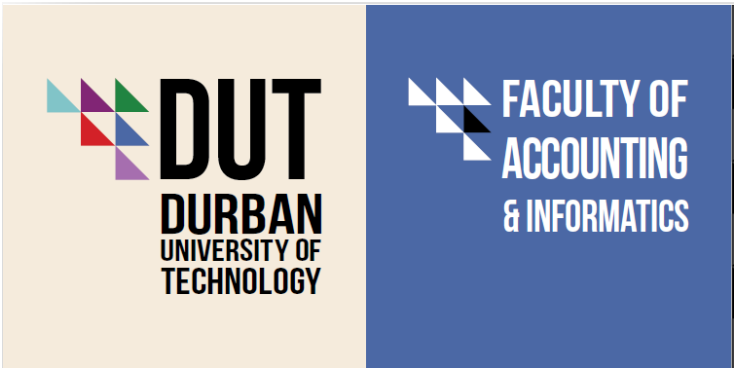

DURBAN UNIVERSITY OF TECHNOLOGY

A Comparative Study of Deep Learning Algorithms for Hate Speech Detection on Twitter

By

Raymond Tapiwa Mutanga

21959486

A dissertation submitted in fulfilment of the requirement for the Master of Information and Communications Technology degree

Faculty of Accounting and Informatics, Department of Information Technology, Postgraduate Studies

Supervisor: Dr. N. Naicker

Co-Supervisor: Prof. O.O. Olugbara

2021

DECLARATION

I, *Raymond T Mutanga*, declare that:

(i) The research reported in this dissertation, except where otherwise indicated, is my original research.

(ii) This dissertation has not been submitted for any degree or examination at any other university.

(iii) This dissertation does not contain other persons' data, pictures, graphs or other information unless specifically acknowledged as being sourced from other persons.

(iv) This dissertation does not contain other persons' writing unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:

- Their words have been re-written, but the general information attributed to them has been referenced.
- Where their exact words have been used, their writing has been placed inside quotation marks and referenced.

(v) This dissertation does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the dissertation and in the Reference Section of this dissertation.

_

Signature: *Name of Student*

Date: 18 March 2021

Approved for final submission

Supervisor/Promoter

Dr N Naicker (PhD)

29 October 2021

Date

ACKNOWLEDGEMENTS

I am immensely grateful to my supervisor, Dr N Naicker, whose mentorship on both professional and personal levels was crucial to completing this work. I sincerely appreciate all the valuable time he spent guiding my research work. I was fortunate to work with such a kind and encouraging supervisor over the years.

Professor O Olugbara, my co-supervisor, deserves special mention for his insightful and positive advice. Despite his busy schedule as the Executive Dean, he always created time for guiding my research work. I was fortunate to learn from such a humble machine learning genius.

I extend my gratitude to Dr Constance Israel and the Writing Studio for proofreading and language editing services.

I am grateful to colleagues at the Durban University of Technology and the ICT and Society Research group members for their support and motivation. The provision of GPU clusters by Baba Geoff Mapiye is greatly appreciated. The material support and motivation from my study partners, namely Kuda Zvareva, Fannie Ndlovu, Freedom Khubisa, Sammy Frimpomg, Stera Mwamba, Kandolo KaMuzombo, Iponeng Chimidza and Mayowa Sowofora, is gratefully acknowledged. I am grateful to the Montlands Library staff for allowing me to use their facilities when the university library was undergoing renovations.

The support from the Durban University of Technology in offering me admission to pursue this degree is greatly appreciated. It indeed was a great experience undertaking my postgraduate studies at such a renowned academic institution.

Lastly, I appreciate my family and friends for their unwavering support towards the completion of my studies. Special mention goes to my mother Mrs A. Mutanga for her encouragement and prayers.

"There is a fine line between free speech and hate speech. Free speech encourages debate whereas hate speech incites violence."

Newton Lee

DEDICATION

To my late father Mr. S Mutanga.

ABSTRACT

Hate speech is an undesirable phenomenon with severe psychological and physical consequences. The emergence of mobile computing and Web 2.0 technologies has increasingly facilitated the spread of hate speech. The speed, accessibility and anonymity afforded by these tools present challenges in enforcing measures that minimise the spread of hate speech. The continued dissemination of hate speech online has triggered the development of various machine learning techniques for its automated detection. However, current approaches are inadequate because of further challenges such as the use of domain-specific language and language subtleties. Recent studies on automated hate speech detection have focused on the use of deep learning as a possible solution to these challenges. Although some studies have explored deep learning methods for hate speech detection, there are no studies that critically compare and evaluate their performance.

This work investigates the use of deep learning algorithms as possible solutions to hate speech detection on Twitter. Three taxonomic classes of deep learning algorithms, namely, Traditional deep learning algorithms, Traditional algorithms with partial attention mechanism and Transformer models, which are entirely based on the attention mechanism, are evaluated for performance, using two publicly available corpora. One of the datasets contained 24 786 tweets annotated into three different classes, while the other dataset contained 2300 tweets annotated into two different classes. All tweets from the two datasets were first preprocessed to rid of them of characters and words deemed irrelevant to the classification decision, for instance, hashtags, stop words and punctuation marks. The preprocessed text was then transformed into feature vectors which were used as input for deep learning algorithms explored in this study. A series of experiments were performed to measure the performance of the deep learning algorithms in hate speech detection. The algorithms were tested on a wide spectrum of tweets containing different forms of hate speech. The efficacy of the deep learning algorithms was objectively evaluated using six state-of-the-art statistical evaluation metrics: precision, F-measure, recall, accuracy, Mathews correlation coefficient and area under the curve. The results from this study indicate that variations in parameters do not impact the efficacy of deep learning algorithms by the same proportions. The findings of this empirical study, therefore, provide deep-learning practitioners with a better understanding of the adaptation of robust deep-learning techniques for automated hate speech detection tasks.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

The following are the most important acronyms in the study:

AUC	Area under the Curve
BERT	Bidirectional Encoder Representations from Transformers
BPN	Back Propagation
CCE	Categorical Cross Entropy
CNN	Convolutional Neural Network
DL	Deep Learning
DISTILBERT	Distilled BERT
FFNN	Feed Forward Neural Networks
GRU	Gated Recurrent Unit
HSO	Hate Speech and Offensive Language
LSTM	Long Short-Term Memory
MHA	Multi-Headed Attention
MCC	Mathews Correlation Coefficient
ML	Machine Learning
MLM	Masked Language Modelling
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
NSP	Next Sentence Prediction
PLM	Pretrained Language Models
POS Tagging	Part of Speech Tagging
ROBERTA	Robustly optimised BERT Approach
Seq2seq	Sequence to Sequence

LIST OF TERMS

The following terms as related to the study are explained:

Hate Speech - Language that negatively stereotypes a particular group or individual on the basis of characteristics such as sexuality, nationality, religious affiliation and race.

Social Media - Internet platforms which allow users to quickly access, generate and share content with the public, such as Twitter and Facebook.

Natural Language Processing - An artificial intelligence division that allows machines to comprehend human language.

Deep Learning - A machine learning technique designed imitate the function of the brain by using a large number of hidden non-linear processing layers for feature extraction and representation.

Attention - Part of a neural network that allows it to dynamically highlight relevant features of the input data, for example, a sequence of textual elements in text processing.

Transformers - A feed-forward deep learning architecture based entirely on attention instead of convolutions and recurrence, where the transformer is able to capture long-term sequential data dependencies while allowing parallelization.

Machine Learning - A group of techniques that allows computers to learn from data without being explicitly programmed.

Classical machine learning - A machine learning technique based on shallow models trained on high dimensional sparse vectors, with methods depending on carefully selected features.

Feature Engineering - The process of combining text processing methods and domain awareness for feature extraction.

Corpus - A large and structured set of machine-readable text that has been produced in a natural communicative setting (plural is corpora).

Optimiser - Techniques for modifying a neural network's properties, for example, learning rate and weights, to minimise losses, and to get results faster.

CHAPTER ONE

INTRODUCTION AND BACKGROUND TO THE STUDY

1.1 Introduction

Hate speech is a collective term for utterances or statements which disseminate, trigger, encourage or justify hatred, segregation and violence against an individual or group of individuals (Whillock and Slayden 1995). Typical forms of hate speech include racism, tribalism, sexism, xenophobia and islamophobia. No single hate speech definition has been unanimously accepted as the gold standard by the research community. However, various researchers concur that it targets underprivileged persons in a way that may be deemed harmful to them (MacAvaney *et al*. 2019). Hate speech promotes prejudice, which can undermine people, sow seeds of discord between different societal groups and eventually lead to deeper social cohesion problems (Pálmadóttir and Kalenikova 2018). Divisions in societal cohesion and attacks on the egos of hate speech victims have the potential to fuel social unrest and hate crimes (Bleich 2011). For example, hate speech fueled xenophobic attacks in the KwaZulu-Natal province of South Africa, where seven immigrants died and approximately 5000 others displaced between March 2015 and May 2015 (Aljazeera 2021).

In the past, the propagation of hate speech has been achieved mainly through the use of traditional electronic and print media such as newspapers, radio and television. For example, the holocaust, which resulted in mass killings of Jews, also had its roots in hate speech propaganda, which was propagated using the technologies of those days. Furthermore, hate speech leading to the Rwandan genocide in 1994 was spread through radio and print media (Schabas 2000). Since then, communication technologies have evolved to include the Internet and mobile devices, allowing rapid exchange of information.

The emergence of Web 2.0 tools such as Twitter and Facebook have transformed communication by allowing users in different parts of the world to seamlessly compile, collaborate and share their content with others. Given the meteoric rise of user-generated content on platforms such as Twitter, the volume of online hate speech is growing (Schmidt and Wiegand 2017). Platforms such as Twitter enable users to instantaneously post different kinds of messages in different formats such as text, images, videos and metadata, sometimes in the form of emojis, mentions, emoticons, uniform resource locators and hashtags for social media users to view, comment and share with other users (Kursuncu *et al*. 2019). Tweets are generally rife with idioms, acronyms, phonemes, homophones and figures of speech like onomatopoeia, which can complicate the understanding of hateful speech. Moreover, the

Twitter restrictions on the number of allowable characters encourage the usage of unconventional and incomprehensible abbreviations, misspellings, grammatical errors and slang terminologies.

Millions of tweets are generated daily, enabling the creation of datasets large enough for analysis (Gaumont, Panahi and Chavalarias 2018). In 2017 alone, Twitter had 330 million active users per month, and 157 million of the users were active daily, sharing approximately 500 million tweets each day (Kursuncu *et al*. 2019). It is projected that at least one-third of the world population will be using social media by the end of 2021 (Pereira-Kohatsu *et al*. 2019).

The large volumes of harmful messages posted on Twitter necessitate the development of techniques to curb their continued dissemination. To address this, some governments in the developed world have instituted laws to prohibit hate speech in face-to-face conversations and on the internet media (Davidson *et al.* 2017). Although such legislation acts as a deterrent, it does not entirely stop determined individuals from posting content containing hate speech.

Besides the broader societal implications of hate speech, uncontrolled propagation also negatively impacts the reputation of online host platforms such as Twitter and Facebook (Yasseri and Vidgen 2019). In response to this challenge, organisations such as Facebook and Twitter currently have employees dedicated to the task of manually deleting content perceived to contain hate speech. In addition, Facebook and Twitter users are advised to label and report content they deem unsuitable or harmful to society. However, such interventions are laborious for human annotators, and they are also prone to subjective human judgement (Pitsilis, Ramampiaro and Langseth 2018). These methods are stressful for human annotators, and they have been linked to post-traumatic stress disorders (Miok *et al*. 2020). Critics have argued that the use of human annotators is insufficient since the messages are only deleted after they have been posted and possibly after the messages have inflicted harm already (Ullmann and Tomalin 2020). Hate speech may also be expressed in slang or other languages that the annotator may not understand. There are 7117s distinct languages used for communicating verbally and in written form (*Ethnologue Languages of the world* 2021). Given this high number of languages, it is not practical for human annotators to understand all the languages used in social media.

Machine learning-based hate speech recognition models have been proposed in response to the shortcomings of human annotators and legislation. Classical machine learning algorithms and deep learning algorithms are the two taxonomic subclasses of machine learning algorithms.

Classical algorithms make use of handcrafted features, which consume much time and are ordinarily insufficient (Young *et al*. 2018). As a result, classical algorithms fail to capture semantic and syntactic representations of text effectively. Deep learning algorithms, on the other hand, carry out end-to-end training, allowing the model to encode salient feature representations. Deep Neural Networks have been proven to outperform classical models based on n-gram features (Holmes and Jain 2006). Furthermore, deep learning algorithms such as Recurrent Neural Networks (RNN) are capable of preserving sequential information over periods of time, which allows easier integration of contextual information in text classification tasks (Wang, Li and Xu 2018). Although context helps distinguish hate from non-hate texts, it has largely been excluded from detection models (Gao and Huang 2017). Deep learning models can capture complex data representations making them applicable to the identification of hate speech, where the language used is highly ambiguous. However, no studies have focused on objective comparative evaluations of deep learning algorithms, making it difficult to understand the most appropriate algorithms in addressing the hate speech phenomenon in online spaces (Fortuna and Nunes 2018b).

This work, therefore, was aimed at finding the best performing deep learning algorithm for detecting hate speech. To achieve this, an experimental comparison of deep learning algorithms for hate speech detection was carried out. Ten deep learning algorithms representing traditional deep learning algorithms and recent transformer-based algorithms were selected for investigation, namely, Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU), Multiplayer Perceptron (MLP), Bidirectional Encoder Representations from Transformers (BERT), Distilled Bidirectional Encoder Representations from Transformers (DistilBERT), Robustly optimised Bidirectional Encoder Representations from Transformers approach (RoBERTa) and XLNet.

1.2 Statement of the Problem

In recent years, hate speech detection has received considerable interest from governments, academics and social media companies (Banks 2010; Schmidt and Wiegand 2017; Brown 2018). However, current solutions to the problem of hate speech detection are inadequate (Mondal, Silva and Benevenuto 2017; Fortuna and Nunes 2018b). To complement existing automated solutions of hate speech detection, this study seeks to carry out an experimental

comparison of deep learning algorithms for hate speech detection. Deep learning has been chosen ahead of classical machine learning due to its ability to handle unstructured input, its ability to deal with sparse features (dimensionality) as well as its reduced need for handcrafted features (Kowsari *et al.* 2019). Major deep learning algorithms that have been used in related NLP tasks, namely, Convolutional Neural Networks, Recurrent Neural Networks, Long Short Term Memory, Recursive Neural Networks and the Multilayer Perceptron, are experimentally compared in this study. These algorithms have performed well in tasks such as Chunking and Parts of Speech Tagging (Collobert *et al.* 2011).

There is no comparative evaluation of hate speech recognition algorithms in the extant literature. A recent survey on the detection of hate speech also noted the absence of comparative evaluation on hate speech detection as an area that needs to be addressed (Fortuna and Nunes 2018a). Therefore scope exists for the investigation proposed by this research topic.

Although traditional deep learning models have better predictive accuracy over classical algorithms in hate speech detection, the length of the text sequences they can handle is limited (Mhammedi *et al.* 2017). It is widely agreed that RNN algorithms such as LSTM have made remarkable progress towards solving the long term dependencies challenge. However, research has shown that the LSTM`s performance drops once the sequence exceeds thirty words (Bahdanau, Cho and Bengio 2014). This limitation presents a huge challenge in hate speech detection on Twitter, considering its recent increase of allowable words per tweet from 140 to 280 words. One of the major computational limitations suffered by traditional deep learning algorithms such as the CNN and the RNN, is the sequential processing of text (Minaee *et al.* 2020). Sequential processing of text requires much time and therefore leads to longer inference times. The transformer model, which is based entirely on attention, has demonstrated its ability to deal with long sequences of textual data while maximising the availability of Graphical Processing Units to make training faster through parallel processing (Vaswani *et al.* 2017). These advantages necessitate the need to include transformer-based models for hate speech recognition.

1.3 Research Aim and Objectives

The aim of this research is to investigate selected deep learning algorithms to determine the algorithm most suited to detecting hate speech on Twitter.

To achieve this aim, the following research objectives [RO] were set:

[RO1]: To comprehensively review relevant publications based on deep learning.

[RO2]: To prepare datasets for building and evaluating deep learning hate speech recognition models.

[RO3]: To experimentally evaluate the performance of deep learning algorithms on imbalanced and balanced datasets.

[RO4]: To evaluate the performance of deep learning algorithms on binary and multiclass datasets.

[RO5]: To evaluate the performance of deep learning algorithms in detecting different forms of hate speech.

1.4 Significance of the Study

It is envisaged that this study will assist in minimising the spread of hate speech in online spaces. More specifically, it will help in the following ways:

- It will help organisations that have an online presence to protect their images or brands by preventing users from posting hate speech messages on their platforms.
- The proposed solution may be adopted by government security services such as the police in tracking and apprehending people who disseminate hate speech.
- Policymakers may use the models to forecast the likelihood of hate crimes like xenophobia.

1.5 Scope and Delimitations of the Study

This study is limited in scope as follows:

- This study is limited to publications in the English language only.
- Only Twitter text data was used in this study.
- Due to time constraints, only a limited number of algorithms could be explored.
- The data used for training the models was limited to 2 publicly available hate speech datasets.
- Only deep learning techniques were explored in this study.

1.6 Research Output

This study will also lead to research products. It has already produced the following publication in an academic journal accredited by the Department of Higher Education and Training: R. Mutanga, N. Naicker, and O. Olugbara, "Hate Speech detection using Transformer Methods," *International Journal of Advanced Computer Science and Applications, vol.11, no 9, 2020. DOI:* 10.14569/IJACSA.2020.0110972

1.7 Structure of the Dissertation

This study is presented in five chapters, which are arranged in the following manner:

Chapter One: Introduction and Background to the Study

The first chapter covers the background study of the research and clearly highlights the need for automating the detection of hate speech in social media. Additionally, the identified research problems which led to the research aim and objectives of the study are clearly described in this chapter.

Chapter Two: Literature Review

A comprehensive review of relevant publications on deep learning and the societal impact of hate speech was carried out. The literature survey covers the general operation of each of the deep learning algorithms in this study, their performance in hate speech detection based on earlier studies, and their strengths and drawbacks.

Chapter Three: Research Methodology

The step-by-step methodology carried out to accomplish the formulated study aims and objectives in this research is described. The major steps involved include the acquisition and preprocessing of the dataset, feature extraction and training using deep learning algorithms, and evaluation of algorithms performance based on selected text classification metrics.

Chapter Four: Presentation of Results and Discussion

The chapter provides both qualitative and statistical metrics-based performance assessment and comparison of the selected deep learning algorithms. The results are discussed extensively, and key patterns in the results are noted, with possible explanations for the patterns.

Chapter Five: Summary, Conclusions and Implications of the Study

A summary of how each of the set objectives was met is presented. Additionally, the limitations of the study are presented and explained. Thereafter, both the practical and the research implications of the study are discussed. The chapter concludes with a discussion of open research issues and recommendations for future research.

1.8 Chapter Summary

This chapter provides a comprehensive background of the problem being solved. It touches on the societal impacts of hate speech and clearly outlines why this problem requires attention. The statement of the problem explains, in brief, the research gap which this study seeks to cover. The stated aims and objectives highlight at a high level the tasks or activities that were carried out in addressing the identified research gap. The scope of the study denotes the limits within which the study was conducted. Lastly, the dissertation synopsis lists the major chapters that make up this study. The next chapter provides an extensive literature review of earlier works as well as the direction this work has taken.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

Historically, text mining was primarily concerned with analysing documents or pieces of text-based on identified topics only (Antai 2016). As the field evolved and the Internet became more accessible, a new branch of text mining has emerged to include analysis and classification of text messages expressed on social networks such as Twitter. There are various challenges in the classification of subjective content because of language subtleties. Text posted on online platforms is generally rife with idioms, acronyms, homophones and phonemes that may complicate understanding and make the classification of text difficult. This is especially true in the realm of hate speech, where people can express their hatred through idiomatic expressions. For example, the English Idiom" kick the bucket" might refer to someone kicking the bucket physically or to someone`s death. Furthermore, the use of sarcastic messages online further complicates the comprehension of implicitly expressed hate speech. For instance, the phrase "Degenerate Anifta rioter attacking black police get a proper response" is a negative statement. However, it may be misinterpreted as a neutral statement by the classification system. An effective system must have feature vectors capable of capturing meaning from language. This can be very challenging since language is highly ambiguous and complex. Other factors that make identifying hate speech difficult include the domain of an utterance, context, author and targeted recipient (Schmidt and Wiegand 2017).

Due to the size of the Internet, automatic computer-based models have been proposed to tackle subjectivity analysis tasks such as hate speech detection. Machine learning algorithms are the gold standard solution to text classification problems such as hate speech detection. The machine learning approaches are classified into classical machine learning and deep learning. Classical machine learning is based on shallow models trained on high dimensional sparse vectors. The discriminative capability of these methods depends on manually crafted features that are often time-consuming and ineffective in capturing hidden patterns in data (Young *et al.* 2018). Classical methods depend on the careful design and conversion of text data into feature vectors used by algorithms such as Naïve Bayes (NB), Logistic Regression, Support Vector Machine (SVM) and Random Forest (RF). Many researchers have explored classical algorithms as a possible solution to the challenge of hate speech detection (Warner and Hirschberg 2012; Mehdad and Tetreault 2016; Waseem 2016; Waseem and Hovy 2016; Gao

and Huang 2017). These studies employed manually engineered features, namely n-grams, bag of words representation, comment embeddings and linguistic features.

Conversely, deep learning methods use several layers to learn hierarchical data representations. This allows for multi-level automated feature representation learning. Traditional deep learning approaches, particularly recurrent neural networks, have been the predominant techniques used for text categorisation tasks such as hate speech detection. Text is viewed as a sequence of words by Recurrent Neural Network (RNN)-based models, which are designed to capture semantics from word dependencies. Nevertheless, the basic RNN model often underperforms feedforward neural networks in text categorisation (Minaee *et al.* 2020). Long Short Term Memory (LSTM) is the most popular variant of the RNN, which is specifically designed to capture long-term dependencies. To recall values over time periods, the LSTM employs a memory cell and three gates. This property allows the LSTM to solve the vanishing gradient and exploding gradient problems inherent in the basic RNN. The shortcomings of the RNN gave birth to the attention-based models based on the correlation of individual words within a block of text. While attention-based models address some of the limitations of Recurrent Neural Networks, they can only process text sequentially, thereby impacting processing time. The latest transformer architecture allows parallel processing of input sequences while capturing long term dependencies better than earlier techniques.

The next section discusses competing definitions of what constitutes hate speech, with the aim of guiding dataset selection and the model building process. Subsequent sections of this chapter discuss various studies that have been undertaken to address hate speech identification in social media, specifically focusing on the aforementioned traditional deep learning approaches, attention-based models and the latest pre-trained transformer models.

2.2 Definitions of Hate Speech

Researchers disagree on what constitutes hate speech, and these disagreements lead to challenges in evaluating hate speech (MacAvaney *et al.* 2019). The annotation of hate speech datasets can be strengthened by a common description of hate speech, leading to improved models of hate speech detection. (Ross *et al.* 2017). However, there is no universally accepted definition due to differences in social norms, subjective individual or group interpretation, as well as context (Mossie and Wang 2020). Several academics and social media firms have

attempted to define hate speech with the aim of providing a framework for the creation of hate speech detection techniques.

Davidson *et al.* (2017) define hate speech as "Language that is used to expresses hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult members of a group". A study by de Gibert *et al.* (2018) defines hate speech as "a deliberate attack directed towards a specific group of people motivated by aspects of the group's identity". A recent study on the challenges and solutions of hate speech detection defines hate speech as "language that attacks or diminishes, that incites violence or hate against groups, based on specific characteristics such as physical appearance, religion, descent, national or ethnic origin, sexual orientation, gender identity or other, and it can occur with different linguistic styles, even in subtle forms or when humour is used."

