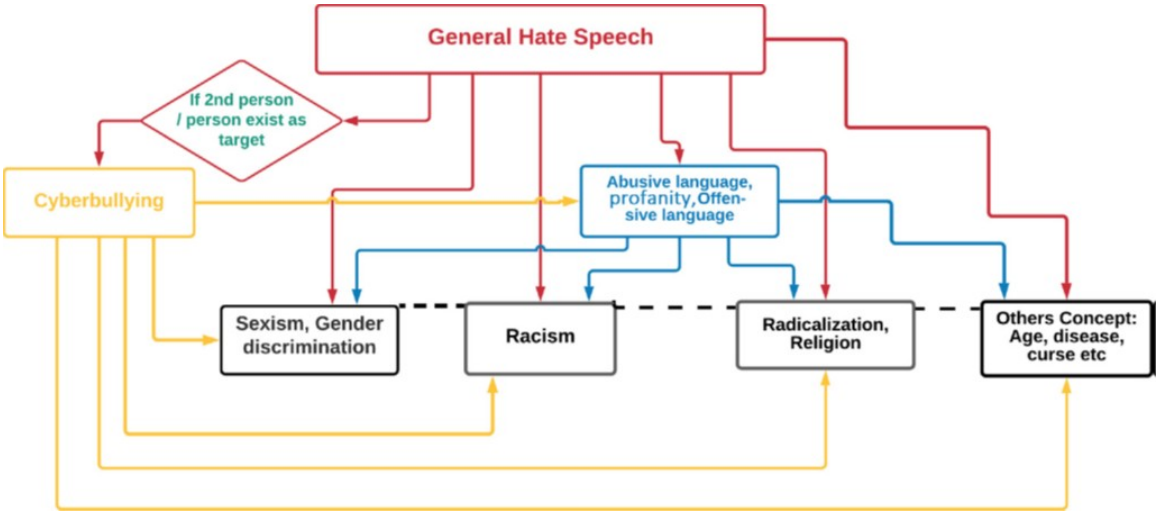


Figure 1.1: *Different type of Hate Speech Concepts*

users has been stated to increase day by day, particularly in developing countries such as India, Sri Lanka, China, etc. (Tarwani et al. 2019).

There were 7.50 million active social media users in Sri Lanka by January 2024 (Kemp 2024), which is 34.2 percent of the total population. Social media can be used to gain many insights such as the popularity of products among customers from a business perspective. With this rapid increase in social media users, the use of hate speech, abusive, and offensive content sharing has increased substantially. These behaviors are called cyberbullying, where people bully others behind closed curtains where no one can see. And the victims can get more affected psychologically as these activities can be seen by a wider audience within a concise period.

Some of the statistics figures related to the internet and social media usage in Sri Lanka are shown in Figures 1.2 and 1.3. According to Figure 1.3, we can see that 3.9% of internet users have increased compared to the past year in January 2024.

According to the Sri Lanka Computer Emergency Readiness Team (SLCERT), there were over 20,000 cybersecurity-related incidents in 2023. During the period of COVID-19 pandemic, there's been a rapid increase in cyber-crimes which includes cyberbullying and hate speech (CERT 2025). Hence the importance of early detection of cyberbullying/ hate speech can result in a decrease in the overall number of such cyber-crimes. The growth of social media cybercrimes in Sri Lanka from 2018-2023 is depicted in Figure 1.4.

Many studies have been done on hate speech (HS) detection for social media, particularly English (Patil et al. 2023, Sossi Alaoui et al. 2022, Malik et al. 2023) and other widely spoken languages. However, there has been limited number of research on hate speech detection in Sinhala and Romanized Sinhala (Muthuthanthri & Smith 2024, Fernando et al.

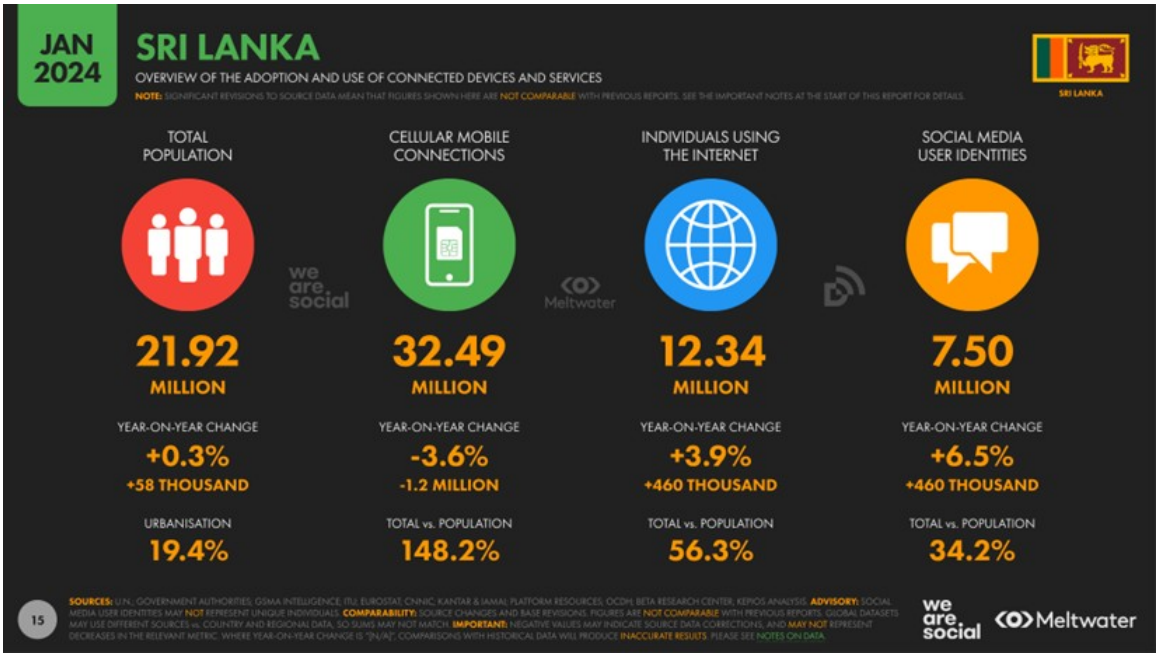


Figure 1.2: *Internet & Social Media usage in Sri Lanka (Source: https://datareportal.com/)*

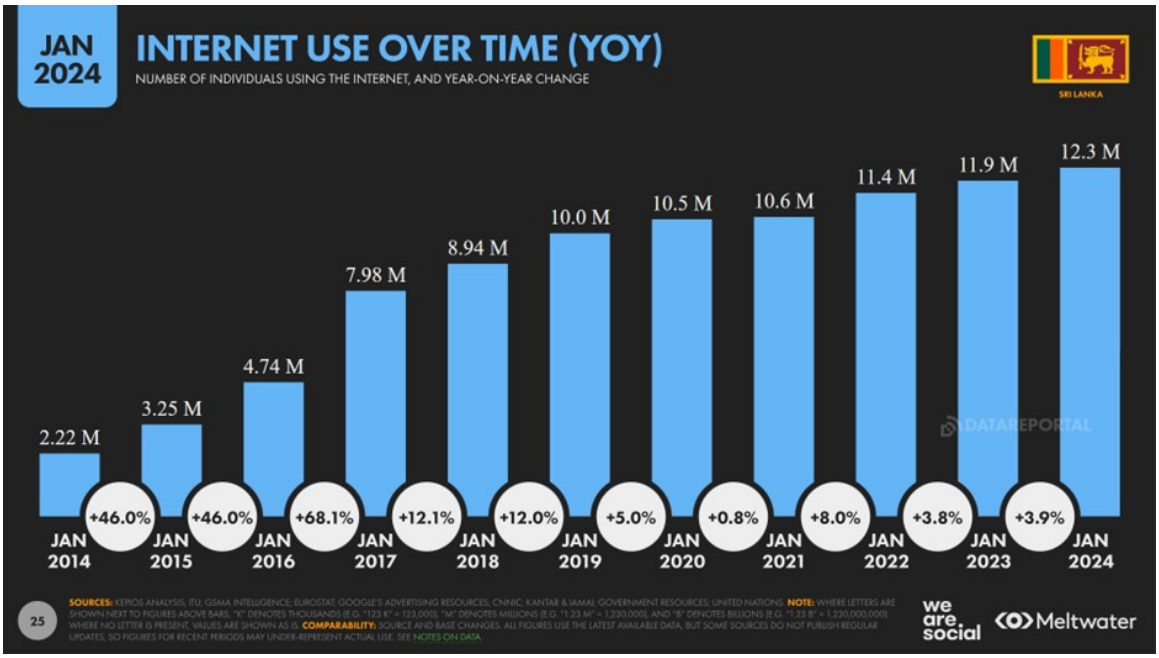


Figure 1.3: *Internet user trends in Sri Lanka*

2022). In Sri Lanka, most of the hate speech content can be seen in code mixed form due to the widespread use of the language on popular social media platforms.

Sentiment analysis for Romanized Sinhala is very difficult due to various challenges. The Sinhala language belongs to the Indo-Aryan branch of the Indo-European languages,

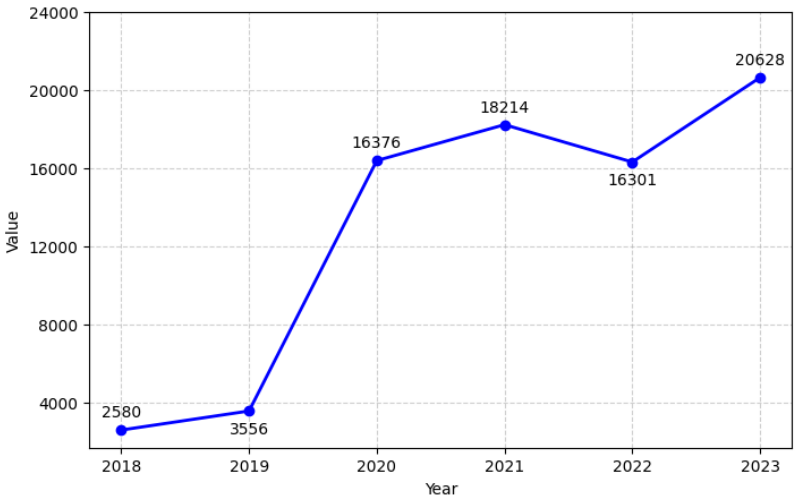


Figure 1.4: *Growth of social media cyber crimes in Sri Lanka from 2018-2023 Source: (CERT 2025)*

and it has also been influenced by colonial-period languages such as Dutch, Portuguese, and English (Ishara Amali & Jayalal 2020). Romanized Sinhala has many different spelling variations; for example, "Ayya" and "Aiya" are the same Romanized Sinhala word for 'brother' in English and the lack of a lexical database such as WordNet for Sinhala words makes this task rather difficult.

Moreover, Sinhala and Romanized Sinhala words can have multiple interpretations for the same word, lack specific grammar rules, and have a different vocabulary in different regions of the country, resulting in difficulties when automating the detection of hate speech comments. For example: "බැල්ලිගේ putha" which means "Son of bitch" in English and "gedara බැල්ලි" which means "female dog at home" indicates two different interpretations of the word "බැල්ලි" where the first use makes it offensive and the second does not. In addition, there can be various trends coming up from time to time in social media which would change how people use their language. Ex - There has been a trend to type Sinhala with the sound of the letter 'm' (i.e., "karamnam", "කියම්න" etc). Furthermore, new words are being added to the vocabulary of people with time (i.e., "Sago" as a word to talk to someone close).

Despite these complexities, there has been minimal research on hate speech detection in Sinhala and Romanized Sinhala code-mixed text with emojis, and no significant studies were found focusing specifically on this area. Therefore, the main focus of this study is to identify harsh or insulting comments targeting a person in social media comments using effective and highly accurate machine learning models.

1.3 Motivation

This research is driven by several important reasons. There is a lack of reliable ways to automatically detect hate speech in Sinhala and Romanized Sinhala with emojis as stated in section 1.2. Organizations need a system that can identify and deal with harmful online content in real-time. Without it, hate speech spreads easily, making the internet a negative place for everyone.

Sinhala and Romanized Sinhala (Singlish) with Emojis have special characteristics that make it difficult to detect hate speech. Romanized Sinhala has many ways of spelling words, different meanings for the same words, and no strict grammar rules that make it difficult to understand the automated system. There is a growing problem of hate speech and cyberbullying in Sri Lanka, especially since COVID-19. More people using the internet and social media means more chances of negative behavior.

Finally, this research aims to make the Internet safer and more welcoming place for everyone, no matter what language they use. By creating a strong model to detect hate speech in Sinhala and Romanized Sinhala with emojis, this study hopes to give people the tools to reduce online negativity and encourage better conversations.

1.4 Statement of the problem and Research Questions

Hate speech (HS) has become a major issue on social media platforms worldwide, and Sinhala language content is no exception. Detecting hate speech in social media comments are written in Sinhala and Romanized Sinhala code-mixed text often used with emojis is particularly challenging due to the unique features, complexity, and nuances of the language. Moreover, the use of colloquial terms, slang, and abbreviations in online communications further complicates the detection process. The problem is further exacerbated by the lack of effective detection mechanisms and tools that are specifically tailored for Sinhala and Romanized Sinhala code-mixed text, making it difficult to monitor and address instances of hate speeches on social media platforms effectively.

This study aims to investigate how machine learning (ML) algorithms can be used to improve the detection of hate speech in social media comments written in Sinhala and Romanized Sinhala code-mixed text with emojis. By leveraging the power of machine learning and natural language processing techniques, we seek to develop a robust and accurate model for detecting and classifying hate speech content. In particular, we explore the performance of various classifiers and feature extraction methods to determine their effectiveness in identifying hate speech content in Sinhala and Romanized Sinhala code-mixed text.

Furthermore, we examine the impact of text pre-processing on the performance of these

classifiers and feature extraction methods, taking into account the unique characteristics of Sinhala and Romanized Sinhala code-mixed text. By understanding the factors that contribute to the success or failure of these detection methods, we hope to contribute to the development of more effective tools and strategies to identify hate speech in the Sinhala-speaking community.

Ultimately, the findings of this study will not only provide valuable insights into the challenges and opportunities in detecting hate speech in the Sinhala and Romanized Sinhala code-mixed language but also serve as a foundation for future research and practical applications in the field of hate speech detection for other languages and contexts. By addressing this critical issue, we can work towards creating a safer and more inclusive online environment for all users, regardless of the language they use to communicate.

1. Is a CNN-BiLSTM model an effective architecture for hate speech detection in Sinhala and Romanized Sinhala text that contains emojis (Muthuthanthri & Smith 2024)?

2. How effective is the proposed CNN-BiLSTM combined model in detecting hate speech in Sinhala and Romanized text in Sinhala, particularly with emojis and considering the complexity of low-resource languages (Muthuthanthri & Smith 2024, Alkasassbeh et al. 2024)?

3. How is the performance of the CNN-BiLSTM model compared to the standalone deep learning models (CNN, BiLSTM, LSTM) and traditional machine learning classifiers (SVM, Random Forest, Logistic Regression, Naive Bayes) in the context of Sinhala hate speech detection, and what are the relative contributions of CNN and BiLSTM components in the integrated architecture (Riyadi et al. 2024)?

1.5 Research Aim and objectives

1.5.1 Aim

The main focus of this study is to identify harsh or insulting comments targeting individuals on social media by leveraging effective and highly accurate model for detecting hate speech in Sinhala and Romanized Sinhala (Singlish) text with emojis, using deep learning techniques.

1.5.2 Objectives

- **Create a Sinhala and Romanized Sinhala code-mixed dataset including emojis, collected from social media and online platforms (Facebook, Instagram, TikTok,**

YouTube): To provide a strong foundation for the study, a comprehensive dataset containing Sinhala and Romanized Sinhala text with emojis from various social media and online platforms will be compiled. This dataset will serve as the basis for training and evaluating the machine learning models to accurately detect hate speech instances in Sinhala and Romanized Sinhala code-mixed.

- **Investigate the feature extraction methods for detecting hate speech in Sinhala and Romanized Sinhala code-mixed text:** The study will explore the effectiveness of features in comparison to other feature extraction methods such as Bag-of-Words and TF-IDF in order to identify the most suitable approach for detecting hate speech in Sinhala and Romanized Sinhala. (Fernando et al. 2022, Smith & Thayasivam 2019).

- **Finding the best-performing machine learning approche for detecting hate speech in Sinhala and Romanized Sinhala code-mixed with emojis in social media/online comments:** Several machine learning classifiers, including CNN, BiLSTM, LSTM, SVM, Random Forest (RF), Logistic Regression (LR) and Naive Bayes, will be evaluated and compared to determine the most effective and accurate classifier for detecting hate speech (Agrawal & Awekar 2018, Abebaw et al. 2022).

- **Assess the impact of text pre-processing on detecting hate speech in Sinhala and Romanized Sinhala code-mixed:** The study will investigate the effects of text pre-processing techniques, on the performance of feature extraction methods. This will help determine the optimal pre-processing pipeline for the hate speech detection.

- **Evaluate the impact of user-based and text-based features on the performance of classifiers in detecting hate speech content in Sinhala and Romanized Sinhala code-mixed:** The study will investigate the performance of various machine learning classifiers using appropriate metrics. This will help in understanding the significance of these features in enhancing the accuracy of hate speech detection in Sinhala and Romanized Sinhala.

1.6 Scope

1.6.1 In Scope

- **Creating a dataset:** Gathering Sinhala and Romanized Sinhala code-mixed text with emojis from various social media and online platforms to train and test machine learning models.

- **Exploring feature extraction:** Investigating the effectiveness of Bag-of-Words and TF-IDF for detecting hate speech in Sinhala and Romanized Sinhala code-mixed text.
- **Evaluating classifiers:** Assessing and comparing the performance of various machine learning classifiers to determine the most effective and accurate classifier for detecting hate speech in Sinhala and Romanized Sinhala code-mixed text.
- **Investigating text pre-processing:** Analysing the effects of text pre-processing techniques on the performance of feature extraction methods.
- **Analysing text-based features:** Examining the impact on the performance of machine learning classifiers.
- **Performing statistical evaluation:** Employ statistical hypothesis evaluate to validate the findings of study and ensure their reliability.
- **Providing insights and recommendations:** Offering insights into the most effective methods for detecting hate speech in Sinhala and Romanized Sinhala code-mixed and suggesting recommendations for future research and practical implementations in the field of hate speech detection.

1.6.2 Out of Scope

- **Analysis of multimedia content:** While hate speech may occur through images, videos, and audio files, this study will solely focus on the detection of hate speech on text based inputs from social media and online platforms.
- **Emotion and sentiment analysis:** Although these techniques may provide additional insights into the nature of hate speech content, the study will not dive into the sentiment analysis or emotion recognition in the text, as the primary focus lies on feature extraction methods for detecting hate speech instances.
- **Hate speech detection in languages other than Sinhala and Romanized Sinhala code-mixed:** The study is specifically focused on the detection of hate speech content in Sinhala and Romanized Sinhala code-mixed text with emojis. Research on hate speech detection in other languages or scripts is considered out of scope.
- **Development of a full-fledged hate speech detection system or application:** The primary objective of this research is to identify the most effective methods and classifiers for detecting hate speech in Sinhala and Romanized Sinhala code-mixed lan-

guage. The development of an end-to-end system or application for practical use is not within the scope of this study.

1.7 Structure of the Thesis

The rapid growth of internet and social media usage in Sri Lanka, as illustrated in figs. 1.2 and 1.3, has led to increased opportunities for communication and information sharing. However, this growth has also contributed to a rise in negative behaviours, such as cyberbullying and hate speech. The Sri Lanka Computer Emergency Readiness Team (SLCERT) reports a sharp increase in cyber crimes, including cyberbullying and hate speech, during the COVID-19 pandemic, emphasizing the importance of early detection in curbing these incidents (CERT 2025).

Although numerous studies have been conducted on hate speech detection in English and other popular languages, limited research has been carried out on the Sinhala and Romanized Sinhala code-mixed language. This study aims to fill this gap by focusing on the detection of hate speech in Sinhala and Romanized Sinhala code-mixed with emojis using machine learning approaches. The main objectives of the research include creating a novel dataset that contain Sinhala and Romanized Sinhala code-mixed language with emojis, investigating effective feature extraction methods, identifying the best-performing machine learning classifiers, assessing the impact of text pre-processing, and performing statistical hypothesis to evaluate and validate the results in detecting hate speech.

CHAPTER 2

LITERATURE REVIEW

2.1 Chapter Overview

In this chapter provides a comprehensive review of existing hate speech detection approaches and the background of the research context is explored. Initially, an introductory background to the context is given and then a comprehensive study is done on the related literature.

2.2 Hate Speech Detection Approaches

Hate Speech is a broad field and studied across a large number of research domains. Hate Speech detection approaches can be divided into three main sub-domains which are Ruled Based, Machine Learning (ML) and Deep Learning (DL). This research is focused on Deep Learning approaches. According to the literature, deep learning approaches can be further categorized based on hate speech patterns, as shown in Figure 2.1.

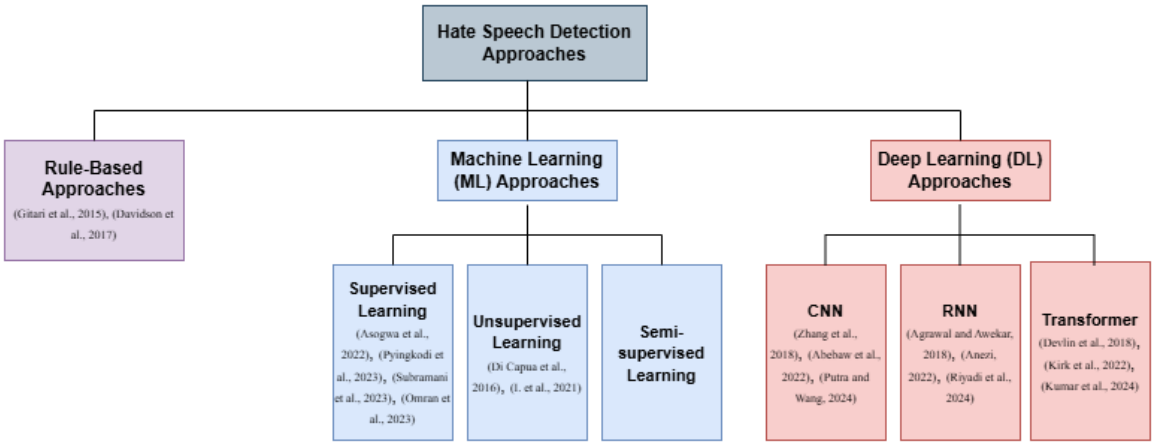


Figure 2.1: Hate Speech detection approaches

2.2.1 Rule-Based Linguistic Approaches

The linguistic rule-based approach for hate speech detection relies on a set of predefined rules and a linguistic engine that understands grammar, morphology, and semantics in a

given language. These approaches often use lexicons, dictionaries, and pattern-matching techniques to identify hateful content (Mohiyaddeen & Siddiqui 2021).