Social media organisations such as Yahoo, Facebook and Twitter established guidelines and definitions of what constitutes hate speech on their respective platforms. These definitions are used as the basis for deciding whether posted messages can be classified as hate speech or not. For example, Facebook prohibits the posting of content that attacks users because of their gender, religion, ethnicity, sexual orientation, race, religion and country of origin. YouTube classifies as hate speech any "content that promotes segregation or humiliates an individual or group of people on the basis of the individual's or group's race, ethnicity, or ethnic origin, nationality, religion, disability, age, veteran status, sexual orientation, gender identity, or other characteristic associated with systematic discrimination or marginalisation".

2.3 Classical Deep Learning Approaches

The predominant deep learning algorithms in hate speech detection are Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), Multi-Layer Perceptron (MLP), Long Short Term Memory (LSTM), and Gated Recurrent Units (GRU). This study explores ten deep learning algorithms for hate speech detection. The selected algorithms include traditional deep learning algorithms, attention enhanced deep learning algorithms and recent transformer-based algorithms. The traditional algorithms explored in this study are Multi-Layer Perceptron (MLP), Recurrent Neural Network, Gated Recurrent Unit (GRU), Long Short Term Memory (LSTM), LSTM with attention, and Convolutional Neural Networks (CNN). In the subsequent subsections, the researcher outlines the theory behind each of the traditional

deep learning algorithms, their advantages and disadvantages, as well as their application in hate speech studies.

2.3.1 Convolutional Neural Network

A typical CNN structure comprises an input layer, pooling layer, convolutional layer, fully connected layers and output layers. The pixel values of the algorithm's input are contained in the input layer. The output of neurons linked to local regions of the input is determined by the convolutional layer, which calculates the scalar product between their weights and the area linked to the input volume. An elementwise activation function is applied by the rectified linear unit to the output of the activation produced by the preceding layer. The pooling layer downsamples along with the spatial dimensionality of the given input, further reducing the parameters in that activation. Class scores from the activations to be used for class allocation are computed by the fully connected layers.

CNN was primarily developed for computer vision tasks (Litjens *et al*. 2017). However, their use has been extended to text processing (Lai et al. 2015; Wadawadagi and Pagi 2020). The CNN is regarded as an efficient feature extractor, specifically in the task of hate speech recognition, where it effectively extracts characters and word combinations (Badjatiya *et al*. 2017; Gambäck and Sikdar 2017). CNN makes use of pooling to reduce output between layers in the network. Convolution layers function as feature extractors, capturing local features by limiting the hidden layers' input space to a local field. This is due to the capability of CNN to handle local spatial correlation among neurons of neighbouring layers. This property is particularly important for hate speech detection, where it is needed to identify predominant local features for deciding the classes to which text messages belong.

Deep CNNs have been explored for hate speech recognition and other task classification tasks (Zhang, Zhao and LeCun 2015; Ordóñez and Roggen 2016; Gambäck and Sikdar 2017; Liu *et al.* 2017). Gambäck and Sikdar (2017) investigated the use of CNN in conjunction with word2vec features for hate speech detection, using a dataset by Waseem and Hovy (2016). Their recall results surpassed results obtained by Waseem and Hovy (2016) in an earlier study using the same dataset by 7.6 %. In their study, Elouali, Elberrichi and Elouali (2020) presented a CNN-based hate speech detection model on a dataset containing seven different languages. Their approach achieved a relatively high accuracy of 83%, despite the complexity that comes with seven different languages in a single dataset. Alshaalan and Al-Khalifa (2020) investigated the use of deep learning algorithms to detect hateful Saudi text on Twitter. The

CNN surpassed other methods explored in that study with an F-measure score of 79% and Area Under the Curve (AUC) score of 89%. A common problem with using CNN for NLP tasks is high dimensionality. The pooling technique can be employed to minimise the output dimensionality while retaining the most salient information. Varying kernel sizes and concatenating their outputs enables the detection of multiple size patterns. These patterns could be phrases such as "I like" or "I hate", and CNN will recognize those phrases in a sentence regardless of their location. This makes CNN suitable for text classification tasks such as hate speech detection.

2.3.2 Recurrent Neural Network

Recurrent Neural Networks(RNN), first proposed by McClelland, Rumelhart and Group (1986) are feed-forward networks designed to handle sequential data such as speech and text (Saksesi, Nasrun and Setianingsih 2018). Connections between nodes in the network create a guided graph along a sequence of neural network blocks. The network operates by giving earlier data points in a sequence higher weights. This property makes it effective for text, string, and sequential data classification tasks (Kowsari *et al.* 2019).

Several studies have investigated the use of RNN in NLP tasks such as spam detection, fake news categorisation and sentiment analysis (Duncan and Zhang 2015; Baktha and Tripathy 2017; Ren and Ji 2017; Ajao, Bhowmik and Zargari 2018; Al-Smadi *et al.* 2018; Bahad, Saxena and Kamal 2019). Recent studies have investigated the use of Recurrent Neural Networks in hate speech detection. In their research, Pitsilis, Ramampiaro and Langseth (2018) used an ensemble of RNN to separate normal text from sexism and racism. Their method outperformed earlier works on the same dataset, such as the work by Badjatiya *et al.* (2017) and Waseem and Hovy (2016). A study by Saksesi, Nasrun and Setianingsih (2018) experimented with the use of RNN to classify hate speech using different batch sizes and noted that the performance of the RNN improved as the batch size increased. Generally, the RNN is regarded as being able to capture long-term dependencies better than other traditional algorithms. Since the RNN considers the information from earlier nodes in a network, it improves semantic analysis of textual information (Zhang, Chen and Huang 2018; Kowsari *et al*. 2019). The RNN performs sequential processing, hence it is capable of capturing the sequential nature of languages (Young *et al*. 2018). Word meanings are derived from previous words in the same sentence, for instance, in the difference in meaning between "kid" and "kid glove". This is particularly important in tasks like hate speech detection, where a statement should be classified as hateful or neutral, depending on context rather than on word occurrence. RNNs are designed to model

these context interdependencies in language. Additionally, the RNN can model variable text length, making it suitable for sequence modelling tasks (Tang, Qin and Liu 2015). The RNN also facilitates time distributed joint processing for tasks such as multilevel categorisation (Chen *et al*. 2017). Despite its dominance and advantages, the RNN has a number of limitations. It is only appropriate for small and fixed-length input sequences, since it is susceptible to exponential increase or decrease of the gradient over lengthy sequences (Nair and Hinton 2010; Pascanu, Mikolov and Bengio 2012; Kowsari *et al.* 2019). This limitation presents a challenge for the detection of hateful text on Twitter since the maximum number of allowable words per tweet has recently increased from 140 word to 280 words. Several studies have proposed enhanced versions of RNN, such as LSTM and GRU, to address the shortcomings associated with standard RNN. These techniques are discussed in other sections of this study.

2.3.3 The Long Short Term Memory (LSTM)

The Long Short Term Memory (LSTM) is a gradient-based special variant of the RNN, introduced by Hochreiter and Schmidhuber (1997), that is capable of modelling ordered sequential input such as textual data (Liu *et al.* 2017). The LSTM was specifically developed to learn long term dependencies while addressing the vanishing and exploding gradient problems prevalent in the vanilla version of the Recurrent Neural Network (Sak, Senior and Beaufays 2014; Kowsari *et al.* 2017; Pouyanfar *et al.* 2018). The architecture of the RNN allows information accumulation as the algorithm operates, and it uses feedback to memorise earlier cell states (Nowak, Taspinar and Scherer 2017). The critical elements of the LSTM cells are the cell state, which is the state of the cell transferred to the succeeding steps in sequence; the forget gate, which determines which information to omit, and the input gate, which determines what should be forwarded to the subsequent activation.

The LSTM has been widely used in classification tasks such as fake news detection and spam detection. Studies have investigated the use of LSTM for detecting hateful content with considerable success (Kwok and Wang 2013; Gao and Huang 2017; Ahluwalia *et al.* 2018; Pitsilis, Ramampiaro and Langseth 2018). (Pitsilis, Ramampiaro and Langseth 2018) experimented with an LSTM- based ensemble to detect hate speech in Twitter using the dataset provided by (Waseem and Hovy 2016). Their method outperformed state-of-the-art approaches which used the same dataset. Furthermore, a recent study on word level and character level modelling proved that LSTM-based architectures outperform more recent models (Jain,

Sharma and Agarwal 2019). This further highlights the superiority of the LSTM on sequence-based tasks such as automated detection of hate speech (Melis, Dyer and Blunsom 2017). Language modelling is one of the fundamental applications of recurrent network-based algorithms, and many recent works have focused on optimising LSTMs for this task (Krueger *et al.* 2016; Merity, Keskar and Socher 2017). Despite its advantages over other deep learning algorithms, the LSTM requires substantial amounts of data to train and validate (Kowsari *et al.* 2019). The LSTM is also computationally expensive since it takes considerable time to train.

2.3.4 Gated Recurrent Unit (GRU)

The Gated Recurrent Unit (GRU) is a simplified version of the LSTM, which was introduced by Cho *et al.* (2014). Like the LSTM, it aims to address the vanishing gradient shortcoming inherent in the standard version of the Recurrent Neural network (Umar *et al.* 2019). In the RNN, fewer computations are required to update its hidden state, making it easier to train than the LSTM. The GRU comprises an input and output gate, while the LSTM comprises the input, output and forget gates (Dey and Salemt 2017). The main difference between the GRU and LSTM is that the GRU exposes the whole-cell state to other units, while the LSTM controls how much of the cell state is revealed to other units.

The GRU gating equations are as follows:

$Zt=\delta(wz,[ht-1,xt])$

$Rt=\delta(wr.[ht-1,xt])$

$Ht=tanh(W.[rt*ht-1,xt])$

$Ht=(1-zt)*(ht-1+zt?*ht)$,

Where zt, rt, w, xt, ht,tanh and represent the update gate, reset gate, weight, input vector, output vector, hyperbolic tangent activation and sigmoid function, respectively.

The GRU has been proven to give good results in sequential tasks such as text classification (Zulqarnain *et al.* 2020). Researchers have explored widely the use of GRUs in classification tasks such as opinion mining and spam detection (Adamson and Turan 2015; Cheng *et al.* 2017; Sung and Jeong 2018; Huang, Xie and Sun 2019; Poomka *et al.* 2019; Yang, Zuo and Cui 2019; Roy, Singh and Banerjee 2020), Several seminal works have explored the use of GRU for hate speech detection in social media. However, only a few of those studies were based on

the English language. Patihullah and Winarko (2019) investigated the use of GRU and word2vec features for hate speech detection in Indonesian Twitter textual data. The efficacy of the GRU was compared against three benchmark methods, namely, Logistic Regression, Random Forest and Naïve Bayes. The findings of the study indicate that a combination of GRU and word2vec features gave the best accuracy of 93%. In their study, Zulqarnain *et al.* (2020) proposed a combination of GRU and Support Vector Machine for the detection of hate speech using a Chinese dataset. The GRU was used for model training, while the Support Vector Machine was used as a replacement for the softmax output layer. Their proposed model produced 95% accuracy surpassing the performance of benchmark methods used in the same study, namely, Deep Auto Encoder Belief Networks and Bidirectional LSTM with a convolutional layer. A study by Van Huynh *et al.* (2019) implemented a Bidirectional GRU-LSTM-CNN classifier for hate speech detection using a Vietnamese language dataset. Their method had a 71% F1 score and came fifth overall position in the VLSP shared task 2019. Mossie and Wang (2020) proposed a GRU-based hate speech detection model for Amharic textual data on Facebook. Their approach produced the best AUC result of 98%, outperforming other methods used in the same study. A few studies have also explored the use of GRU networks for hate speech detection using English corpora (Zhang, Robinson and Tepper 2018; Zhang and Luo 2019).

2.3.5 The Multi-Layer Perceptron (MLP)

The MLP is a series of interconnected layers of the perceptron. The first layer output is used as the second layer input. This process is repeated for each of the remaining layers, with each receiving input from the previous layer and supplying its output as input to the next layer in the network before output reaches the last layer (output layer) (Singh and Husain 2014). The layers between the input and output layers are known as hidden layers. MLP has adjustable weights for training the system.

The MLP is a simple Neural network that acts as a universal function approximation (Singh and Husain 2014). The MLP has been investigated with success to address classification tasks from different domains such as audio processing (Zabidi *et al.* 2010), medical diagnosis (Nasir, Mashor and Hassan 2013) and pattern classification (Hu 2010). According to literature, only a few studies investigated the use of MLP for the task of hate speech detection. A recent study by Putri *et al.* (2020) compared algorithms for the task of hate speech detection using a dataset comprising 4302 Indonesian tweets related to race, ethnicity, religion and politics. The Multinomial Naïve Bayes classifier outperformed the MLP overall, while the MLP had the best

result of 84,3% achieved using unigrams and SMOTE features. Silva and Roman (2020) investigated the use of MLP with three hidden layers to detect hate speech in Portuguese tweets. The MLP achieved an F-measure score of 85%, surpassing results from an earlier study by Fortuna *et al.* (2019), in which the LSTM produced an F-measure score of 83% on the same dataset. However, the MLP `s performance was achieved using unprocessed tweets. This suggests that the performance of the MLP on hate speech detection may be improved by preprocessing the data. The Multi-Layer Perceptron has several advantages, which include computational efficiency, adaptive learning and simplicity (Wankhede 2014). Adaptive learning may particularly be useful for the task of hate speech detection since it allows it to learn based on given tasks. However, the number of layers in the MLP is manually set, and failure to set the optimal number of layers may result in overfitting or underfitting. This limitation calls for regularisation techniques like dropout, which are known to minimise overfitting (Srivastava *et al.* 2014). Additionally, the MLP has slow convergence, and its local minima can affect the training process.

2.4 Attention-Based Deep Learning Algorithms

The performance of deep learning techniques has been improved by techniques such as attention and transformer modelling (Vaswani *et al*. 2017; Devlin *et al*. 2018; Mozafari, Nematbakhsh and Fatemi 2019). However, the use of these techniques for hate speech detection is still limited. This technique was first introduced in 2015 (Bahdanau, Cho and Bengio 2014). Attention is a deep learning technique that has been investigated with considerable success in tasks such as opinion mining. Recent adaptations of the Attention mechanism have seen models progress from RNNs to Self-Attention and the Transformer models (Devlin *et al*. 2018).

2.4.1 Models with Attention Mechanism

Recurrent Neural Networks have been the gold standard in text processing tasks such as hate speech detection (Mutanga, Naicker and Olugbara 2020). While these techniques have performed well by capturing long-term sequences as compared to earlier methods, they fail to capture input sequences to arbitrary lengths. The LSTM, a special time of the RNN, was designed to capture long-term sequences better than earlier methods. However, the vanilla LSTM`s performance falls when the length of the sequence exceeds thirty words (Bahdanau, Cho and Bengio 2014). The extension of Twitter`s number of allowable characters from 140 to 280 calls for techniques that may be able to capture long-range sequences. Consequently, techniques such as attention have been explored to effectively derive meaning from long sequences of text such as those found on Twitter texts. The attention mechanism handles and

quantifies the interdependence between the input and output elements and within the input elements of a network. The attention mechanism ensures that the encoder codes task-specific information by enabling the decoder to make reference to the input sequence. Additionally, the decode is also conditioned on a context vector computed based on the hidden state sequence of the input. For each input sequence, the bidirectional LSTM generates a series of annotations (*(h1,….., hTx)*. Joining the forward and backward hidden states in the encoder results in the vectors h1,h2..htTx. The weights for the attention model are learned by a Feed-Forward Neural Network, and the output word context vector is generated using the weighted sum of annotations.

In NLP, the attention mechanism was first applied to the task of machine translation by Bahdanau, Cho and Bengio (2014), where it achieved superior results compared to earlier methods. Since then, it has been explored for other text processing tasks such as language understanding, answering questions, and text classification (Sukhbaatar, Weston and Fergus 2015; Vinyals *et al.* 2015; Kumar *et al.* 2016; Shen *et al.* 2017; Liu and Guo 2019). A few studies have incorporated the attention mechanisms in the detection of hateful content. Gao and Huang (2017) used the LSTM with attention model to detect hate speech. Their LSTM with attention model outperformed other baselines, including bidirectional LSTM by 3 to 4% in F-measure score. Additionally, the authors also noted that the LSTM was effective in detecting hate speech expressed implicitly compared to the other models. Santosh and Aravind (2019) used a Hierarchical model with attention based on phonemic words to detect hate speech in English-Hindi code-mixed textual data. Their results showed that the Hierarchical model with attention significantly outperformed the Support Vector Machine. Some studies also used the attention mechanism to enhance their models during data science competitions (De la Pena Sarracén *et al.* 2018; Baruah, Barbhuiya and Dey 2019; Wang *et al.* 2019).

2.5 Transformers

Improvements to attention the mechanism have gradually moved models from partial attention to Transformer models, which are based on full attention (Devlin *et al*. 2018). One major challenge of using RNN for text classification is that it processes text in a sequential manner. Despite the fact that CNNs are less sequential than RNNs, their computational cost to capture relationships between words in a sentence grows as sentence length also increases in the same manner as RNN. The transformers address this problem by applying self-attention for parallel computation of each word in a document or sentence. Transformers consist of encoders and decoders. Every encoder has a self-attention layer and a Feed Forward Neural Network. As

illustrated in Figure 2.1, the encoder`s input flows through the self-attention layer, which helps the encoder to look at other words in the input sequence as it encodes a particular word. On the other hand, the decoder has an attention layer between the Feed Forward Neural Network layer and the Self Attention layer. This allows the decoder to focus on the salient parts of the input sequence. Transformers allow for better parallel processing as compared RNNs and CNN, which enables efficient training of models on large volumes of data using GPU clusters. Transformers use much deeper network architectures and are pre-trained on vast corpora to learn contextual text representations by predicting words conditioned on their context (Radford *et al*. 2019). The transformer is increasingly becoming the preferred architecture for text processing outperforming traditional deep learning algorithms in natural language comprehension and natural language generation (Melis, Dyer and Blunsom 2017; Mutanga, Naicker and Olugbara 2020). Pretrained word and sentence embeddings used with traditional deep learning algorithms are capable of retaining the semantics of words used in sentences; however, they lack context mutability (Mishev *et al*. 2020). To illustrate the importance of context, we analyse the following two sentences that contain the word *Springbok*. "Springboks had a great Rugby World Cup" and "Springboks are nearing extinction in South Africa". In the first sentence, the term Springbok refers to the South African National Rugby Team, while in the second sentence, the same term refers to a wild animal. Nonetheless, the encoders would generate the same encoding for the two words, irrespective of their contextual usage. This traditional deep learning limitation necessitates the need for contextual embeddings.

2.5.1 Architecture of the Transformer

The transformer architecture performs sequence to sequence conversion using the encoder and decoder. Sequence transduction using transformers is based on multi-headed self-attention mechanisms only.

Figure 2.1 shows the general architecture of the NLP transformer.

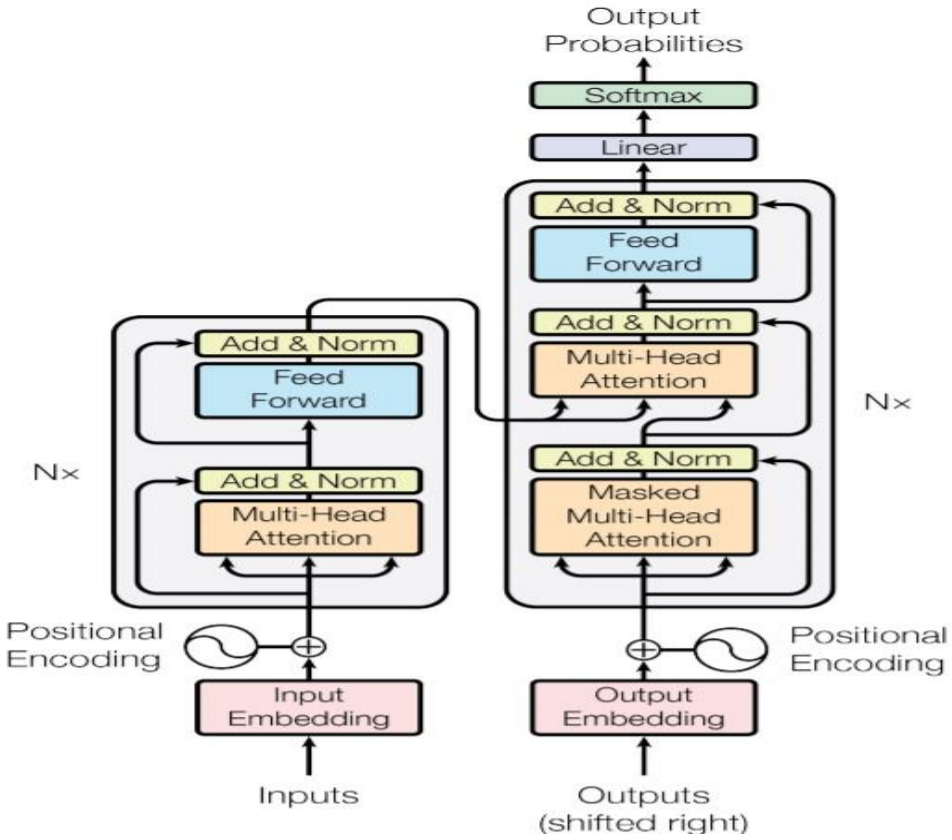

Figure 2.1: Architecture of the transformer model

(Vaswani *et al.* 2017)

2.5.2 Types of Transformers

Transformer Pre-trained Language Models (PLMs) can be grouped into two categories, autoregressive and autoencoding PLMs. Several architectures of transformer models have been successfully investigated for Natural Language processing tasks. In particular BERT has outperformed earlier performance benchmarks in NLP tasks (Mikolov *et al.* 2013b; Devlin *et al.* 2018). BERT uses vast unlabeled corpora to create pre-trained models whose parameters are tunable to enhance efficiency. The success of BERT has inspired the development of several algorithms based on BERT`s architecture. These models include RoBERTa, XLNet and DistilBERT. (Mikolov *et al.* 2013b). RoBERTa is an enhanced version of BERT trained on a larger dataset to maximise predictive accuracy, whereas DistilBERT is trained on a streamlined version of BERT. XLNet. is a generalised autoregressive pretraining approach that rebuilds original data from corrupted input.

2.5.3 BERT

BERT is an autoregressive transformer model that leverages the vanilla transformer architecture to produce an enhanced transformer model (Devlin *et al.* 2018; Sohangir *et al.* 2018). BERT employs the unsupervised learning technique to pre-train deep bidirectional representations from vast unlabeled text corpora by using a masked language model (MLM) and next sentence prediction (NSP). BERT addresses the limitations of earlier previous language models by building a bidirectional masked language model capable of predicting randomly masked words in the sentence, thereby enriching the contextual information of the words.

2.5.4 DistilBERT

DistilBERT is a streamlined variant of BERT that uses half the parameters of BERT while retaining the performance of the full version of BERT in several NLP tasks (Sanh *et al*. 2019). Streamlining of BERT is achieved by removal of the pooler and token type embeddings from BERT`s architecture (Büyüköz, Hürriyetoğlu and Özgür 2020). Furthermore, the model only uses half the number of layers used by BERT. The compact (student) model is trained to replicate the entire output distribution of the more extensive (teacher) model or ensemble of models. The compact model acquires the knowledge based on a distillation loss over the soft-target probabilities of the teacher instead of training with a cross-entropy over the hard-targets (one-hot encoding of the classes). The resultant model comprises 66 million parameters pre-trained on the Toronto Book Corpus and the Wikipedia in an unsupervised manner.

2.5.5 RoBERTa

The RoBERTa model is an enhanced version of BERT developed by the Facebook research team in 2019 (Liu *et al.* 2019). The model was improved by pre-training the model on a dataset tenfold the size of BERT using a different set of hyperparameters and an enhanced training methodology. During the training epochs, RoBERTa also eliminates Next Sentence Prediction (NSP) and introduces dynamic masking of words. These modifications and features outperform BERT in a variety of NLP functions, including text categorisation (Mishev *et al*. 2020).

2.5.6 XLNeT

The XLNet. transformer model is a generalised autoregressive pretraining for language understanding developed by Carnegie Mellon University and Google Brain in 2019 (Yang *et al*. 2019). XLNet seeks to address the limitations of BERT with an architectural design for pre-training. It utilises a generalised autoregressive model where the next token is determined by

all earlier tokens, thereby minimising corrupted input caused by word masking prevalent in BERT. BERT`s shortcomings include disregarding the interdependency between masked tokens since it assumes that they are jointly independent variables. XLNet, on the other hand, takes these tokens into account during context construction and assumes that masked words are mutually dependent. (Mishev *et al.* 2020). Furthermore, XLNet employs Permutation Language Modeling (PLM), which maximises the log-likelihood of a sequence given all possible permutations of words in a sentence. This means that XLNet uses tokens from positions found on the left and the right sides of the token to enrich the contextual information of each position.

2.5.7 Transformers in Hate Speech Detection

Some researchers have investigated the use of transformers for hate speech detection in social media. However, most of the explored techniques are computationally expensive, and it is not practical to run on devices with low computing power, such as handheld devices. Mozafari, Nematbakhsh and Fatemi, 2019 implemented a transformer-based transfer learning approach. They investigated BERT's effectiveness at capturing hate speech by using new CNN optimisation techniques based on transfer learning. They achieved a 92% F1 score outperforming earlier works (Waseem and Hovy 2016). In their study. MacAvaney *et al.* (2019) optimised their BERT model to detect hate speech on Twitter. The authors criticise the approach for its lack of interpretability, despite the fact that its results matched state of the art. Mutanga, Naicker and Olugbara (2020) investigated the performance of deep learning algorithms on a publicly available multiclass dataset with 24 783 tweets. The DistilBERT algorithm produced the overall best results with an F1 score of 75% outperforming baseline transformer and attention-based methods. The result also surpassed results achieved by another study that used the same dataset (Davidson *et al.* 2017). Although the DistilBERT approach outperformed other transformer-based approaches in that study, it has significantly fewer parameters and layers as compared to other transformers. The superior performance of DistilBERT could have been due to the chosen parameters. This, therefore, implies that the performance of other transformer models may be improved by fine-tuning. In a SemVal Competition, Liu, Li and Zou (2019) preprocessed textual data according to the language behaviour on social media before fine-tuning Bidirectional Encoder Representation from Trans-former (BERT) pre-trained by Google`s Artificial Intelligence Language team (Devlin *et al*. 2018). They obtained an F1 score of 0.8286, outperforming other competitors in SubTask A at SemVal 2019 competitions.

2.6 Chapter Summary

The chapter comprehensively reviewed relevant publications based on hate speech and deep learning. This serves as a foundation for the work reported in this dissertation. This chapter gives a detailed description of relevant topics directly related to the current study in five sections, namely the introduction, definitions of hate speech, traditional deep neural techniques, partial attention-based approaches, and full attention (transformer) approaches. Special emphasis was on the different approaches proposed in the literature by identifying their strengths and pitfalls. As described in the introductory part of this chapter, hate speech detection remains a significant but complex challenge that triggered the development of several hate speech detection algorithms to address the problem. From the review, it can be observed that there is no single solution to the problem of hate speech dissemination, despite the persistent efforts of researchers in this field. This inspired the researcher to experimentally investigate and evaluate the performance of deep learning algorithms. The next chapter explains the steps followed to accomplish the set research aim and objectives of this study. Specifically, the researcher discusses the corpora, the hardware, the programming language and the parameter settings for the methods which are the subject of this study.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Introduction

This chapter details the steps taken to meet the set objectives. The systematic approach employed in this study is known as experimentation. Firstly, the hate speech dataset acquisition process is discussed. Thereafter, the preprocessing of the acquired datasets is clearly explained, followed by a discussion of the selected feature representation method. Lastly, the training and classification process of the selected deep artificial neural networks is described. The subsequent sections elaborate on each of the methodological steps involved in this study.

3.2 Datasets

In Chapter Two of this study, the researcher mentioned that the performance of several algorithms would be evaluated on more than one dataset. Essentially, a suitable method should be able to perform optimally over diverse datasets to draw both qualitative and objective conclusions. Therefore, in this study, the researcher chooses to explore two publicly available hate speech data sets. Moreover, the selection of two different datasets would inject diversity to avoid biased classification results. The datasets used in this study include the hate speech and offensive language dataset and the Kaggle Twitter Hate Speech dataset. These datasets are selected based on the following attributes:

1. They are public and easily accessible.
2. They cover all types of hate speech and, as such, can train models which can be used to detect different kinds of hate speech.
3. They differ in size, allowing the analysis of the effect of dataset size in hate speech detection models.
4. They include both binary and multiclass instances.

In this work, it was essential to consider the efficacy of algorithms on diverse data. Consequently, the machine learning algorithms were applied to two datasets of significantly different sizes, as discussed in the next section.

3.2.1 The Hate Speech and Offensive Language (HSO) Dataset

The multiclass hate speech and offensive language dataset originally created by Davidson *et al.* (2017) was chosen for the experimental comparisons reported in this research because it contains different types of hate speech, and it has a comparatively high number of instances.

The dataset, as well as the results achieved by Davidson *et al*. (2017), provides a platform to measure improvements that could be achieved with the dataset and compare results, using various deep learning-based models developed by researchers who used the same dataset. This dataset had 24 783 Twitter text messages categorised and labelled into three classes: neutral speech, offensive language and hate speech. 77.4% of the instances are labelled as neutral, 16.8% as offensive and 5.8% as hate. The tweets in the dataset were manually annotated by Crowd Flower (CF) employees. The employees were asked to label each tweet as either containing hate or not. In labelling the datasets, they were guided by the definition of Davidson *et al*. (2017, p.512), which describes hate speech as "*language that is used to expresses hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group* ". Annotators were advised not only to look at the presence of certain words in a given tweet but also to consider the context surrounding words or phrases. A minimum of three annotators was assigned to code each tweet. The intercoder-annotator agreements score provided by Crowd Flower is 92%.

3.2.2 Kaggle Dataset

The original Kaggle dataset is made up of 8 778 neutral messages and 1 155 hate messages. The dataset was grossly imbalanced. It was essential to measure the efficacy of the selected deep learning algorithms on a smaller dataset. To enable the testing of algorithm performance on smaller datasets, the dataset was reduced programmatically to 2 300 messages. The new balanced dataset consisted of 1 150 hate messages and 1 150 neutral messages. The original dataset is publicly available on the Kaggle site:

https://www.kaggle.com/pandeyakshive97/datasets.

3.3 Visualisation of Datasets

Figure 3.1 and Figure 3.2 illustrate the distribution of the classes for each of the datasets used.

Figure 3.1 shows the Class distribution of the HSO dataset visualisation.

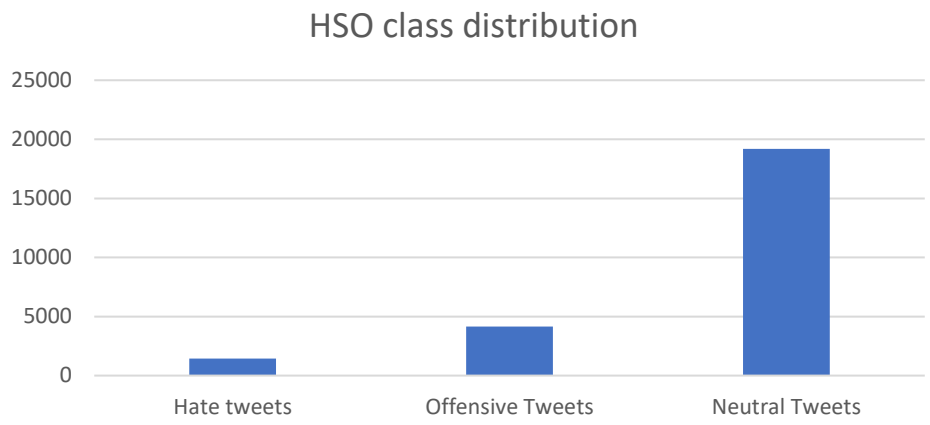

Figure 3.1: Dataset class distribution

As shown in Figure 3.1, the HSO dataset comprises 1 437 tweets labelled as hate tweets, 4 163 tweets labelled as offensive, and 19 182 tweets labelled as neutral.

Figure 3.2 shows the Class distribution of the Kaggle dataset visualisation.

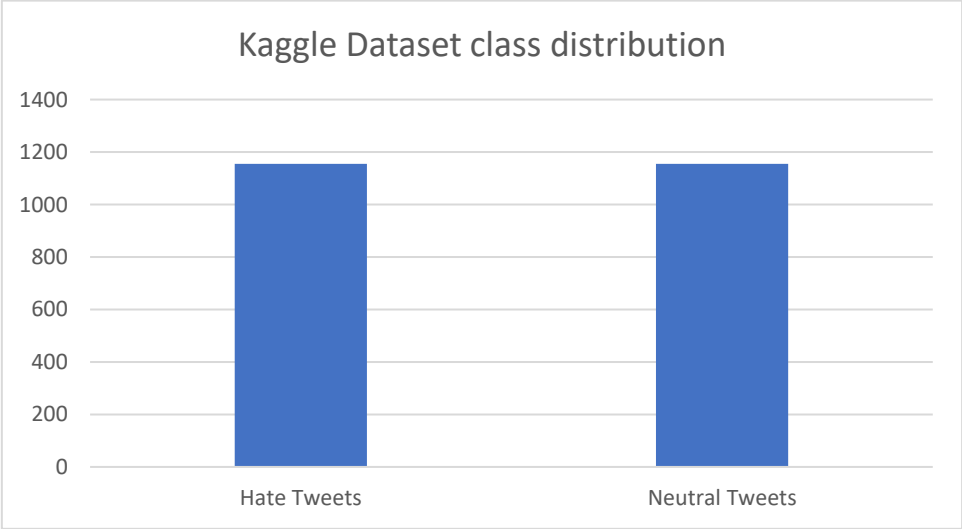

Figure 3.2: Kaggle Dataset Class Distribution

As shown in Figure 3.2, the Kaggle dataset is well balanced, comprising 1 150 tweets labelled as hate tweets and 1 150 tweets labelled as neutral.

3.4 System Setting

Two publicly available annotated datasets from Twitter social media platform were used for the experiments. This enabled us to evaluate our models based on the ground truth. All the

algorithms examined in this study were coded in Python, a versatile programming language with built-in machine learning libraries. The major libraries used in this work include Keras with theano backend, Natural Language Toolkit (NLTK), Hugging face`s transformers, scikit and matplotlib. Since Python is an open-source platform, its machine learning libraries continue to evolve as new functionalities are being added by the programming community. The experiments for traditional deep learning algorithms explored in this study were carried out on a laptop with Ubuntu 16.10 Linux distro, 8GB Random Access Memory and Core i7 CPU@1.6GHZ (8 CPUs),1.8GHz processor specifications. The experiments in this study required a minimum of 5GB free disk space, and the available disk space was 80GB. The experiments for transformer-based models were performed on a computer with Windows 10 operating system, GEO Force Graphical Processing Unit (GPU) and 64 GB RAM. The Graphical Processing Unit was used to take advantage of the parallelisation capability of Transformer algorithms explored in this study. The 64GB RAM also allows us to timeously save the pretrained transformer models onto the computer`s local storage. This allows the programs to access the pre-trained model locally rather than downloading it whenever it is needed.

3.5 Data Pre-processing

Text preprocessing is an essential part of NLP tasks which transforms text into a form ready for input into text classification algorithms. Due to the conversational nature of Twitter texts, preprocessing was applied to convert the Tweets into a format that is more predictable and analysable for the task of automated hate speech detection. Preprocessing also minimised feature sparsity in the feature representations. Preprocessing is a proven technique for improving the predictive capability of classifier algorithms (Uysal and Gunal 2014). Furthermore, preprocessing reduces the computational resources needed by a classifier while minimising the overall training time (Kadhim 2018). The processes involved in preprocessing the tweets in the datasets include tweet cleaning, text normalisation, stop word removal and removal of null values, as explained in the next section.

3.5.1 Cleaning up of Tweets

This stage involved deleting URLs prefixed with "https:" and http://, tags (i.e., "@user") and any text incompatible with the American National Standards Institute convention (ANSI). This is because such URLs do not add any information that is relevant to the classification decision. For example, if the tweeting history of the author is known, it might be useful in the classification decision. However, since no history is given regarding the author, the inclusion

of tags is not useful. The cleaning up of tweets was achieved using a regular expression-based string manipulation, coded using Python programming language version 3.7.

3.5.2 Data Normalisation

Twitter text data is generally noisy and is conversational in nature; therefore, it is necessary to remove the noise and standardise the data before passing it through machine learning algorithms. To mitigate twitter text noise and standardise input, the researcher applied state-of-the-art lemmatisation and Stemming techniques using Python s Natural Language Toolkit Library (NLTK). The Wordnet lemmatiser, which reduces words to their morphological root forms, was used, thereby reducing the number of features. This was achieved by converting words conveying the same meaning related to a single word; for example, the words *go*, *going*, *went*, and *gone* are all reduced to the word *go*.

3.5.3 Lower Casing

All word tokens in the dataset were converted to lower case to avoid capitalised versions of words being recognised as different features to lower case versions of the same word. For instance, the terms *Nigga* and *nigga* would be regarded as entirely different features if words are not converted to the same case. Switching words to the same case reduces feature space, thereby improving the performance of the model.

3.5.4 Stop Word Removal

Stop words refers to word tokens that do not carry important meaning, and their presence or absence is deemed not to influence the meaning of tweets. For example, in the tweet "Muslims are a disgusting group", the removal of Stop-words "are" and "a" will not change the overall sentiment of the statement. In this work, stop words were removed using Natural Language Toolkit (NLTK), a python libraries suite designed for natural language processing. A standard list of English words from NLTK was removed from the dataset before processing.

3.5.5 Word Length

Words with less than three characters were removed from the dataset. Short words generally contain no meaning; therefore, their removal does not change the overall meaning of tweets; for example, the words "be" and "is" are generally used to connect ideas in language. A custom python programmed method to delete these words was used for this task.

3.5.6 Removal of Null Values

All rows containing null values in the tweet column were removed using a built-in pandas library in the Python programming language. This process was performed to eliminate programming bugs associated with null values.

Table 3.1 shows an example of text transformation during the preprocessing stage.

Table 3.1: Preprocessing Steps

Action	Expected output
Preliminary stage: Read a tweet from the dataset	"!!! RT @mayasolovely: As a woman, you shouldn't complain about cleaning up your house. & as a man, you should always take the trash out..."
Step 1: remove unwanted text patterns from text such as special characters, hashtags, user tags, etc.	RT As a woman you shouldn t complain about…
Step 2: removing short words (less than three characters)	woman shouldn complain about cleaning your house
Step 3: tokenisation and stemming	woman, shouldn, complain, about, clean, your house

3.6 Feature Representation for Traditional Deep Learning Algorithms

Machine learning algorithms accept features in numerical form only. Therefore, it was necessary to convert word features into a numerical format for input into classifier algorithms. Conversion of word features into numerical form can be achieved using different techniques such as word embedding s and the Bag of Words approach. Traditional feature representation methods such as the Bag of Words (BOW) approach suffer from several disadvantages as compared to word embedding s. For example, the lack of word order leads to loss of contextual meaning of words, thereby negatively impacting the quality of feature representation (Le and Mikolov 2014). Additionally, the BOW approach uses sparse representations, which lead to high dimensionality (Yogarajan *et al.* 2020).

Word embeddings have been shown to be effective in capturing contextual similarities, and because of reduced feature space, they are particularly computationally efficient in processing NLP tasks (Young *et al.* 2018). Moreover, the dimensionality of word embeddings can be varied according to the problem being solved. Smaller dimensionality is more appropriate for syntactic tasks such as Named entity recognition (Melamud *et al.* 2016). On the other hand, a larger dimensionality is appropriate for tasks dependent on semantics, such as sentiment analysis and hate speech detection (Ruder, Ghaffari and Breslin 2016). The deep learning algorithms evaluated in this study use word embedding feature vectors as input for training. Keras deep learning library with theano backend was used to code the models. Keras is a wrapper to TensorFlow, a deep learning library that allows the use of a few lines of code. The algorithms used in this study belong to the many-to-one architecture, implying that they are trained on several input sequences to predict one output. In this study, the Keras embedding layer was used, which was initialised with generated weights that learn an embedding for the vocabulary in the training dataset. The Keras embedding layer was implemented as part of a deep learning model where the embedding was learned along with the model itself for each algorithm. Since the input data must be integer encoded, each word is represented by a unique integer. This data preparation step was performed using Keras` built-in tokeniser. The embedding layer is the network's first hidden layer, and it has three important arguments, namely:

- ***Input dimension****:* The vocabulary size of the dataset is defined by this integer. The vocabulary size would be 10 if the data is integer encoded with values ranging from 0 to 9.
- ***Output dimension****:* This is the vector space size in which words were embedded with. It determines the size of each word's output vector in this layer. Different values between 16 and 100 were experimented with to find the optimal figure for the task.
- ***Input length****:* This is the length of the input sequences used to specify any Keras model's input layer. If all of the input documents contain 1000 sentences, for example, the input length will be 1000.

3.7 Tuning and Training for Traditional Deep Learning Algorithms

Tuning deep neural networks can be challenging. Careful selection of parameters in neural networks can be the difference between superior and inferior performance (Reimers and Gurevych 2017). Poorly selected hyperparameters may lead to poor learning by algorithms.

The next section presents the steps taken into consideration while tuning and training the selected traditional deep learning algorithms.

3.7.1 Minibatch Size

A minibatch is a collection of instances from a training dataset, used at a time to calculate the gradients and parameter updates. It is crucial to maintain the minibatch size bigger than one to maximize parallelisation. The algorithms were tested with minibatch sizes of 16, 32, 64 and 128. Initially, a minibatch size of 16 was used. The output of the networks was monitored for various minibatch sizes before deciding on size (64), which produced the best results for all of the experiments.

3.7.2 Number of Epochs and Number of Iterations

An epoch is a complete pass of the whole dataset during training (Brownlee 2018). On the other hand, iteration refers to the number of parameter updates in a row per minibatch. Multiple epochs and one iteration were employed for training the network. Deciding on a suitable number of epochs to use is important, as using too few epochs may not allow the algorithm sufficient time to learn appropriate parameters, whereas too many epochs can result in overfitting. In the experiments of this study, 10 epochs across all algorithms were used.

3.7.3 Learning Rate

Learning rate determines the step size at every iteration as it progresses towards a minimum loss function. A poor learning rate may lead to slow network learning. A network`s learning rate is usually determined by the training as well as the network architecture. For comparison purposes in this work, the researcher used a learning rate of 0,1 with early stopping.

3.7.4 Activation Function

In deep learning, the activation function is responsible for converting the activation level of a neuron into an output signal (Karlik and Olgac 2011). The activation function is dependent on the layer type. Functions such as tanh, sigmoid are susceptible to the vanishing gradient problem, thereby negatively affecting the learning process in deep neural networks. For the hidden (non-output) layers, ‘rectified linear Unit (relu) activation functions are often preferred since they are less susceptible to vanishing gradient as compared to the aforementioned activation functions. The activation functions for the output largely depends on the kind of output expected from the network. Since the researcher was using a binary and a multiclass datasets classification problem, the softmax and the Rectified Linear Unit (RELU) activation functions were used.

3.7.5 Loss Function

Deep neural networks employ loss functions at each layer, which can be used for pre-training to learn more relevant weights. The loss function aids the classification decision when used in the output layer. The loss function to be used is essentially determined by the purpose of the neural network. In our network, the researcher used the binary cross-entropy loss function for the Kaggle dataset, whereas the categorical cross-entropy was employed for the multiclass HSO dataset.

3.7.6 Regularization using Dropout

Dropout is a regularisation technique meant to help models prevent overfitting by improving the model's generalisation capability. In this research, the effect of varying dropout values between 0,1 and 0,9 was investigated. The dropout technique is the gold standard regularisation technique in NLP tasks (Srivastava *et al.* 2014). The optimal value for dropout was 0,5. Other researchers in NLP have also found a dropout of 0.5 to be the most optimal (Kim 2014).

3.7.7 Optimisation Algorithm

In this work, the default optimiser for the algorithms was Adam. The use of Adam as the default optimiser was inspired by its widespread use and success in related tasks (Ruder 2016). Additionally, the effect of rmsprop and adagrad were evaluated. To complement the optimisers` automatic internal learning rate tuning, the " Reduce Learning rate on Plateau" approach was implemented, where a learning rate was manually set until convergence. Reduce Learning rate on the plateau was set with validation loss as the monitor. The Reduce Learning rate on the plateau was configured with a factor of 0.6, verbose of 1 and patience of 1.

3.7.8 Layers

Choosing the appropriate type of layer connection was essential to keep the gradient significant enough to allow the weight to update its value. In this study, the researcher used the dense connection for the layers between the input and output. Dense connections add direct connections from each layer to all subsequent layers as opposed to the successive addition of layers (Gao and Huang 2017). The choice of the dense layer was inspired by its recent success in Natural Language Processing tasks (Ruder 2016). The effect of varying the number of layers on algorithm performance was also explored. The number of dense layers was varied from 3 to 7 for every algorithm.

3.7.9 Dataset Split

The default train-test split ratio was set at 80:20. Additionally, the impact of varying the train-test split ratio was further explored by investigating the 70:30 train-test split ratio.

Table 3.2 summarises the parameter settings used in this study.

Table 3.2: Parameter Settings used

Parameter	Embedding Parameters	Epochs	Layers	Train/Test split	Activation Function	Optimiser	Loss Function
Setting	Input Length: 2000 Input Dimension: 200 Output Dimension: 16	20	3, 5 and 7	Default: 80:20, Variation: 70:30	Relu and Sigmoid	Default: Adam Variations (RMsProp and Adagrad)	Binary cross-entropy Categorical Cross-Entropy

Figure 3.3 illustrates the major steps followed in implementing each of the deep learning algorithms.