A key characteristic of rule-based approaches is their ability to detect words with similar meanings by leveraging synonym expansion techniques. For instance, if the system is programmed to detect the word ”bad”, it can also identify semantically related words like ”terrible”, ”awful”, or ”unsatisfactory” by using linguistic rules and thesauri.

Similarly, Gitari et al. (2015) proposed a rule-based method for sentiment analysis of social media text, categorizing hate speech into religion, nationality, and race. Their classification model utilized sentiment analysis to detect subjective sentences and rank the polarity of sentiment phrases, linking semantic and subjective features to hate speech. Their lexicon-based approach achieved 71.55% precision, highlighting the potential of sentiment-driven rule-based classification in hate speech detection.

Davidson et al. (2017) implemented a lexicon-based filtering method combined with rule-based pattern matching to classify hate speech. Their approach achieved F1-score of 0.91 for detecting offensive language, demonstrating the effectiveness of predefined hate speech terms. However, distinguishing between offensive and hateful speech remained a challenge.

2.2.2 Machine Learning Approaches

Machine learning involves training a classifier or regression model using a given dataset and evaluating its performance on a test dataset. It enables systems to learn patterns from data and make predictions or classifications. Machine learning can be broadly categorized into Supervised Learning, where labeled data is used to train models; Unsupervised Learning, which identifies patterns and structures in unlabeled data; and Semi-Supervised Learning, which leverages a small amount of labeled data along with a large pool of unlabeled data to improve model performance (Mohiyaddeen & Siddiqui 2021).

2.2.2.1 Supervised Learning Approaches

Asogwa et al. (2022) explored the effectiveness of the Support Vector Machine (SVM) and the Naïve Bayes (NB) algorithms in detecting hate speech. Their study utilized a dataset comprising 62,485 instances, which underwent preprocessing and feature extraction to enhance model performance. The empirical evaluation revealed that the SVM classifier achieved a classification accuracy of approximately 99%, while the NB classifier attained around 50% accuracy on the test set. These findings suggest that SVM is a more effective method for hate speech detection in this context.

Pyingkodi et al. (2023) used various supervised machine learning algorithms to detect hate speech in social media content. They trained models using a dataset from Kaggle, applying preprocessing techniques such as tokenization, stop-word removal, and lemmatization. The study evaluated classifiers including Logistic Regression, Naïve Bayes, Random Forest, and Support Vector Machines (SVM). Among these, Random Forest achieved the highest accuracy of 90.1%, demonstrating its effectiveness in classifying hate speech.

Subramani et al. (2023) proposed a supervised machine learning approach for hate speech classification using Gradient Boosting, Random Forest, and Naïve Bayes algorithms. They trained their model on a Kaggle dataset containing hate speech, offensive, and neutral comments. After applying data preprocessing techniques such as tokenization, stop-word removal, and feature extraction, they found that Gradient Boosting achieved the highest accuracy of 86.04%, making it the most effective classifier in their study.

Omran et al. (2023) conducted a comprehensive comparative analysis of machine learning algorithms for hate speech detection in social media. Their study aimed to identify an optimal algorithmic combination that balances simplicity, ease of implementation, computational efficiency, and strong performance metrics. Through meticulous pre-processing and rigorous evaluation, they found that combining Naïve Bayes and Decision Tree algorithms achieved a high accuracy of 0.887 and an F1-score of 0.885, demonstrating its effectiveness in hate speech detection.

Alfreihat et al. (2024) developed an Emoji Sentiment Lexicon (Emo-SL) that is specifically designed for Arabic tweets. They addressed the challenges of feeling analysis in informal Arabic texts, which include morphological complexity and dialectic variations. The Emo-SL was constructed using a corpus of 58K Arabic tweets containing emojis, and sentiment scores were calculated for 222 frequently occurring emojis. By combining emoji-based features with machine learning (ML) to classify feelings, the authors demonstrated performance improvements. The paper mentions that the integrated emoji-aware approach achieves an 89% F1 score, exceeding the VADER sentiment analysis based on rules. Furthermore, the results show that adding Emo-SL derived emoji functions to ML classifiers can significantly improve the accuracy of 26.7% compared to only text functions. They give an example of translated arabic sentiment X's tweets such as I loved a movie so much-a great acting, poignant story 😍🎥, I was disappointed with today's service I expect the best 😞💔.

2.2.2.2 Unsupervised Learning Approaches

Di Capua et al. (2016) proposed an unsupervised cyberbullying detection approach using Growing Hierarchical Self-Organizing Maps (GHSOMs). They applied semantic and syn-

tactic feature clustering to identify bullying traces in social network data without requiring labeled datasets. The model was fine-tuned for Twitter and tested on YouTube and Formspring, achieving an accuracy of 69%, a recall of 94%, and a precision of 60% on the YouTube dataset. Their results demonstrated that unsupervised learning can effectively detect cyberbullying patterns across different social media platforms.

I. et al. (2021) proposed an unsupervised hybrid approach for detecting cyberbullying on Instagram. They utilized the InstaCities1M dataset, a collection of Instagram posts associated with major U.S. cities, to evaluate their method. By extracting linguistic features such as idioms, sarcasm, irony, and voice (active or passive), alongside user and network attributes, their approach achieved an accuracy of 88% and an F-measure of 0.962. This demonstrates the effectiveness of unsupervised learning in identifying cyberbullying without the need for labeled data.

2.2.2.3 Deep Learning Approaches

The deep learning approach leverages artificial neural networks to automatically learn and extract complex patterns from data. When provided with a dataset, deep learning models process the information through multiple layers of neurons, each refining the understanding of features (Mohiyaddeen & Siddiqui 2021). Convolutional Neural Networks (CNNs) excel in image recognition by detecting edges, shapes, and textures, while Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks are effective for sequential data like text (Agrawal & Awekar 2018). Transformer-based models such as BERT and GPT further enhance language processing by capturing contextual meaning (Putra & Wang 2024). With sufficient training, deep learning models achieve high accuracy in classification tasks, making them powerful tools for hate speech detection and other NLP applications.

Convolutional Neural Networks (CNN)

Earlier studies experimented with CNNs for hate speech detection. Zhang et al. (2018) proposed a convolutional-GRU deep neural network for detecting hate speech on Twitter. Their model combined convolutional layers for feature extraction with GRU layers to capture sequential patterns in short text, enhancing accuracy. The approach outperformed existing models, achieving up to a 13% improvement in F1-score across six out of seven datasets. The authors also introduced a new dataset covering topics like religion and refugees, expanding the scope of available data for hate speech detection in social media contexts.

Abebaw et al. (2022) proposed a multichannel Convolutional Neural Network (CNN) for hate speech detection in social networks. Their model utilized word2vec embeddings

across multiple channels to enhance feature extraction and improve the model's sensitivity to nuanced textual patterns. The multichannel approach significantly outperformed single-channel models, achieving an accuracy of 92.5% and an F1-score of 0.89, demonstrating its effectiveness in detecting hate speech across limited datasets in social media platforms.

Putra & Wang (2024) proposed an advanced BERT-CNN model for hate speech detection, combining BERT's contextual embeddings with CNN for spatial feature extraction. This hybrid approach achieved high performance, with an accuracy of 94.5% and an F1-score of 0.937 across multiple datasets, outperforming traditional models and demonstrating its effectiveness in identifying hate speech on social media platforms.

Recurrent Neural Network (RNN)

Multiple studies have evaluated RNNs, LSTMs and GRUs, given their strength in sequential modelling. Agrawal & Awekar (2018) investigated cyberbullying detection using RNN-based models, including LSTM (Long Short-Term Memory) and BiLSTM (Bidirectional LSTM) with attention. Their study demonstrated that deep learning models significantly outperform traditional machine learning techniques. By leveraging transfer learning across multiple social media platforms, their BiLSTM with attention model achieved an F1-score of 0.94, highlighting its effectiveness in detecting cyberbullying with minimal feature engineering.

Anezi (2022) developed a deep Recurrent Neural Network (RNN) model, named DRNN-2, for detecting hate speech in Arabic social media content. The architecture comprises 10 layers with a batch size of 32 and performs classification over 50 iterations. The model achieved an accuracy of 94% and an F1-score of 0.91, demonstrating its effectiveness in identifying hate speech within Arabic textual data.

Riyadi et al. (2024) proposed a double-layer hybrid CNN-RNN model to enhance hate speech detection on imbalanced datasets. This approach integrates Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to effectively capture spatial and temporal features in textual data. The double-layer architecture addresses data imbalance by enhancing feature extraction and representation. The model achieved an accuracy of 82.7%, a precision of 79.7%, a recall of 75.9%, and an F1-score of 77.8%, highlighting its potential in mitigating online toxicity.

Transformer

Recent studies have evaluated Transformer networks pre trained on large text corpora. Devlin et al. (2018) introduced BERT (Bidirectional Encoder Representations from Transformers), a groundbreaking language representation model that captures context from both

directions in text. By pre-training on large text corpora and fine-tuning on specific tasks, BERT achieved state-of-the-art results across various NLP benchmarks. Notably, it improved the GLUE benchmark score to 80.5%, MultiNLI accuracy to 86.7%, SQuAD v1.1 question answering Test F1 to 93.2, and SQuAD v2.0 Test F1 to 83.1, demonstrating its effectiveness in understanding and processing natural language.

Kirk et al. (2022) addressed the challenge of detecting hate speech that incorporates emojis by introducing HatemojiCheck, a test suite of 3,930 short-form statements designed to evaluate the performance of hate detection models on emoji-based content. Their findings revealed significant weaknesses in existing models when processing such content. To mitigate these issues, they developed HatemojiBuild, a dataset comprising 5,912 adversarial examples generated through a human-and-model-in-the-loop approach. Models trained on this dataset demonstrated a substantial improvement in detecting emoji-based hate, achieving an accuracy increase from 60.0% to 76.2% on the HatemojiCheck test suite, while maintaining strong performance on text-only hate speech detection. They give an example of racist use of emoji following England's defeat in the Euro 2020 football final, with emojis such as 🐵, 💩, and 🍌.

Kumar et al. (2024) investigated the effectiveness of leading large language models (LLMs) in generating synthetic biased and cyberbullying data and evaluated the proficiency of Transformer AI models in detecting bias and cyberbullying within both authentic and synthetic contexts. The study involved semantic analysis and feature engineering on a dataset of over 48,000 sentences related to cyberbullying collected from Twitter. Leveraging state-of-the-art LLMs such as ChatGPT-4o, Pi AI, Claude 3 Opus, and Gemini-1.5, synthetic biased, cyberbullying, and neutral data were generated to deepen the understanding of bias in human-generated data. AI models including DeBERTa, Longformer, BigBird, HateBERT, MobileBERT, DistilBERT, BERT, RoBERTa, ELECTRA, and XLNet were initially trained to classify Twitter cyberbullying data and subsequently fine-tuned, optimized, and quantized for multilabel classification (detecting both biases and cyberbullying). The outcome of the study include a prototype of a hybrid application that combines a Bias Data Detector and a Bias Data Generator, validated through extensive testing.

2.3 Comparative Analysis of Hate Speech Detection Approaches

Various studies have been conducted in the field of hate speech (HS) detection. This study has categorized these approaches into three main sections: Linguistic Rule-Based Approaches, Machine Learning Approaches, and Deep Learning Approaches. Table 2.1 summarizes key research work, highlighting the algorithms and features used in each approach, along with their reported performance metrics.

Table 2.1: Comparative Analysis

Author & Year	Platform	Algorithm	Precision	Recall	Accuracy	F1-score
Rule-Based Linguistic Approach						
Gitari et al. (2015)	Social Media	Lexicon-based, Sentiment Analysis	73.42%	68.42%	-	70.83%
Davidson et al. (2017)	Twitter	Logistic regression with L2 regularization	91%	90%	-	90%
Supervised Learning Approach						
Asogwa et al. (2022)	Social Media	SVM, NB	-	-	SVM: 99% NB: 50%	-
Pyingkodi et al. (2023)	Twitter	LR, NB, RF, SVM	-	-	90.1%	-
Subramani et al. (2023)	Social Media	Gradient Boosting Algorithm	92%	80%	86.04%	94%
Omran et al. (2023)	Social Media	SVM, KNN, RF, NB, DT	84%	86%	86%	83%
Alfreihat et al. (2024)	Twitter	SVM, KNN, RF, NB, KNN	-	-	26.7%	89%
Unsupervised Learning Approach						
Di Capua et al. (2016)	Twitter, YouTube	GHSOM	81%	26%	72%	40%
I. et al. (2021)	Instagram	HAN, GAE	-	-	88%	96%
Deep Learning Approaches - CNN						
Zhang et al. (2018)	Twitter	CNN, GRU	-	-	94%	-
Abebaw et al. (2022)	Social Media	Multichannel CNN	83.2%	87.6%	-	85.3%
Putra & Wang (2024)	Social Media	CNN, BERT	-	-	91%	-
Deep Learning Approaches - RNN						
Agrawal & Awekar (2018)	Social Media	CNN, LSTM, SVM, LR, RF, NB	-	-	98%	95%
Anezi (2022)	Social Media	Deep RNN	-	-	95.38%	-
Riyadi et al. (2024)	Social Media	hybrid CNN-RNN	79.7%	78.8%	82.7%	79.2%
Deep Learning Approaches - Transformers						

Continued on next page

Table 2.1 – continued from previous page

Author & Year	Platform	Algorithm	Precision	Recall	Accuracy	F1-score
Devlin et al. (2018)	Social Media	BERT	-	-	80.5%	-
Kirk et al. (2022)	Social Media	BERT	-	-	82.8%	84.7%
Kumar et al. (2024)	Twitter	DeBERTa, Longformer, HateBERT, RoBERTa, XLNet	-	-	-	-

2.4 A Literature Review of the Domain

As stated previously, many studies have been carried out during the past decade in detecting hate speeches covering many areas. Nevertheless, only very few have been carried out for the Sinhala language and specifically for Sinhala and Romanized Sinhala. This section will be focused on previous work on hate speech detection which used machine learning and deep learning.

2.4.1 Related work for English and other languages

Hate speech detection has been widely researched for English and several code-mixed languages. Much of the research done in this area had used mainly either machine learning and deep learning models.

In Davidson et al. (2017) address the challenge of distinguishing hate speech from offensive language on Twitter. They labeled tweets as hate speech, offensive, or neither, and trained a classifier to differentiate between these categories. The study revealed that racist and homophobic tweets were more likely to be classified as hate speech, while sexist tweets were often seen as merely offensive. This highlights the difficulty in automatically detecting hate speech due to the nuances of human interpretation. The research underscores the need for context-aware and nuanced approaches in automated hate speech detection.

Another research carried out by Badjatiya et al. (2017) explored deep learning for hate speech detection in English tweets, classifying them as racist, sexist, or neither. They experimented with deep learning architectures like CNNs and LSTMs to learn semantic word embeddings. Their approach outperformed character n-gram methods by approximately 18 F1 points, achieving a macro F1 score of 93%. This research demonstrated the effectiveness of deep learning in capturing the complexities of hate speech.

In Haidar et al. (2017) proposed a system for detecting cyberbullying in Arabic content, using machine learning techniques such as Naive Bayes, Support Vector Machine (SVM), and Decision Tree. The authors used a dataset of Arabic comments collected from social media platforms and achieved an accuracy of 82.2% with the SVM classifier.

Similar to Sinhala, Hindi is also a morphologically rich language, and Tarwani et al. (2019) was conducted to detect cyberbullying using sentiment classification for the Romanised Hindi language. Here they first created an annotated Romanised Hindi (Hinglish)

cyberbullying comments dataset taken from Instagram and YouTube and then developed a hybrid machine-learning model. Here they have used stemming and lemmatization as pre-processing techniques and TFIDF vectorizing and N-grams as feature extraction methods. They selected five best-performing classifiers namely, Multinomial Naive Bayes, Logistic Regression, SVM, Decision Tree, and Passive Aggressive classifiers out of 9 classifiers they tested individually and then developed a hybrid model by combining these 5 which outperformed all the other classifiers. They have also created a spelling variation list to replace similar words in their dataset. The main limitation of this study is that we cannot exactly consider content to be cyberbullying/ not cyberbullying solely based on their positive and negative sentiments.

In Ali & Syed (2020) authors have incorporated sarcastic elements in detecting cyberbullies which were not discussed in most of the previously done research in this area. For sarcasm detection, they have used various features such as symbols: '!' and '?', repeated letters, intense adjectives, capital letters, etc. They have used several machine learning classifiers and an ensemble approach where SVM and soft voting ensemble classifiers have given better results. However, in this research, only textual features were considered where contextual, and network were ignored.

Deepasree Varma et al. (2022) investigated hate speech detection in code-mixed English and Malayalam text using the BERT language model for embedding. The study employed DistilBERT, a smaller version of BERT, to extract features and classify tweets as hate or non-hate. The authors trained various machine learning models, including Logistic Regression, SVM, and Naive Bayes, using the BERT embeddings. They achieved an accuracy of 90.5% for the code-mixed dataset using Logistic Regression with Grid Search.

Patil et al. (2023) focus on enhancing the accuracy of hate speech detection by using a deep learning model called Recurrent Convolutional Neural Network (RCNN). The study compares the performance of RCNN with a standard Long Short-Term Memory (LSTM) model on a dataset of tweets labeled as hate speech or non-hate speech. The authors demonstrate that the RCNN model surpasses the LSTM model in terms of accuracy, precision, recall, and F1-score, achieving an accuracy of approximately 92%. This improved performance is attributed to the RCNN's ability to capture both the long-term dependencies and local features of the text, which are crucial for understanding the nuances of hate speech.

Although many studies can be found in the sentiment analysis domain, only a few can be found for sentiment analysis-based hate speech detection, specifically for Sinhala and Romanized Sinhala. However, there is limited research addressing Sinhala and Romanized Sinhala code-mixed content.

2.4.2 Related work for the Sinhala and Romanized Sinhala languages

Though many studies have been done on hate speech detection for English and other languages. There were only few studies were related to the Sinhala language along with Romanized Sinhala (Singlish). Some of the studies done for the Sinhala and Romanized Sinhala Language related to hate speech detection are discussed below.

Dias et al. (2018) conducted a study to determine racist and non-racist content in Sinhala by training a SVM model and they have received a precision of 100% and an accuracy of 70.8%. Data were collected only from Facebook and have only considered racist comments

which can be considered as a subset of hate speech cyberbullying context.

In the research done by Smith & Thayasivam (2019) to detect Sinhala and English code mixed content they have used word n-grams, character n-grams, and BoW as features to train deep neural networks, CNN, recurrent neural networks machine learning models, where XGM model has given the best results compared with the other models with an accuracy of 92.1% with bi-gram features. However, this study was conducted only to detect Sinhala-English Code-mixed language. (Ex- ”ඈ ම shopping යනවා” - English and Sinhala Unicode mixed data)

In Jayasuriya et al. (2020) have used both machine learning, lexical-based approaches, and an ensemble classifier with a combination of both. For the lexical-based approach, they have created a subjective lexicon using the Sinhala-English dictionary and using the Levenshtein ratio analysis has given the best results. For the ML-based approach, they have used n-grams with unigrams, bigrams, and trigrams for feature extraction and used several classifiers to train the model where they observed unigrams are most effective in terms of feature extraction and logistic regression with stop word removal and unigrams outperformed other models. Their hybrid approach has provided the best results out of all the approaches.

In Ishara Amali & Jayalal (2020), authors have developed a rule-based feature extraction model for the classification of cyberbullying for social media comments in Sinhala. Their source of data was Twitter and have created rules like having a combination of the first-person pronoun, bad word, and a second-person pronoun, percentage of bad words in tweets, etc. as features for their classification model. They have labelled by giving four options as very non-cyberbullying, non-cyberbullying, cyberbullying, and cyberbullying where they have finally calculated the weight of them to get the final label. As for machine learning, the approach with the SVM classifier has given the highest F1 score and they have also observed that the larger the dataset F1 score increases as well, when the dataset was larger than 600 the accuracy was around 93% while when the dataset was lesser than 200 it was around 85%.

Samarasinghe et al. (2020) utilized deep learning techniques to detect hate speech in the Sinhala language. They have used convolutional neural networks (CNN) where a) will determine if the text contains hate speech or not and b) detected hate speech will be categorized into multiple levels based on the hate level. According to their study, CNN has shown higher accuracy than the SVM model when detecting hate speech and during hate-level classification, SVM has surpassed CNN performance. Their major limitation is the dataset is too small and since it was collected from gossip sites mainly, it was biased towards politics and racism hence hate speech due to other areas like disability, and sexual orientation was ignored. And they have suggested in the study that it may improve results if they have used part of speech tagging (POS) rather than using complete sentences to train the models.

Another research is done to detect abusive comments in Sinhala (Sandaruwan et al. 2020) trained three machine learning models Multinomial Naïve Bayes (MNB), Support Vector Machine (SVM) and, Random Forest Decision Tree (RFDT) using a Sinhala comment corpus consist of 2000 evenly distributed offensive and neutral comments. They have done feature extraction using a bag of words (BoW), word skip-gram model, character n-gram model, and word n-gram models. After training each model testing has been done using

200 comments where MNB has shown the highest performance with 96.5% accuracy and 96% for recall. In addition to that they have also used two lexicon-based approaches called the cross-lingual lexicon approach and corpus-based lexicon approach and the corpus-based lexicon approach has given the highest results with an accuracy of 90.5%. Here also they have only considered features like word n-grams and character n-grams while micro features like the word count of sentences, and POS tagging were not considered.

In Hettiarachchi et al. (2020), here they have manually annotated a dataset with labels and no proper method of labelling has been used. For example, one person can view a certain comment as hate speech while someone can label it otherwise. In our approach, we will be getting the assistance of different people with sound knowledge of the Sinhala language to label the dataset. Here they have used a hybrid approach and have selected the best-performing classifier where MNB with TF-IDF vectorizer(bigrams) have outperformed others.