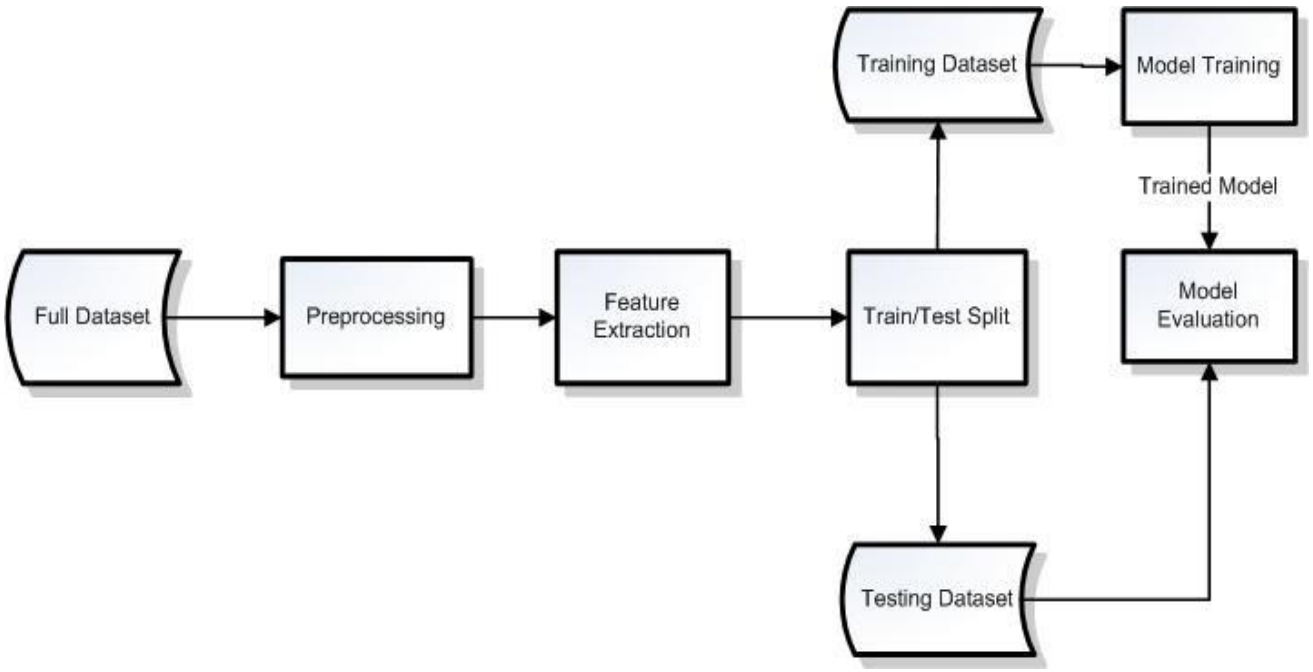

Figure 3.3: Illustration of steps followed in hate speech detection using traditional deep learning algorithms

Figure 3.3 illustrates the general architecture of the traditional deep learning hate speech detection models explored in this study. The complete pipeline of the processes at each stage is shown. At the initial stage, tweets from the datasets are preprocessed. This is followed by feature extraction, which is performed automatically by the hate deep learning algorithm. Lastly, the model is trained and evaluated using the training and testing proportions of the dataset, respectively.

3.8 Implementation of the Attention Mechanism

The attention mechanism was implemented in conjunction with the Bidirectional LSTM at the word level. At every phase t, the Bidirectional -LSTM receives as input word vector containing semantic and syntactic information, known as word embedding (Mikolov *et al.* 2013a). Thereafter, an attention layer was applied over each hidden state hˆ t. The model's attention weights are learned through the joining past hidden state of the Post-Attention LSTM (Pos-Att-LSTM and the current hidden state of the Bidirectional LSTM. The Post-Att-LSTM network detects the presence or absence of hate within a text.

The bidirectional LSTM operates basically the same way as the vanilla LSTM, but the processing of the incoming text is from both the left and right as opposed to one way. The Bidirectional-LSTM was investigated with the aim of capturing long-range and backwards dependencies based on its success in earlier studies (De la Pena Sarracén *et al.* 2018; Ren, Wan and Ren 2018; Do *et al.* 2019). Two fundamental building blocks of the attention-based LSTM for hate speech detection are the attention layer and the Post attention Layer.

3.8.1 Attention Layer

The attention mechanism allows the Bidirectional LSTM to decide parts of the tweet on which the model should focus. The model learns what to focus on based on the input tweet and what it will have produced to date. The goal of the attention layer is to get a context vector capable of capturing salient information and input it to the subsequent level (De la Pena Sarracén *et al.* 2018).

3.8.2 Post Attention Layer

The Post-Attention-LSTM is responsible for assigning tweets to either the hate or neutral category. At each time step, the network receives the context vector, propagated until the final

hidden state. This vector is text representation which is used in the final softmax layer as follows:

$y\hat{} = softmax(Xg * sTx + zg)$ *where Xg and zg are the parameters for the softmax layer.*

For the loss function, the researcher used cross-entropy, which is denoted as:

$L = - X\, n\, yn * log(\hat{}yn),$

Where yn is the correct classification of the n-th tweet.

The full architecture of the bidirectional LSTM with attention used in this study is summarised in Figure 3.4. As Illustrated, the model includes the bidirectional LSTM layer, the attention layer and the post attention layer.

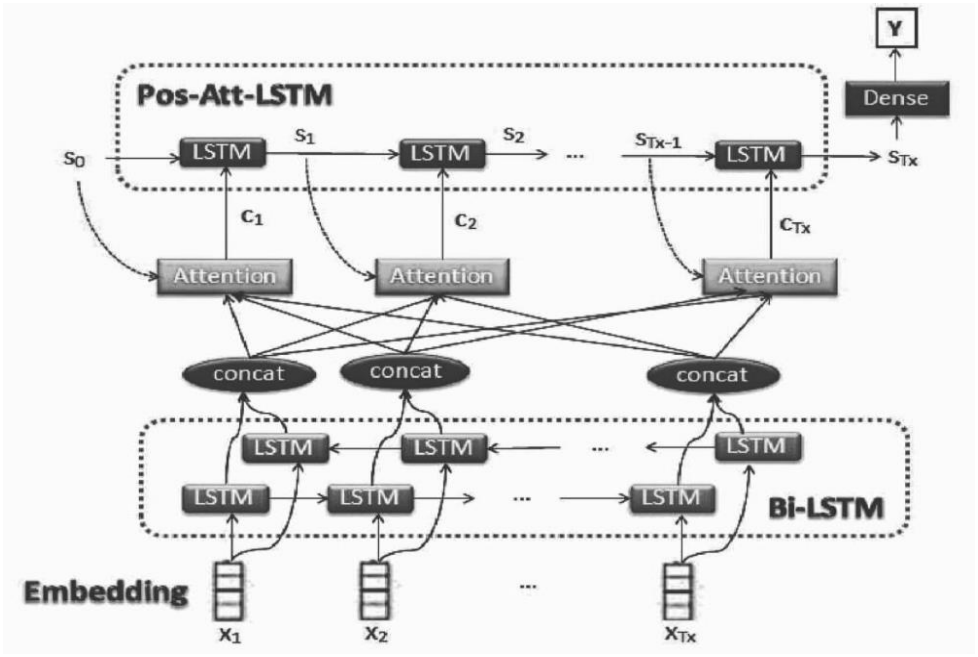

Figure 3.4: Architecture of the bidirectional LSTM with attention for hate speech detection in Twitter

3.9 Implementation of Transformer Algorithms

Transformers mirror the standard text classification, which includes preprocessing the text, model training and predictions on unseen data. The transformer methods are selected in this study due to their built-in self-attention feature, which facilitates the capture of long-term dependencies while enabling parallel processing of input features. The capture of long-term

dependencies enables anaphora resolution, which has been identified as a major limitation encountered when classifying subjective text (Cambria *et al*. 2017). Nevertheless, the majority of transformers are resource-intensive, making them less applicable in environments with scarce resources (Sajjad *et al.* 2020). Given this background, in this study, full transformer methods were investigated alongside transformer methods that have been streamlined and customized for resource-constrained environments, for example, DistilBERT, which is a streamlined version of the BERT architecture. Other transformer methods explored in this study are RoBERT, XLNet, and BERT.

3.9.1 Pretraining Transformer Models

Through transformers, pre-existing models built on very large datasets can be customised for different tasks classification tasks. This minimises the cost of training a new deep learning model every time. Additionally, the datasets on which the transformer models are trained meet industry-accepted standards, and we are assured of a model that has been vetted for quality. Transformers are trained in an unsupervised manner on corpora such as the English Wikipedia and the Toronto book Corpus. After Pre-training, the model the researcher customised each transformer model to the task of hate speech detection using the parameters discussed in Section 3.9.2 of this study.

3.9.2 Parameter Tuning of Transformer Models

The tuning of hyperparameters is a fundamental phase when customising pre-trained models to given assignments. As outlined in Table 3.3, the hate speech detection model was improved by tweaking the parameters of each transformer method during the training session. The same optimal hyperparameter values were configured for the number of epochs, early stopping patience, batch size and sequence length. To match Twitter's cap of 280 characters per tweet, the maximum sequence length parameter was set at 280 characters. The optimum number of epochs in our experiments was set at four. Additionally, overfitting was minimised through the use of early stopping method. The early stopping patience parameter was configured at 4, implying that training is interrupted as soon as there is no further reduction of validation loss for four epochs. The evaluation batch size specifies the number of instances that can be processed simultaneously. The researcher configured the evaluation batch size at 256 since it was the maximum batch size compatible with the computer`s processor.

The other hyperparameters used for each of the transformers were summarised in Table 3.3.

Table 3.3: Hyperparameters for transformer methods

Hyperparameters	**Method**			
	BERT	**RoBERT**	**XLNET**	**DistilBERT**
Learning rate	4e-5	4e-2	4e-3	4e-5
Epochs	4	4	4	4
Sequence length	280	280	280	280
Evaluation batch size	256	256	256	256
Training batch size	256	256	256	256
Gradient accumulation steps	1	2	2	0
Early stopping patience	4	4	4	4

After parameter tuning, the hate speech detection experiments using transformer methods was concluded by evaluation of the trained models. The performance of the transformer models was objectively measured and analysed using well known statistical metrics, as discussed in Section 3.10 of this study. The complete pipeline of hate speech detection using transformer methods is illustrated in Figure 3.5, showing that the dataset was first divided into training and testing parts. A pretrained transformer model was customised or fine-tuned for the hate speech detection task using the training component or part of the dataset. After fine-tuning the model for hate speech detection, it was evaluated using state of the art metrics. The evaluation was based on test data which was 20% or 30% of the full dataset, depending on the train-test split ratio used. The evaluation metrics used in this study were accuracy, precision, recall, F-measure, area under the curve (AUC) and Mathews Correlation Coefficient (MCC).

Figure 3.5 shows the architecture of the transformer-based model hate speech detection models explored in this study.

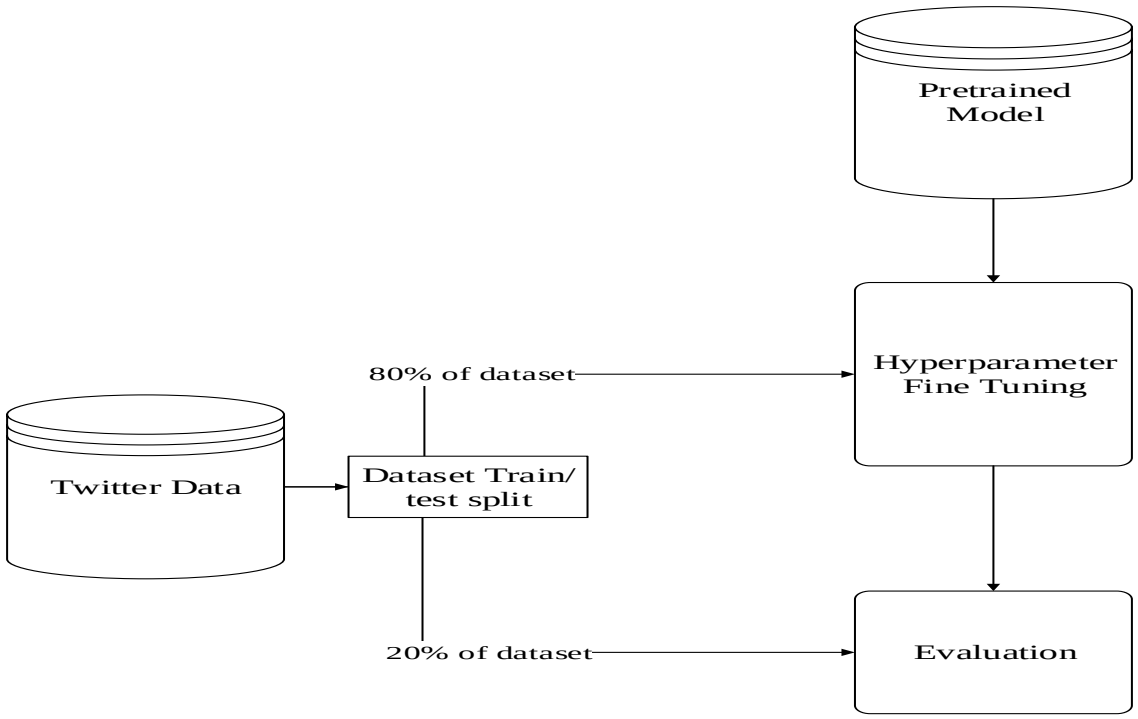

Figure 3.5: Transformer architecture for hate speech detection

3.10 Metrics

Evaluation metrics were used to measure the quality and performance of machine learning models. Evaluating machine learning models or algorithms are crucial for any study because they present an objective way of assessing model performance. In the literature, there is no consensus on the best evaluation metrics for classification problems. Performance evaluation of hate speech detection algorithms has largely been based on classic metrics such as precision, recall and F-measure (Badjatiya *et al.* 2017; Gambäck and Sikdar 2017; Park and Fung 2017; Zhang, Robinson and Tepper 2018; Charitidis *et al.* 2020). In this study, the Mathews correlation coefficient metric, a contingency matrix method of calculating the Pearson product-moment correlation coefficient, was also included (Powers 2020). The MCC is less affected by imbalanced datasets compared to more popular metrics such as accuracy, which overstate the model's performance (Sokolova, Japkowicz and Szpakowicz 2006; Gu, Zhu and Cai 2009; Akosa 2017). However, the evaluation was based on widely used metrics, which are accuracy, precision, recall, F-measure and area under the curve. Accuracy, precision, recall and F-measure can be calculated from values of the confusion matrix, which are True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives.TP represents actual positives predicted as positive, TN represents Actual Negatives that are predicted correctly as Negative,

FP represents actual negatives that are wrongly predicted as positives, and FN represents actual positives that are wrongly predicted as negatives.

The general formula for accuracy is $\frac{TP+TN}{TP+TN+FP+FN}$ (Kowsari *et al.* 2019).

In this study, accuracy represents the proportion of the number of tweets correctly detected to the total number of tweets in the dataset.

The general formula for recall is $\frac{TP}{TP+FN}$ (Grandini, Bagli and Visani 2020).

Recall is expressed as the proportion of the number of hate tweets correctly detected to the number of tweets in the hate speech class.

The general formula for precision is $\frac{TP}{TP+FP}$ (Grandini, Bagli and Visani 2020).

Precision denotes the extent to which a model can be trusted when it predicts an instance in the dataset as being positive (Grandini, Bagli and Visani 2020). In this study, precision is the proportion of the number of hate tweets correctly detected to the total number of hate tweets.

The general formula for F-measure is $2 * \frac{precision*recall}{precision+recall}$ (Kowsari *et al.* 2019).

F-measure-Score is the harmonic mean measure that seeks a balance between recall and precision (Grandini, Bagli and Visani 2020).

The general formula for MCC is $\frac{TP.TN-FP.FN}{\sqrt{(TP+FP)}.(TP+FN).(TN+FP).(TN+FN)}$

Area under the curve is created by plotting the true positive rate against the false-positive rate. AUC indicates the extent to which positive and negative classes are well in regard to the decision index (Kowsari *et al.* 2019).

For an objective analysis of algorithm performance, it was essential to use this combination of metrics together since the weaknesses of an evaluation metric may be addressed by another evaluation. For example, accuracy may not be a useful evaluation metric for highly imbalanced data as it does not specify performance per class. On the other hand, precision and recall show class-specific performance.

3.11 Chapter Summary

This chapter outlined the steps in sequence taken to achieve the second, third and fourth objectives of this research study. The chapter discusses in detail the implementation of the deep learning algorithms for detecting hateful speech. The first phase of the chapter covered the acquisition of the training datasets. The datasets explored in this study contain carefully selected Twitter messages of a maximum of 280 characters as per Twitter restrictions policy. This was followed by data preprocessing, feature representation and algorithmic implementation of the classical deep learning algorithms, attention-based deep learning algorithms and transformer algorithms. RMsProp, adagrad and adam optimisation algorithms were tested in several experiments to find the best performing algorithm. All these processes were crucial for achieving the principal contribution of this research work. The next chapter covers the evaluation experiments, experimental results and their analysis.

CHAPTER FOUR

PRESENTATION OF RESULTS AND DISCUSSION

4.1 Introduction

This chapter presents the evaluation experiments, the experimental results and their corresponding analysis to accomplish the third objective of this study, which aimed at objectively measuring the performance of deep learning algorithms for detecting hate speech. The first section of this chapter begins with an analysis of the performance evaluation process. This is followed by the visual analysis of the training process for the datasets explored in this work. The following section presents the quantitative analysis of the binary classification results. In the quantitative analysis section, the six statistical evaluation metrics used in this study and their mathematical representations are presented. Next is the quantitative results obtained from each of the evaluation metrics and their respective interpretations.

4.2 Analysis of Performance Evaluation

The universal problem in the development of any text classification algorithm, which is achievable through experimentations, is a comprehensive measure of its accuracy to validate that one algorithm is better than the other. Moreover, if a given text classification algorithm is better than its counterparts, will it be consistent under varying parameters and other factors? These questions are hereby addressed by using the following procedures in the experiments:

- More than one dataset with various forms of hate speech was used to observe the behaviour of a classification algorithm to ensure its consistency across the hate speech datasets.
- Both qualitative and quantitative evaluation techniques were adopted to cater for the weaknesses of relying on one evaluation technique.
- Evaluation of model behaviour during the training phase is of vital importance since it was a good measure for validation to determine whether a model was underfitting or overfitting.
- During the quantitative evaluation, a minimum number of six evaluation metrics was used to ensure that the overall performance of a classification algorithm is significant to an extent.

4.3 Training Process Visual Analysis

A learning curve's shape reflects the behaviour of a machine learning model. It indicates properties such as how representative the dataset is and the parameter tuning required to improve learning, classification performance or both.

There are three typical dynamics that may be deduced from observing learning curves: Underfit, Overfit and Good Fit (Brownlee 2021). Overfitting occurs when a machine learning model learns the training dataset too well, including the statistical noise and random variations in the training dataset (James *et al.* 2013). Underfitting occurs when a model fails to achieve a sufficiently low error rate on training data (Bengio, Goodfellow and Courville 2017). The target of any learning algorithm is a perfect fit, and it exists between an overfit and an underfit model. (Brownlee 2021). For visual analysis of the training process, algorithms were explored using the bigger dataset. The relationship between training and validation accuracy gave a good idea about model training behaviour.

Training and Validation accuracy Graphs for datasets for deep learning algorithms used in this study are shown in the next section. The training and validation graphs were generated from the Hate Speech and Offensive Language dataset only.

Figure 4.1 shows the training and validation accuracy for CNN over eight epochs.

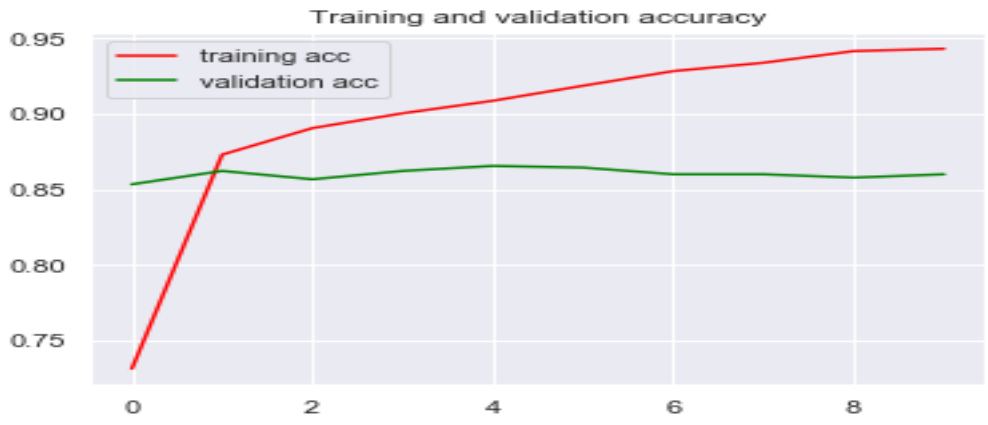

Figure 4.1: Training and Validation Graph for CNN

It can be observed from Figure 4.1 that training accuracy is consistently higher than validation accuracy throughout the epochs shown. From the first epoch to the eighth epoch, training accuracy consistently increased from 87% to 94%, while validation accuracy fluctuated between 85% and 86%. This trend clearly shows that the CNN model was failing to capture

the discriminative features from the dataset. This led to overfitting, as evidenced by the continuous drop in performance on unseen data.

Figure 4.2 shows the training and validation accuracy for the MLP over eight epochs.

Figure 4.2: Training and Validation accuracy for MLP

Figure 4.2 shows the training and validation accuracy graph for the MLP method. Throughout the eight epochs, the training accuracy is greater than the validation accuracy. The gap between the training and validation accuracy progressively increases with each epoch. This indicates the model's failure to learn the salient features distinguishing hate speech from neutral speech through experience.

Figure 4.3 shows the training and validation accuracy for the RNN over eight epochs.

Figure 4.3: Training and Validation accuracy Graph for RNN

Figure 4.3 depicts the training and validation accuracy graph for the MLP method. We can observe that after the first epoch, validation accuracy remains fluctuates around 87% while

training accuracy continues to rise. The gap between the training and validation accuracy gradually increases with each epoch, indicating overfitting.

Figure 4.4 shows the training and validation accuracy for the LSTM over eight epochs.

Figure 4.4: Training and Validation accuracy for LSTM

From Figure 4.4, we observe that both training and validation accuracy steadily increase from the first epoch until the sixth epoch. This steady increase indicates that the LSTM can learn from its experience during training. This may be attributed to the model's inherent capability to handle sequential data such as text. After the 6th epoch, training and validation accuracy continue to improve, albeit at a slower rate as compared to the period between the first and the sixth epoch.

Figure 4.5 shows the training and validation accuracy for the GRU over eight epochs.

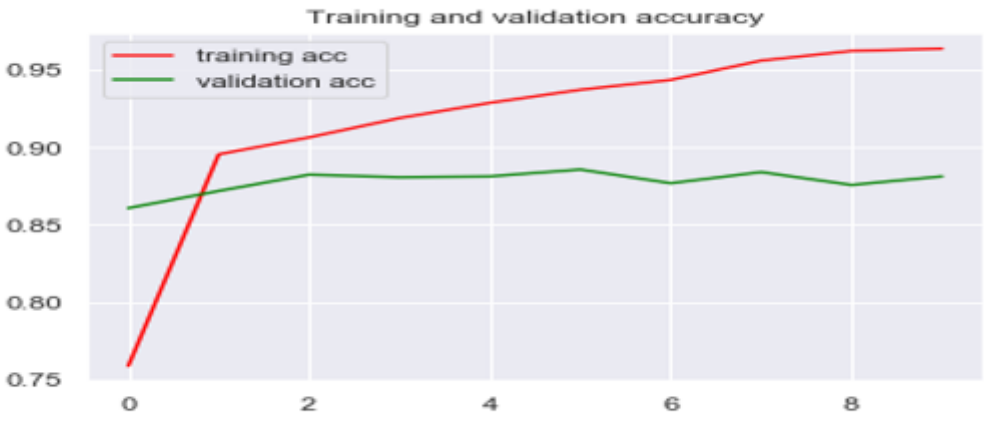

Figure 4.5: Training and Validation accuracy for GRU

It can be observed from Figure 4.5 that training accuracy is consistently higher than validation accuracy. As a matter of fact, from the second epoch onwards, validation accuracy continues to decrease while training accuracy increased with each successive epoch. This clearly shows that the GRU model was overfitting as its performance on unseen data kept dropping.

Figure 4.6 shows the training and validation accuracy for RoBERTa over eight epochs.