The research conducted by Liyanage & Jayakumar (2021) has trained several models to detect Sinhala - English code-mixed language. Sinhala - English code-mixed language can be of two types.

1. Sinhala words written in English letters (Romanized English) i.e., ”Mama gedara yanawa.”

2. English and Sinhala Unicode mixed language i.e., ”Mama church යනවා”

They have created a Sinhala - English code-mixed hate and non-hate comment dataset collected from Facebook and YouTube, and they have trained eight machine learning models and three ensemble models (two voting classifiers using hard and soft voting methods and a stacking classifier using SVM, MNB, Ada boost classifiers as base classifiers) to detect hate speech in Singlish and has evaluated against precision, F1 score, accuracy, and recall. The hard voting classifier has outperformed all the other models by giving an accuracy of 84%. During feature extraction, they only considered the word n-grams.

Fernando et al. (2022) tackled the issue of hate speech in Sinhala social media using machine learning and deep learning. They preprocessed text data and used various methods to extract meaningful features, including Bag-of-Words, TF-IDF, and word embeddings. They trained various models, including Logistic Regression, Naïve Bayes, SVM, and deep learning models like CNN, RNN, and LSTM. Their best-performing model was an RNN with FastText embeddings, achieving 70% accuracy. This research highlights the potential of deep learning for Sinhala hate speech detection.

Muthuthanthri & Smith (2024) conducted research on hate speech detection for transliterated English and Sinhala code-mixed data, which they called ”Singlish”. The study focused on transliterated English and Sinhala code-mixed data from Facebook, carefully annotated to maintain dataset accuracy. The data underwent preprocessing and feature extraction, with techniques like Term Frequency (TF) - Inverse Document Frequency (IDF) and fastText Word Embeddings applied to grasp the complexities of the mixed-code language. Various models were evaluated, ranging from conventional ones like Logistic Regression and Support Vector Machine (SVM) to advanced architectures like Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM). The study also incorporated gradient-boosting frameworks and transformer-based

models like BERT and GPT2. The effectiveness of these models was rigorously assessed using metrics like accuracy, precision, recall, F1-score, confusion matrices, and Receiver Operating Characteristic area under curve value (ROC AUC) values. BERT stood out, achieving 82% accuracy and a 90% ROC AUC value, proving highly effective for this detection task.

2.5 Summary

A summary of Related work for English, Sinhala and Romanized Sinhala and Other languages used in the literature is provided in Table 2.2.

Table 2.2: Summary of Literature Surveys

Title	Authors	Year	Description	Gap
Related work for English and Other languages				
Automated Hate Speech Detection and the Problem of Offensive Language	Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber	2017	Explored the distinction between hate speech and offensive language on Twitter using crowdsourced data and a multiclass classifier.	Focused on Twitter data; generalizability to other platforms is unclear. Future research could explore the impact of context and user demographics on hate speech classification.
Deep Learning for Hate Speech Detection in Tweets	Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma	2017	Investigated deep learning for hate speech detection in English tweets, classifying them as racist, sexist, or neither, using CNNs and LSTMs to learn semantic word embeddings.	Focused on detecting hate speech but did not explore identifying specific target groups or the severity of hate speech. Future work could incorporate target group identification.
A Multilingual System for Cyberbullying Detection: Arabic Content Detection using Machine Learning	Batoul Haidar, Maroun Chamoun	2017	ML models have been trained to detect Arabic Cyberbullying content	Trained only SVM and Naive Bayes to detect cyberbullying in Arabic content.
Cyberbullying detection in hindi-english code-mixed language using sentiment classification	Jethanandani Tarwani, Manan Kanth	2019	Done for a romanized hindi dataset and has trained 5 models where SVM has produced the best results	Has decided if the content is cyberbullying or not solely based on positive and negative sentiment.
Cyberbullying detection using machine learning	Aaminah Ali, Adeel Sayeed	2020	Sarcasm element is considered when training ML models	Have only considered textual features.

Continued on next page

Table 2.2 – continued from previous page

Title	Authors	Year	Description	Gap
Hate Speech detection in English and Malayalam Code-Mixed Text using BERT embedding	Deepasree Varma P, Dr Vinod P, Dr M Nandakumar, Akhil Madhu, Akshay K	2022	Investigated hate speech detection in code-mixed English and Malayalam text using the BERT language model for embedding. They achieved an accuracy of 90.5% for the code-mixed dataset using Logistic Regression with Grid Search.	The research focused on binary classification (hate or non-hate) but did not explore the detection of different types of hate speech, such as racism, sexism, or homophobia. Future work could extend the model to multi-class classification and explore the use of other transformer-based models.
Hate Speech Detection using Deep Learning and Text Analysis	Prachi Patil, Sakshi Raul, Dhanisha Raut, Dr. Tatwadashi Nagarhalli	2023	Proposed a deep learning model for hate speech detection using a RCNN. The study found that the RCNN model outperforms the LSTM model in terms of accuracy, precision, recall, and F1-score.	The research focused on binary classification (hate speech or non-hate speech) and did not explore the detection of different types of hate speech or the severity of hate speech. Future work could extend the model to multi-class classification and explore the use of other deep learning models.
Related work for Sinhala and Singlish				
Identifying racist social media comments in sinhala language using text analytics models with machine learning	Dulan Dias, Madushi Welikala	2018	Conducted the study to determine racist and non-racist content in Sinhala by training SVMs	Data were collected only from Facebook and have only considered racist comments which can be considered as a subset of hate speech/ cyberbullying context.
Language Detection in Sinhala-English Code-mixed Data	Uthayasanker Thayasivam, Ian Smith	2019	Have used word n-grams, character n-grams, and BoW as features to train deep neural networks, CNN, recurrent neural networks ML models	Conducted only to detect Sinhala-English Code-mixed language
Sentiment classification of Sinhala content in social media	Pradeep Jayasuriya, Sarith Ekanayake	2020	Have used both machine learning and lexicon approached and a hybrid approach where latter has produced best results	The dataset is only limited to Sinhala and YouTube content

Continued on next page

Table 2.2 – continued from previous page

Title	Authors	Year	Description	Gap
Classification of cyberbullying Sinhala language comments on social media	Shantha Jayalal. Amali Ishara	2020	Have developed a rule-based feature extraction model for the classification of cyberbullying for social media comments in Sinhala and SVM has produced best results	Dataset is too small which is around 200 and twitter is the only source of data
Machine learning approach for the detection of hate speech in sinhala unicode	SWAMD Samarasinghe, RGN Meegama	2020	Have utilized deep learning techniques to detect hate speech in the Sinhala language	Dataset is too smaller and was collected from gossip sites mainly where it is biased toward racism and politics
Identification of abusive sinhala comments in social media using text mining and machine learning techniques	Supun Tharaka Sandaruwan	2020	Done to detect abusive comments in Sinhala where MNB, SVM, RFDT models were trained	Features like word count and POS tagging were ignored
Detecting hate speech in social media articles in romanized sinhala	Nimali Hettiarachchi, Ruvan Weerasekera	2020	Have used a hybrid approach where MNB with TF-IDF vectorizer(bi-grams) have outperformed others	The dataset is only limited to Facebook and no proper data annotation methods have been used. Only textual features have been considered.
Hate Speech Detection in Sinhala-English Code-Mixed Language	Oshadhi Liyanage, Krishnakripa Jayakumar	2021	Created a Sinhala - English code-mixed hate and non-hate comment dataset to retrain 8 different ML Models where ensemble approach with Hard voting classifier has produced best results	Only textual features were considered
Sinhala Hate Speech Detection in Social Media Using Machine Learning and Deep Learning	W.S.S. Fernando, Ruvan Weerasinghe, E.R.A.D. Bandara	2022	This study explores hate speech detection in Sinhala social media using various machine learning and deep learning models, including RNN, LSTM, and CNN, with FastText word embeddings. The RNN with FastText achieved the highest accuracy of 70%.	The research focuses on detecting hate speech but does not identify the target groups of the hate speech, which could be valuable information for mitigation strategies. Future research could incorporate target group identification and explore the use of more advanced deep learning models like Transformers.

Continued on next page

Table 2.2 – continued from previous page

Title	Authors	Year	Description	Gap
Hate Speech Detection for Transliterated English and Sinhala Code-Mixed Data	Meuru Muthuthanthri and Roy Ian Smith	2024	This study explores hate speech detection in Sinhala-English code-mixed data from Facebook, using TF-IDF, fastText, and various machine learning models including BERT and GPT2. BERT achieved 82% accuracy and a 90% ROC AUC value.	The research primarily focuses on Facebook data, leaving a gap in understanding hate speech detection in other platforms like Twitter and YouTube. Future work could explore these platforms and address transliteration-induced spelling variations, symbol-based evasions, and the integration of emojis and symbols into detection models.

All of the studies mentioned above focused on improving the detection of hate speech in various ways. If we consider the accuracy of the detection of Sinhala hate speech, there is potential for improvement using deep learning techniques. In addition to that, many studies say that symbols such as emojis play a critical role in extracting the semantic meaning from a text phrase. Thus, this research will be focused on improving the accuracy of hate speech detection in Sinhala and Romanized Sinhala by considering semantic meanings of symbols such as emojis.

2.6 Chapter Summary

There are three types of hate speech detection techniques: rule-based, machine learning, and deep learning. Prior studies on the hate speech detection in Sinhala and other languages have investigated a number of methods, such as SVM, deep neural networks and CNN. By creating a hybrid CNN-RNN model for detecting hate speech in code-mixed text with emojis in both Sinhala and Romanized Sinhala, this study seeks to expand on previous research.

CHAPTER 3

METHODOLOGY

3.1 Chapter Overview

The methodology used in this study is described in this chapter. The data collection process, including manual data collection and integration of existing data sets, is detailed. The data annotation process, data preprocessing techniques, functional engineering methods, and the development of the CNN-RNN hybrid model are also described in the chapter. In addition, hyperparameter tuning, model training, and assessment strategies are discussed.

3.2 Research High-level design

The high level design for the hate speech detection is structured into five major phases: data acquisition, preprocessing, model development, training, and evaluation, all adapted to the unique characteristics of Sinhala data and the challenges of the detection of hate speech in low-resource settings. The main classification model that was developed and evaluated in this study is a combination of **Convolutional Neural Network(CNN) and Bidirectional Long Term Short Term Memory (BiLSTM)** named **CNN-BiLSTM** model. An independent bidirectional long-term memory network (BiLSTM) was involved in the initial research, but the methodology evolved to focus on CNN-BiLSTM architectures to exploit the strength of the convolutionary and recurrent layers to optimize the performance of Sinhala hate speech detection. Each phase will be described in this chapter, along with the adaptations and considerations necessary to efficiently process the text of Sinhala and Romanized Sinhala with emojis. In addition to that, robust models for this task will be highlighted. The key components of the research design are illustrated in Figure 3.1.

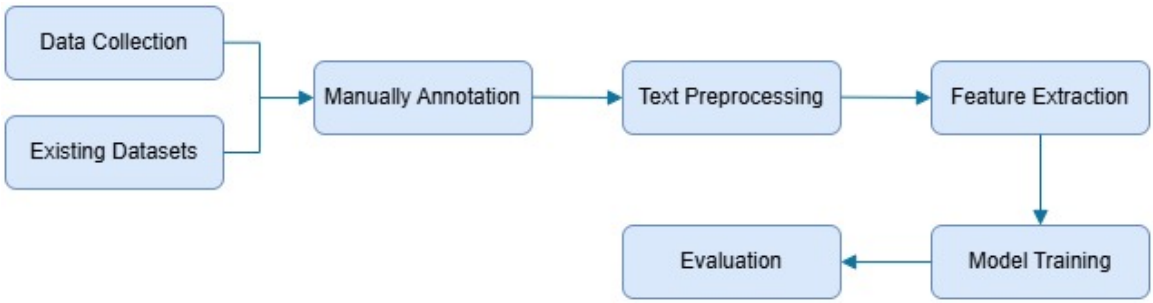


Figure 3.1: High-level design of research methodology

3.3 Data Collection

The dynamic evolution of language in the context of social media was recognised in this study, and the inherent challenge of failure of data sets in the detection of hatred was addressed. Rapid linguistic innovations are characterized by modern social media platforms in which new vocabulary representations, sign languages and evolving communication styles are constantly introduced by users. As a result, reduced effectiveness may be experienced by existing datasets when applied to real-time hate speech detection scenarios, which necessitates the creation of a current and relevant data set.

Furthermore, a critical gap in the public availability of datasets specifically adapted for Sinhala language content has been identified, especially covering Unicode and Romanized Sinhala scripts, along with the nuanced expression transmitted through emojis. To overcome this limitation and ensure the relevance and representativeness of the dataset, a two-sided approach to data acquisition was adopted, which included both manual data collection and the integration of existing datasets as described in Figure 3.2.

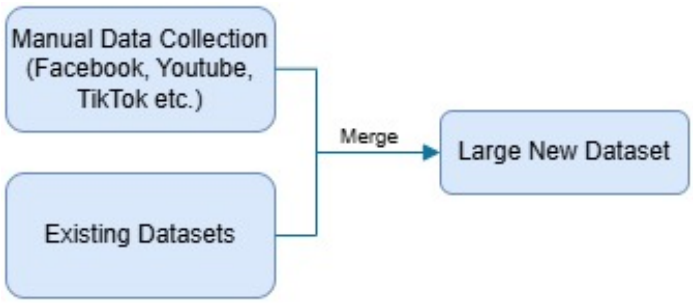


Figure 3.2: Dataset creation

3.3.1 Manual Data Collection

Due to constraints imposed on privacy policies on social media platforms and constraints on automated scraping tools and APIs for access to user-generated content on a large scale, manual data collection methods have been adopted. This approach is resource intensive but considered essential to capture the latest linguistic trends and contextual nuances of online Sinhala and mixed language communications.

The data were meticulously collected from prominent social media platforms, particularly Facebook, TikTok, and YouTube. These platforms were selected because they are widespread in Sri Lanka and make an important contribution to the online discourse in which hate speech manifests itself. Manual data collection allowed a targeted approach, focusing on comments sections and public forums where hate speech is more prevalent despite Figure 3.3 (Kemp 2024). This methodology also allowed the inclusion of data that reflect current events and trending topics, thus ensuring the temporal relevance of the data set.

Figure 3.3: Social Media platforms usage in Sri Lanka (Source: https://datareportal.com/)

3.3.2 Integration of Existing Datasets

In order to increase manually collected data and create a robust and comprehensive resource, annotated data from three existing hate speech datasets were incorporated. The combination of these datasets has helped increase the total size of the data set and potentially increase the linguistic and contextual diversity represented Ziyaden et al. (2024). The specific datasets used were: Jayasuriya (2020) and Weerathunga (2020). Refer to Appendix A for a comparison of dataset analysis.

Before integration, these datasets have been carefully examined to ensure compatibility with annotation schemes and data formats. When necessary, data transformation and label harmonization have been performed to create unified data sets suitable for modeling training.

This study is aimed to combine the collection of manual data focusing on the capture of contemporary online languages with the inclusion of existing data sets and to establish comprehensive and representative data sets. These datasets can be supported the development of effective and robust hate speech detection models in dynamic social media environments, especially in the context of Sinhala and Romanized Sinhala.

3.4 Data Annotation/ Data Labelling

As explained in Section 2.4 Sinhala is a morphologically rich language that has many hyponyms and synonyms for one word. When it comes to Romanized Sinhala, it would be rather difficult with the different types of spelling variations available. In addition to that, there could be different kinds of trends going on that changes the pronunciations of words as well (i.e. There is a trend going on by adding ”thee” sound to the end of questions - ”Kamuthee”, ”Karamuthee” etc.) Therefore, the manually annotation is performed on the dataset using crowd-sourcing to avoid such scenarios. Class labels that are used for the annotation process are depicted in Table 3.1.

Weights	Description
1	Offensive
0	Non-Offensive

Table 3.1: Data Annotation

These labels will be assigned to the comments that were collected from various social media platforms as depicted in Table 3.2.

Comment	Label
සෙල්ලම් baduwa aran dunne naane අද	Non-Offensive
Tho ne ගොන් baduwa	Offensive
U nam හෙන ponnayek	Offensive
Hena hathama ගහපන් ponnaya 😑	Offensive

Table 3.2: Examples for Labelled Dataset

In order to enrich the text data with emoji emotions, a weighted Gazetteer file was created. A list of emojis and a corresponding weight (of 10) indicating the potential of the emoji to contribute to hate speech is contained in this file. It is indicated that a weight of 10 signifies a strong correlation with the speech of hatred when used in a sentence. The weights were allocated taking into account the density of emoji use in sentences classified as hate speech, with a focus on sentences that contain at least one emoji. The formula has been used to calculate the weight of the emoji is shown in Equation (3.1).

The final emoji weight calculated using the formula that shown in Equation (3.1) was rounded to the nearest integer. This rounding ensures that emojis are grouped into a few distinct weight categories, allowing for more effective categorization and analysis. By doing so, it becomes easier to identify and interpret the role of emojis in conveying hate speech, especially when they share similar emotional intensities.

$$\text{Emoji Weight} = \left(\frac{\text{EOHS}}{\text{TOE}}\right) \times \left(\frac{\text{THS}_{\text{emoji}}}{\text{TAS}_{\text{emoji}}}\right) \times 10 \tag{3.1}$$

Where:

- EOHS = Emoji Occurrences in Hate Speech Sentences
- TOE = Total Occurrences of Emoji in Dataset
- $\text{THS}_{\text{emoji}}$ = Total Hate Speech Sentences with at least one emoji
- $\text{TAS}_{\text{emoji}}$ = Total Annotated Sentences with at least one emoji

For example, for the emoji 😏:

- EOHS 😏 = 90 (The emoji 😏 appears 90 times in hate speech sentences)
- TOE 😏 = 100 (The emoji 😏 appears 100 times in the entire dataset)
- $\text{THS}_{\text{emoji}}$ = 400 (There are 400 sentences annotated as hate speech that contain at least one emoji)
- $\text{TAS}_{\text{emoji}}$ = 600 (There are 600 total annotated sentences that contain at least one emoji)

Then, the Emoji Weight for "😏" calculated in Equation (3.2). A sample of extracted emojis and related calculated weights are shown in Tables 3.3 and 3.4.

$$\text{Emoji Weight (😏)} = \left(\frac{90}{100}\right) \times \left(\frac{400}{600}\right) \times 10 = 6 \tag{3.2}$$

Emoji	Weight
😤	8
😑	6.43
😈	6.85
😏	6

Table 3.3: Emojis with assigned weights

Emoji	Weight
💔	2.35
👉 👈	3.1
🍆	6.9
🍑	7

Table 3.4: Emojis with assigned weights

3.5 Data Pre-Processing

To prepare the collected Sinhala and Romanized Sinhala text data, including emojis, for effective analysis with our primary CNN-BiLSTM combined model, a carefully designed preprocessing pipeline has been implemented. The goal of this pipeline is to transform Sinhala comments into formats that highlight meaningful patterns for the detection of hate speech in Sinhala while minimizing noise and irrelevant information. The preprocessing strategy consists of a series of steps that are intentionally applied to ensure optimal text refinement and feature extraction specifically for the Sinhala and Romanized Sinhala text and to maximize the performance of the CNN-BiLSTM model. The flow of data pre-processing is illustrated in Figure 3.4.

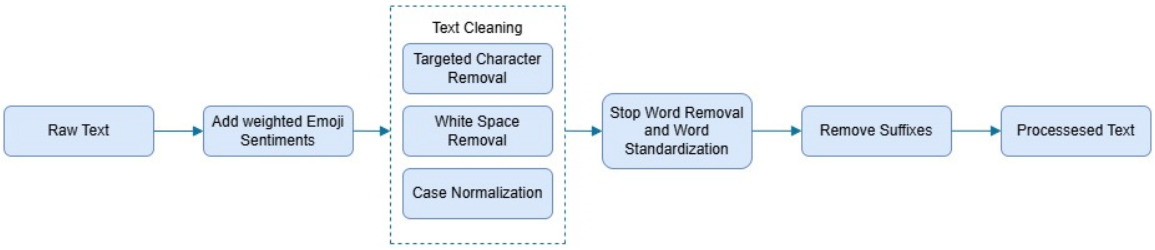


Figure 3.4: Data Preprocessing Flow

3.5.1 Enriching Text with Weighted Emoji Sentiment

Recognizing that emojis are far from simply decorations in online communication, often with significant emotional weight and contextual hints. As an Example: "oya කරලා thiyenne hondata 🥰" vs "oya කරලා thiyenne hondata 😡" The data preprocessing stage began by addressing these visual elements. Instead of simply throwing away emojis (a common approach that risks losing valuable sentiment information) this study adopted a strategy

to embed their feelings directly into the text. This was achieved by using a custom-built lexicon "emoji_weights.csv". This lexicon assigned a numerical sentiment weight to each emoji, reflecting its perceived emotional intensity (Alfreihat et al. 2024). Sample emoji weights are depicted in Section 3.4.

During the preprocessing, each emoji encountered in a comment was replaced by a unique token that can be read by humans. This token, formatted as __EMOJI_WEIGHT_-[weight]__, effectively transcribed the sentimental weight of the emoji into text data itself. For example, if an 😂 emoji is associated with a weight of 2.0, it will be converted to __-EMOJI_WEIGHT_2_0__. This initial step ensured that the feeling conveyed by the emojis was not lost, but rather became an integral part of the text representation, ready for the model to learn. This subtle approach recognizes the evolving nature of online communication and the critical role of non-verbal signs such as emojis in conveying meaning (Alfreihat et al. 2024).

Category	Emojis
Anger / Disgust	🤯 😡 😠 😤 🤢 🤮 😐 🥶 😖 😈 👹 👺
Animals	🐕 🦮 🐖 🐖 🐎 🐄 🐄 🐂 🐃 🐵 🐒 🙈 🐑 🐸
Inanimate Things	💩 🥾 👠
Disrespect / Dislike	🖕 👎 🙄 😒
Violence / Threat	✊ 🔪 💣 🔥
Sexual / Inappropriate	💦 🍆 🍑 🔥 😏 😉 😈 👅 👄 👙 🩲 🚿 🛌

Table 3.5: Common Offensive Emojis

3.5.2 Text Cleaning: Laying a Solid Foundation

After incorporation of emoji sentimental information, the next phase was focused on systematically cleaning text content to eliminate elements that might hinder the learning process of the model. This cleaning process was the basic step to ensure that the text was as easily streamlined and concentrated as possible before further processing. Cleaning activities are included:

- **Target Character Removal:** To focus on analysis of linguistic content related to the communication of Sinhala and Romanized Sinhala, characters outside the alphanumeric range of these languages have been systematically removed. This process eliminated characters from other scripts that were considered less relevant for the detection of hate speech in this particular context. Regular expressions have been used to perform this selective removal efficiently and accurately.