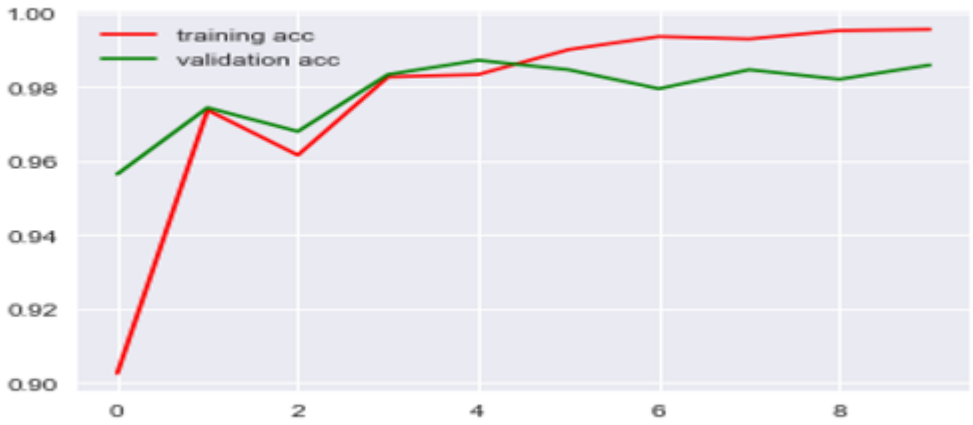

Figure 4.6: Training and Validation accuracy for RoBERTa

Figure 4.6 illustrates the training and validation accuracy for the RoBERTa algorithm. The validation accuracy is marginally above the training accuracy from the first epoch to the fourth epoch, indicating model underfitting during the early epochs. However, as training progressed, training accuracy increased at a faster rate than validation accuracy, surpassing the validation accuracy on the fifth epoch. From the 6th epoch until the eighth epoch, the gap between the validation and training accuracy is negligible, indicating a stable model. The stability is attributed to the fact that RoBERTa is optimised for robustness and performance since it is trained on corpora which is ten times larger than BERT.

Figure 4.7 shows the training and validation accuracy for DistilBERT over eight epochs.

Figure 4.7: Training and Validation accuracy graph for DistilBERT

Figure 4.7 depicts the training and validation accuracy for the DistilBERT algorithm. From the first epoch to the third epoch, the validation accuracy is higher than the training accuracy, indicating model underfitting during the early epochs. However, as training progressed, the gap between training and validation accuracy increasingly reduced until the third epoch, when training accuracy surpassed validation accuracy. As weights continued to be updated, the optimal performance was achieved in the fourth epoch. Thereafter the model started overfitting.

Figure 4.8 the training and validation accuracy for XLNet. over eight epochs.

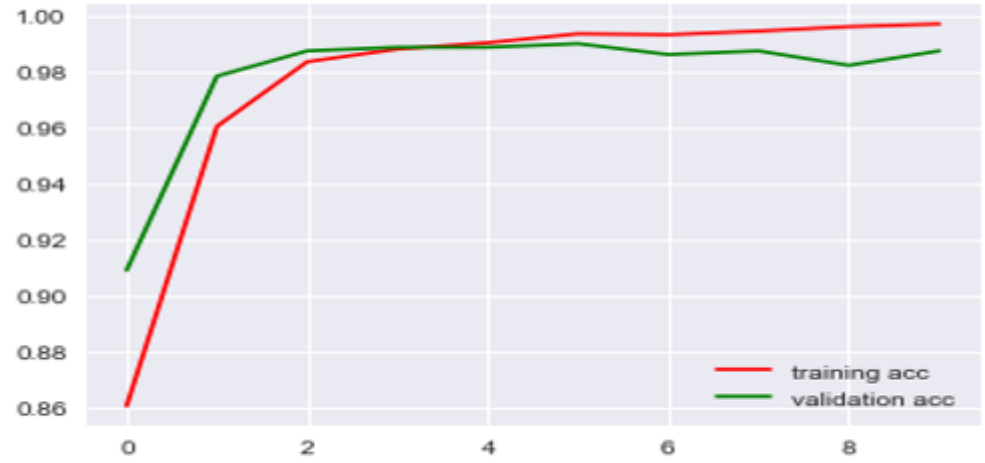

Figure 4.8: Training and validation accuracy graph for XLNET

Figure 4.8 depicts the training and validation curves for the XLNet algorithm over eight epochs. After the third epoch, training accuracy continued to increase steadily, while validation accuracy fluctuated around 98,5%. However, the gap between the training and validation curve is negligible, suggesting that the XLNet model managed to learn the salient features of hate speech in the dataset to make accurate predictions on unseen data. After the eighth epoch, validation accuracy increases slightly, further confirming that the model is close to a good fit.

Figure 4.9 shows the training and validation accuracy for BERT over the eight epochs.

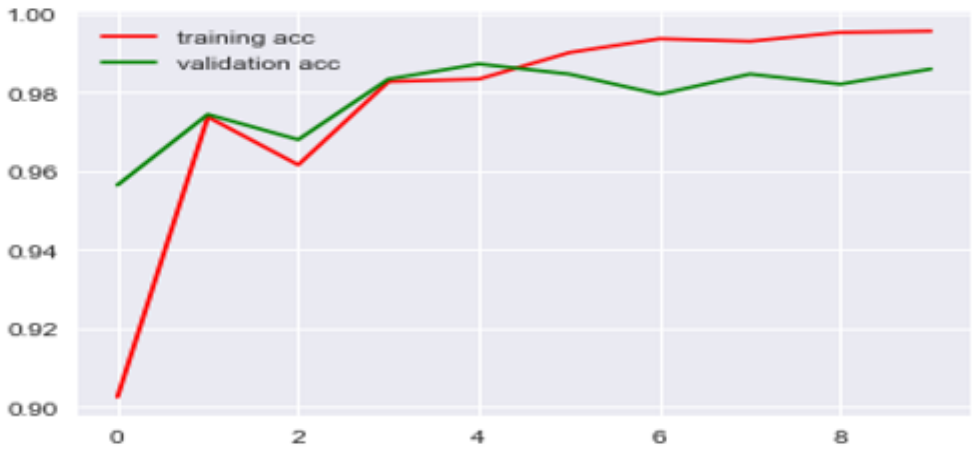

Figure 4.9: Training and Validation accuracy graph for BERT

Figure 4.9 shows the training and validation accuracy plot for the BERT algorithms. During the early phases of training, the training accuracy is higher than validation accuracy, indicating underfitting. As training progresses, the model updates weights to improve performance. From the fifth epoch, the model starts to overfit, and the optimal performance is achieved at the seventh epoch when the gap between the training and validation accuracy is minimal.

Figures 4.1, 4.2, 4.3, 4.4, 4.5, 4.6,4.7, 4.8 and 4.9 present a visual comparison of the training curves for each of the deep learning algorithms explored in this study. These graphs clearly show the correlation between the training accuracy and validation accuracy for each of the algorithms. The graphs show that for traditional deep learning algorithms, as the number of epochs increases, the difference between the training and validation accuracy also increases. This suggests that the algorithms could be overfitting. In particular, the MLP shows a marked difference between training and validation accuracy from as early as the third epoch. On the other hand, on observing graphs of transformer algorithms, it can be noted that the difference

between training and validation is maintained at roughly the same value, indicating that the algorithms are close to a good fit on training data. In particular, XLNet shows good training performance as the gap between training and validation is very small from the first epoch until the eighth epoch. A closer inspection of traditional deep learning algorithms graphs shows that RNN based algorithms such as LSTM and GRU make more robust models compared to other deep learning algorithms. On traditional deep learning algorithms, the LSTM has the best fit, while MLP has the worst fit.

4.4 Quantitative Analysis of Deep Learning Algorithms

In this section, the statistical values used to assess the performance of the deep learning algorithms are presented and interpreted. A total of ten deep learning algorithms were evaluated using the evaluation metrics discussed next.

4.4.1 Evaluation Metrics

There are several ways to quantitatively measure the agreement between the final classification result and ground truth as labelled in the dataset. The main objective of a classification algorithm was to classify textual data into predefined categories accurately. The performance of each of the algorithms investigated in this study needed to be proven for quality using suitable evaluation metrics. In this study, to measure the quality of the results of the various deep learning algorithm, a total number of six universally agreed, standard, and easy-to-understand statistical evaluation metrics, namely precision, recall, F-measure, accuracy, Mathews correlation coefficient and area under the curve (AUC) were used to objectively assess the classification performance of the algorithms. It should be noted that the MCC metric was used to assess the performance of the deep learning algorithms on the Kaggle binary dataset only. This is because the MCC metric was designed to measure the quality of binary classifications only. Comprehensive descriptions and methods of computation for these metrics are discussed in Chapter Three of this study,

4.5 Effect of Train-Test Split on Performance

This section investigated the effect of varying the proportion of data used for training the algorithms on the algorithm performance in this study. The experiment compares using 70% training data against using 80% training data for both the Kaggle and the HSO datasets. Six state-of-the-art metrics, namely accuracy, precision, recall, F- measure, area under the curve and Mathews correlation coefficient, were used to assess the impact of proportion data size on performance.

4.5.1 Effect of Train-Test Split Ratio on Accuracy

Table 4.1 shows the accuracy results of the algorithms investigated in this work recorded while varying train-test split ratio.

Table 4.1: Effect of train-test split ratio on accuracy

Method	**HSO**		**Kaggle**	
	70/30	**80/20**	**70/30**	**80/20**
CNN	87	87,4	87,3	90,4
RNN	88,5	89,1	88,6	91,2
MLP	86,4	86,7	86,9	87,5
GRU	88,7	89,2	88,9	91
LSTM	88,9	91,2	85	91,8
LSTM with Attention	88,6	91	89,5	92
BERT	89	90	90	90
RoBERTa	90	91	90	92
XLNET	90	91	92	93
DistilBERT	91	92	93	93,5

Table 4.1 shows that for both the HSO dataset and the Kaggle dataset, the overall accuracy scores of all traditional deep learning algorithms improves significantly as the train-test split ratio increases. For instance, the performance of the convolutional neural network algorithm improved when the proportion of training data was increased from 70% to 80% on the HSO dataset. The performance further confirms the literature position by (Kowsari *et al.* 2019) that deep learning algorithms perform well with bigger datasets or training sets. From Table 4.1, it can also be observed that traditional deep learning algorithms recorded better accuracy scores on the smaller Kaggle dataset as compared to the bigger HSO dataset. This trend is due to the fact that the Kaggle dataset is more balanced than the HSO dataset; therefore, this allows the

algorithms to learn patterns from both datasets equally. The differences in performance of the transformer algorithms between the bigger HSO dataset and the smaller Kaggle dataset are negligible. This is attributable to the fact that transformer algorithms are based on pre-trained models. Therefore, they learn general English corpora features instead of features from the actual hate speech datasets explored in this study. This may also be due to dataset characteristics such as the proportion of code-mixed words. It can be observed that when using the 80:20 train-test split ratio, the overall performance of all traditional deep learning algorithms improved by an average of 1,6% in response to changing the dataset from HSO to Kaggle. Figure 4.10 depicts the impact of the train-test split ratio on accuracy.

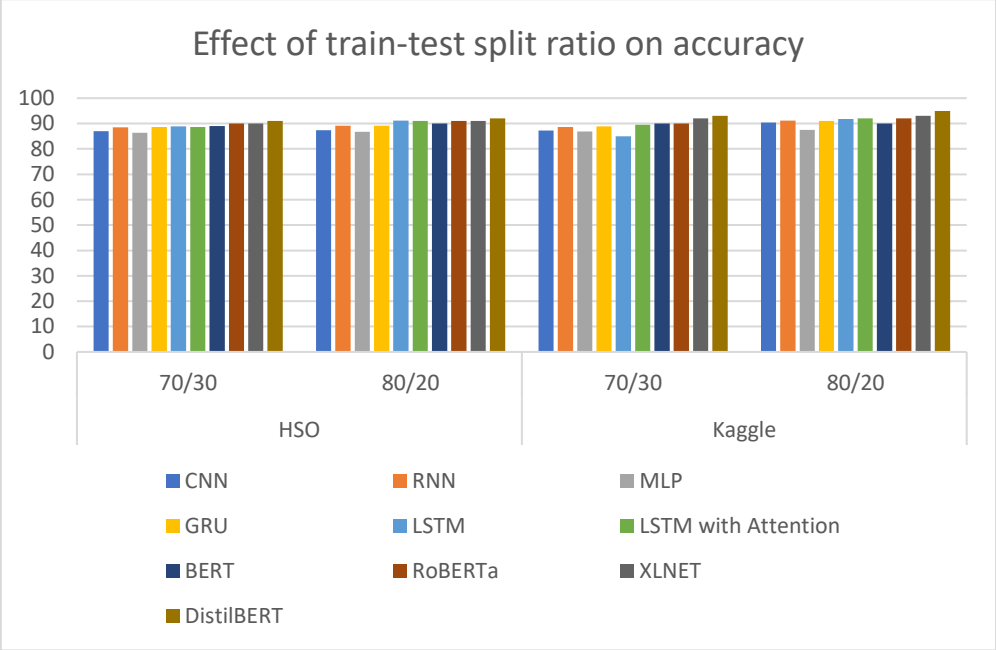

Figure 4.2: Illustration of the effect of the train-split ratio on accuracy

Figure 4.10 graphically illustrates the effect of the train-test split ratio on accuracy. It can be observed that the effect of dataset training size on transformer algorithm performance was negligible. Only small differences can be noted between the 80:20 train-test split ratio and the 70:30 train-test split ratio on both datasets. This trend is due to the fact that transformer algorithms are pretrained on general English corpora and are only fine-tuned for a given task. This means that they are not affected by dataset size and are therefore suitable for both large and small amounts of data. Figure 4.10 shows that other traditional deep learning algorithms had higher accuracy scores on the 80:20 dataset train-test split ratio as compared to the 70:30 train-test split ratio. This trend is attributable to the high number of training instances in the

80:20 train-test split ratio, which allows the algorithm to learn more patterns from the datasets, thereby producing a better model.

4.5.2 Effect of Train-Test Split Ratio on Precision

Table 4.2 displays the precision results of the algorithms investigated in this study under different train-test split ratios.

Table 4.2 Effect of train-test split ratio on precision

Algorithm	**HSO**		**Kaggle**	
	70/30	**80/20**	**70/30**	**80/20**
CNN	83	87,2	84,3	88,4
RNN	85,5	90	86,5	90,2
MLP	76,4	81	77,3	82
GRU	85,6	88,3	86	89
LSTM	86,4	90,4	87,4	90,8
LSTM with attention	88	91	88	91,9
BERT	88,7	91,2	89	89,8
RoBERTa	89	90,9	90,9	90
XLNET	93	95	93	94
DistilBERT	92	95	94	95

Table 4.2 shows the precision scores for all algorithms under the 70:30 and the 80:20 train-test split ratio. The overall precision scores improve as the training proportion of the dataset is increased from 70:30 to 80:20 train-test split ratio. Precision for all algorithms improved by 3,2% and 2,5% on the HSO and Kaggle dataset, respectively. A similar relationship can be observed on the test and split table, where the 80:20 train-test split ratio gives better

precision results for all algorithms across both datasets. The difference in the rate of precision improvement is attributed to the fact that deep learning algorithms perform better with bigger datasets.

Figure 4.11 graphically illustrates the effect of train-test split ratio split on precision scores.

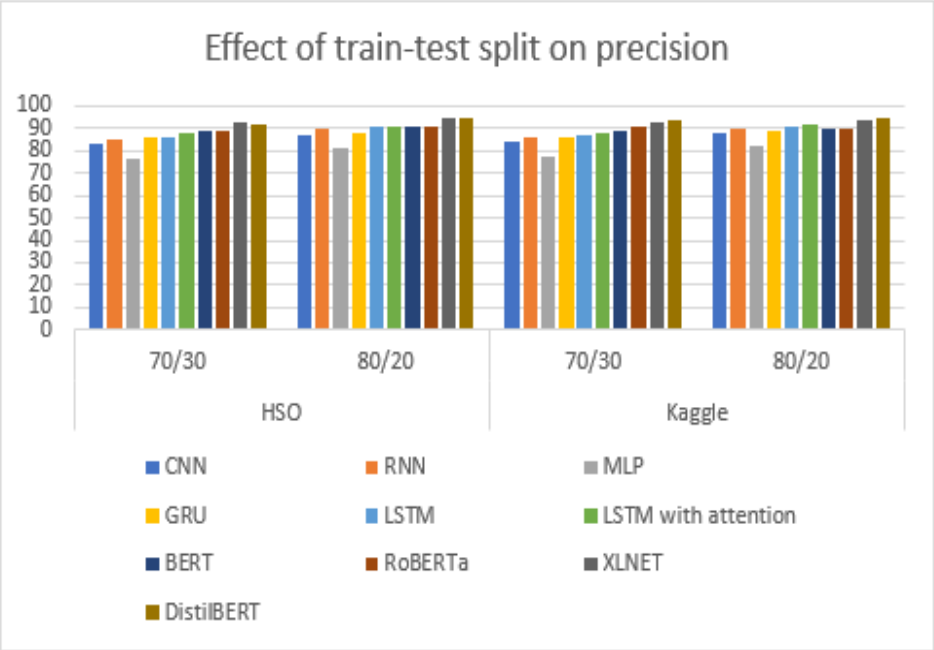

Figure 4.11: The effect of train-test split ratio on precision

Figure 4.11 depicts the effect of the train-test split ratio on precision scores. It can be observed that generally, algorithms performed better using the 80:20 train-test split ratio as compared to the 70:30 train-test split ratio. This trend substantiates the literature position by Kowsari *et al.* (2019) that the performance of machine learning algorithms is directly proportional to the amount of training data. However, increasing the training size only had a negligible effect on the performance of transformer-based algorithms. This is because transformer algorithms learn patterns from general English Corpora instead of the training datasets. Another pattern that can be observed from the figure is that algorithms performed generally better on the HSO dataset in comparison to the Kaggle dataset. This is because larger datasets have a greater number of training instances than smaller datasets. The high number of training instances allows the algorithms to learn the salient features necessary for classification. Training examples in the 80:20 train-test split ratio allow the algorithm to learn more patterns from the datasets, thereby producing a better model.

4.5.3 Effect of Train-Test Split on Recall

Table 4.3 shows the recall results of algorithms investigated in this study while varying the train-test split ratio.

Table 4.3: Effect of train-test split ratio on recall

Algorithm	**HSO**		**Kaggle**	
	70/30	**80/20**	**70/30**	**80/20**
CNN	83,2	87,4	78,3	81,9
RNN	86,5	89,8	84,5	86,7
MLP	77,4	81,2	71,3	69,7
GRU	85,6	86,3	81,2	83,4
LSTM	88,4	90,4	83,0	84,6
LSTM with attention	89,4	90,2	90,2	91
BERT	92,4	92,6	92,8	93
RoBERTa	92,6	93	92,8	93,6
XLNET	93,8	94	94	94,7
DistilBERT	94	97,5	95	96,4

From Table 4.3, it can be observed that the 80:20 train-test split ratio offers better recall performance as compared to the 70:30 train-test split ratio. The CNN gained the highest 3,6 % recall points from increasing the training test set by 10% on the Kaggle dataset. However, the performance of the MLP on the same dataset dropped by 0,6%. The reduction in performance of the MLP indicates that it may have failed to extract predictive features from the additional data. Recall for all algorithms was increased by an average of 2,8% on the bigger HSO dataset and 1.6% on the smaller dataset.

Figure 4.12 graphically depicts the effect of train-test split ratio on recall.

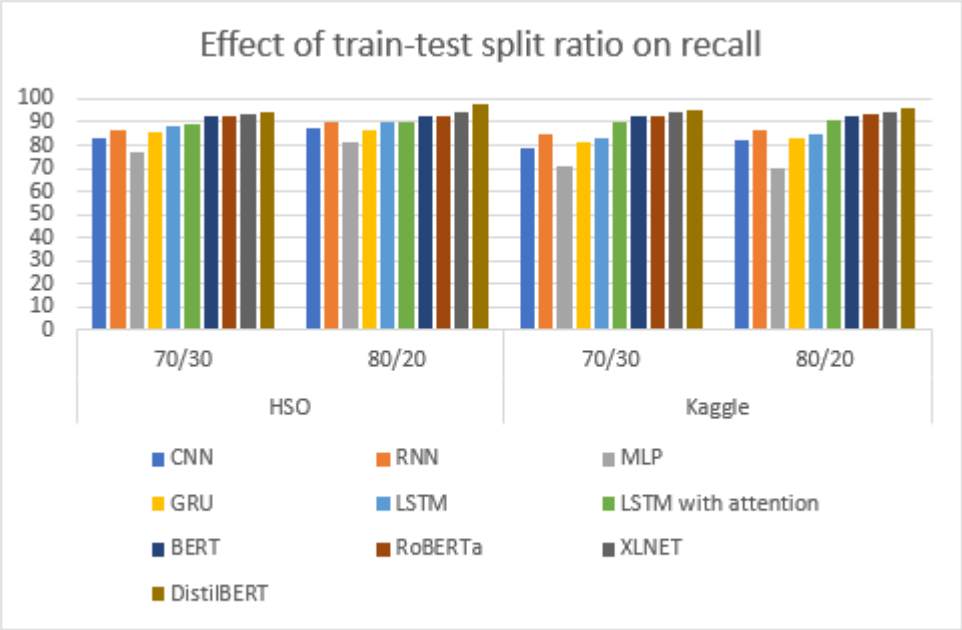

Figure 4.12: The effect of train-test split ratio on recall

Figure 4.12 presents the impact of the proportion of training data on recall scores. It can be observed that the 80:20 train-test split ratio gave a better performance in comparison to the 70:30 train-test split ratio. For example, the recall score of the CNN algorithm rose from 83,2 % to 87,4% when the training data was increased by 10% on the HSO dataset. It can also be observed that transformer algorithms consistently outperformed other algorithms regardless of the train-test split ratio used. This finding substantiates previous findings by Mutanga, Naicker and Olugbara (2020) who also observed similar patterns. This is attributed to the fact that transformer algorithms are pretrained on large English corpora, making them suitable for different tasks when required.

4.5.4 Effect of Train-Test Split Ratio on F-Measure Scores

Table 4.4 shows the F-measure results of the algorithms investigated in this work recorded while varying train-test split ratio.

Table 4.4: Effect of the train-test split ratio on F-measure

Method	**HSO**		**Kaggle**	
	70/30	**80/20**	**70/30**	**80/20**
CNN	83,1	87,3	77,8	82,1
RNN	86	89,9	84	86,4
MLP	76,9	81,1	70,8	69,7
GRU	85,6	87,3	80,6	82,4
LSTM	87,4	90,4	82,5	82,4
LSTM with attention	88,4	89	89	90
BERT	90,6	90,8	90,7	91
RoBERTa	91	93	92	92,2
XLNET	92,8	94	92,8	92,9
DistilBERT	93	94	92,8	93

The F-measure score is a harmonic mean of precision and recall, making it a more objective metric compared to most metrics used in text classification tasks. Although accuracy is the most popular classification metric, it can be primarily contributed by a large number of True Negatives, which may not be relevant for practical applications, whereas False Negatives and False Positives usually have negative implications that may need to be avoided. This is particularly true in the domain of hate speech detection, where misclassifying hate as non-hate is of concern more than the categorisation of neutral tweets as hate speech. Information from Table 4.4 reveals that the F-measure score is positively affected by an increase in a dataset's training proportion. All the algorithms perform consistently better with the 80:20 train-test split ratio as compared to the 70:30 train-test split ratio. The CNN and the MLP produced the most

significant improvement as a result of a 10% increase in training data. The MLP and CNN gained 4,2% on the HSO dataset as a result of the increase in training data,

Figure 4.13 graphically illustrates the effect of the train-test split ratio on F-measure.