- **Whitespace convergence:** Changing the whitespace between words or extra spaces at the beginning and end of comments can create inconsistencies. In order to standardize text formatting, all instances of multiple whitespace characters were arranged

into single spaces, and the whitespaces leading/dragging were trimmed. This standardization ensures cleaner and consistent text input in subsequent steps.

- **Case normalization:** To reduce the complexity of vocabulary and ensure that models treat semantically identical words, all text is converted to lowercase. This step effectively divides capitalization variations such as ”Hate” and ”hate” into a single representation and simplifies the model learning task (Zhang et al. 2022).

3.5.3 Tokenization: Breaking Down Text for Analysis with Contextual Awareness

Once the text is cleaned up and prepared, the next critical step is tokenization, which decomposes the continuous text into a single unit (token) that the model can process. In order to capture both the meaning of individual words and the crucial context provided by word sequences, a specialized tokenization strategy was adopted that combined unigrams and bigrams (Katona et al. 2021).The combination of unigrams and largerams allowed the model to take into account the meaning of individual words and the contextual relationship between words, providing a richer input representation.

3.5.4 Refining Token Sets: Stop Word Removal and Word Standardization

After tokenization, the token set was further refined through two important processes: ending the deletion of words and word standardization. These steps are designed to focus the attention of the model on the most informative words and reduce redundancy in the vocabulary.

Strategic removal of stop words

Stop words are common words in a language, but often semantically light, can dilute the signal of richer words. To mitigate this situation, a two-tier approach to halt the removal of words has been adopted. Custom Stop Word lists loaded from StopWords_425.txt were incorporated. This custom list was specially curated to include words such as ”මේ”, ”එක”, and ”වගේ”.

In the Table 3.6, the most common terms were chosen, and some words were removed from the list because they changed the polarity of a phrase (LTRL n.d.).

Word standardized for consistency

variations in word forms, such as different spelling and orientations, can lead to unnecessary complexity and data sparsity. In order to address this, a word standardization process was used, using word_variations.csv files. This file acts as a lexicon mapping the spelling and spelling forms of words to a single standardized base form. This approach will be improved the generalization of the model by treating the different surface forms of the same word as equivalent (Jahjah et al. 2016). Sample word variations that included in word_variations.csv file is depicted in Table 3.7.

Removed words from stop word list		
නොමැති	නැත	නැහැ
නෑ	නිසා	නොහැකි
නොව	බැහැ	තමයි
හොඳ	වුනා	ඉතින්

Table 3.6: Removed Words from Stop Word List

Word	**Variations**
walaththaya	wal | wala | walattaya | walaa
wassa	wassa | wassek | gon wassa
wesi	weesi | weasi | wesige | wesey | wesa | wesiyo | vesi | vsigput | wsgpthata
yako	yako | yakoo | yaki | yaku | yakuu
වේසී	බේසික් | බේසියොන්ට | වේසියෝ
පුක	පුක | පුකේ | පු#කේ | පුනෑ | පු# දිදී | පුකක් | පුකට | පු# | පුප | පු⋆ | හිකෙන් පුනා | පුප

Table 3.7: Sinhala word variations

3.5.5 Sinhala-Specific Suffix Removal

Given the potential existence of morphologically rich Sinhala text, the preprocessing pipeline included a step of removing suffixes. This was motivated by the understanding that Sinhala words often derive different meanings or grammatical functions by suffixes (Sandaruwan et al. 2020). A list of Sinhala consonants was loaded from ”Suffixes-413.txt”. When removing suffixes, it needs to be ensured that the remaining ”root” words contained at least two Sinhala characters after the suffix was removed. This limitation was introduced to prevent excessive aggression and result in meaning loss and the creation of non-verbal meanings, which are particularly important in a morphologically rich language such as Sinhala. This step focuses on the root of the word, aimed at reducing lexical variations and potentially improving the ability of the model to recognize semantic relationships in the text of Sinhala. Some sample suffixes are listed in Table 3.8.

ක	ය	ක්	යක	කක
කක්	යක්	යකි	කකට	යකගෙන්

Table 3.8: Sample set of sinhala suffixes

3.5.6 Handling Class Imbalance

To resolve the imbalanced data set, random oversampling was used to manipulate the data (Riyadi et al. 2024). It involved the random duplication of data from the minority class. Resampling with different parameters was adopted to ensure the desired class balance. These methods have often been used to deal with imbalanced data sets in machine learning, despite their potential shortcomings, such as information loss and over-adaptation. Figure 3.5 shows the oversampling technique to balance the data set.

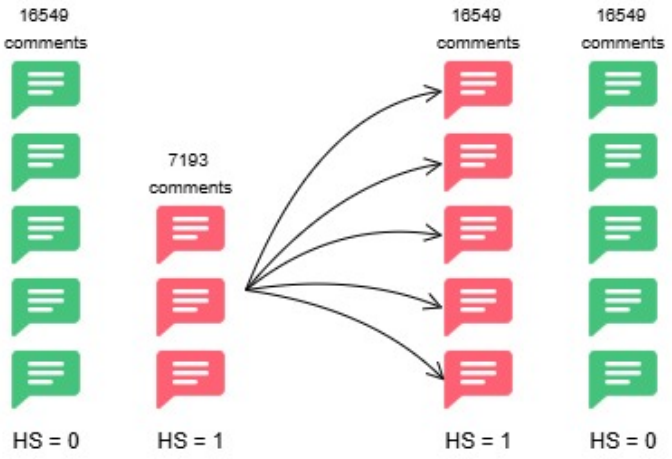


Figure 3.5: Illustration of Oversampling

3.6 Feature Extraction: Transforming Text into Meaningful Representations

Feature extraction is a key step in natural language processing (NLP) tasks and serves as a bridge between raw text data and machine learning models. This includes the conversion of text into a numerical format that captures the necessary information relevant to the task. In the context of hate speech detection, effective feature extraction of characteristics is essential to distinguish models from subtle patterns and linguistic signals indicating abusive content. There are several ways to extract features, each with its strengths and limitations. This section will discuss common methods and then justify the method selected for this research.

3.6.1 Traditional Feature Extraction Methods in Text Classification

Before the advent of deep learning, machine learning models for text classification were largely based on traditional feature extraction techniques. These methods often include the creation of sparse high-dimensional vector representations of texts based on word count or frequency. Among the most well-known traditional methods are:

- **Bag-of-Words (BoW):** This is probably the simplest and most fundamental technique. BoW represents a document as an unevenly assembled collection of words,

without considering grammar and word order but tracking word frequency. Each document is vectorized on the basis of the number of words in a predefined vocabulary. Although simple to implement, BoW has limitations such as ignoring word order and semantic relationships, leading to very high and scarce features vectors (Smith & Thayasivam 2019, Fernando et al. 2022).

- **Term Frequency-Inverse Document Frequency (TF-IDF):** The TF-IDF optimizes the Bag-of-Words approach by weighting words according to their importance in the document and in the entire corpus. It increases the weight of the words that are common in a particular document but that are rare in the entire corpus, effectively highlighting the terms that are unique to a particular document. TF-IDF is effective in many text classification tasks and addresses some of the limitations of BoW by emphasizing the importance of terms. However, unlike BoW, it still does not capture the order of words or deeper semantic meanings (Ranasinghe et al. 2024, Fernando et al. 2022).

- **N-grams:** N-grams go beyond a single word (Unigram) to consider a sequence of n consecutive words (e.g. Bigram, Trigram) as characteristics. This method partially addresses the limit of BoW and TF-IDF word order by incorporating local word contexts. For example, the word ”not good” as a bigram has a different meaning than simply considering ”not” and ”good” as a unigram. Although n-grams improve context awareness, they can still lead to high dimension and rarity, and do not fully capture long-term dependency and semantic similarities between words (Liyanage & Jayakumar 2021).

3.6.2 Word Embeddings: Capturing Semantic Meaning in Dense Vectors

Unlike traditional methods, word embedded offers a dense and semantically rich method of feature extraction (Malik et al. 2022). Word embeddings represent words as a low-dimensional continuous vector in vector spaces, and semantically similar words are placed closer together. These embedded programs are usually learned from large text files using techniques such as Word2Vec, GloVe, or FastText. The main advantages of word embedding compared to traditional methods are:

- **Semantic Representation:** Word embedding captures semantic relationships between words. Similar meaning words have a similar vector representation, allowing models to understand similarity of words, which is essential for nuanced tasks such as hate speech detection, where subtle semantic variations can be significant.

- **Dense and low-dimensional vectors:** Compared to low-dimensional vectors from BoW, TF-IDF, and even n-grams, the word embedding is dense and has a lower dimension (e.g., 100-300 dimensions). This reduces computational complexity and can improve the generalization of models, especially in limited data sets.

- **Contextual information (to some extent):** While basic word embeddings, such as Word2Vec and GloVe, are context-dependent (a word has the same embedding regardless of the context), they still encode distribution semantic information derived from the contexts in which words usually appear in the training body.

3.6.3 Embedding Layer for Feature Extraction in Neural Network Models

This research utilizes word embeddings through an Embedding layer in proposed CNN-BiLSTM and other neural network models for Sinhala and Romanized Sinhala hate speech detection. Embedding layers are a standard technique in neural NLP for feature extraction, which transforms discrete tokens into continuous vector spaces. This layer extracts features by the following approaches. Related input parameters are captured using the hyper parameter tuning approach described in Section 3.7.2.

- **Vocabulary Lookup (Sinhala):** The Embedding layer uses an internal search table built from Sinhala and Romanized Sinhala texts to learn and store a unique vector for each Sinhala token in this vocabulary. Vocabulary-based search is a fundamental step in the processing of text for neural networks.
- **Dense Vector Representation (Sinhala):** Each Sinhala token (word, emoji) is assigned to a dense vector that captures semantic features more effectively than sparse one-hot representations, which is particularly beneficial for the Sinhala language with its nuances and morphology. Dense vector representations are essential to capture semantic relationships.
- **Sequence Input Handling (Sinhala Comments):** The embedded layer obtains the pre-acquired vector for each token index in the Sinhala comment sequence, allowing a consistent input process for neural network architectures. The handling of variable-length sequences by padding and fixed input lengths is a common practice in deep-text learning.
- **Learnable and Task-Tuned Embedding (Sinhala Hate Speech):** The Embedding for Sinhala tokens is learned during CNN-BiLSTM and training of neural network models. These embeddings, which are randomly initialized, are adjusted through backpropagation to minimize the loss function in the Sinhala hate speech detection. This task-specific learning creates embedded materials optimized for discriminating against Sinhala hate speech, which are expected to be superior to fixed or generic embedded materials, especially for nuanced Sinhala language and low-resource scenarios (Alkasassbeh et al. 2024). Learning task-specific embeddings allows the model to focus on the most relevant features for a particular classification task.

By incorporating the Embedding layer as the initial layer into our CNN-BiLSTM, BiLSTM and LSTM models, this study directly integrates the extraction of characteristics into the model training process. This end-to-end learning approach enables models to learn the best vector representations of Sinhala words and emojis, representations that are specifically designed for the task of finding hatred words in Sinhala. Furthermore, the density

and semantic richness of learned word insertions are expected to be especially beneficial in dealing with the complexities of Romanized Sinhala and the use of emoji within the text.

3.7 Model Training

The training process of machine learning models is essential to achieve the objective tasks precisely. In this study, the training phase included several key steps: defining the model architecture, optimizing its settings by high-parameter tuning, and finally training models on prepared data sets. This section describes these steps in detail and explains the choices made and their rationale.

3.7.1 Model Architecture: CNN-BiLSTM Combined Network for Hate Speech Detection (Primary Approach)

The primary model architecture for our Sinhala and Romanized Sinhala hate speech detection system is the combined Convolutional Neural Network - Bidirectional Long-Term Memory (CNN-BiLSTM). This architecture is specifically designed to effectively capture both local and sequential patterns in Sinhala text, which we would have assumed would be crucial to accurately identify hate speech, especially in a low-resource language context and with the complexity of the use of Romanized scripts and emoji. The combination of CNN and BiLSTM models were selected because they offer complementary strengths in the processing of text data, with CNNs gaining in the extraction of local features and BiLSTMs gaining long-term dependencies (Riyadi et al. 2024).

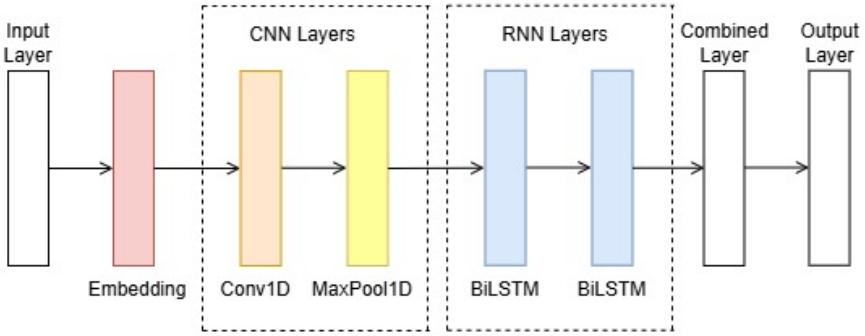


Figure 3.6: The Architecture of the proposed model

The CNN-BiLSTM model combines two main branches and processes the input Sinhala comment in parallel, and then integrates the learning representation into the final classification. Here is a simplified architecture breakdown:

1. Input and Embedding Layer (shared by both branches): The model starts with an Input Layer that accepts a symbolized and a Romanized Sinhala comment. This is followed immediately by an Embedding Layer. As described in detail in section 3.6.3, this layer transforms every Sinhala word or emoji into a dense vector representation, creating a rich, numerical ”code” that captures the semantic meaning of the Sinhala vocabulary. This embedding layer is shared by both CNN and BiLSTM branches, so that both branches begin with the same word representations.

2. CNN Branch (Local Feature Extraction): One branch of the model is the Convolutional Neural Network (CNN). CNN is effective in tasks such as text classification because it automatically learns hierarchical features and detects local patterns such as n-grams. This branch is intended to find important local patterns, such as the combination of words (n-grams) that represent hate speech in Sinhala texts. It consists of:

    - **Conv1D layer:** This layer applies convolutionary filters through word embeddings, looking for patterns in small windows of text. It learns to recognize the combination of specific words that are characteristic of offensive or non-offensive Sinhala comments.
    - **GlobalMaxPooling1D Layer:** This layer summarizes the most important features discovered by the Conv1D layer and selects the strongest signals associated with the local pattern within the Sinhala text by preserving the maximum activation of each feature map.

3. **Concatenation Layer (Feature Fusion):** After extracting local features from CNN branches and extracting sequential contexts from BiLSTM branches, the two types of information are combined by using the Concatenation Layer. This layer merges the outputs of both branches into a single, richer characteristic vector that represents both the local patterns and the broader context of the Sinhala text. Feature concatenation is a common strategy for combining information from different model components.

4. **Dense Layer (Classification):** Finally, Dense Layers (fully connected layers) are used to perform the classification based on the combined features:

    - Dense layer activation: This layer further processes the combined characteristics in a nonlinear way and learns a complex relationship. ReLU is a popular feature of deep learning activation.
    - Dropout Layer: Another dropout layer is used for regularization.
    - Output density layer (sigmoid): The final layer is a density layer with the sigmoid activation function. It takes the processed properties and outputs a probability between 0 and 1, indicating whether Sinhala comments are offensive or not. sigmoid function is the standard for binary classification tasks to generate probabilities.

The CNN-BiLSTM architecture combines local feature extraction from CNN with sequential context interpretation from BiLSTM to effectively detect hate speech in Sinhala and Romanized Sinhala texts, including emojis. Parallel branches enable models to capture various patterns in Sinhala data, thereby providing more comprehensive and robust classification, especially for the complexity of low-resource languages and complex tasks such as hate speech detection.

3.7.2 Hyperparameter Tuning with Keras Tuner

To find the best configuration for the BiLSTM model in the combined CNN-BiLSTM model, it is used a hyperparameter adjustment using Keras Tuner. Hyperparameters are models' settings that are not derived from data but set before training (e.g. LSTM layer number, graduation rate, learning rate). The search for the optimal combination of hyperparameters can significantly affect the performance of the model. We use the Hyperband Tuner optimization algorithm provided by Keras Tuner. Hyperband is an efficient hyperparameter optimization algorithm that adjusts resources to more promising hyperparameter configurations and accelerates the tuning process relative to simple grid search or random search.

The following key hyperparameters in the BiLSTM model define the search space:

- **embedding_dim:** The dimensions of the word embedded (64 to 256 steps, 32 steps). Larger dimensions can capture more complex semantic information, but also increase the complexity of the model.
- **lstm_units_1:** The number of units in the first layer of BiLSTM (32 to 128, in 32 steps). More units can learn more complicated patterns, but can also increase the risk of overload.
- **dropout_1:** Dropout rate after the first layer of BiLSTM (0.1 steps from 0.2 to 0.7). Higher dropout rates increase regularisation, but may lead to under-supply if it is set too high.
- **lstm_units_2:** The number of units in the second layer of BiLSTM (16 to 64, step 16).
- **dense_units:** The number of units of the first density layer (32 to 128 in the 32 steps).
- **dropout_2:** The dropout rate after the dense layer (from 0.2 to 0.7, 0.1 steps).
- **learning_rate:** Learning rate of Adam optimizer (elected from values 1e-2, 1e-3, 1e-4). The learning rate controls how much weight the model adjusts during each training step.

The Hyperband tuner systematically explores the different combinations of these hyperparameters, forms a model with each configuration and evaluates its performance on the validation data set. The objective='val_accuracy' is set to guide the tuner to find configurations that maximize accuracy on the validation set. The max_epochs=10 and factor=3 were set for the Hyperband algorithm to control the duration and resource allocation of the tuning process. Early stop is used during the adjustment process itself (with patience=3 to prevent validation loss) to prevent the waste of resources on poorly implemented configurations that were not improved.

3.7.3 Model Training with Best Hyperparameters and Early Stopping

After the hyperparameter tuning process, the best hyperparameter configuration discovered by the Keras Tuner was recovered. Then a new CNN-BiLSTM combined model was built using these best hyperparameters. This final model was trained on the entire training data set for a maximum of 20 periods, with a batch size of 64. The Adam optimizer is used with the best learning rate identified by the tuner and the binary loss function of entropy crossing, which is the standard for binary classification problems. Accuracy was used as an evaluation metric during training.

To further refine the training process and prevent overfitting during the final training phase, EarlyStopping callback is used. Keeping track of the validation loss, with a patience of 3 epochs, EarlyStopping automatically stopped the training process if the validation loss did not improve for three consecutive epochs. Importantly, restore_best_weights=True was set in the EarlyStopping callback, ensuring that the model weights were returned to those of the epoch that achieved the lowest validation loss.

This comprehensive training process, which incorporated CNN-BiLSTM architecture, hyperparameter adjustment with Hyperband, and early suspension, was designed to develop a robust and accurate model of hatred speech detection optimized for the characteristics of the collected data set and the nuances of online Sinhala and Romanized Sinhala language communication.

3.8 Chapter Summary

Research methodology includes the collection of data from various sources, the annotation of hate speech and the pre-processing of model training. Feature engineering techniques are used to extract meaningful representations from text data. A hybrid CNN-RNN model is designed and implemented, and the hyper parameters are adjusted by the Keras Tuner. The model is trained and evaluated using appropriate metrics and the results are analyzed to assess its performance in the detection of hate speech.

CHAPTER 4

DESIGN AND IMPLEMENTATION

4.1 Chapter Overview

This chapter describes in detail the design and implementation of a hate speech detection system. This study provides an overview of system design and describes the implementation of data preprocessing techniques, characteristic extraction methods and the CNN-BiLSTM model architecture. The chapter also discusses the training and optimization process, model assessment procedures and the implementation environment and libraries used in the study.

4.2 System Design Overview

The Sinhala hate speech detection system is designed to process text comments in Sinhala and Romanized Sinhala, including emojis, collected from social media platforms. This is because social media are a major source of hate speech online. The system follows a structured work, which consists of different stages to effectively identify the content of hate speeches, as detailed in Figure 3.1 of Chapter 3. The main stages are: data preprocessing, feature extraction, CNN-BiLSTM model training, and hate speech detection (classification). The rationale for this step-by-step approach and the overall system architecture are described in Chapter 3 and illustrated in Figure 3.1 of Chapter 3.

In order to achieve this process, the system is architected into five key modules that each deal with a specific phase of the hate speech detection process as stated in Chapter 3:

- Data Collection Module: This module collects Sinhala text comments from social media and integrates existing data sets to create a data set for training and evaluation. The need for a representative data set is critical for the performance of the model. The data collection strategy is described in Section 3.3 of Chapter 3.

- Data Preprocessing Module: This module performs text cleaning, standardization and enrichment steps. Raw text is noisy and requires preparation for machine learning. The main steps, as justified in Section 3.5 of Chapter 3, include emoji processing, text cleaning, tokenization, word stoppage and word standardization.

- Feature Extraction Module: This module converts preprocessed text into numerical feature vectors using word embeddings, using a Keras embedding layer. Machine learning models require numerical input, and word embedding captures semantic meaning. The choice of word embeddings as features is justified in Section 3.6 of Chapter 3.

- CNN-BiLSTM Model Module: The core classification component, using a CNN-BiLSTM network to learn text patterns for detecting hate speech. This hybrid model

combines CNNs for local features and BiLSTMs for sequential context to improve performance. The rationale for choosing this hybrid architecture is detailed in Section 3.7.1 of Chapter 3.

- Training and Evaluation Module: This module handles model training, hyperparameter tuning and performance evaluation. Thorough training and evaluation are essential to ensure the effectiveness and generalisation of the model. The training and evaluation methodology is described in Section 3.7 of Chapter 3.

4.3 Data Preprocessing Implementation

As described in Section 3.5 of Chapter 3, data preprocessing is essential for the preparation of raw and Romanized Sinhala text comments for effective detection of hate speech. This is because raw social media texts are often noisy and inconsistent, requiring cleaning and standardization for effective model training. This section describes the implementation of these preprocessing steps, which directly correspond to the methodology in Section 3.5.

4.3.1 Enriching Text with Weighted Emoji Sentiment

As indicated in Section 3.5.11 of Chapter 3, emojis are incorporated by enriching the text with weighted feelings. Emojis convey emotion in online communication, and capture of this emotion improves sentiment analysis. A custom lexicon, emoji_weights.csv, loaded with the load_emoji_weights function, assigns sentimental weights to emoji. The emoji weights loading function is depicted in Code 4.1.

```
# Load Emoji Weights
def load_emoji_weights(filepath):
    try:
        df_emoji = pd.read_csv(filepath)
        # Convert to dictionary: {emoji: weight}
        emoji_weights = dict(zip(df_emoji['Emoji'],
            df_emoji['weight']))
        return emoji_weights

    except FileNotFoundError:
        print(f"Error: Emoji weights file not found at {
            filepath}")
        return {}  # Return empty dict to avoid errors
            later
    except Exception as e:
        print(f"Error loading emoji weights: {e}")
        return {}
```

Code 4.1: Python code for load emoji weights

The preprocess_text function then replaces emojis with __EMOJI_WEIGHT_[weight]_-_ tokens, as shown in Code 4.2:

```
# Function to clean and preprocess text
```

```
            def preprocess_text(text, emoji_weights,
                high_weight_threshold=7.0):
                processed_text = text
                has_high_weight_emoji = 0

                # --- Emoji Replacement (FIRST) ---
                for emoji, weight in emoji_weights.items():
                    if emoji in processed_text:
                        weight_str = str(weight).replace('.', '_')
                        replacement_token = f"__EMOJI_WEIGHT_{weight_str
                            }__"
                        processed_text = processed_text.replace(emoji, f
                            " {replacement_token} ")  # Replace emojis
                        if weight >= high_weight_threshold:
                                has_high_weight_emoji = 1
                emoji_feature = has_high_weight_emoji
```

Code 4.2: Python code for emoji weights

This ensures emoji sentiment information is preserved within the text representation, as justified in Section Section 3.5.1.

4.3.2 Text Cleaning

Text cleaning, as described in Section 3.5.2 of Chapter 3, eliminates noise and inconsistencies. Noise and inconsistencies in the text can prevent the learning of the model. This implementation includes the removal of target characters, white space convergence, and case normalization.

Characters outside alphanumeric Sinhala and English ranges, and underscores are removed using regular expressions, as implemented in preprocess_text function. This focuses the model on the relevant linguistic content, as described in Section 3.5.2. The next step of the text cleaning, removing white spaces are achieved through re.sub() and .strip() in-bult functions in python language. The final step Case Normalization is achieved through converting all characters to lower case using python .lower() function. The related dode segments are shown in Code 4.3.

```
            def preprocess_text(text, emoji_weights,
                high_weight_threshold=7.0):
                # ... (Emoji enrichment code from Subsection 4.3.1) ...

                # --- Text Cleaning (AFTER Emoji Replacement) ---
                processed_text = re.sub(r'[^A-Za-z0-9_\u0D80-\u0DFF]+',
                    ' ', processed_text)
                processed_text = re.sub(r'\s+', ' ', processed_text).
                    strip()
                processed_text = processed_text.lower()
                # ... rest of preprocess_text function ...
```

Code 4.3: Python code for text cleaning

4.3.3 Tokenization (with Bigrams)

The detailed explanation of tokenization and related approaches are given in Section 3.5.3. Tokenization is essential for machine learning models to process text. This implementation uses tokenize_with_bigrams (code below) to generate unigrams and bigrams while preserving emoji tokens to capture word sequences and individual word meanings. The python code shown in Code 4.4 used to apply the tokenization for a given input.

```
            def tokenize_with_bigrams(text):
                """Tokenizes text into unigrams and bigrams, preserving
                    emoji tokens."""

                # Split the text, preserving the emoji tokens:
                parts = re.split(r'(__EMOJI_WEIGHT_[0-9_]+__|\s+)', text
                    )
                # Remove empty strings and whitespace-only strings from
                    the split result
                parts = [part for part in parts if part.strip()]

                unigrams = []
                for part in parts:
                if part.startswith('__EMOJI_WEIGHT_'):
                        unigrams.append(part)  # Keep emoji tokens whole
                    else:
                        unigrams.extend(word_tokenize(part))  # Tokenize
                             other parts with nltk
                bigrams = list(ngrams(unigrams, 2))
                bigram_strings = [" ".join(bigram) for bigram in bigrams
                    ]
                return unigrams + bigram_strings

            def preprocess_text(text, emoji_weights,
                high_weight_threshold=7.0):
                # ... (Preprocessing steps from previous subsections)
                    ...

                processed_text = ' '.join(processed_words) # Assuming
                    processed_words list

                # --- Tokenization (with Bigrams and Emoji Preservation)
                     ---
                words = tokenize_with_bigrams(processed_text)
                processed_text = ' '.join(words)

                return processed_text, emoji_feature
```

Code 4.4: Python code for tokenization with bigrams

4.3.4 Stop Word Removal and Word Standardization

Token sets are refined using stop word removal and word standardization, as detailed and justified in Section 3.5.4 of Chapter 3. The refining of tokens reduces noise and improves

semantic focus.

Stop words are removed using a custom list in StopWords_425.txt, loaded by load_stopwords function shown in Code 4.5. Stop word removal logic is included in preprocess_text function as shown in Code 4.6.

```
# Detect the file encoding
with open("StopWords_425.txt", 'rb') as f:
    result = chardet.detect(f.read())
    encoding = result['encoding']
    print(f"Detected encoding: {encoding}")

# Load stopwords from file
def load_stopwords(file_path):
    with open(file_path, 'r', encoding=encoding) as f:
        return set(f.read().splitlines())
```

Code 4.5: Python code for loading Stop Words

```
def preprocess_text(text, emoji_weights,
   high_weight_threshold=7.0):
# ... (Preprocessing steps from previous subsections)
    ...
words = processed_text.split()
processed_words = []
for word in words:
    standardized = standardize_word(word,
        word_variations) # Standardize word
    if standardized not in stop_words_custom:
        processed_words.append(standardized) # Keep word
            if not a stop word
processed_text = ' '.join(processed_words)
# ... (Tokenization and subsequent steps) ...
```

Code 4.6: Python code for removing Stop Words

As the next step, Word variations are standardized using word_variations.csv, loaded by load_word_variations, mapping variations to base words using standardize_word. The impact of the word standardization is ecplained in Section 3.5.4. Word standardization is applied in preprocess_text before stop word removal as shown in the code in Subsection Section 4.3.4 This addresses spelling variations and improves generalization, as shown in Code 4.7.

```
# Load word variations from CSV file
def load_word_variations(file_path):
    word_variations = {}
    # Read the CSV file
    df_variations = pd.read_csv(file_path)
    for _, row in df_variations.iterrows():
        word = row['word']
        variations = row['Spelling variations'].split('
            | ')  # Split variations by ' | '
        word_variations[word] = set(variations)
```

```
        return word_variations

    # Function to standardize words using word variations dictionary
    def standardize_word(word, word_variations):
        for base_word, variations in word_variations.items():
            if word in variations:
                return base_word
        return word
```

Code 4.7: Python code for word standardization

4.3.5 Sinhala-Specific Suffix Removal

Sinhala-specific suffix removal, as an optional step detailed in Section Section 3.5.5 of Chapter 3, is implemented using remove_suffixes and a suffix list in Suffixes-413.txt loaded by load_suffixes. Suffix removal can reduce word variations in morphologically rich languages like Sinhala. The related python code implementation is depicted in Code 4.8.

```
def load_suffixes(file_path):
"""Loads suffixes from file."""
with open(file_path, 'r', encoding=encoding) as f:
    return set(f.read().splitlines())

suffixes = load_suffixes('Suffixes-413.txt')

def remove_suffixes(word, suffixes):
    """Removes Sinhala suffixes from a word, with validation."""
    for suffix in sorted(suffixes, key=len, reverse=True):  # Sort longest suffix first
        if word.endswith(suffix):
            root_word = word[:-len(suffix)].strip()  # Remove suffix

            # Count Sinhala characters in the remaining word
            sinhala_chars = re.findall(r'[\u0D9A-\u0DC6]', root_word)
            if len(sinhala_chars) >= 2:
                return root_word  # Valid reduction, return root word
            else:
                return word  # Not enough Sinhala characters, keep original

    return word  # No suffix matched, return original

def preprocess_text(text, emoji_weights, high_weight_threshold=7.0):
    # ... (Preprocessing steps including stop word removal and word standardization) ...
```

```
                    processed_words_suffix_removed = [remove_suffixes(
                        word, suffixes) for word in processed_words] #
                        Apply suffix removal
                    processed_text = ' '.join(
                        processed_words_suffix_removed)

                    # ... (Tokenization and subsequent steps) ...
```

Code 4.8: Python code for removing suffixes

4.3.6 Handling Class Imbalance (Data Balancing)

Class imbalance, as discussed in Section 3.5.6 of Chapter 3, is handled using random oversampling. Class imbalance can bias models to favor the majority class. The implementation uses sklearn.utils.resample as shown in Code 4.9 balance the dataset. This oversampling strategy and its rationale are detailed in Section 3.5.6 in Chapter 3.

```
from sklearn.utils import resample

# --- Data Balancing ---
# Separate majority and minority classes
df_majority = df[df['Is offensive'] == 0]
df_minority = df[df['Is offensive'] == 1]

# Upsample minority class
df_minority_upsampled = resample(df_minority,
                                 replace=True,      #
                                     sample with
                                     replacement
                                 n_samples=len(
                                     df_majority), # to
                                     match majority
                                     class
                                 random_state=42)   # for
                                      reproducible
                                     results

# Combine majority class with upsampled minority class
df_balanced = pd.concat([df_majority,
    df_minority_upsampled])
```

Code 4.9: Python code for data oversampling

4.4 Feature Extraction Implementation

Feature extraction, as justified in Section Section 3.6 of Chapter 3, uses word embeddings. Word embeddings capture semantic relationships, improving model understanding. This section describes the implementation using Keras tools.

the featuer extraction implementation contains two major steps as Tokenization and padding as explaind in Section 3.6. In the tokenization implementation, text is converted to numerical sequences using Keras Tokenizer since the Neural networks require numerical input.

In the text padding implementation, word sequences are padded to a uniform length using pad_sequences since the Neural networks need fixed-length inputs. The related python code is shown in Code 4.10.

```
from tensorflow.keras.preprocessing.text import
    Tokenizer
from tensorflow.keras.preprocessing.sequence import
    pad_sequences

# --- Tokenization and Padding ---
tokenizer = Tokenizer(num_words=10000, oov_token='<OOV>'
    )
tokenizer.fit_on_texts(df_balanced['Comment_cleaned'])

# Convert text to sequences (from Subsection 4.4.1)
X = tokenizer.texts_to_sequences(df_balanced['
    Comment_cleaned'])

# --- Padding ---
X = pad_sequences(X, maxlen=50, padding='post',
    truncating='post')
```

Code 4.10: Python code for tokenization

4.5 Model Architecture Implementation

The CNN-BiLSTM model architecture, justified in Chapter 3 Section 3.7.1, is implemented with the Keras Functional API. This architecture combines local and sequential feature extraction to improve hate speech detection.

4.5.1 CNN-BiLSTM Model Structure

The model structure is implemented in the create_cnn_bilstm_model function, combining CNN and BiLSTM branches for feature extraction and dense layers for classification. The overview of the architecture is shown in Section 4.2 of this chapter and Figure 3.6 in Chapter 3.

The input layer and embedding layer are implemented by converting integer sequences to dense vectors. Embedding layers map words to vector space, capturing semantic information. Parameter choices are detailed in Section 3.7.2.

The CNN branch, designed for local feature extraction, is implemented using Conv1D and GlobalMaxPooling1D layers. Layer parameters (filters, kernel size, activation) are detailed in Section 3.7.1.

The BiLSTM branch, for capturing sequential context, is implemented using two Bidirectional(LSTM) layers and a Dropout layer as shown in . BiLSTMs capture long-range

dependencies by processing sequences bidirectionally.

The outputs of the CNN and BiLSTM branches are concatenated and fed into dense layers for classification. Dense layers perform the final classification based on combined features. All the related python code implementation is shown in Code 4.11

```
from tensorflow.keras.layers import Embedding, LSTM,
    Bidirectional, Dense, Dropout, Conv1D,
    GlobalMaxPooling1D, Input, concatenate
from tensorflow.keras.models import Model

def create_cnn_bilstm_model(input_dim, output_dim,
    input_length):
    input_layer = Input(shape=(input_length,))
    embedding_layer = Embedding(input_dim=input_dim,
        output_dim=output_dim, input_length=input_length
        )(input_layer)

    # CNN Branch
    cnn_branch = Conv1D(128, 5, activation='relu')(
        embedding_layer)
    cnn_branch = GlobalMaxPooling1D()(cnn_branch)

    # BiLSTM Branch
    bilstm_branch = Bidirectional(LSTM(32,
        return_sequences=True))(embedding_layer)
    bilstm_branch = Dropout(0.5)(bilstm_branch)
    bilstm_branch = Bidirectional(LSTM(32))(
        bilstm_branch)

    # Concatenate Branches
    merged = concatenate([cnn_branch, bilstm_branch])

    # Dense Layers
    dense_layer = Dense(64, activation='relu')(merged)
    dropout_layer = Dropout(0.4)(dense_layer)
    output_layer = Dense(1, activation='sigmoid')(
        dropout_layer)

    model = Model(inputs=input_layer, outputs=
        output_layer)
    return model

cnn_bilstm_model = create_cnn_bilstm_model(input_dim
    =10000, output_dim=128, input_length=50)
```

Code 4.11: Python code for tokenization

4.6 Training and Optimization Implementation

Model training and optimization follow the methodology outlined in Section 3.7.1 of Chapter 3.

Data splitting into training and testing sets is implemented using train_test_split function using python. Train-test split allows for evaluating generalization on unseen data.

Model compilation is implemented using model.compile(), configuring optimizer, loss function, and metrics. Compilation configures the learning process for the model.

Model training with early stopping is implemented using model.fit() and the EarlyStopping callback, as detailed in Section 3.7 of Chapter 3. Early stopping prevents overfitting and selects the best model during training. Parameters for EarlyStopping and fit() (epochs, batch_size, validation data, callbacks) are set as detailed in Section 3.7 of Chapter 3. Related python code is implemented in Code 4.12

```
from sklearn.model_selection import train_test_split
# --- Train-test split ---
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)

from tensorflow.keras.optimizers import Adam
def train_and_evaluate_keras_model(model, X_train, y_train,
    X_test, y_test, model_name):
    # ... (Function definition) ...
    model.compile(loss='binary_crossentropy', optimizer=Adam
        (learning_rate=0.001), metrics=['accuracy'])

     # --- Early Stopping ---
    early_stopping = EarlyStopping(monitor='val_loss',
        patience=3, restore_best_weights=True)

    # --- Model Training ---
    history = model.fit(X_train, y_train, epochs=20,
        batch_size=64, validation_data=(X_test, y_test),
        callbacks=[early_stopping])
```

Code 4.12: Python code for model training and evaluation

4.7 Model Evaluation

Model evaluation is performed using model.evaluate(), classification_report, and confusion_matrix to assess performance on the test set. Evaluation provides insights into the model's effectiveness in detecting hate speech. The related functionality is implemented in python as shown in Code 4.13

```
from sklearn.metrics import classification_report,
    confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

def train_and_evaluate_keras_model(model, X_train, y_train,
    X_test, y_test, model_name):
    # ... (Model compilation and training code from previous
        subsections) ...

```

```
# --- Model Evaluation ---
loss, accuracy = model.evaluate(X_test, y_test, verbose
    =0)
print(f"{model_name} Accuracy: {accuracy:.4f}")

y_pred_prob = model.predict(X_test)
y_pred = (y_pred_prob > 0.5).astype("int32")

print(f"{model_name} Classification Report:\n{
    classification_report(y_test, y_pred)}")

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Non-Offensive', 'Offensive'],
            yticklabels=['Non-Offensive', 'Offensive'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title(f'{model_name} Confusion Matrix')
plt.show()
```

Code 4.13: Python code for model evaluation

4.8 Implementation Environment and Libraries

The implementation environment and libraries used are as follows, and are standard for deep learning and NLP tasks. Using standard tools ensures reproducibility and leverages established best practices. The full code implementation can be found in Appendix B.

- Programming Language: Python (version 3.12.9)
- Deep Learning Framework: TensorFlow (version 2.18.0) with Keras API
- NLP Libraries: NLTK (version 3.9.1)
- Data Manipulation and Analysis Libraries: Pandas (version 2.2.3), NumPy (version 2.0.2)
- Machine Learning and Evaluation Libraries: Scikit-learn (version 1.6.1) (sklearn)
- Visualization Libraries: Matplotlib (version 3.10.0), Seaborn (0.13.2)
- Development and Execution Environment: Windows 10 Pro 64 bit on Intel(R) Xenon(R) Gold 6238R CPU @ 2.20GHz (4 CPUs)

These choices reflect common tools in NLP and deep learning as justified generally in relevant literature.

4.9 Chapter Summary

This chapter provided a detailed account of the design and implementation of the Sinhala hate speech detection system, building upon the methodology of Chapter 3. It described the practical steps to create a functional system, covering data preprocessing, feature extraction, model architecture, training, and the implementation environment. The following chapter will present the results of the system's evaluation.

CHAPTER 5

EVALUATION AND RESULTS

5.1 Chapter Overview

This chapter presents a comprehensive evaluation of our main approach, the CNN-BiLSTM combined model, for the detection of hate speech. To contextualize its performance, it is considered all other models as comparison models. This section describes in detail the evaluation metrics used, presents the results for the CNN-BiLSTM model and all comparison models on the test data set, and interprets these findings in relation to the research objectives, highlighting the comparative effectiveness of our combined approach and the insights obtained from the study of the standalone CNN, BiLSTM and LSTM architectures.

5.2 Evaluation Metrics: Quantifying Model Performance

To rigorously evaluate the performance of the primary CNN-BiLSTM combined model and comparison models for detection of hate speech in Sinhala and Romanized Sinhala text (including emojis), a number of standard measures for binary classification was employed. These metrics provide complementary perspectives on the effectiveness of each model in distinguishing between offensive and non-offensive Sinhala and Romanized Sinhala comments. The measures used in this assessment are detailed as follows:

Accuracy: Accuracy is calculated as a ratio of the correctly classified instance to the total number of instances, and provides a comprehensive measure of the correctness of a model. Although information is available for balanced data sets, accuracy alone may not be sufficient, especially in the case of low-resource languages such as Sinhala, where data sets may be smaller or show specific characteristics. In such cases, a high accuracy score can be misleading if the model mainly excels in classifying the majority class while performing badly on the minority class. Thus, additional measures are crucial to a more nuanced understanding, especially in the context of Sinhala hate speech detection, where class imbalance can be a concern.

Precision (for 'Offensive' Class): Precision, especially for the 'offensive' class (label '1'), quantifies the proportion of comments that the Sinhala and Romanized Sinhala had predicted to be offensive and actually offensive. In Sinhala's hate speech detection, high accuracy in identifying offensive content is essential to minimize false positives – cases in which benign Sinhala comments are mistakenly marked as hate speech. False positives can lead to unreasonable censorship and suppression of legitimate expression in Sinhala's online spaces, raising important ethical concerns about content moderation (Davidson et al. 2017). As Davidson et al. (2017) highlights that using too much content, on the side of caution, can have significant impacts on freedom of expression, a particularly sensitive issue in linguistic communities such as those using Sinhala. The precision is calculated as

follows:

$$Precision = TruePositive/(TruePositive + FalsePositive)$$

Recall (Sensitivity) (for "Offensive" Class): The recall, also focusing on the "Offensive" class, measures the proportion of true offensive Sinhala and Romanized Sinhala comments that were correctly identified as offensive by the model. High recall is essential to minimize false negatives – cases where Sinhala's actual hate speech is present but missed by the model. The inability to detect Sinhala hate speech can have serious consequences in the real world, allowing harmful content to spread within Sinhala-speaking online communities and potentially harm individuals or groups. Recall is calculated as follows:

$$Recall = TruePositives/(TruePositives + FalseNegatives)$$

F1-Score (for "Offensive" Class): F1-Score represents the harmonic average of precision and recall and provides a balanced metric comparing false positive and false negative values. It is particularly valuable when class distribution is uneven, or when a balanced balance between accuracy and recall is required, which is a common problem in the detection of hate speech, especially in low-resource languages such as Sinhala. The F1 score deals with the limitations of the use of accuracy and recall in isolation, which is especially relevant in the evaluation of the Sinhala text classification model. The score of F1 is calculated as follows:

$$F1 - Score = 2 * (Precision * Recall)/(Precision + Recall)$$

Confusion Matrix: Confusion matrix is a visual tool to summarize the performance of classification models. It presents a detailed breakdown of True Positives, True Negatives, False Positives, and False Negatives in a matrix format. The confusion matrix provides a granular view of the predictions of a model for each class, allowing in-depth analysis of where the model excels and where it needs improvement in the classification of Sinhala comments. In this research, we visually present the confusion matrices of the main models to facilitate the interpretation of their classification patterns in the context of the recognition of hate speech in Sinhala.

By evaluating this comprehensive number of measures, we aim to achieve a holistic and balanced understanding of the detection capabilities of hate speech in Sinhala and Romanized Sinhala of CNN-BiLSTM models and comparison models. This approach goes beyond simple precision to evaluate their practical effectiveness and ethical implications in the actual moderation scenario of Sinhala content, especially given the low resources of Sinhala languages in NLP.

5.3 Overall Model Performance on Test Dataset

This section shows the overall performance of our CNN-BiLSTM combined model and the comparison models on the retained test data set for Sinhala and the Romanized Sinhala language hate speech detection. First, we focus on accuracy as a primary and high-level metric for assessing the predictive capability of models in the context of the Sinhala text. Although

accuracy is a general indication of predictive capacity, it is essential to remember, as mentioned in Section 5.2, that accuracy itself offers a limited perspective, especially in tasks such as the detection of hate speech, especially when dealing with low-resource languages such as Sinhala, where the characteristics of the database may influence the accuracy score. Therefore, while we start with accuracy for a general comparison of Sinhala hate speech detection models, subsequent sections will focus on more precise metrics to provide a richer evaluation.