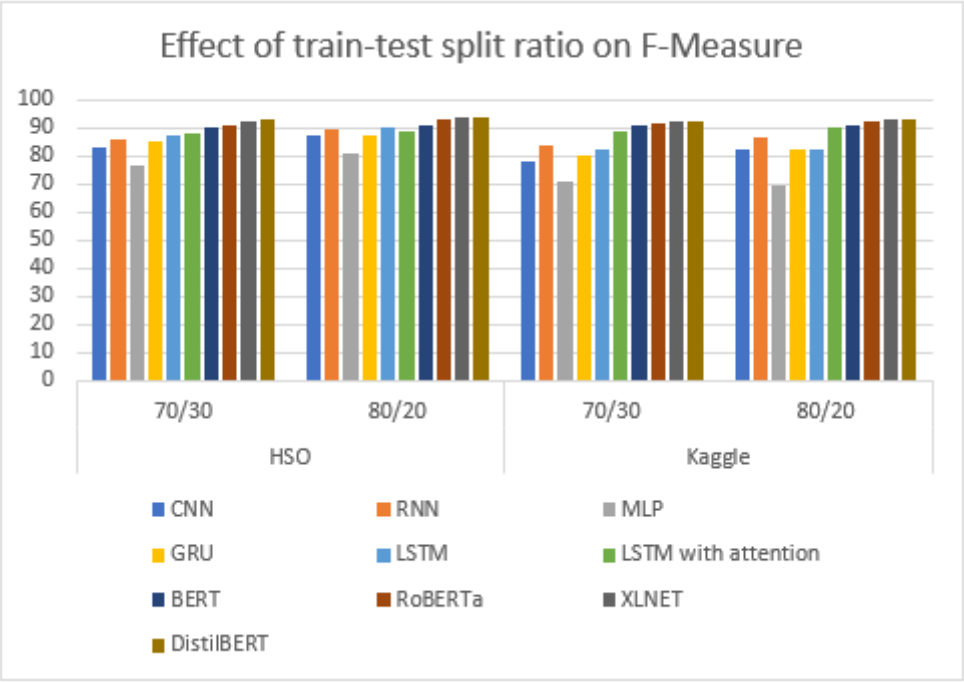

Figure 4.3: The effect of the train-test split ratio on F-measure

Figure 4.13 presents, in pictorial form, the effect of varying training dataset: testing dataset ratio on F-measure score. We can observe that all traditional deep learning algorithms performed better using the 80:20 train-test split ratio as compared to the 70:30 train-test split ratio. Additionally, one can observe that traditional algorithms performed consistently better on the HSO dataset in comparison to the Kaggle Dataset. This is attributable to a higher number of training examples which allow the algorithms to learn sufficient relevant patterns for the eventual classification task. On the other hand, the performance transformer algorithms fluctuated around the same levels regardless of dataset size or train-test split ratio.

4.5.5 Effect of Train-Test Split Ratio on Area under the Curve

Table 4.5 shows the AUC results of the algorithms investigated in this work recorded while varying the train-test split ratio.

Table 4.5: Effect of train-test split ratio on AUC

Method	HSO		Kaggle	
	70/30	**80/20**	**70/30**	**80/20**
CNN	92	93,1	84	88,4
RNN	95,2	96,7	88,5	99,2
MLP	85,4	87,3	81,3	85
GRU	92,6	93,3	89,9	90,2
LSTM	96,4	97,1	95,0	94,8
LSTM with attention	96,4	96,8	94,5	92
BERT	96,4	96,8	96,4	97
RoBERTa	96,5	96,4	96,6	96,7
XLNET	96,7	96,4	96,7	96,5
DistilBERT	97	97,3	97,4	97,5

Table 4.5 reflects the impact of varying the proportion of the training data on the AUC score for all datasets and algorithms. It can be observed that the AUC score is directly affected by an increase in training data. All the algorithms recorded increases in AUC after increasing the training set by 10%. However, transformer algorithms recorded a negligible improvement in AUC of 0,7% increase on the bigger HSO dataset. This confirms the literature position that transformer methods may be practical to use when training data is limited as compared to other deep learning algorithms (Antoun, Baly and Hajj 2020). On the other hand, RNN produced the most significant improvement as a result of dataset size increase, gaining 10,7% when it moved from 88,5% to 99.2% on the smaller Kaggle dataset.

Figure 4.14 graphically depicts the effect of the train-test split ratio on AUC scores for all algorithms.

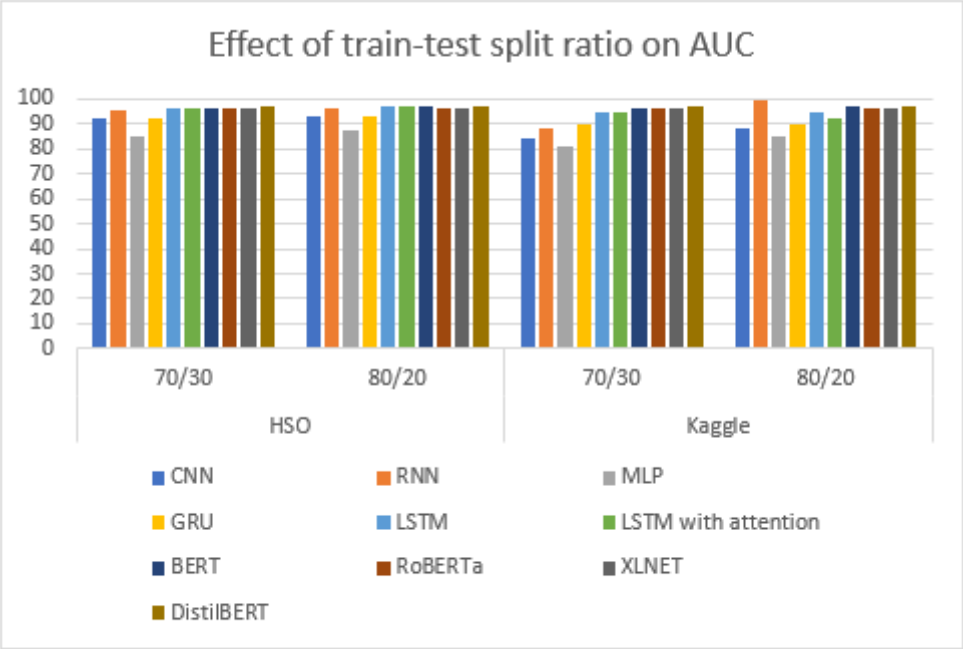

Figure 4.14: The effect of train-test split ratio on AUC

Figure 4.14 graphically depicts the impact of the train-test split ratio on AUC scores. It can be observed that generally, algorithms performed better using the 80:20 train-test split ratio as compared to the 70:30 train-test split ratio. For example, the AUC score of the RNN algorithm rose by more than 10% when the training set was increased by only 10%. Nevertheless, increasing the training size had a negligible effect on the performance of transformer-based algorithms. This is because transformer algorithms learn patterns from general English Corpora such as the Toronto Book Corpus instead of the training datasets. Figure 4.14 displays a clear trend where the LSTM with attention algorithm outperformed other traditional deep learning algorithms. This may be due to the algorithm's attention mechanism, which allows it to capture context, which is a salient characteristic of hate speech detection.

4.5.6 Effect of Train-Test Split Ratio on MCC

Table 4.6 shows MCC results for the algorithms investigated in this study recorded while varying train-test split ratio.

Table 4.6: Effect of train-test split ratio on MCC

Method	**Kaggle**	
	70/30	**80/20**
CNN	76	81,4
RNN	83,8	86,6
MLP	73,1	75
GRU	80,9	83
LSTM	83,2	87,6
LSTM with Attention	87,7	89,4
BERT	89	90
RoBERTa	88	89
XLNET	91	91,4
DistilBERT	91,6	93,2

Table 4.6 details the MCC scores for all algorithms based on the two-class dataset only computed from the overall test tweets. The table clearly shows that transformer methods outperformed other deep learning algorithms investigated in this work. Performance for all traditional deep learning algorithms improved when the training set was increased from 70% to 80%. This is because the selected algorithms had more data to learn the discriminative features from. However, the difference in MCC scores between the 70:30 train-test split ratio and the 80:20 train-test split ratio is negligible for transformer-based algorithms. This is because pretrained transformer models do not learn discriminative features from the dataset but from pretrained corpora such as Wikipedia and Toronto book corpus.

Figure 4.15 graphically illustrates the effect of the train-test split ratio on MCC.

Figure 4.15: Illustration of the effect of train-test split ratio on MCC

Figure 4.15 graphically depicts the effect of the train-test split ratio on MCC. It can be observed that the impact of dataset training size on transformer algorithm performance was negligible. Only small differences can be noted between the 80:20 train-test split ratio and the 70:30 train-test split ratio on the Kaggle Dataset. In particular, the XLNET. algorithm's score only changed by 0,4 % accuracy when the training set was increased from 70% to 80%. This trend confirms the literature position by Mutanga, Naicker and Olugbara (2020) that the impact on performance of training data size on pretrained transformer models is negligible. Figure 4.15 shows that other traditional deep learning algorithms had higher accuracy scores on the 80:20 dataset train-test split ratio compared to the 70:30 train-test split ratio. This trend is attributable to the high number of training instances in the 80:20 train-test split ratio, which allows the algorithm to learn more patterns from the datasets, thereby producing a better model.

4.6 Effect of Number of Layers on Performance

This section investigated the effect of varying the number of dense layers on the performance of deep learning algorithms explored in this study. The experiment looked at how using three dense layers, five dense layers, and seven dense layers affected each of the ten algorithms examined in this analysis. The effect of varying the number of layers

was evaluated based on six state-of-the-art metrics: accuracy, precision, recall, F-measure, AUC, and Mathews correlation coefficient.

4.6.1 Effect of Number of Layers on Accuracy

Table 4.7 shows the accuracy results of the algorithms investigated in this work recorded while varying the number of layers.

Table 4.7: Effect of number of layers on accuracy scores

Method	**HSO**			**Kaggle**		
	3 layers	**5 layers**	**7 layers**	**3 layers**	**5 layers**	**7 layers**
CNN	88,6	89,3	89	89	88	89
RNN	89,9	91	91,2	90	91,2	92
MLP	79,9	79,7	79,8	75,3	75	75,4
GRU	89.8	91,4	91	88	91	90
LSTM	91,2	91,4	92.1	87	89,2	90
LSTM with attention	91	89	92	89	91	91
BERT	91	92	92	94	93	93
RoBERTa	92	93	92	92	93	94
XLNET	92	94	94	94	93	95
DistilBERT	95	93	95	94	94	95

We observe from Table 4.7 that the performances of the accuracy scores of the majority of algorithms that the addition of layers has a positive impact on accuracy. Nevertheless, an increase in the number of layers beyond five leads to insignificant increases and, in some cases, a slight drop in performance. For instance, the addition of 2 more layers from 5 to 7 layers led to a decrease in the accuracy score of CNN and the MLP. Only the DistilBERT had its accuracy score falling when the number of layers were increased from three to five layers.

Figure 4.16 graphically depicts the effect of the number of layers on accuracy.

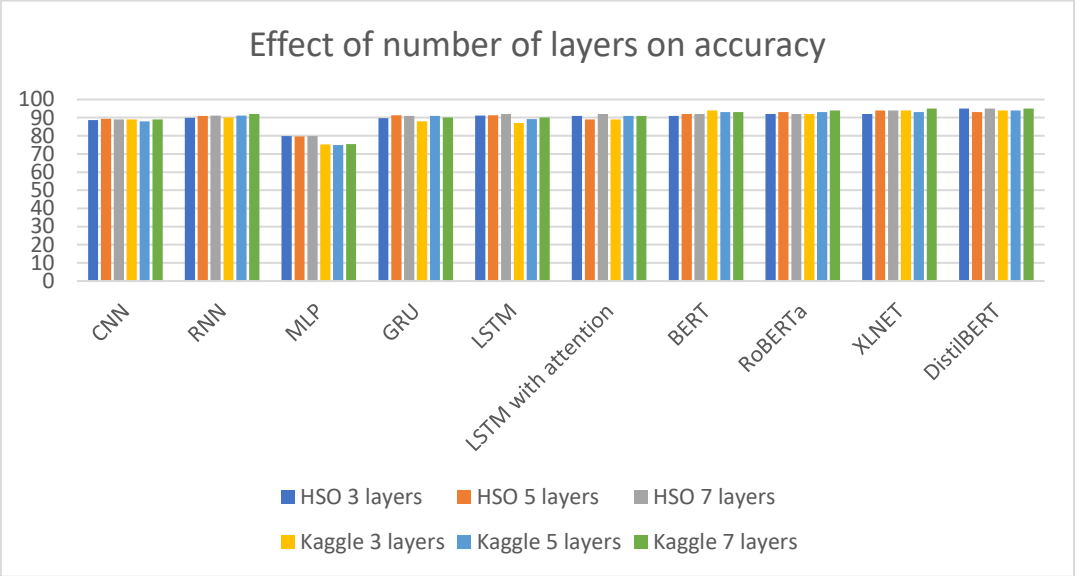

Figure 4.4: The effect of the number of layers on accuracy

Figure 4.16 shows the impact of the number of layers on the accuracy score for all algorithms across the two datasets used in this study. Increasing the number of layers improves the efficiency of most algorithms, as seen in the graph. However, a further increase of layers beyond five leads to either negligible increases or a reduction in the accuracy scores of the algorithms. One important trend to note is that, although scores from both datasets improved with the addition of layers, accuracy scores for the HSO dataset increased with a bigger margin compared to the Kaggle dataset. This is because the HSO dataset is a multiclass dataset, hence it is more complex and therefore requires more layers for the algorithms to learn the salient features.

4.6.2 Effect of Number of Layers on Precision Scores

Table 4.8 shows precision results for algorithms investigated in this work recorded while varying the number of layers.

Table 4.8: Effect of number of layers on precision

Method	HSO			Kaggle		
	3 layers	**5 layers**	**7 layers**	**3 layers**	**5 layers**	**7 layers**
CNN	84,6	82,3	80,	74	74,6	55
RNN	85,9	87	88,2	84,8	85,2	88,1
MLP	73,9	74,7	73,4	70,3	71	70,4
GRU	87,8	87,8	87,9	82,4	84,2	85,7
LSTM	88,2	92,4	92.5	87,2	89,2	91.0
LSTM with attention	89	88	88	90	89	89
RoBERTa	90	92	90	91	90	91
DistilBERT	91	89	90	92	90	91
BERT	94	95	94	95	94	95
XLNET	95	95	94	95	96	95

The effect of the number of layers is illustrated in Table 4.8. We note that the performance of most algorithms does not improve significantly after the 5th layer. In fact, the MLP and CNN's precision scores actually dropped after the addition of the 7th layer. For instance, the performance of CNN drops from 74,6% to 55% on the Kaggle dataset in response to an increase in layers from 5 to 7.

Figure 4.17 depicts the effect of the number of layers on precision.

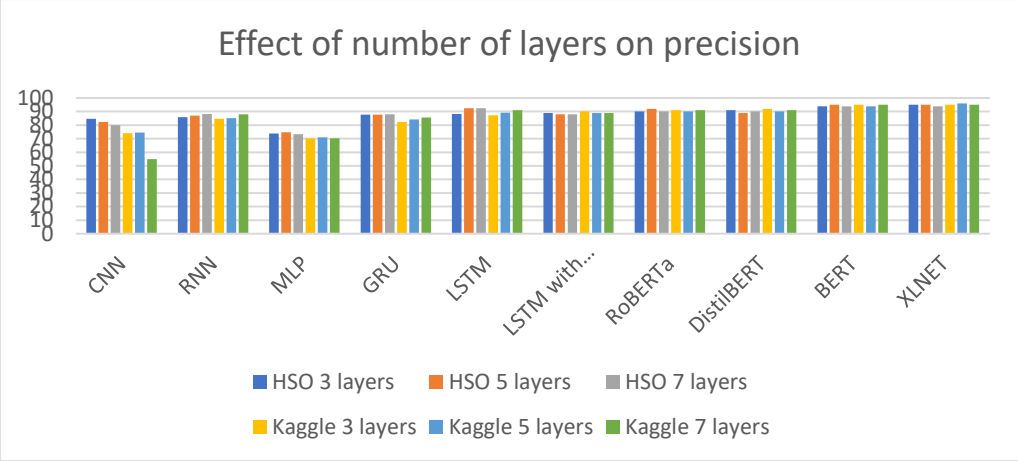

Figure 4.5: The effect of the number of layers on precision

Figure 4.17 shows the effect of the number of layers on the precision score for all algorithms across the two datasets used in this study. The graph clearly illustrates that performance increased with the addition of dense layers. However, the precision of CNN dropped drastically when the layers were increased from five to seven. This may have been caused by the model having become too complex given the training and validation datasets used in the researcher`s experiments.

4.6.3 Effect of Number of Layers on Recall Scores

Table 4.9 shows the recall results of the algorithms investigated in this work recorded while varying the number of layers.

Table 4.9: Effect of number of layers on recall

Method	**HSO**			**Kaggle**		
	3 layers	**5 layers**	**7 layers**	**3 layers**	**5 layers**	**7 layers**
CNN	85,6	84,3	82,0	74	74,8	75,0
RNN	86,9	86,0	87,2	83,8	84,2	86,1
MLP	73,7	74,8	73,6	70,6	71,2	71,4
GRU	87,6	87,8	88,9	82,4	84,4	85,2
LSTM	88,3	92,3	92,7	87,8	89,4	92,3
LSTM with attention	92,8	92	92	92,7	92,7	92,6
RoBERTa	93,7	93	93	92,8	92,7	92,6
DistilBERT	94	93,6	94	93,4	93,2	93,4
BERT	94	93	94	93,9	93	92,7
XLNET	97	97,5	96	97	96	96

From Table 4.9, it can be observed that there is no distinct pattern or correlation between performance and the number of layers that may be drawn from the recall scores of all algorithms. For instance, an increase in layers from three to five using RNN led to a decrease in performance from 86,9% to 86%. Further addition of number of layers resulted in a 1,2 % improvement in the recall score of the RNN. The performance of other algorithms such as CNN and RoBERTa was also fluctuating as the number of layers was changed.

Figure 4.18 shows a graphical representation of the effect of the number of layers on recall.

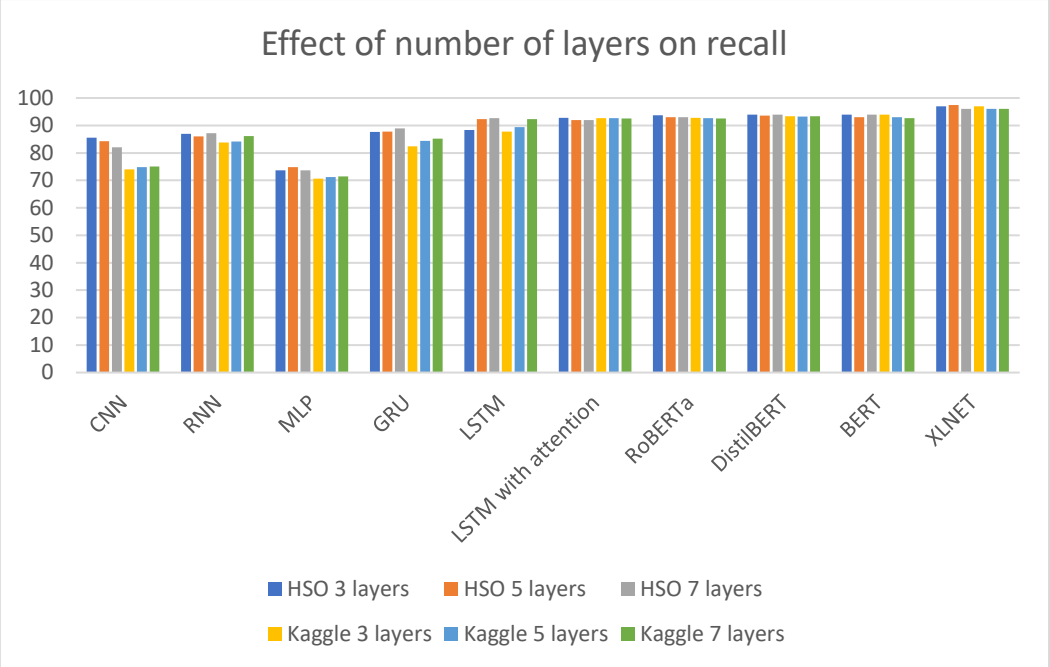

Figure 4.6: The effect of the number of layers on recall

Figure 4.18 illustrates the impact of varying the number of layers on the recall score for all algorithms across the two datasets used in this study. From the graph, we can see that the recall score fluctuated with the addition of dense layers. The optimal number of layers for each algorithm is not easily recognisable by observing the graph. Therefore, no distinct pattern or correlation between the number of layers and recall scores can be deduced from the findings of the experiments.

4.6.4 Effect of Number of Layers on F-Measure

Table 4.10 shows the F-measure results of the algorithms investigated in this work while varying the number of layers.

Table 4.10: Effect of number of layers on F-measure

Method	**HSO**			**Kaggle**		
	3 layers	**5 layers**	**7 layers**	**3 layers**	**5 layers**	**7 layers**
CNN	86	83,3	81	74	74,7	63,5
RNN	86,4	86,5	87,7	84,3	84,7	87,1
MLP	73,8	74,7	73,5	70,4	71,1	70,9
GRU	87,7	87,8	88,4	82,4	84,3	85,4
LSTM	88,2	92,3	92,6	87,5	89,3	91,6
LSTM with attention	93	93	91	92	91	90
RoBERTa	93	92,9	93,1	93	93	92,8
DistilBERT	93,8	93,5	93,2	93,9	93,8	93,7
BERT	94	94	93	94,3	94,2	94,3
XLNET	94	94	94,4	94,4	94,5	94,3

It can also be observed from Table 4.10 that no distinct correlation or pattern can be deduced from the results. There is no significant positive or negative impact of the number of layers on the F-measure score. The F-measure scores fluctuate as the number of layers increased.

Figure 4.19 shows a graphical representation of the effect of the number of layers on F-measure.

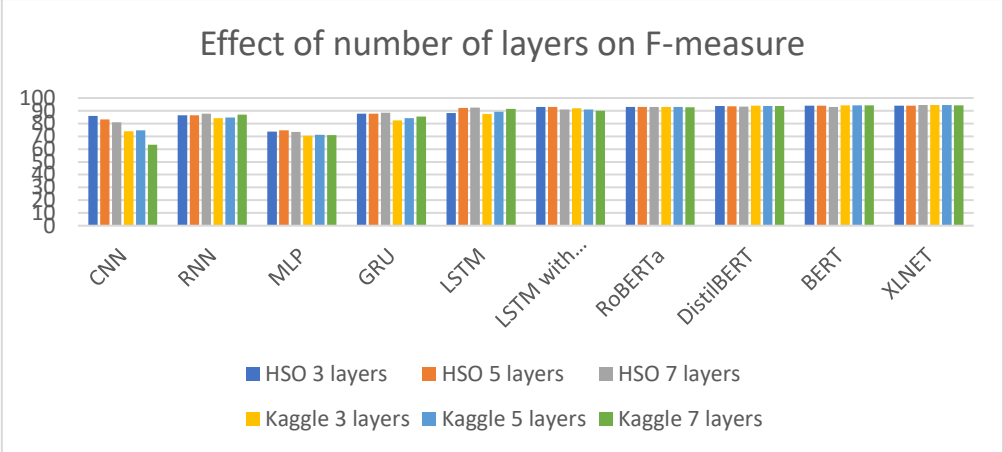

Figure 4.7: The effect of the number of layers on F-measure

Figure 4.19 presents the impact of varying the number of layers on the F-measure score for all algorithms across the two datasets investigated in this study. From the graph, we can see that the F-measure score for methods investigated in this work fluctuated with the addition of dense layers. Therefore, no distinct correlation can be deduced between the F measure score and the number of layers. This lack of correlation calls for further research in finding the optimal number of layers for the best F- Measure score.