Test Accuracy of Models on Sinhala Dataset	
CNN-BiLSTM Model	0.9251
CNN Model	0.9234
Standalone BiLSTM Model	0.9139
LSTM Model	0.8983
SVM	0.8991
Random Forest	0.8931
Logistic Regression	0.8352
Naive Bayes	0.7891

Table 5.1: Accuracy of models on sinhala dataset

The combined **CNN-BiLSTM** model and the **CNN** model show the highest total accuracies in the test data for Sinhala hate speech detection, reaching **0.9251** and **0.9234**, respectively. This is particularly encouraging for the processing of Sinhala languages, suggesting that both convolutionary and combined convolutional-recurrent neural network architectures are highly effective for the detection of hate speech even in low-resource environments such as Sinhala.

The standalone **BiLSTM** model also showed a strong performance with an accuracy of **0.9139** on Sinhala text, compared to the simplest **LSTM** model **(0.8983)**. Among traditional machine learning comparison models, SVM and Random Forest achieved relatively competitive accuracy in Sinhala (approximately 0.89-0.90), while Logistic Regression and Naive Bayes demonstrated significantly lower performance in terms of accuracy in Sinhala language, indicating their limitations in capturing the complexity of the patterns of hate language in Sinhala text. However, to fully understand the performance of the Sinhala hate speech detection model, it is necessary to examine the accuracy, recall and F1 scores presented in the following sections.

5.4 Detailed Classification Performance: Confusion Matrix Analysis

To gain a broader understanding of the performance of our first CNN-BiLSTM combined model for Sinhala and Romanized Sinhala hate speech detection, we analyzed its confusion

matrix on the test data set. The confusion matrix provides a detailed breakdown of the model's predictions for Sinhala comments, revealing not only the overall accuracy, but also the specific types of classifications and misclassifications performed by this combined architecture in the processing of Sinhala texts.

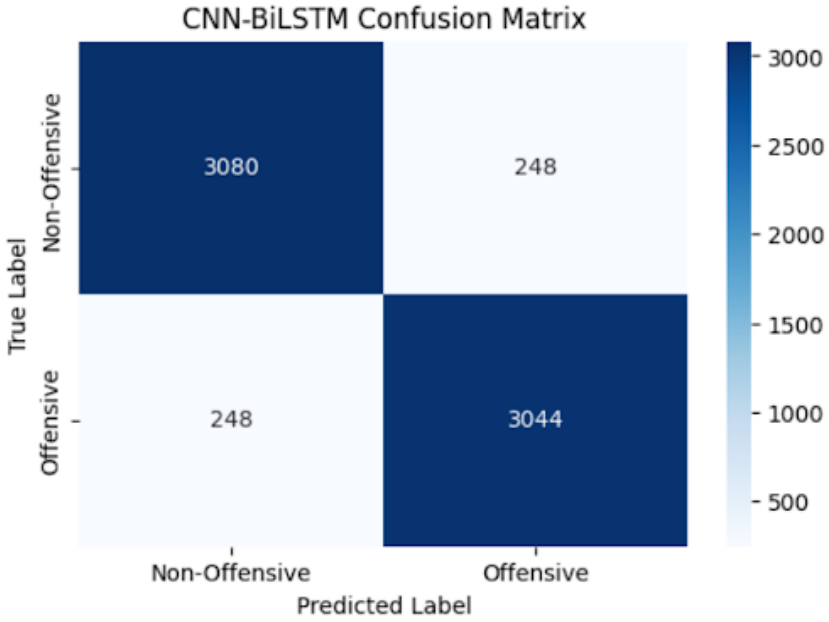


Figure 5.1: Visual representation of confusion matrix for Sinhala hate speech detection

Interpretation of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) for CNN-BiLSTM Model in Sinhala Context:

- **True Positives (TP) in Sinhala Hate Speech Detection:** The confusion matrix reveals that the CNN-BiLSTM combined model correctly identified 3044 instances of offensive Sinhala and Romanized Sinhala comments as offensive. This substantial TP value highlights the combined model's robust capability in accurately detecting Sinhala hate speech present in the text. This is particularly significant in the context of a low-resource language like Sinhala.

- **True Negatives (TN) for Sinhala's non-offensive content:** The model correctly classified the comments of 3080 non-offensive Sinhala and Romanized Sinhala as non-offensive. This even greater TN value reflects the excellent performance of the combined model in identifying legitimate Sinhala content and avoiding unnecessary markings of benign Sinhala texts.

- **False Positives (FP) in Sinhala Text Classification:** The model mishandled 248 non-offensive Sinhala and Romanized Sinhala comments as offensive. These false positives, although relatively low, represent instances where the combined model, despite its strong overall performance in Sinhala, still exceeds the legitimate Sinhala language. In content moderation systems for Sinhala online platforms, false

positives can lead to unreasonable censorship of acceptable content and may have adverse effects on the user experience of Sinhala speakers. As a result, minimizing false positives is an important consideration for Sinhala content moderation.

- **False Negatives (FN) - Lost Sinhala Hate Speech:** The confusion matrix indicates that the CNN-BiLSTM combined model has missed 248 cases of real offensive and Romanized Sinhala comments, classifying them as non-offensive. These false negatives, equal to false positives in this case, are a critical concern in the Sinhala hate speech detection. While the combined model effectively detects a large proportion of Sinhala hate speech, these missing cases represent inability to identify harmful content in Sinhala, which can have harmful effects by allowing hate speech to spread within online Sinhala communities. Efforts to further reduce false negatives are essential to maximize the security and inclusion of online environments for Sinhala language users.

Based on the confusion matrix, CNN-BiLSTM combined models show remarkable balanced performance for sinhala hate speech detection. The almost identical numbers of false positives (248) and false negatives (248) indicate that the model does not exhibit significant bias towards offensive Sinhala content in its current configuration. This almost equal distribution of errors requires a well-defined model that effectively balances the competing priorities of precision and recall in this Sinhala hate speech detection task. The overall low number of FP and FN further strengthens the strong performance of the combined model and its potential for practical application in Sinhala content moderation systems, especially considering the difficulties of processing low-resource languages.

5.5 Comparative Performance Analysis

CNN-BiLSTM Models vs. Comparison Models (for Sinhala Hate Speech Detection) To contextualize the performance of our first CNN-BiLSTM combined model for Sinhala and Romanized Sinhala Hate Speech Detection, this section presents a comparison analysis of the comparison models assessed in this study, all evaluated on the same Sinhala test dataset. We will examine the key evaluation criteria of all models to identify the relative strengths and weaknesses of each approach to detection of Sinhala hate speech. Table 5.2 summarizes performance metrics for the detection of Sinhala hatred speech, focusing on accuracy, precision, recall and F1 score for the ”offensive” class (label ”1”) as key indicators of model effectiveness in the context of the Sinhala language.

As shown in Table 5.2, the CNN-BiLSTM combined model and the CNN model demonstrate the highest overall accuracy in Sinhala hate speech detection, reaching 0.9251 and 0.9234, respectively. This confirms our previous observation in that both convolutional and combined convolutionary-recurrent architectures are very effective for the detection of Sinhala hate speech. In particular, the CNN-BiLSTM model achieves the highest accuracy and F1 score for the ”Offensive” class, indicating a better balance of accuracy and recall compared to all comparison models in the process of Sinhala text.

Although the standalone CNN model offers very close performance to the combined model in terms of Sinhala accuracy and accuracy, the slight edge of the CNN-BiLSTM

Model	Accuracy	Precision (Offensive)	Recall (Offensive)	F1-Score (Offensive)
CNN-BiLSTM	**0.9251**	**0.92**	**0.92**	**0.92**
CNN	0.9234	0.92	0.92	0.92
Standalone BiLSTM	0.9139	0.92	0.91	0.91
LSTM Model	0.8983	0.92	0.87	0.9
SVM	0.8991	0.89	0.91	0.9
Random Forest	0.8931	0.86	0.93	0.9
Logistic Regression	0.8352	0.84	0.83	0.83
Naive Bayes	0.7891	0.73	0.91	0.81

Table 5.2: Performance Comparison of CNN-BiLSTM Model and Comparison Models on Test Set

model suggests that the inclusion of the BiLSTM layer may provide a marginal benefit for recording sequential dependency in Sinhala text data, possibly leading to a slightly improved generalization of hate speech in the Sinhala language.

The standalone BiLSTM model, while it works slightly lower than the CNN and CNN-BiLSTM models in Sinhala accuracy, still outperforms the simpler LSTM model in Sinhala context, highlighting the advantage of bidirectional context processing for Sinhala hate speech detection. The unidirectional LSTM, which processes Sinhala text only in a forward direction, appears to be less effective in capturing the nuances of Sinhala's hate language than its bidirectional counterpart and the convolutionary approaches, especially in a low-resource linguistic scenario.

Among the traditional machine learning comparison models, the SVM and Random Forest achieved relatively competitive accuracies in Sinhala, but the accuracy of the ”offensive” category is much lower than that of the CNN-BiLSTM and CNN models when applied to Sinhala texts. Interestingly, the Random Forest model obtained the highest recall for the ”Offensive” category (0.93), even surpassing CNN-BiLSTM models to minimize the false negatives in the detection of Sinhala's hate speech. However, this higher recall costs a much lower accuracy (0.86), indicating that random forests tend to display Sinhala content without offensive, leading to a less desirable compromise between accuracy and recall, which is reflected in its lower F1 scores in the ”offensive” category compared to CNN-BiLSTM for Sinhala text classification.

Logistic Regression and Naive Bayes show significantly lower performance in all metrics than neural network models and even SVM and Random Forest in the context of Sinhala hate speech detection. This suggests that these simpler linear models are not adequate to effectively capture the complex patterns and contextual dependencies inherent in Sinhala hate speech in this data set. Their lower precision, in particular, indicates a higher rate of false positives, and their lower recall suggests a greater tendency to miss actual cases of

Sinhala hate speech.

Overall, the comparison analysis clearly positions CNN-BiLSTM combined models as leading approaches to the detection of Sinhala and Romanized Sinhala hate speech and demonstrates superior or comparable performance in key metrics, in particular, F1 scores for ”offensive” classes that effectively balance accuracy and recall of Sinhala text. Although the CNN-BiLSTM architecture has outstanding performance in Sinhala, the CNN-BiLSTM architecture offers a small but consistent advantage. Neuronal network architectures (CNN, BiLSTM, LSTM, and CNN-BiLSTM) generally outperform traditional machine learning models for detection of Sinhala hate speech, highlighting the effectiveness of deep learning techniques for the complexity of detection of Sinhala's hate speech in low-resource text data. These results indicate that CNN-BiLSTM combined models are a robust and promising solution to the detection system of Sinhala hate speech and can contribute to creating a safer and more inclusive online environment for Sinhala speakers.

5.6 Model evaluation againsts sample inputs

This section presents the output of the CNN-BiLSTM model developed and the baseline models on specific input examples. Sample inputs were selected considering Sinhala, Romanized Sinhala and emojis inclusions for the considered text. This showcases the model's performance on diverse examples, highlighting its ability to handle different linguistic nuances and the presence of emojis. The phrase relating to captured text is shown in Appendix C.

5.6.1 Sinhala and Romanized sinhala inputs without emojis

This subsection presents examples of input text with Sinhala and Romanized Sinhala phrases without emojis and the corresponding outputs from all models, including the CNN-BiLSTM model. This helps illustrate the models' ability to detect hate speech based solely on textual cues. Selected text phrases and predictions are shown in Table 5.3.

Table 5.3: Model evaluation against selected text phrases without emojis

Input Text	**CNN-BiLSTM**	BiLSTM	CNN	LSTM	SVM	Logistic Regression
mala මිනියක් වගේ	**Offensive**	Non-Offensive	Offensive	Non-Offensive	Offensive	Offensive
Doni thamai hodatama kare patiya adarei manke	**Non-Offensive**	Non-Offensive	Offensive	Offensive	Offensive	Offensive
සිරාවටම කියන්නෙ මේ ජීවිතේ අහපු ලස්සනම වොයිස් එක	**Non-Offensive**	Offensive	Offensive	Offensive	Non-Offensive	Non-Offensive

Continued on next page

Table 5.3 – continued from previous page

Input Text	**CNN-BiLSTM**	BiLSTM	CNN	LSTM	SVM	Logistic Regression
සිරාවටම කියන්නෙ මේ ජීවිතේ අහපු කැතම වොයිස් එක	**Offensive**	Offensive	Offensive	Offensive	Non-Offensive	Non-Offensive
අපි වගේ හ## ඉන්නකම් මුංට සල්ලි හොයන්න පුලුවන් ටික් ටොක් එකෙන්	**Offensive**	Non-Offensive	Non-Offensive	Offensive	Offensive	Offensive

5.6.2 Sinhala and Romanized sinhala inputs with emojis

This subsection presents examples of input text with Sinhala and Romaized Sinhala phrases with emojis and the corresponding outputs from all models, including the CNN-BiLSTM model. This helps illustrate the models' ability to detect hate speech based solely on textual cues. Selected text phrases and predictions are shown in Table 5.4.

Table 5.4: Model evaluation against selected text phrases with emojis

Input Text	**CNN-BiLSTM**	BiLSTM	CNN	LSTM	SVM	Logistic Regression
හුලෙන් බකලා වගේ නිකන් 😂💔 හරිම තාත්විකකයි	**Offensive**	Non-Offensive	Offensive	Non-Offensive	Offensive	Offensive
oya කරලා thiyenne hondata 🥰	**Non-Offensive**	Non-Offensive	Non-Offensive	Non-Offensive	Non-Offensive	Offensive
oya කරලා thiyenne hondata 😡	**Offensive**	Non-Offensive	Non-Offensive	Non-Offensive	Non-Offensive	Offensive
මෙලෝ වෙනසක් නෑ හැමදාම එක වගේ 🤮	**Offensive**	Non-Offensive	Non-Offensive	Non-Offensive	Non-Offensive	Non-Offensive
Giji giji pojjak කරලා celebrate pojja කරන්න 😈	**Offensive**	Non-Offensive	Non-Offensive	Non-Offensive	Non-Offensive	Offensive

5.7 Qualitative Verification Through Application Prototype

In addition to the above-mentioned quantitative performance metrics, a working prototype has been developed and tested to verify the effectiveness of the proposed hate speech detection system in the real world. This prototype provides an interactive interface, in which users can enter the text of Sinhala and Romanized Sinhala, often enriched with emojis, and

receive immediate classification results based on CNN-BiLSTM models, as well as comparative outputs of LSTM, CNN, and BiLSTM models.

To demonstrate the practical performance of the system, two real-world comments extracted from public social media platforms were evaluated through the application. The selected text inputs and their corresponding predictions are summarized in Table 5.5.

Table 5.5: Qualitative evaluation with best performing models

Input Text	**CNN-BiLSTM**	**LSTM**	**CNN**	**BiLSTM**
සින්දු හානවා 😅 මේවට හොදයි කියන උන්ටනම් සිරාම ට්‍රැක්. Personality හොදයි ඒත් සින්දු කියන්න බෑ. 😅 සිංහල සිංදුනම් කොහොමත් බෑ ටකරම් වොයිස් එකක්.	**Offensive (51%)**	Offensive (95%)	Non-Offensive (50%)	Offensive (77%)
Mahesha Jayaweera ඉන්දියාව Talent Round එකේ Asia Finalists Select උනා ඒත් ❤️ Talent Round එකයි head To head round එකෙයි දෙකේම final ආපු එකම Asia candidater අනුදි ❤️	**Non-Offensive (96%)**	Non-Offensive (91%)	Non-Offensive (100%)	Non-Offensive (89%)

Illustrative screenshots showing the model predictions for the above comments are included in Appendix D for further reference.

5.8 Chapter Summary

The performance of the hate speech detection model is evaluated using multiple measures such as accuracy, precision, recall, and F1-score. The results illustrate the efficiency of the model in reliably identifying hate speech in Sinhala and Romanized Sinhala code-mixed text with emojis. Comparison analysis with baseline models shows that the proposed CNN-BiLSTM model exceeds existing methods. This chapter also analyzes the impact of various hyperparameters and feature engineering techniques on model performance and provides a study of factors that contribute to its effectiveness.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Chapter Overview

This chapter concludes our study of Sinhala hate speech detection using a CNN-BiLSTM approach. This summarizes our main findings directly addressing research questions, summarizing the overall conclusions on the research problem, acknowledging the limitations of the study, and finally highlighting promising implications for future research and development in this vital area.

6.2 Conclusions on Research Questions

- **Is a CNN-BiLSTM model an effective architecture for hate speech detection in Sinhala and Romanized Sinhala text that contains emojis?**

  Our findings strongly indicate that the **CNN-BiLSTM** model is actually an effective architecture for hate speech detection in Sinhala and Romanized Sinhala text. The model has shown high overall accuracy **(0.9251)** and balanced F1-score **(0.92 for the "Offensive" class)** in our Sinhala test data set. This performance exceeds that of many traditional machine learning models and independent deep learning models, suggesting that the **CNN-BiLSTM** architecture is well adapted to capture the complex patterns inherent in Sinhala hate speech. The effectiveness of this architecture is further supported by the detailed confusion matrix and class-specific accuracy and recall metrics, which consistently show strong performance.

- **How effective is the proposed CNN-BiLSTM combined model in detecting hate speech in Sinhala and Roman text in Sinhala, particularly with emojis and considering the complexity of low-resource languages?**

  The proposed **CNN-BiLSTM** combined model shows a high efficiency in detecting hate speech within Sinhala and Romanized Sinhala texts, even if there is an emoji. Continuously high performance measures in terms of accuracy, accuracy, recall and F1 scores (all greater than 0.92 in CNN-BiLSTM models) show the strong capability to handle Sinhala's nuances, including its romantic variants and the integration of common emoji in online communication. Furthermore, the ability to achieve such performance in a low-resource language context, such as Sinhala, highlights the efficiency and potential of real-world applications where data and resources may be limited. The balance of the model's performance in minimizing both positive and negative falsehoods is especially important for ethically responsible detection of hate speech in this linguistic environment.

- **How is the performance of the CNN-BiLSTM model compared to the standalone deep learning models (CNN, BiLSTM, LSTM) and traditional machine learning classifiers (SVM, Random Forest, Logistic Regression, Naive Bayes) in the context of Sinhala hate speech detection, and what are the relative contributions of CNN and BiLSTM components in the integrated architecture?**

  The comparative analysis reveals that the **CNN-BiLSTM** model shows superior or highly competitive performance for Sinhala hate speech detection compared to both independent deep learning models and traditional machine learning classifiers. While the CNN model is close to the CNN-BiLSTM in terms of accuracy, the CNN-BiLSTM has achieved a slightly better balance of accuracy and recall, which is reflected in its higher F1 score for the ”Offensive” class. Independent BiLSTM and LSTM models showed strong performance, but were generally outperformed by CNN-based architectures, suggesting that convolutional layers effectively contribute to capturing relevant local features in Sinhala texts. Traditional machine learning models (SVM, Random Forest, Logistic Regression, Naive Bayes) generally lacked deep learning approaches, especially in precision and F1 score, indicating the advantage of deep learning models, particularly the CNN-BiLSTM combined architecture, for taking into account the complex patterns and contextual dependencies of Sinhala hate speech. The success of CNN-BiLSTM highlights the synergistic effect of combining convolutionary and recurrent neural network components for this task.

6.3 Conclusion on Research Problem

The research problem addressed in this paper is the critical need for an effective mechanism to detect hate speech in comments on online social media platforms written in Sinhala and Romanized Sinhala, often including emojis. The increasing prevalence of hate speech online, particularly in under-resourced languages such as Sinhala, requires robust automated detection to protect online users and cultivate a more inclusive digital space for Sinhala speakers. This research directly addresses this critical problem by developing and rigorously assessing a new approach to the detection of Sinhala hate speech, providing valuable knowledge contributions and positive social consequences.

- **Contribution towards knowledge:**

  This study makes several important contributions to the field of hate speech detection, particularly for the Sinhala and Romanized Sinhala language. These contributions include:

  1. **The effectiveness of the CNN-BiLSTM Combined Model:** This research has analyzed in detail the effectiveness of the CNN-BiLSTM Combined Model and has proved its significant effectiveness in detecting hate speech in Sinhala and romanticized Sinhala texts.
  2. **Established CNN-BiLSTM as a high performance architecture through comparative analysis:** Through rigorous comparative analysis with a variety of machine learning and deep learning classifiers, this study established the

CNN-BiLSTM architecture as a robust and high-performance solution for Sinhala hate speech detection.

3. **Focused on the role of deep learning feature extraction in Sinhala NLP:** This research aims to better understand the effects of deep learning-based feature extraction through embedded layers in CNN-BiLSTM models, which are crucial to capturing ambiguous semantic and contextual information for effective detection of Sinhala hate speech.
4. **Provided baseline results for future Sinhala NLP research:** This thesis provides valuable baseline results by rigorously assessing and recording the performance of CNN-BiLSTM and comparing models on a Sinhala hate speech dataset. These benchmarks can serve as a basis for future research efforts in Sinhala NLP, especially in the field of online content moderation and other low-resource languages facing similar challenges.

- **Contribution towards society**

  The research conducted in this study also has several societal implications, including:

  1. **Offers a mechanism to improve the online experience for Sinhala users:** The development and validation of an effective CNN-BiLSTM model for detecting hate speech in Sinhala provides a critical technical mechanism that can be used to improve the online experience for Sinhala users by enabling automated identification and potential mitigation of hate content.
  2. **Sensitivity to Sinhala Online Hate Speech Challenges:** This work contributes to raising critical awareness of the specific challenges posed by online hate speech in the context of Sinhala. This increased awareness can encourage the development of targeted community guidelines, prevention programmes and intervention strategies to combat online negativity in Sinhala-speaking communities.
  3. **Encouraged Further Development for Sinhala and Low-Resource Languages:** Finally, this research aims to promote the further development and broader deployment of effective tools and resources for the detection of hate speech specifically for Sinhala and other low-resource languages. This progress contributes to the creation of a more equitable and respectful global digital environment that supports linguistic diversity and actively contributes to mitigate the harm caused by online speakers of all languages.

In summary, this research addresses the problem of detecting hate speech content in Sinhala and Romanized Sinhala social media comments by investigating the application of machine learning classifiers and feature extraction methods. The findings of this study contribute to the knowledge base in the field and provide practical solutions for fostering a safer online environment for the Sinhala-speaking community.

6.4 Limitations

Despite the promising results obtained from our CNN-BiLSTM model for the detection of Sinhala hate speech, this study recognises certain limitations that are important to consider for future work and interpretation:

1. **Dataset size and representativeness:** The size of the Sinhala hate speech dataset, although sufficient for model training and evaluation, is still relatively limited compared to data sets available for high-resource languages such as English. A larger and more diverse set of data, including a broader range of online platforms and Sinhala text styles, could potentially lead to further improvements in model generalization and robustness, as well as a more comprehensive evaluation of performance across different types of Sinhala online content.

2. **Focus on text comments and emoji context:** This research focuses mainly on the detection of hate speech within text comments, which incorporate emojis as features. Model performance on other forms of online content (such as images, videos, audio, etc.) in different contexts of Sinhala (news articles, formal writings, etc.) remains unexplored. Furthermore, emojis are considered, but a more in-depth analysis of how emoji interacts with the Sinhala text in the context of hate speech could be beneficial.

3. **Challenges of Romanized Sinhala and Code Mixing:** Although the model was trained to handle both the Sinhala script and the Romanized Sinhala, the complexity of Romanized Sinhala and potential code mixes with English and other languages pose constant challenges. The variation in romantic style and the unpredictable nature of the mixing of codes could influence the performance of the model, especially when it comes to visible or less frequent language variations in Sinhala's online text.

4. **Potential biases in training data:** Despite efforts to create a balanced set of data, there is always the potential for inherent biases to exist in training data. If the dataset unexpectedly overrepresents certain demographics or perspectives, the trained CNN-BiLSTM model may unexpectedly show biases in its predictions, potentially leading to unfair or unequal performances between different subgroups within the Sinhala online community. Continuous monitoring and bias mitigation strategies are crucial.

6.5 Implications for further research

The research on CNN-BiLSTM in Sinhala to detect Sinhala hate speech provides some promising opportunities for future research and development, based on the results of this study and considering the limitations of current research.

1. **Expanding Dataset Size and Diversity:** Future research should prioritize the creation and use of significantly larger and more diverse data sets for Sinhala hate speech detection. This includes the collection of data from a wider range of online platforms popular with Sinhala speakers, the inclusion of various text styles and genres beyond social media comments, and the better representation of different demographic

groups within the Sinhala online community to improve model generalization and reduce potential prejudices.

2. **Multimodal Hate Speech Detection in Sinhala:** Future work could explore multimodal approaches to Sinhala's hate speech detection, which goes beyond text analysis alone. The study of the integration of image, video and audio data, together with text comments, could lead to a more comprehensive and accurate detection, especially as online hate speech often manifests itself in several ways. This may include the development of models that can process and merge information from text, images and potentially even audio signals in Sinhala's online content.

3. **Improved model Interpretability and Explainability:** Future research should make the interpretability and explainability of deep learning models such as CNN-BiLSTM priority for the detection of hate speech in Sinhala. Developing and applying techniques to interpret models, such as attention mechanisms, layer-wise relevance distribution or SHAP values, can provide useful insights into why models make certain predictions in the Sinhala text. This would help to debug, refine models, and strengthen user trust in the automatic Sinhala content moderation system.

4. **Low-Resource Language Transfer Learning and Adaptation:** Given Sinhala's low resource nature, future research could explore and adapt language transfer learning techniques in greater depth. Research into pre-training on larger, related datasets (even in other languages or general web text) and fine-tuning on smaller Sinhala hate speech datasets could improve model performance and data efficiency. The study of cross-lingual transfer learning approaches could also be valuable to use resources from higher resource languages.

In summary, the research on the hate speech detection in Sinhala and Romanized Sinhala mixed text with emojis opens up a number of opportunities for future research. Future efforts may focus on extending the data sets to other languages and methods, such as images and videos, in order to create a more complete and inclusive detection system. By addressing these challenges and opportunities, future research could make significant progress in detecting online hate speech and promoting a safer and more positive digital environment for all.

BIBLIOGRAPHY

Abebaw, Z., Rauber, A. & Atnafu, S. (2022), 'Design and Implementation of a Multichannel Convolutional Neural Network for Hate Speech Detection in Social Networks', *Revue d'Intelligence Artificielle* **36**(2), 175–183.
URL: *https://www.iieta.org/journals/ria/paper/10.18280/ria.360201*

Agrawal, S. & Awekar, A. (2018), 'Deep Learning for Detecting Cyberbullying Across Multiple Social Media Platforms'. arXiv:1801.06482 [cs].
URL: *http://arxiv.org/abs/1801.06482*

Alfreihat, M., Almousa, O. S., Tashtoush, Y., AlSobeh, A., Mansour, K. & Migdady, H. (2024), 'Emo-SL Framework: Emoji Sentiment Lexicon Using Text-Based Features and Machine Learning for Sentiment Analysis', *IEEE Access* **12**, 81793–81812.
URL: *https://ieeexplore.ieee.org/document/10483059/*

Ali, A. & Syed, A. M. (2020), 'Cyberbullying Detection Using Machine Learning'.

Alkasassbeh, M., Almomani, A., Aldweesh, A., Al-Qerem, A., Alauthman, M., Nahar, K. & Mago, B. (2024), Cyberbullying Detection Using Deep Learning: A Comparative Study, *in* '2024 2nd International Conference on Cyber Resilience (ICCR)', IEEE, Dubai, United Arab Emirates, pp. 1–6.
URL: *https://ieeexplore.ieee.org/document/10533166/*

Alkomah, F. & Ma, X. (2022), 'A literature review of textual hate speech detection methods and datasets', *Information* **13**(6).
URL: *https://www.mdpi.com/2078-2489/13/6/273*

Anezi, F. Y. A. (2022), 'Arabic Hate Speech Detection Using Deep Recurrent Neural Networks', *Applied Sciences* **12**(12), 6010.
URL: *https://www.mdpi.com/2076-3417/12/12/6010*

Asogwa, D. C., Chukwuneke, C. I., Ngene, C. C. & Anigbogu, G. N. (2022), 'Hate Speech Classification Using SVM and Naive BAYES'. Publisher: arXiv Version Number: 1.
URL: *https://arxiv.org/abs/2204.07057*

Badjatiya, P., Gupta, S., Gupta, M. & Varma, V. (2017), Deep Learning for Hate Speech Detection in Tweets, *in* 'Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion', pp. 759–760. arXiv:1706.00188 [cs].
URL: *http://arxiv.org/abs/1706.00188*

CERT (2025), 'Sri Lanka Computer Emergency Readiness Team'.
URL: *https://cert.gov.lk/resources*

Davidson, T., Warmsley, D., Macy, M. & Weber, I. (2017), 'Automated Hate Speech Detection and the Problem of Offensive Language'. Version Number: 1.
URL: *https://arxiv.org/abs/1703.04009*

Deepasree Varma, P., Vinod, P., Nandakumar, M., Akshay, K. & Madhu, A. (2022), Hate Speech detection in English and Malayalam Code-Mixed Text using BERT embedding, *in* '2022 International Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS)', IEEE, Kochi, India, pp. 1–6.
URL: *https://ieeexplore.ieee.org/document/9885339/*

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018), 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding'. Version Number: 2.
URL: *https://arxiv.org/abs/1810.04805*

Di Capua, M., Di Nardo, E. & Petrosino, A. (2016), Unsupervised cyber bullying detection in social networks, *in* '2016 23rd International Conference on Pattern Recognition (ICPR)', IEEE, Cancun, pp. 432–437.
URL: *http://ieeexplore.ieee.org/document/7899672/*

Dias, D. S., Welikala, M. D. & Dias, N. G. (2018), Identifying Racist Social Media Comments in Sinhala Language Using Text Analytics Models with Machine Learning, *in* '2018 18th International Conference on Advances in ICT for Emerging Regions (ICTer)', IEEE, Colombo, Sri Lanka, pp. 1–6.
URL: *https://ieeexplore.ieee.org/document/8615492/*

Fernando, W., Weerasinghe, R. & Bandara, E. (2022), Sinhala Hate Speech Detection in Social Media Using Machine Learning and Deep Learning, *in* '2022 22nd International Conference on Advances in ICT for Emerging Regions (ICTer)', IEEE, Colombo, Sri Lanka, pp. 166–171.
URL: *https://ieeexplore.ieee.org/document/10024082/*

Gitari, N. D., Zhang, Z., Damien, H. & Long, J. (2015), 'A Lexicon-based Approach for Hate Speech Detection', *International Journal of Multimedia and Ubiquitous Engineering* **10**(4), 215–230.
URL: *http://gvpress.com/journals/IJMUE/vol10$_n$o4/21.pdf*

Haidar, B., Chamoun, M. & Serhrouchni, A. (2017), 'A Multilingual System for Cyberbullying Detection: Arabic Content Detection using Machine Learning', *Advances in Science, Technology and Engineering Systems Journal* **2**(6), 275–284.
URL: *https://astesj.com/v02/i06/p34/*

Hettiarachchi, N., Weerasinghe, R. & Pushpanda, R. (2020), Detecting Hate Speech in Social Media Articles in Romanized Sinhala, *in* '2020 20th International Conference on Advances in ICT for Emerging Regions (ICTer)', IEEE, Colombo, Sri Lanka, pp. 250–255.
URL: *https://ieeexplore.ieee.org/document/9325465/*

I., A., M., K., R., R., J.I., S. & S., P. D. (2021), 'Unsupervised Hybrid approaches for Cyberbullying Detection in Instagram', *International Journal of Computer Applications* **174**(26), 40–46.
URL: *http://www.ijcaonline.org/archives/volume174/number26/abishak-2021-ijca-921191.pdf*

Ishara Amali, H. M. A. & Jayalal, S. (2020), Classification of Cyberbullying Sinhala Language Comments on Social Media, *in* '2020 Moratuwa Engineering Research Conference (MERCon)', IEEE, Moratuwa, Sri Lanka, pp. 266–271.
URL: *https://ieeexplore.ieee.org/document/9185209/*

Jahan, M. S. & Oussalah, M. (2023), 'A systematic review of hate speech automatic detection using natural language processing', *Neurocomputing* **546**, 126232.
URL: *https://www.sciencedirect.com/science/article/pii/S0925231223003557*

Jahjah, V., Khoury, R. & Lamontagne, L. (2016), Word normalization using phonetic signatures, *in* R. Khoury & C. Drummond, eds, 'Advances in Artificial Intelligence', Springer International Publishing, Cham, pp. 180–185.

Jayasuriya, P., Ekanayake, S., Munasinghe, R., Kumarasinghe, B., Weerasinghe, I. & Thelijjagoda, S. (2020), Sentiment classification of Sinhala content in social media, *in* '2020 International Research Conference on Smart Computing and Systems Engineering (SCSE)', IEEE, Colombo, Sri Lanka, pp. 136–141.
URL: *https://ieeexplore.ieee.org/document/9313023/*

Jayasuriya, S. (2020), 'Sinhala Unicode Hate Speech'.
URL: *https://www.kaggle.com/datasets/sahanjayasuriya/sinhala-unicode-hate-speech*

Katona, E., Buda, J. & Bolonyai, F. (2021), 'Using N-grams and Statistical Features to Identify Hate Speech Spreaders on Twitter'.

Kemp, S. (2024), 'Digital 2024: Sri Lanka'.
URL: *https://datareportal.com/reports/digital-2024-sri-lanka*

Kirk, H. R., Vidgen, B., Röttger, P., Thrush, T. & Hale, S. A. (2022), 'Hatemoji: A Test Suite and Adversarially-Generated Dataset for Benchmarking and Detecting Emoji-based Hate'. arXiv:2108.05921 [cs].
URL: *http://arxiv.org/abs/2108.05921*

Kumar, Y., Huang, K., Perez, A., Yang, G., Li, J. J., Morreale, P., Kruger, D. & Jiang, R. (2024), 'Bias and Cyberbullying Detection and Data Generation with Transformer AI Models and top LLMs'.
URL: *https://www.preprints.org/manuscript/202407.0411/v1*

Liyanage, O. & Jayakumar, K. (2021), Hate Speech Detection in Sinhala-English Code-Mixed Language, *in* '2021 21st International Conference on Advances in ICT for Emerging Regions (ICter)', IEEE, Colombo, Sri Lanka, pp. 225–230.
URL: *https://ieeexplore.ieee.org/document/9774816/*

LTRL (n.d.), 'LTRL – Language Technology Research Lab'.
URL: *https://ltrl.ucsc.lk/*

Malik, J. S., Qiao, H., Pang, G. & Hengel, A. v. d. (2022), 'Deep Learning for Hate Speech Detection: A Comparative Study'. Version Number: 2.
URL: *https://arxiv.org/abs/2202.09517*

Malik, J. S., Qiao, H., Pang, G. & van den Hengel, A. (2023), 'Deep learning for hate speech detection: A comparative study'.
URL: *https://arxiv.org/abs/2202.09517*

Mohiyaddeen, M. & Siddiqui, D. (2021), 'Automatic hate speech detection: A literature review', *International Journal of Engineering and Management Research* **11**, 116–121.

Muthuthanthri, M. & Smith, R. I. (2024), Hate Speech Detection for Transliterated English and Sinhala Code-Mixed Data, *in* '2024 4th International Conference on Advanced Research in Computing (ICARC)', IEEE, Belihuloya, Sri Lanka, pp. 155–160.
URL: *https://ieeexplore.ieee.org/document/10499768/*

Omran, E., Al Tararwah, E. & Al Qundus, J. (2023), 'A comparative analysis of machine learning algorithms for hate speech detection in social media', *Online Journal of Communication and Media Technologies* **13**(4), e202348.
URL: *https://www.ojcmt.net/article/a-comparative-analysis-of-machine-learning-algorithms-for-hate-speech-detection-in-social-media-13603*

Patil, P., Raul, S., Raut, D. & Nagarhalli, T. (2023), Hate speech detection using deep learning and text analysis, *in* '2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS)', pp. 322–330.

Putra, C. D. & Wang, H.-C. (2024), 'Advanced BERT-CNN for Hate Speech Detection', *Procedia Computer Science* **234**, 239–246.
URL: *https://linkinghub.elsevier.com/retrieve/pii/S1877050924003569*

Pyingkodi, M., Thenmozhi, K., Chitra, K., Karthikeyan, M., Pradeep, M., Suriya, P. T. D. & Viswanathan, T. J. (2023), Hate Speech Analysis using Supervised Machine Learning Techniques, *in* '2023 International Conference on Computer Communication and Informatics (ICCCI)', IEEE, Coimbatore, India, pp. 1–6.
URL: *https://ieeexplore.ieee.org/document/10128591/*

Ranasinghe, T., Anuradha, I., Premasiri, D., Silva, K., Hettiarachchi, H., Uyangodage, L. & Zampieri, M. (2024), 'SOLD: Sinhala offensive language dataset', *Language Resources and Evaluation* .
URL: *https://link.springer.com/10.1007/s10579-024-09723-1*

Riyadi, S., Divayu Andriyani, A. & Noraini Sulaiman, S. (2024), 'Improving Hate Speech Detection Using Double-Layers Hybrid CNN-RNN Model on Imbalanced Dataset', *IEEE Access* **12**, 159660–159668.
URL: *https://ieeexplore.ieee.org/document/10737089/*

Samarasinghe, S. W. A. M. D., Meegama, R. G. N. & Punchimudiyanse, M. (2020), Machine Learning Approach for the Detection of Hate Speech in Sinhala Unicode Text, *in* '2020 20th International Conference on Advances in ICT for Emerging Regions (ICTer)', IEEE, Colombo, Sri Lanka, pp. 65–70.
URL: *https://ieeexplore.ieee.org/document/9325493/*

Sandaruwan, H. M. S. T., Lorensuhewa, S. A. S. & Kalyani, M. A. L. (2020), 'Identification of Abusive Sinhala Comments in Social Media using Text Mining and Machine Learning Techniques', *International Journal on Advances in ICT for Emerging Regions (ICTer)* **13**(1), 13–25.
URL: *https://account.icter.sljol.info/index.php/sljo-j-ijaicterict/article/view/7213*

Smith, I. & Thayasivam, U. (2019), Language Detection in Sinhala-English Code-mixed Data, *in* '2019 International Conference on Asian Language Processing (IALP)', IEEE, Shanghai, Singapore, pp. 228–233.
URL: *https://ieeexplore.ieee.org/document/9037680/*

Sossi Alaoui, S., Farhaoui, Y. & Aksasse, B. (2022), 'Hate speech detection using text mining and machine learning', *International Journal of Decision Support System Technology* **14**, 1–20.

Subramani, D. K., Hemalatha, P., Vinmathi, D., Kavya, S. & T Aswini (2023), 'Prediction Of Hate Speech Classification Using Secured Supervised Machine Learning With Nlp', **30**(5).

Tarwani, S., Jethanandani, M. & Kant, V. (2019), Cyberbullying Detection in Hindi-English Code-Mixed Language Using Sentiment Classification, *in* M. Singh, P. Gupta, V. Tyagi, J. Flusser, T. Ören & R. Kashyap, eds, 'Advances in Computing and Data Sciences', Vol. 1046, Springer Singapore, Singapore, pp. 543–551. Series Title: Communications in Computer and Information Science.
URL: *http://link.springer.com/10.1007/978-981-13-9942-8*$_5$1

Weerathunga, C. (2020), 'sinhala-hate-speech-detection'. original-date: 2020-01-25T06:13:12Z.
URL: *https://github.com/chashikajw/sinhala-hate-speech-detection*

Zhang, H., Cheng, Y.-C., Kumar, S., Huang, W. R., Chen, M. & Mathews, R. (2022), 'Capitalization Normalization for Language Modeling with an Accurate and Efficient Hierarchical RNN Model'. Version Number: 1.
URL: *https://arxiv.org/abs/2202.08171*

Zhang, Z., Robinson, D. & Tepper, J. (2018), Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network, *in* A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai & M. Alam, eds, 'The Semantic Web', Vol. 10843, Springer International Publishing, Cham, pp. 745–760. Series Title: Lecture Notes in Computer Science.
URL: *https://link.springer.com/10.1007/978-3-319-93417-4*$_4$8

Ziyaden, A., Yelenov, A., Hajiyev, F., Rustamov, S. & Pak, A. (2024), 'Text data augmentation and pre-trained Language Model for enhancing text classification of low-resource languages', *PeerJ Computer Science* **10**, e1974.
URL: *https://peerj.com/articles/cs-1974*

Appendix A: Datasets Analysis

Name	Details	No of Data	Resource Links
SOLD and SEMI-SOLD (2024)	• SOLD consists of 10,000 Twitter posts manually annotated for offensive and non-offensive content at both sentence-level and token-level. • It is the largest offensive language dataset available for Sinhala. • SemiSOLD, a larger dataset with over 145,000 Sinhala tweets, is also introduced, annotated using a semi-supervised approach. • Annotation scheme includes sentence-level labeling for offensive and non-offensive posts and token-level labeling for tokens contributing to offense. • Data is available through HuggingFace and can be loaded into pandas dataframes.	10 000 / 145 000	DataSet Sold: https://huggingface.co/datasets/sinhala-nlp/SOLD DataSet Semi Sold: https://huggingface.co/datasets/sinhala-nlp/SemiSOLD
chashikajw / sinhala-hate-speech-detection (2020)	• The data set was done for both Sinhala English mixed content. • The data were annotated as offensive data and non offensive data using a bag of words approach and linear regression model. • The data set contains 2520 data rows.	2520	Github: https://github.com/chashikajw/sinhala-hate-speech-detection/tree/master
renuka-fernando/sinhalese_language_racism_detection (2019)	• Data collection is done by using both Twitter Standard and Premium search APIs. • Tweets were searched with pre-identified key words collected via surveys and experts. • Total tweets: 1411 (Neutral: 1081, Racist: 108, Sexism: 222)	1412	Github: https://github.com/renuka-fernando/sinhalese_language_racism_detection
Sinhala Unicode Hate Speech (2020)	• Sinhala Annotated Data Set Based on Sinhala Unicode based comments on Facebook. • label which can be used to identify whether the comment is a hate speech or not.	6309	https://www.kaggle.com/datasets/sahanjayasuriya/sinhala-unicode-hate-speech/data
NLPC-UOM/ Sinhala-English-Code-Mixed-Code-Switched-Dataset (2022)	• Sinhala English mixed content and Annotated dataset. • There are both sentence level annotations and word level annotations. • This dataset contains 10,000 comments that have been annotated at the sentence level for sentiment analysis, humor detection, hate speech detection, aspect identification, and language identification.	10 000	https://huggingface.co/datasets/NLPC-UOM/Sinhala-English-Code-Mixed-Code-Switched-Dataset
oshadhi-liyanage / Hate-Speech-Dataset (2021)	• Sinhala-English code mixed language (Singlish). • Offensive and Non-Offensive are annotated as 1 and 0.	1375	Github: https://github.com/oshadhi-liyanage/Hate-Speech-Dataset