4.6.5 Effect of Number of Layers on AUC

Table 4.11 displays the AUC results of the algorithms investigated in this work recorded while varying the number of layers.

Table 4.11: Effect of number of layers on AUC

Method	**HSO**			**Kaggle**		
	3 layers	**5 layers**	**7 layers**	**3 layers**	**5 layers**	**7 layers**
CNN	88,7	90,3	93,0	79	79,6	79,8
RNN	93,9	93,8	93,2	86,5	86,2	90,6
MLP	79,9	79,7	79,8	81,3	88	82,4
GRU	89,8	90,4	90	87,4	81,2	88,7
LSTM	91,2	91,4	91,8	87	89,2	92
LSTM with attention	93,4	93,2	93	88	87,9	88
RoBERTa	95,2	95,3	95	95,4	95,5	95,1
DistilBERT	95,2	95,1	95,3	95,6	95,7	96
BERT	95,7	95	95,6	96	96,3	95,5
XLNET	96	95	96	96,6	95,7	96,5

Table 4.11 shows that AUC scores fluctuate as the number of hidden layers is increased. In particular, RNN based algorithms such as simple RNN, GRU, LSTM and LSTM with attention. For instance, GRU's AUC increased by 0,6% in response to the increase in layers from 3 to 5 layers. When the number of layers was further increased to 7, the AUC score decreased by 0,4% on the HSO dataset. On the Kaggle Dataset, the AUC score for MLP decreased by 0.3% after increasing the number of layers from 3 to 5. Further increase in the number of layers to 7 led to an increase of 4.4%on the AUC score.

Figure 4.20 shows a graphical representation of the effect of the number of layers on AUC.

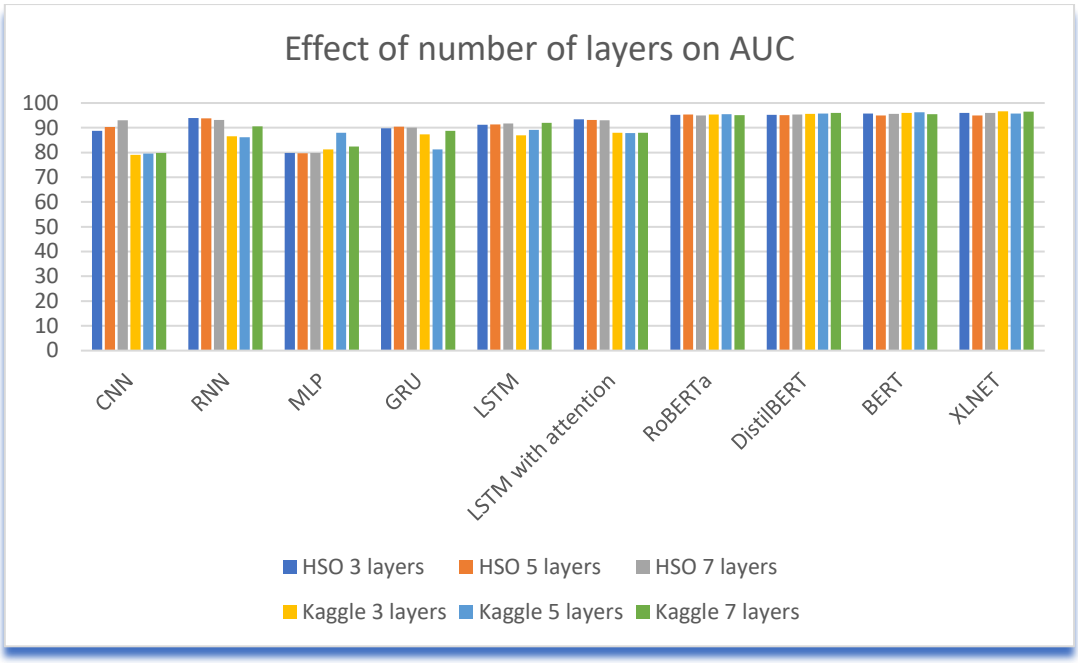

Figure 4.8: The effect of number of layers on AUC

Figure 4.20 illustrates the impact of varying the number of layers on the AUC score for all algorithms across the two datasets explored in this study. From the graph, we observe that the F-measure score for algorithms investigated in this work fluctuated with the addition of dense layers. In particular, RNN based algorithms such as GRU and LSTM displayed the highest instability levels as their AUC scores fluctuated. No correlation between the number of dense layers and AUC scores can be deduced from the graph. Therefore, it is difficult to ascertain the optimal number of layers for each of the algorithms explored in this study.

4.6.6 Effect of Number of Layers on MCC

Table 4.12 shows the MCC results of the algorithms investigated in this work recorded while varying the number of layers.

Table 4.12: Effect of number of layers on MCC

Method	**Kaggle Dataset**		
	3 layers	**5 layers**	**7 layers**
CNN	87	86	87,6
RNN	88	89,4	90
MLP	73,5	73	73,4
GRU	86	89,2	87,9
LSTM	85,2	87,1	88,2
LSTM with attention	89	90	89,8
BERT	92	91,2	91,3
RoBERTa	90	91,2	92,3
XLNET	92,2	91,5	92,8
DistilBERT	92	92,4	93

Table 4.12 show the relationship between the number of dense layers and algorithm performance. No distinct pattern or correlation can be obtained from varying the number of dense layers. For example, when the number of layers was increased from three to five, the performance of XLNET improved, but when the number of layers was increased to seven, the performance decreased. When the number of layers was increased from three to five to seven, only RoBERTa and BERT showed modest improvements in efficiency.

Figure 4.21 shows a graphical representation of the effect of the number of layers on MCC.

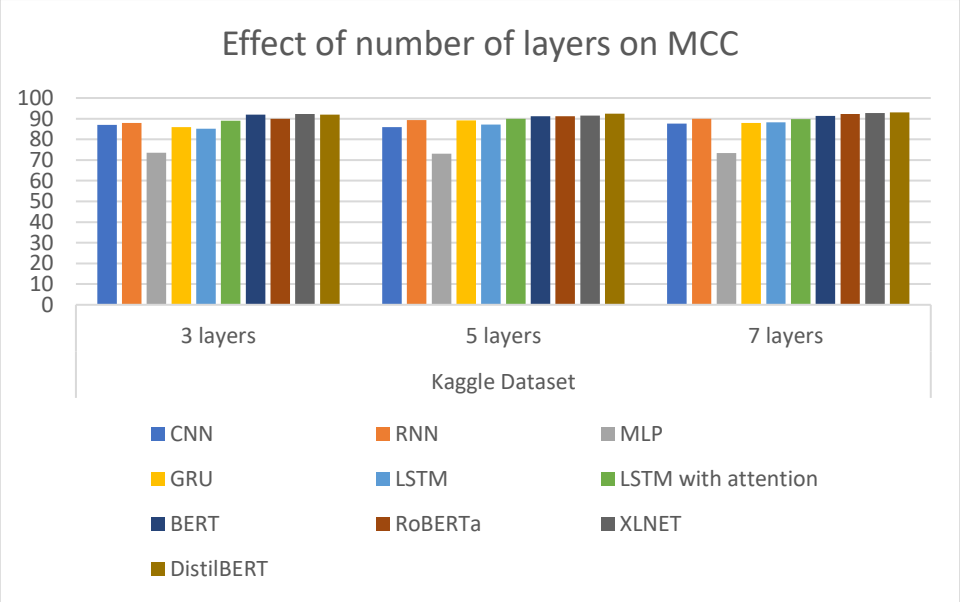

Figure 4.9: The effect of the number of layers on MCC

Figure 4.21 presents results on the impact of varying the number of layers on the MCC score for all algorithms on the Kaggle Dataset. From the graph, it can be observed that the MCC scores for the majority of the algorithm fluctuated as more layers were added. Nevertheless, the MCC score of the BERT and RoBERTa algorithms increased gradually with the addition of layers. The consistent performance of BERT and RoBERTa is because they are built on the same basic architecture as propounded by Liu *et* al (2019).

4.7 Effect of Optimiser on Performance

This section looked at the impact of different optimisers on the efficacy of deep learning algorithms. The experiment investigated the impact of using three different optimisers, namely, adam optimiser, adagrad optimiser and rmsprop optimiser, on each of the ten algorithms explored in this study. The impact of changing the layer count was assessed using six cutting-edge metrics: accuracy, precision, recall, F-measure, AUC, and Mathews correlation coefficient.

4.7.1 Effect of Optimiser on Accuracy

Table 4.13 shows the accuracy results of the algorithms investigated in this work recorded while varying optimisers.

Table 4.13: Effect of optimiser on accuracy

Method	**HSO**			**Kaggle**		
	Adam	**adagrad**	**rmsprop**	**adam**	**adagrad**	**rmsprop**
CNN	88,7	89,2	89,1	89,3	88,7	89,2
RNN	90,6	90	90,1	88,7	89,2	87,6
MLP	80,9	79,7	79,9	76,3	75	75,4
GRU	91,8	89,4	91	90,4	89,2	88,6
LSTM	92,2	91,4	91,1	90,1	89,2	90,0
LSTM with attention	92	91	90	92	92	91,6
BERT	92,6	92	91	93,6	93	92,5
RoBERTa	93	92	91	94	93	92,5
XLNET	93	92,6	92	94	93,6	92
DistilBERT	94	93	92,8	95	94	93,6

It can be noticed from Table 4.13 that the use of the adam optimiser produced superior accuracy scores as compared to other optimisers, performing in the majority of algorithms. The adam optimiser only failed to produce the best results using the CNN. The rmsprop produced the overall second-best accuracy score ahead of the adagrad, which produced the least accuracy scores. Adam optimiser's superior performance can be attributed to its ability to combine moment term for update of weight parameter and the sum of squared gradients to scale current gradients for weight updates (Ruder 2016).

Figure 4.22 illustrates the effect of optimiser on accuracy.

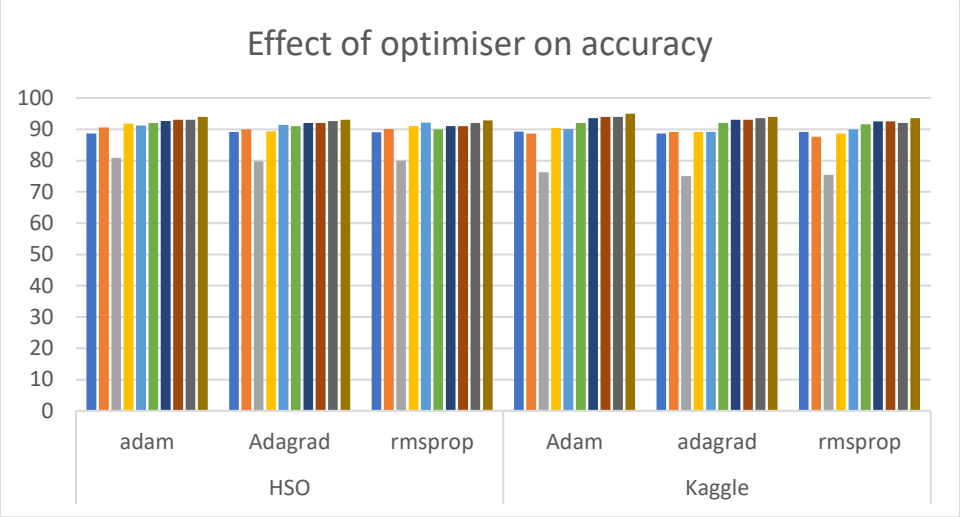

Figure 4.10: The effect of optimiser on accuracy

Figure 4.22 shows the performance of deep learning algorithms under different optimisers. The adam optimiser gave the best accuracy score on both datasets for all algorithms. The rmsprop optimiser gave the second-best accuracy scores, and the adagrad optimiser gave the lowest accuracy scores. This trend clearly shows that the adam optimiser is better suited to sequence-based tasks such as Natural Language Processing compared to rmsprop and adagrad. This trend further validates the findings by Mutanga, Naicker and Olugbara (2020), where the adam optimiser produced the best accuracy scores. Figure 4.22 shows a clear trend where the adam optimiser consistently gave higher accuracy scores on the HSO dataset as compared to the Kaggle dataset. This may be attributed to having more training examples in the HSO dataset.

4.7.2 Effect of Optimiser on Precision

Table 4.14 shows the precision results of the algorithms investigated in this work recorded while varying optimisers.

Table 4.14: Effect of optimiser on precision

Method	**HSO**			**Kaggle**		
	Adam	**adagrad**	**rmsprop**	**adam**	**adagrad**	**rmsprop**
CNN	86,7	86,7	86,1	79.3	78,7	79,2
RNN	89,6	88	89,2	88,7	86,2	85,6
MLP	78,6	76	78,2	72,3	72	72,1
GRU	90,6	90,4	90,6	86,4	84,2	85,6
LSTM	91,2	90,3	90,89	89,1	88,2	88,7
LSTM with attention	90	90,1	90	90	90,4	89,7
BERT	91	91	88	91	90,7	90
RoBERTa	92,4	92	91,8	92	92	90
XLNET	93	93,6	92,8	93	92	92
DistilBERT	94	89	93	94	93	92

Table 4.14 shows the impact of different optimisation algorithms on precision. It is noted that the adam optimiser performed consistently superior to other algorithms, which are rmsprop and adagrad. Adam had an average precision score of 88.6 per cent ahead of adagrad and rmsprop, which had 87.8% and. 87.7% respectively across all datasets. It should be, however, noted that the effect of changing optimisation algorithms did not have a significant impact on the Multi-Layer Perceptron algorithm, suggesting that its performance may be improved by varying other parameters.

Figure 4.23 depicts the effect of optimiser on precision.

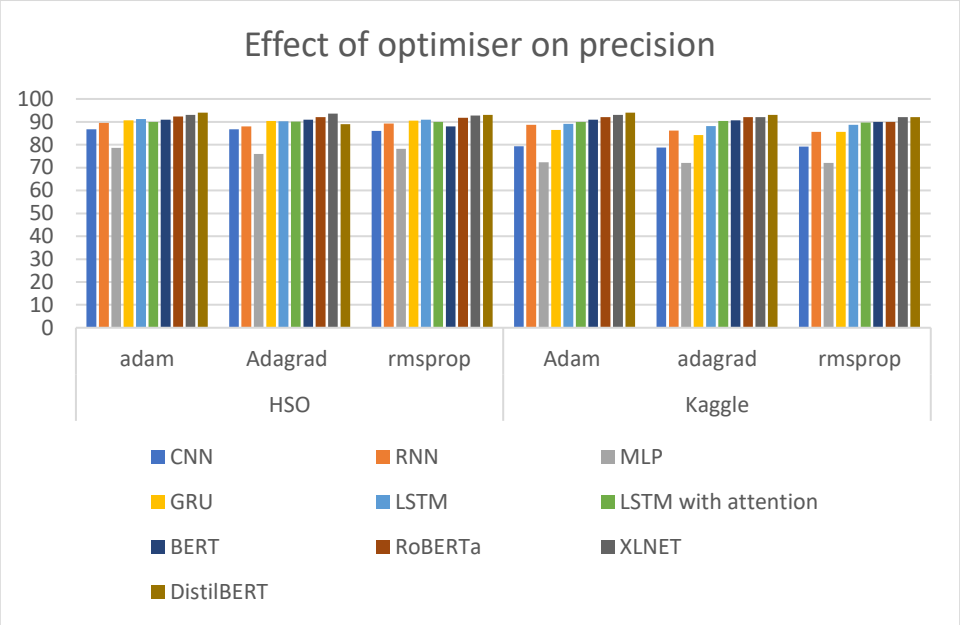

Figure 4.11: The effect of optimiser on precision

Figure 4.23 presents the impact of different optimisers on precision score under three different optimisers. The adam optimiser achieved the best score on both datasets. The adam optimiser also performed consistently higher across all ten algorithms on the larger HSO dataset as compared to the Kaggle Dataset. This trend may be attributable to having more training examples in the larger dataset. The differences in performance when using the adagrad and the rmsprop optimisers, are negligible, as shown in Figure 4.23. A closer look at the performance of the MLP shows that changing optimisation algorithms did not have a significant impact on the Multi-Layer Perceptron algorithm, suggesting that its performance may be improved by varying other parameters.

4.7.3 Effect of Optimiser on Recall

Table 4.15 shows the recall results of the algorithms investigated in this work recorded while varying optimisers.

Table 4.15: Effect of optimiser on recall

Method	**HSO**			**Kaggle**		
	Adam	**adagrad**	**rmsprop**	**adam**	**adagrad**	**rmsprop**
CNN	85,7	85,7	86,1	79,3	78,7	79,2
RNN	89,6	88,0	89,2	88,7	86,2	85,6
MLP	78,6	76	78,2	72,3	72	72,1
GRU	90,6	90,4	90,55	86,4	84,2	85,6
LSTM	91,2	90,3	90,89	89,1	88,2	88,7
LSTM with attention	91	91,1	91	91	91,4	90,8
BERT	92	92	89	92	91,7	91
RoBERTa	93,4	93	92,8	93	93	91
XLNET	94	94,6	93,7	94	93	93
DistilBERT	95	90	94	95	94	93

It can be noticed from Table 4.15 that the use of the adam optimiser produced superior recall scores as compared to other optimisers explored in this study, performing in best in nine of the ten algorithms. The adam optimiser only failed to produce the best results using the LSTM with attention. The rmsprop produced the overall second-best recall score ahead of the adagrad, which produced the least accuracy scores. Adam recorded an average of 90,2 % on the HSO dataset, while rmsprop had an average score of 89,1% on the HSO dataset. Adam's superior performance can be due to its capability to combine moment term for the update of weight parameter and the sum of squared gradients to scale current gradients for weight updates (Ruder 2016).

Figure 4.24 displays a graphical representation of the effect of optimiser on recall.

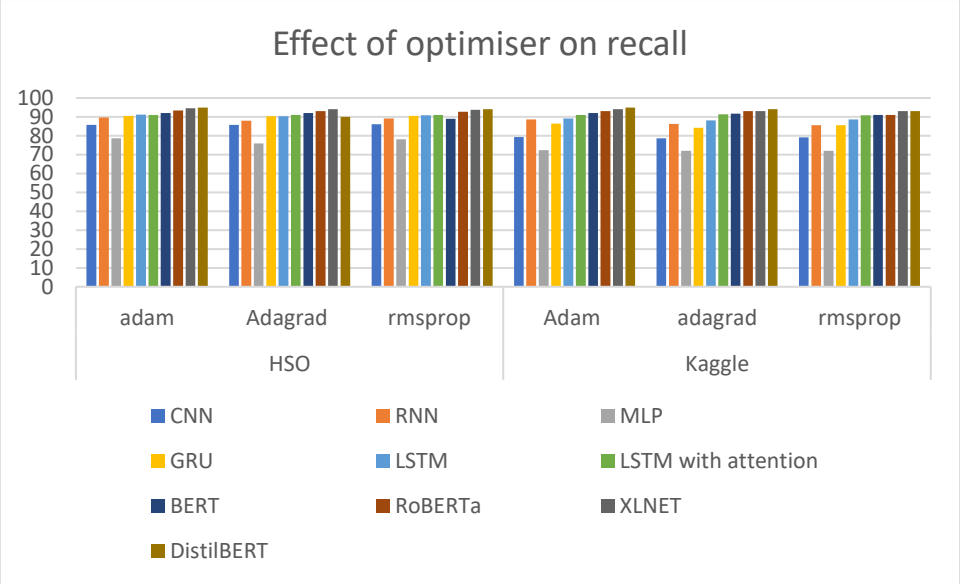

Figure 4.12: The effect of optimiser on recall

Figure 4.24 provides a pictorial illustration of the effect of optimiser on recall score. It is apparent that the best result was obtained using the adam optimiser on both datasets. On the HSO dataset, it can be observed that nine of the ten algorithms had a recall score above 84%. Only the MLP scored below 80%. The RMS prop had the second-best performance after adam. However, on closer inspection, it can be noted that the difference between the performance of the adagrad and rmsprop was negligible. This minor difference calls for fine-tuning of parameters such as learning rate, early stopping patience and Network dropout to improve performance.

4.7.4 Effect of Optimiser on F-measure scores

Table 4.16 shows the F-measure results of the algorithms investigated in this work recorded while varying optimisers.

Table 4.16: Effect of optimiser on F-measure

Method	**HSO**			**Kaggle**		
	Adam	**adagrad**	**rmsprop**	**adam**	**adagrad**	**Rmsprop**
CNN	87,8	86,7	86,1	79,3	78,7	79,2
RNN	89,6	88	89,2	88,7	86,2	85,6
MLP	78,6	76	78,2	72,3	72,	72,0
GRU	90,6	90,4	90,55	86,6	84,4	85,6
LSTM	91,3	90,3	90,9	89,1	88,1	88,7
LSTM with attention	91,7	91,6	91,2	90	89	89,4
BERT	92,5	92,4	92,4	92	91	90
RoBERTa	92,9	92,6	92,5	93	92,8	92,7
XLNET	93	92,9	92,7	93,5	93	92,8
DistilBERT	93,8	93,2	93	94	93,6	93

From Table 4.16, we observe that the adam optimisation algorithm consistently outperforms other optimisation techniques investigated in this work. The best result was produced by the LSTM while using the adam optimiser. This indicates the LSTM's suitability for hate speech detection where false negatives and should be minimised. It is interesting to note that transformer models outperformed other deep learning algorithms under varying conditions. LSTM with attention had the second-best performance after the transformer models, consistently outperforming other traditional deep learning algorithms. It can be concluded that the superior result of the LSTM with attention over traditional algorithms is due to LSTM with attention's ability to capture more context through long term dependencies.

Figure 4.25 shows a graphical representation of the effect of optimiser on F-measure.

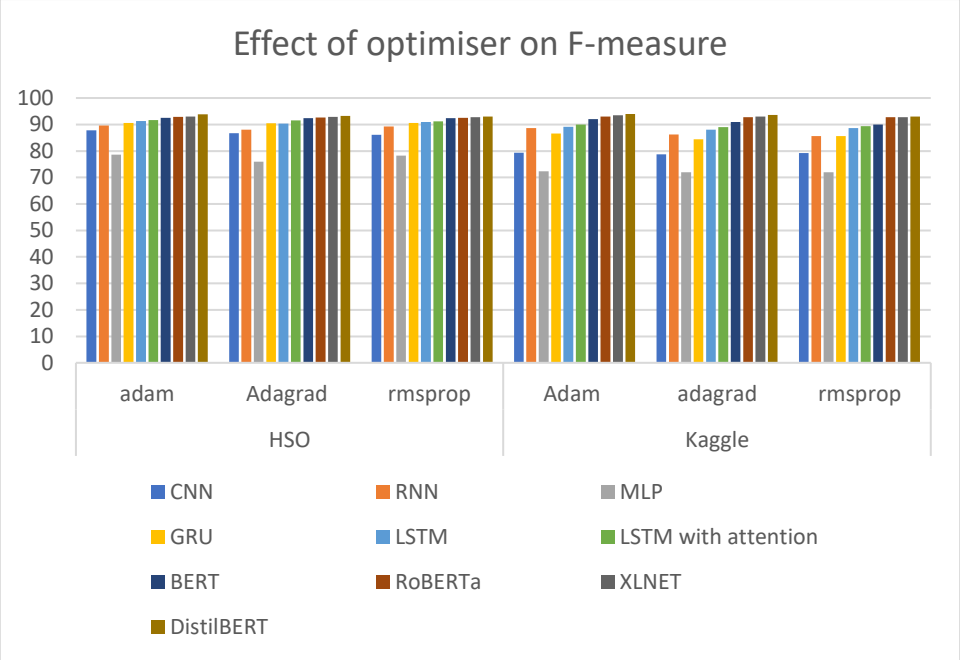

Figure 4.13: The effect of optimiser on F-measure

Figure 4.25 depicts the influence of optimiser on the F- Measure score. It is apparent that the best result was obtained using the adam optimiser. On the HSO dataset, it can be observed that seven of the ten algorithms gave an F measure score above the 90% line. Only the MLP scored below 80%. This may be attributed to other factors, such as the fewer number of layers within the MLP thereby leading to inferior discriminative capabilities. This is particularly true for highly unstructured and complex problems such as classification of subjective text.