Figure A.1: Datasets feasibility analysis

Appendix B: Code Level Implementations

```
import pandas as pd
import numpy as np
import re
import nltk
import os  # For path operations
import pickle  # For tokenizer
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import
    pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM,
    Bidirectional, Dense, Dropout, Conv1D, GlobalMaxPooling1D
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.utils import resample
from sklearn.metrics import accuracy_score,
    classification_report, confusion_matrix
import chardet  # For encoding detection
from nltk.util import ngrams
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from tensorflow.keras.callbacks import EarlyStopping

# Detect the file encoding
with open("StopWords_425.txt", 'rb') as f:
    result = chardet.detect(f.read())
    encoding = result['encoding']
    print(f"Detected encoding: {encoding}")

# Load stopwords from file
def load_stopwords(file_path):
    with open(file_path, 'r', encoding=encoding) as f:
        return set(f.read().splitlines())

# Load suffixes from file
def load_suffixes(file_path):
    with open(file_path, 'r', encoding=encoding) as f:
        return set(f.read().splitlines())

# Load word variations from CSV file
def load_word_variations(file_path):
    word_variations = {}
```

```
        # Read the CSV file
        df_variations = pd.read_csv(file_path)
        for _, row in df_variations.iterrows():
            word = row['word']
            variations = row['Spelling variations'].split(' | ')  # Split variations by ' | '
            word_variations[word] = set(variations)
        return word_variations

    # Function to standardize words using word variations dictionary
    def standardize_word(word, word_variations):
        for base_word, variations in word_variations.items():
            if word in variations:
                return base_word
        return word

    # Load Emoji Weights
    def load_emoji_weights(filepath):
        try:
            df_emoji = pd.read_csv(filepath)
            # Convert to dictionary: {emoji: weight}
            emoji_weights = dict(zip(df_emoji['Emoji'], df_emoji['weight']))
            return emoji_weights

        except FileNotFoundError:
            print(f"Error: Emoji weights file not found at {filepath}")
            return {}  # Return empty dict to avoid errors later
        except Exception as e:
            print(f"Error loading emoji weights: {e}")
            return {}

    def tokenize_with_bigrams(text):
        """Tokenizes text into unigrams and bigrams, preserving emoji tokens."""

        # Split the text, preserving the emoji tokens:
        parts = re.split(r'(__EMOJI_WEIGHT_[0-9_]+__|\s+)', text)  # Important
        # Remove empty strings and whitespace-only strings from the split result
        parts = [part for part in parts if part.strip()]

        unigrams = []
        for part in parts:
            if part.startswith('__EMOJI_WEIGHT_'):
                unigrams.append(part)  # Keep emoji tokens whole
            else:
                unigrams.extend(word_tokenize(part))  # Tokenize other parts with nltk

        bigrams = list(ngrams(unigrams, 2))
        bigram_strings = [" ".join(bigram) for bigram in bigrams]
```

```
94              return unigrams + bigram_strings
95
96          def remove_suffixes(word, suffixes):
97              for suffix in sorted(suffixes, key=len, reverse=True):  # Sort longest suffix first
98                  if word.endswith(suffix):
99                      root_word = word[:-len(suffix)].strip()  # Remove suffix
100
101                     # Count Sinhala characters in the remaining word
102                     sinhala_chars = re.findall(r'[\u0D9A-\u0DC6]', root_word)
103                     if len(sinhala_chars) >= 2:
104                         return root_word  # Valid reduction, return root word
105                     else:
106                         return word  # Not enough Sinhala characters, keep the original
107
108             return word  # No suffix matched, return original
109
110         # Function to clean and preprocess text
111         def preprocess_text(text, emoji_weights, high_weight_threshold=7.0):
112             processed_text = text
113             has_high_weight_emoji = 0
114
115             # --- Emoji Replacement (FIRST) ---
116             for emoji, weight in emoji_weights.items():
117                 if emoji in processed_text:
118                     weight_str = str(weight).replace('.', '_')
119                     replacement_token = f"__EMOJI_WEIGHT_{weight_str}__"
120                     processed_text = processed_text.replace(emoji, f" {replacement_token} ")  # Replace emojis
121                     if weight >= high_weight_threshold:
122                         has_high_weight_emoji = 1
123             emoji_feature = has_high_weight_emoji
124
125             processed_text = re.sub(r'[^A-Za-z0-9_\u0D80-\u0DFF]+', ' ', processed_text)
126             processed_text = re.sub(r'\s+', ' ', processed_text).strip()
127             processed_text = processed_text.lower()
128
129             words = processed_text.split();
130
131             # Standardize words
132             processed_words = [standardize_word(word, word_variations) for word in words if word not in stop_words_custom]
133
134             # Remove stopwords and apply suffix removal
135             processed_words = [remove_suffixes(word, suffixes) for word in processed_words if word not in stop_words_custom]
136
137             processed_text = ' '.join(processed_words)
```

```
138
139             # --- Tokenization (with Bigrams and Emoji Preservation) ---
140             words = tokenize_with_bigrams(processed_text)
141             processed_text = ' '.join(words)
142             return processed_text, emoji_feature
143
144         # --- Data Loading and setup ---
145         df = pd.read_csv('hate_speech_dataset_v2.csv') #Give your
                dataset file
146
147         # Load external stopwords and suffixes.
148         stop_words_custom = load_stopwords('StopWords_425.txt')
149         suffixes = load_suffixes('Suffixes-413.txt')
150         word_variations = load_word_variations('word_variations.csv')
151         emoji_weights = load_emoji_weights("emoji_weights.csv")
152
153         # --- Apply Preprocessing ---
154         # Create 'Comment_cleaned' and 'emoji_feature' columns
155         df['Comment_cleaned'], df['emoji_feature'] = zip(*df['Comment'].
                apply(lambda x: preprocess_text(x, emoji_weights)))
156
157         text_with_emoji ="Hto  "
158         processed_text, emoji_feature = preprocess_text(text_with_emoji,
                 emoji_weights)
159         print(f"Original: {text_with_emoji}")
160         print(f"Processed: {processed_text}")
161         print(f"Emoji Feature: {emoji_feature}")
162
163         # --- Data Balancing ---
164         # Separate majority and minority classes
165         df_majority = df[df['Is offensive'] == 0]
166         df_minority = df[df['Is offensive'] == 1]
167
168         # Upsample minority class
169         df_minority_upsampled = resample(df_minority,
170                                          replace=True,     # sample with
                                              replacement
171                                          n_samples=len(df_majority),
                                             # to match majority class
172                                          random_state=42) # reproducible
                                              results
173
174         # Combine majority class with upsampled minority class
175         df_balanced = pd.concat([df_majority, df_minority_upsampled])
176
177         # --- Tokenization and Padding ---
178         tokenizer = Tokenizer(num_words=10000, oov_token='<OOV>')
179         tokenizer.fit_on_texts(df_balanced['Comment_cleaned'])
180
181         # Convert text to sequences
182         X = tokenizer.texts_to_sequences(df_balanced['Comment_cleaned'])
183         X = pad_sequences(X, maxlen=50, padding='post', truncating='post
                ')  # Ensure uniform length
184
```

```
        # Convert labels to numpy array
        y = np.array(df_balanced['Is offensive'])

        # --- Train-test split ---
        X_train, X_test, y_train, y_test = train_test_split(X, y,
            test_size=0.2, random_state=42)

        # --- Model Training and Evaluation Functions ---
        def train_and_evaluate_keras_model(model, X_train, y_train,
            X_test, y_test, model_name):
            """Trains and evaluates Keras models."""
            early_stopping = EarlyStopping(monitor='val_loss', patience
                =3, restore_best_weights=True)
            model.compile(loss='binary_crossentropy', optimizer=Adam(
                learning_rate=0.001), metrics=['accuracy'])
            history = model.fit(X_train, y_train, epochs=20, batch_size
                =64, validation_data=(X_test, y_test), callbacks=[
                early_stopping])
            loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
            print(f"{model_name} Accuracy: {accuracy:.4f}")
            y_pred_prob = model.predict(X_test)
            y_pred = (y_pred_prob > 0.5).astype("int32")
            print(f"{model_name} Classification Report:\n{
                classification_report(y_test, y_pred)}")
            cm = confusion_matrix(y_test, y_pred)
            plt.figure(figsize=(6, 4))
            sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=['Non-Offensive', 'Offensive'], yticklabels
                =['Non-Offensive', 'Offensive'])
            plt.xlabel('Predicted Label')
            plt.ylabel('True Label')
            plt.title(f'{model_name} Confusion Matrix')
            plt.show()

        def train_and_evaluate(model, X_train, y_train, X_test, y_test,
            model_name):
            """Trains and evaluates traditional machine learning models.
                """
            model.fit(X_train, y_train)
            y_pred = model.predict(X_test)
            accuracy = accuracy_score(y_test, y_pred)
            print(f"{model_name} Accuracy: {accuracy:.4f}")
            print(f"{model_name} Classification Report:\n{
                classification_report(y_test, y_pred)}")
            cm = confusion_matrix(y_test, y_pred)
            plt.figure(figsize=(6, 4))
            sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=['Non-Offensive', 'Offensive'], yticklabels
                =['Non-Offensive', 'Offensive'])
            plt.xlabel('Predicted Label')
            plt.ylabel('True Label')
            plt.title(f'{model_name} Confusion Matrix')
            plt.show()

```

```
# --- CNN Model ---
cnn_model = Sequential([
    Embedding(input_dim=10000, output_dim=128, input_length=50),
    Conv1D(128, 5, activation='relu'),
    GlobalMaxPooling1D(),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])
train_and_evaluate_keras_model(cnn_model, X_train, y_train,
    X_test, y_test, "CNN")

# --- LSTM Model ---
lstm_model = Sequential([
    Embedding(input_dim=10000, output_dim=128, input_length=50),
    LSTM(64),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])
train_and_evaluate_keras_model(lstm_model, X_train, y_train,
    X_test, y_test, "LSTM")

# --- BiLSTM Model ---
bilstm_model = Sequential([
    Embedding(input_dim=10000, output_dim=160, input_length=50),
    Bidirectional(LSTM(32, return_sequences=True)),
    Dropout(0.5),
    Bidirectional(LSTM(32)),
    Dense(64, activation='relu'),
    Dropout(0.4),
    Dense(1, activation='sigmoid')
])
train_and_evaluate_keras_model(bilstm_model, X_train, y_train,
    X_test, y_test, "BiLSTM")

# --- Combined CNN and BiLSTM Model ---
from tensorflow.keras.layers import Embedding, LSTM,
    Bidirectional, Dense, Dropout, Conv1D, GlobalMaxPooling1D,
    Input, concatenate  # Import Input and concatenate
from tensorflow.keras.models import Sequential, Model  # Import
    Model

def create_cnn_bilstm_model(input_dim, output_dim, input_length)
    :
    input_layer = Input(shape=(input_length,))
    embedding_layer = Embedding(input_dim=input_dim, output_dim=
        output_dim, input_length=input_length)(input_layer)

    # CNN Branch
    cnn_branch = Conv1D(128, 5, activation='relu')(
        embedding_layer)
    cnn_branch = GlobalMaxPooling1D()(cnn_branch)

```

```
        # BiLSTM Branch
        bilstm_branch = Bidirectional(LSTM(32, return_sequences=True))(embedding_layer)
        bilstm_branch = Dropout(0.5)(bilstm_branch)
        bilstm_branch = Bidirectional(LSTM(32))(bilstm_branch)

        # Concatenate Branches
        merged = concatenate([cnn_branch, bilstm_branch])

        # Dense Layers
        dense_layer = Dense(64, activation='relu')(merged)
        dropout_layer = Dropout(0.4)(dense_layer)
        output_layer = Dense(1, activation='sigmoid')(dropout_layer)

        model = Model(inputs=input_layer, outputs=output_layer)
        return model

cnn_bilstm_model = create_cnn_bilstm_model(input_dim=10000, output_dim=128, input_length=50)
cnn_bilstm_accuracy = train_and_evaluate_keras_model(cnn_bilstm_model, X_train, y_train, X_test, y_test, "CNN-BiLSTM")

# --- TF-IDF for Traditional ML Models ---
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X_tfidf = tfidf_vectorizer.fit_transform(df_balanced['Comment_cleaned'])
X_train_tfidf, X_test_tfidf, y_train_tfidf, y_test_tfidf = train_test_split(
    X_tfidf, df_balanced['Is offensive'], test_size=0.2, random_state=42
)

# --- Train and Evaluate Traditional ML Models ---
# Naive Bayes
nb_model = MultinomialNB()
train_and_evaluate(nb_model, X_train_tfidf, y_train_tfidf, X_test_tfidf, y_test_tfidf, "Naive Bayes")

# SVM
svm_model = SVC(probability=True)
train_and_evaluate(svm_model, X_train_tfidf, y_train_tfidf, X_test_tfidf, y_test_tfidf, "SVM")

# Logistic Regression
lr_model = LogisticRegression(max_iter=1000)
train_and_evaluate(lr_model, X_train_tfidf, y_train_tfidf, X_test_tfidf, y_test_tfidf, "Logistic Regression")

# Random Forest
rf_model = RandomForestClassifier()
train_and_evaluate(rf_model, X_train_tfidf, y_train_tfidf, X_test_tfidf, y_test_tfidf, "Random Forest")

```

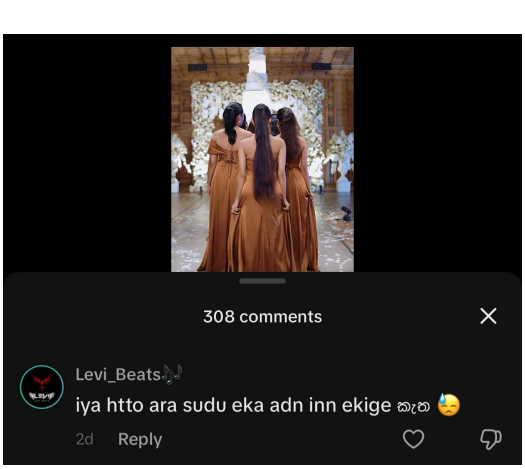
```
313         def predict_all_models(comment, emoji_weights, tokenizer,
                cnn_bilstm_model, bilstm_model, cnn_model, lstm_model,
                tfidf_vectorizer, nb_model, svm_model, lr_model, rf_model):
314             """
315             Predicts hate speech using all trained models and returns a
                    DataFrame with predictions.
316
317             Args:
318                 comment (str): The text comment to predict.
319                 emoji_weights (dict): Dictionary of emoji weights.
320                 tokenizer (Tokenizer): Keras tokenizer used for text
                        preprocessing.
321                 bilstm_model (Sequential): Trained BiLSTM model.
322                 cnn_model (Sequential): Trained CNN model.
323                 lstm_model (Sequential): Trained LSTM model.
324                 tfidf_vectorizer (TfidfVectorizer): TF-IDF vectorizer.
325                 nb_model (MultinomialNB): Trained Naive Bayes model.
326                 svm_model (SVC): Trained SVM model.
327                 lr_model (LogisticRegression): Trained Logistic
                        Regression model.
328                 rf_model (RandomForestClassifier): Trained Random Forest
                         model.
329
330             Returns:
331                 pd.DataFrame: DataFrame with predictions from all models
                        .
332             """
333             processed_text, _ = preprocess_text(comment, emoji_weights)
334             comment_sequence = tokenizer.texts_to_sequences([
                    processed_text])
335             comment_padded = pad_sequences(comment_sequence, maxlen=50,
                    padding='post', truncating='post')
336             tfidf_comment = tfidf_vectorizer.transform([processed_text])
337
338             cnn_bilstm_pred = cnn_bilstm_model.predict(comment_padded)
                    [0][0]
339             bilstm_pred = bilstm_model.predict(comment_padded)[0][0]
340             cnn_pred = cnn_model.predict(comment_padded)[0][0]
341             lstm_pred = lstm_model.predict(comment_padded)[0][0]
342             nb_pred = nb_model.predict_proba(tfidf_comment)[0][1]
343             svm_pred = svm_model.predict_proba(tfidf_comment)[0][1]
344             lr_pred = lr_model.predict_proba(tfidf_comment)[0][1]
345             rf_pred = rf_model.predict_proba(tfidf_comment)[0][1]
346
347             predictions = {
348                 "CNN-BiLSTM": "Offensive" if cnn_bilstm_pred > 0.5 else
                        "Non-Offensive",
349                 "BiLSTM": "Offensive" if bilstm_pred > 0.5 else "Non-
                        Offensive",
350                 "CNN": "Offensive" if cnn_pred > 0.5 else "Non-Offensive
                        ",
351                 "LSTM": "Offensive" if lstm_pred > 0.5 else "Non-
                        Offensive",
```

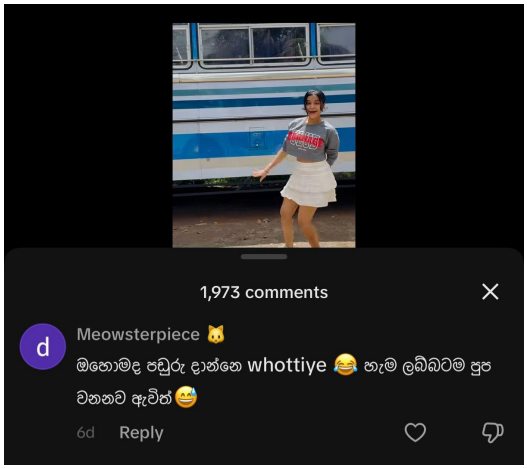
```
                "Naive Bayes": "Offensive" if nb_pred > 0.5 else "Non-
                    Offensive",
                "SVM": "Offensive" if svm_pred > 0.5 else "Non-Offensive
                    ",
                "Logistic Regression": "Offensive" if lr_pred > 0.5 else
                     "Non-Offensive",
                "Random Forest": "Offensive" if rf_pred > 0.5 else "Non-
                    Offensive",
            }

            return pd.DataFrame(predictions, index=[comment])

        # Example usage
        input_comment = "oyanam maha   godak"
        predictions_df = predict_all_models(input_comment, emoji_weights
            , tokenizer, cnn_bilstm_model, bilstm_model, cnn_model,
            lstm_model, tfidf_vectorizer, nb_model, svm_model, lr_model,
             rf_model)
        print(predictions_df)
```

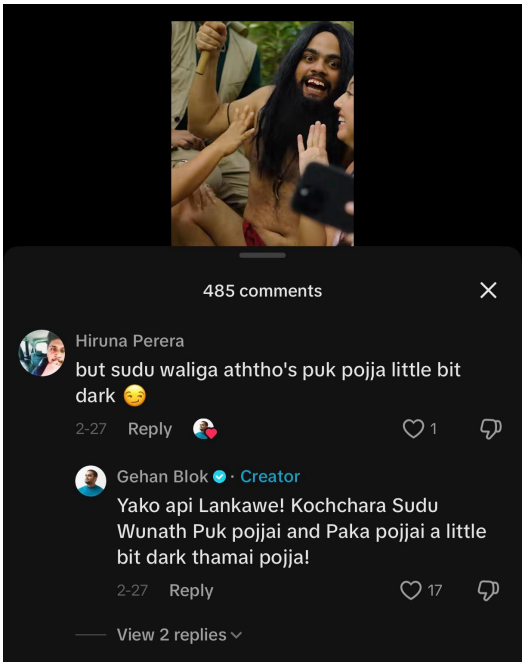
Code B.1: Full Python code for model training and evaluation

Appendix C: Recent Social Media Posts

These comments illustrate the ongoing discussions from various social media platforms.







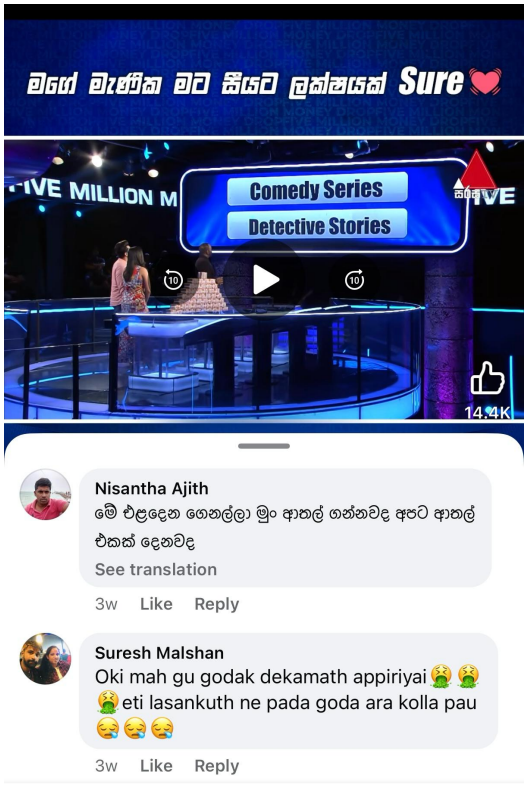


Figure C.2: Recent social media posts

Appendix D: Recent Social Media Posts Evaluated Through the Application

Evaluation results from the application for two recent social media posts. The first example demonstrates a borderline offensive comment, while the second illustrates a supportive, non-offensive remark.

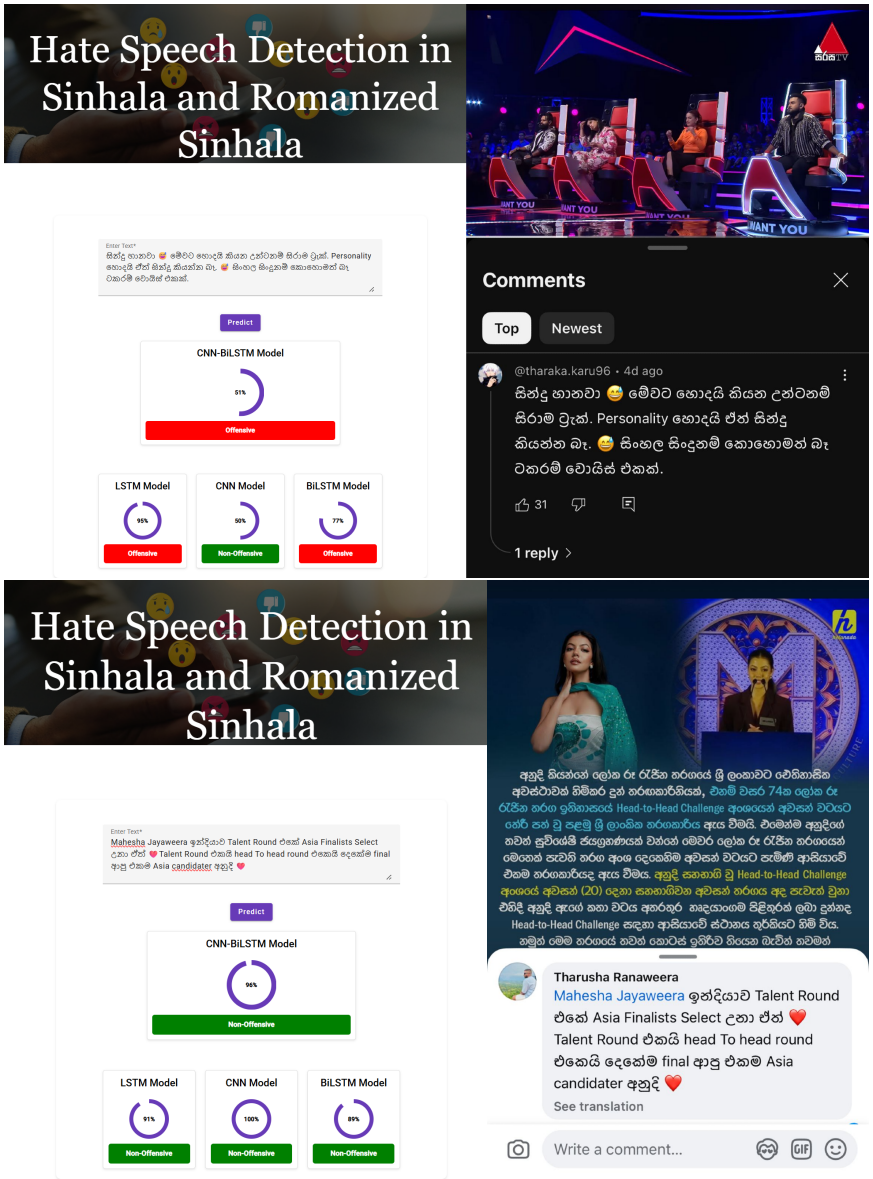


Figure D.1: Recent social media posts evaluated through the application