4.7.5 Effect of Optimiser on Area under the Curve

Table 4.17 shows AUC results for the algorithms investigated in this work recorded while varying optimisers.

Table 4.17: Effect of optimiser on AUC

Method	**HSO**			**Kaggle**		
	adam	**adagrad**	**rmsprop**	**adam**	**adagrad**	**rmsprop**
CNN	89,7	89,2	89,4	79,3	78,7	79,2
RNN	91,8	89,9	91,1	89,7	87,2	89,8
MLP	81	81	76	78	75	75,6
GRU	97,8	90,4	93	89,4	88,2	85,6
LSTM	91,2	91,4	92,1	90,1	89,2	90,0
LSTM with attention	92,2	92,1	92	92	91	89
BERT	93,8	93	92,6	94	94,1	94
RoBERTa	95,1	95	94,7	95,8	95,6	95
XLNET	95,2	95	94,8	95,9	95,6	95
DistilBERT	95,4	94,8	94,6	95,7	94,9	95

Table 4.17 displays the experimental results of varying the optimisation algorithm for each of the algorithms. It can be seen from the table that the adam optimiser consistently outperforms other optimisers. Adam optimiser produced the highest average AUC score of 92,3 % and 90 % on the HSO and Kaggle datasets, respectively. There were very small differences in the AUC scores of rmsprop and adagrad. On the HSO dataset, adagrad recorded 92%, while rmsprop recorded an average score of 91%. On the Kaggle dataset, adagrad recorded an AUC score of 89%, while rmsprop also recorded 89%. Expectedly transformer methods outperformed other deep learning algorithms explored in this study. DistilBERT produced the highest AUC score of 97,5%, which was achieved using 7 dense layers on the smaller HSO dataset.

Figure 4.26 shows a graphical representation of the effect of optimiser on AUC.

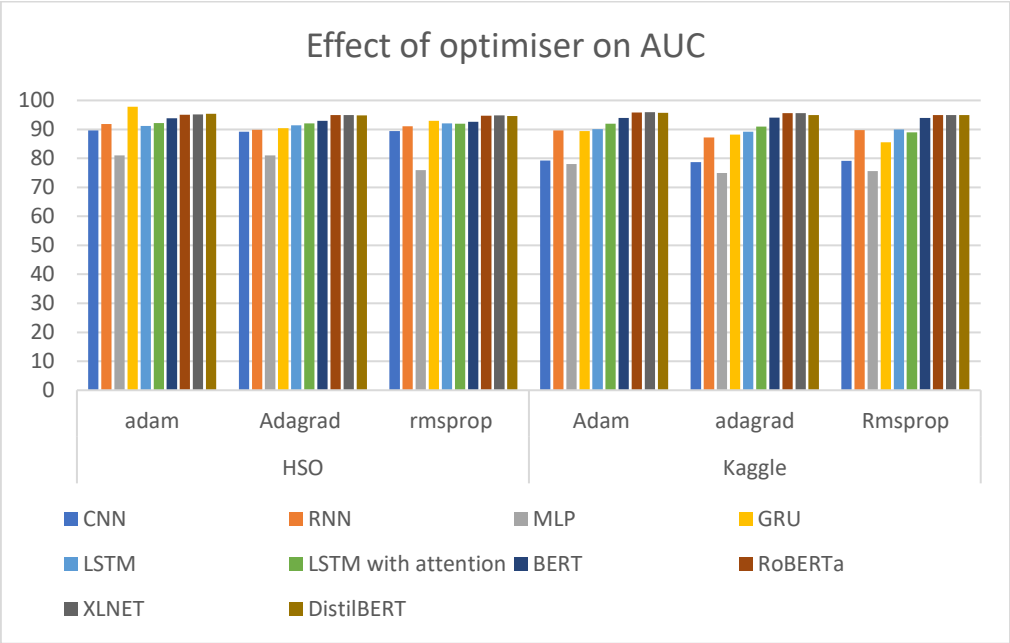

Figure 4.14: The effect of optimiser on AUC

Figure 4.26 shows the effect of varying the optimiser on the AUC score for all the deep learning methods investigated in this work. It is noted that the adam optimiser gave the best AUC score on both datasets for all algorithms. It is apparent that the DistilBERT algorithm gave the best AUC score of close to 100 % achieved under the HSO dataset. The rmsprop optimiser gave the second-best AUC scores after the adam optimiser, while the adagrad optimiser gave the least accuracy scores. The superior performance of the adam optimiser in hate speech detection is consistent with its success in related NLP tasks (Ruder 2016). Transformer-based models gave higher AUC scores as compared to other methods investigated in this work.

4.7.6 Effect of Optimiser on MCC

Table 4.18 shows the MCC results of the algorithms investigated in this study recorded while varying optimisers.

Table 4.18: Effect of optimiser on MCC

Method	**Kaggle**		
	adam	**Adagrad**	**rmsprop**
CNN	87,3	86,7	87,2
RNN	86,7	87,2	85,6
MLP	74,4	73	73,4
GRU	88,4	87,1	86,7
LSTM	88,1	87,2	88,1
LSTM with attention	90	90,2	89,5
BERT	91,5	91,2	90,4
RoBERTa	92	91,1	90,6
XLNET	92,2	91,7	90,1
DistilBERT	93	92,2	91,4

Table 4.18 demonstrates that the algorithms achieved their best performance using the adam optimiser while they performed worst using the adagrad optimiser. The worst performing algorithm was MLP, which achieved an MCC score of 73% using adagrad. On the other hand, DistilBERT had the best MCC score of 93% achieved using the adam optimiser.

Figure 4.15 shows a graphical representation of the effect of optimiser on MCC.

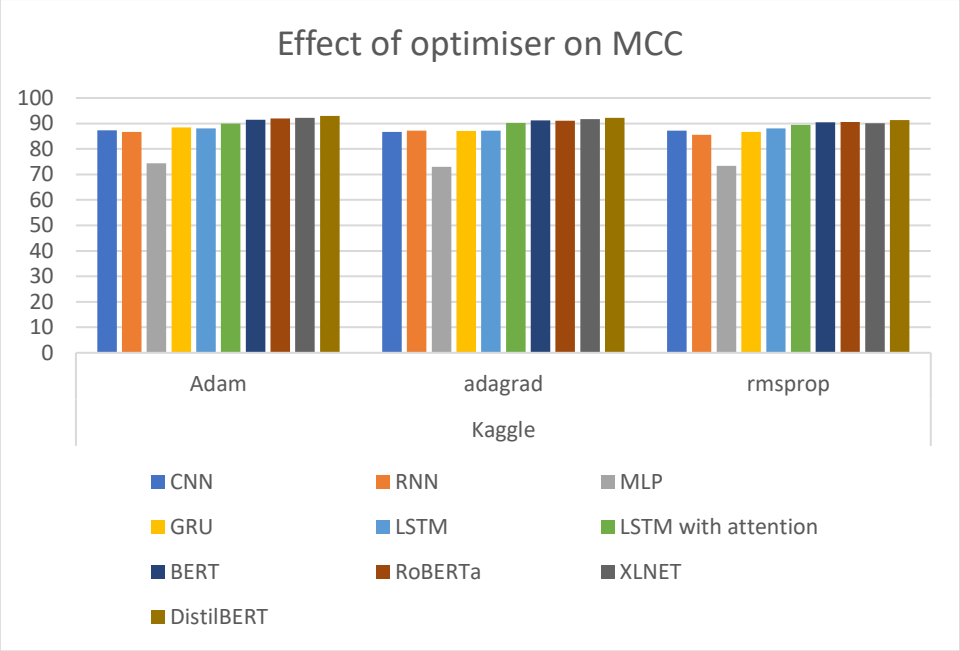

Figure 4.15: The effect of optimiser on MCC

Figure 4.27 provides a pictorial Illustration of the effect of optimiser on the MCC score. It is apparent that the best result was obtained using the adam optimiser on both datasets. The DistilBERT algorithm gave the best MCC score, while the MLP gave the least MCC score. The poor performance of the MLP may be due to the complexity of the dataset, as it is known that the MLP struggles with complex datasets. The superior performance of the DistilBERT may be due to its ability to generalise different instances of hate speech since it is pretrained on general corpora.

4.8 Chapter Summary

This chapter covered the empirical evaluation of ten cutting edge deep learning algorithms for detecting hateful text messages. To systematically evaluate the efficacy of each algorithm, six metrics were used under different parameters and settings, such as the number of layers, train-test split ratio as well as different optimisation algorithms. Two publicly available datasets were used in the experiments. The evaluation methods quantitatively express the extent to which the algorithms are capable of detecting text-based hate speech. The experiments conducted point to a high correlation between the ground truth text labels and the binary classification results produced by all the algorithms. The computed scores across the six evaluation metrics scored relatively high values across the data sets.

CHAPTER FIVE

SUMMARY, CONCLUSIONS AND IMPLICATIONS OF STUDY

5.1 Introduction

This chapter concludes the dissertation and provides valuable insights into current and future research in the same area. Firstly, the summary of the study gives a brief overview of each of the five chapters making up this study. Thereafter, the dissertation revisits each of the study`s objectives and further explains how each of the objectives was accomplished. The contributions of the study section highlight the significant contributions this study is making to the body of knowledge. The next section highlights the implications of the study to both the research community and practice. The limitations and future work section state the restrictions and scope of this study and propose possible directions for future work. The chapter concludes with a summary, which gives a brief overview of the chapter.

5.2 Summary of the Study

In this study, ten deep learning algorithms were experimentally evaluated for their efficacy in automatically detecting hate speech. The study`s overarching aim was to investigate and evaluate the efficacy of deep learning algorithms in detecting hate speech on Twitter.

Chapter One provided a background to the societal effects of hate speech and clearly explained why this phenomenon requires automated solutions. The statement of the problem briefly outlined the research gap which this report seeks to fill. Furthermore, the aims and objectives of this study were clearly highlighted. The limits within which the study was conducted were also explained in the scope of the study section of the first chapter.

Chapter Two reviewed relevant publications based on the societal impact of hate speech and the application of deep learning algorithms for detecting hateful speech. The first section of this chapter clearly outlined the adverse effects of hate speech on society. Practical examples were cited, and the inadequacy of manual human annotators for this task was explained. The need for automated hate speech detection methods was justified. Literature based on deep learning for hate speech detection was comprehensively surveyed, and ten deep learning algorithms were selected for experimental comparisons. The absence of an experiment-based comparative analysis was identified as a research gap based on the literature surveyed. The ten chosen algorithms for comparative evaluation were: Recurrent Neural Networks, Long Short Term Memory, Gated Recurrent Unit, MultiLayer Perceptron, Convolutional Neural Network, RoBERTa, XLNET., BERT and DistilBERT. The selection of machine learning methods

included classical deep learning algorithms, deep learning with partial attention and transformer algorithms which are based on full attention.

Chapter Three covered the methodological steps followed in setting up and carrying out the experiments. The datasets used in this study were presented. A Python-based function was developed to preprocess the datasets by removing less predictive words and characters from the tweets. The preprocessing phase aimed to improve algorithm performance by reducing the feature space of the dataset. The word embedding feature representation method, which was used to convert text into feature vectors by deep learning, was presented and explained, along with the experimental setup for training and evaluating the selected deep learning algorithms. In particular, the parameter settings for the experiments were specified. The experiments were carried out while varying optimisation algorithms, the number of dense layers and the proportion of the training data.

Chapter Four of this study covered the presentation and discussion of the experimental results. Firstly, training and validation accuracy graphs were presented and analysed. Secondly, the impact of number parameters such as layers, optimisation algorithms and the proportion of training data on model performance was presented and analysed. Six state-of-the-art performance metrics, accuracy, precision, recall, F-measure, area under the curve and Mathews correlation coefficient, were used to objectively evaluate each of the selected algorithms. Results were presented in tabular and graphic form for all the algorithms under different conditions.

Chapter Five concludes the dissertation by firstly presenting an overview of the work carried out in the study. This is followed by a discussion of the extent to which set objectives were met. Major contributions of the dissertation to the body of knowledge are articulated. Furthermore, the implications of the study to both future research and practice are outlined. Limitations of the study and possible directions for future work conclude the chapter.

5.3 Conclusions

This dissertation investigated hate speech detection as a text classification task. A comprehensive review of the research work was presented. Despite concerted efforts from researchers, hate speech propagation remains an important problem that is yet unsolved. The work reported in this study has experimentally investigated ten deep learning algorithms for

hate speech detection in Twitter text. A comprehensive literature survey unearthed state of the art in deep learning and hate speech. In particular, the inclusion of transformer-based models in this study was motivated by their success in other text classification tasks, as reported in the literature. Twitter-specific preprocessing was applied to the data to reduce the feature space and training time for each of the models. During training parameters such as optimisation algorithms, the number of dense layers and the proportion of training data were varied to determine their impact on performance. Our findings show that different algorithms respond differently to the same parameter tuning. The algorithms were tested on two benchmark data sets to measure their accuracy, effectiveness and validate their performance. Results from this study are comparable to and even outperform other existing state-of-the-art hate speech detection models. Moreover, the performance of the newer deep learning techniques such as attention has indeed shown that the methods can be feasible in live environments where resources may be constrained.

In order to tackle the identified research issues, the following research objectives were formulated:

[RO1]: To comprehensively review relevant publications based on deep learning.

[RO2]: To prepare datasets for building and evaluating deep learning hate speech recognition models.

[RO3]: To experimentally evaluate the performance of deep learning algorithms on imbalanced and balanced datasets.

[RO4]: To evaluate the performance of deep learning algorithms on binary and multiclass datasets.

[RO5]: To evaluate the performance of deep learning algorithms in detecting different forms of hate speech.

To address the first objective [RO 1], an extensive review of relevant publications based on the societal impact of hate speech and the application of deep learning methods to address hate speech was conducted. The first section of Chapter Two clearly outlined the adverse effects of hate speech on society. Practical examples were cited, and the inadequacy of manual human annotators for this task was explained. The need for automated hate speech detection methods was clearly justified. Literature based on deep learning for hate speech detection was comprehensively surveyed, and ten deep learning algorithms were selected for experimental

comparisons. The absence of an experiment-based comparative analysis was identified as a research gap based on the literature surveyed. The ten chosen algorithms for comparative evaluation were: Recurrent Neural Networks, Long Short Term Memory, Gated Recurrent Unit, MultiLayer Perceptron, Convolutional Neural Network, RoBERTa, XLNET., BERT and DistilBERT.

To address the second objective [RO 2], the development of a preprocessing method for textual Twitter hate speech data was presented in Chapter Three of this study. The proposed Twitter preprocessing program was implemented, taking into consideration the conversational nature of Twitter data. A python-based function was developed to remove characters and words that may be considered less predictive in determining tweet classes. Firstly, the Twitter handles (@user were removed from all tweets in the dataset. After that, a function to remove punctuation marks and special characters was applied to the dataset. Python s Natural Language Toolkit library was then used to identify and remove stop words within the datasets. Lastly, stemming and tokenisation were applied to the datasets. All of these steps aimed to improve algorithm performance by reducing the feature space of the dataset.

To meet the third objective [RO 3] of this study, two datasets were selected for use in this study`s experiments. The first dataset was highly imbalanced, with only 5,8% of the instances in the dataset representing the *hate* class while the rest of the tweets were either neutral or offensive. The second dataset was well balanced as it contained 1 150 hate speech instances 1 150 neutral instances. The same experiments explained in Chapter Three of this work were conducted on both datasets to get an objective assessment of the effect of class imbalance.

The fourth objective, [RO 4], was achieved by selecting and experimenting with two datasets with different number of classes. The HSO dataset contained three classes labelled as hate speech, neutral speech, and offensive speech. Both datasets were exposed to the same conditions during experimentation.

To accomplish the fifth objective, [RO 5] of this study, multiple experiments were done to assess the efficacy of the deep learning algorithms selected for investigation in this study. Both qualitative and quantitative methods were used to provide an exhaustive evaluation of the deep learning algorithms. During the training process, graphs outlining the algorithms' learning curve were plotted to show visual clues of properties like dimensionality. Six state-of-the-art metrics, namely, area under the curve, precision, recall, accuracy, F-measure and MCC, were used for quantitative evaluation and comparison of deep learning algorithms performance. The

experiments were conducted under different conditions. In particular, different train-test split ratios, optimisers, and the number of dense layers were investigated to determine their effect on the algorithms' overall performance. To get an objective assessment of algorithm performance, two datasets of significantly different sizes were used in the experiments. The HSO dataset contained 24 784 tweets labelled into three classes, while the Kaggle dataset contained 2 300 tweets labelled into two classes. The researcher managed to determine the effect of dataset size on the efficiency of deep learning algorithms due to the availability of two datasets of significantly different sizes.

5.4 Contributions of the Study

The study aimed to empirically evaluate the efficacy of deep learning methods in detecting hate speech. This dissertation has investigated the hate speech detection problem by carrying out an experimental comparison of ten selected deep learning algorithms on Twitter data. An explanation of the key contributions made in this study is given next.

Despite significant progress in the detection of hate speech on social media, to the best of the researcher’s knowledge, no prior study has considered experimentally evaluating deep learning methods, which was accomplished in this work. Most importantly, the idea of carrying out a deep learning comparative study was inspired by the growing availability of large, annotated datasets. Earlier studies were limited due to the lack of large datasets with sufficient training examples for learning by deep learning algorithms. Results from this study provide a guideline on which algorithms are the best in addressing the problem of hate speech dissemination.

Pretrained transformer models have achieved state of the art results in Natural Language Processing tasks. Nevertheless, to the best of the researcher's knowledge, no work had proposed the use of pre-trained models for detecting hateful speech in the English Language. In this study, the inclusion of pretrained transformer-based models represented a new paradigm in automated hate speech detection. Since transformers are trained in general English corpora and fine-tuned to suit the task of hate speech detection, they allow the development of models with better generalisation capabilities as compared to models trained on specific hate speech datasets. Furthermore, streamlined transformer models such as DistilBERT are ideal for use in environments where computing power is limited since they are computationally inexpensive.

5.5 Implications of the Study

The implications of this study are presented in two broad categories, implications for research and implications for practice.

5.5.1 Implications for Research

Researchers need to actively harvest multimodal data as it constitutes a significant proportion of user-generated content on social media. This will enable them to include such data in future datasets, allowing the development of models capable of capturing hate speech expressed in video or image formats.

5.5.2 Implications for Practice

Social media has become a popular advertising platform for various organisations. Many organisations have Twitter handles and Facebook pages which they use to market their products as well as communicate with current and prospective customers. However, some users abuse these platforms by posting harmful content, which affects the user experience of other customers. Organisations might mitigate this phenomenon by adopting automatic machine learning-based models to detect and delete harmful content on their platforms.

5.6 Limitations and Future Work

This section presents an evaluation of the work presented in this study. The limitations of this study and envisaged future work are highlighted. Some of the challenges and concerns highlighted in this section may not be directly linked to the objectives of this study, but their significance in the automated detection of hate speech and text classification should be noted. Despite significant research progress in hate speech detection, the need to improve the efficacy of future models still exists. The researcher presents possible research recommendations for further improvements based on this study.

Because of inadequate resources and time, only ten deep learning algorithms were experimentally evaluated for the task of hate speech detection. Deep learning is a growing area, and there are many deep learning algorithms that can be explored for text classification tasks, such as hate speech detection. Recently, algorithms such as Spiking Networks and Deep Belief Networks have been investigated for pattern recognition tasks. A more extensive experiment-based comparative study which includes Deep Belief Networks and Spiking Networks, could lead to more useful findings.

This research was restricted to Twitter text only. It should, however, be noted that hate speech on Twitter may also be conveyed via videos and images. For instance, one can upload a xenophobic video without being detected by a model trained on a text dataset. Such a challenge necessitates the inclusion of images and videos in future datasets. The development of models capable of capturing both textual and non-textual could improve hate speech detection.

Due to time constraints, only one feature representation method was used for all traditional deep learning algorithms in this work. In future, there is a need to examine the impact of varying feature representation methods for detecting hate speech using deep learning algorithms. Feature representation has a direct impact on properties such as dimensionality. Hence it may also affect performance.

The majority of freely accessible hate speech datasets are in English. It is vital that datasets for other widely spoken languages like Mandarin and Swahili are developed. This will help to detect hate speech expressed in other languages.

Hate speech may be expressed in more than one language, particularly in Africa, where most people mix their native language with either French or English. The development of multilingual datasets will help to capture hate speech expressed in more than one language.

From the experiments carried out, it was observed that different deep learning algorithms respond differently to changes in parameters. This makes it very difficult to come up with a totally objective assessment of the performance of deep learning algorithms. In the future, there is a need to develop a universal framework for the objective evaluation of deep learning algorithms.

5.7 Chapter Summary

The chapter offered a comprehensive overview of how each of this study's objectives was met. The formulation of the research problem, literature review, study experiments and analysis of results were summarised. Hate speech detection remains an active research problem that requires concerted effort from industry and academia. The chapter concluded with the limitations of the research and points the way to future work in this research area.

REFERENCES

Adamson, A. and Turan, V. D. 2015. *Opinion Tagging Using Deep Recurrent Nets with GRUs*. Available: https://cs224d.stanford.edu/reports/AdamsonAlex.pdf (Accessed 20 January 2021).

Ahluwalia, R., Soni, H., Callow, E., Nascimento, A. and De Cock, M. 2018. Detecting hate speech against women in english tweets. *EVALITA Evaluation of NLP and Speech Tools for Italian*, 12: 194-199.

Ajao, O., Bhowmik, D. and Zargari, S. 2018. Fake news identification on twitter with hybrid cnn and rnn models. In: Proceedings of *the 9th International Conference on Social Media and Society*. Copenhagen, Denmark, 226-230. Available: https://doi.org/10.1145/3217804.3217917 (Accessed 22 January 2021).

Akosa, J. 2017. Predictive accuracy: A misleading performance measure for highly imbalanced data. In: Proceedings of *SAS Global Forum*. 2-5.

Al-Smadi, M., Qawasmeh, O., Al-Ayyoub, M., Jararweh, Y. and Gupta, B. 2018. Deep Recurrent neural network vs. support vector machine for aspect-based sentiment analysis of Arabic hotels' reviews. *Journal of Computational Science*, 27: 386-393.

Aljazeera. 2021. *What caused xenophonic attacks in South Africa*. Available: https://www.aljazeera.com/news/2016/4/6/what-caused-the-xenophobic-attacks-in-south-africa (Accessed 14/01/21).

Alshaalan, R. and Al-Khalifa, H. 2020. Hate Speech Detection in Saudi Twittersphere: A Deep Learning Approach. In: Proceedings of *Proceedings of the Fifth Arabic Natural Language Processing Workshop*. 12-23.

Antai, R. 2016. A New Hybrid Approach to Sentiment Classification. University of Essex.

Antoun, W., Baly, F. and Hajj, H. 2020. AraBERT: Transformer-based model for Arabic language understanding. *arXiv preprint arXiv:2003.00104*.

Badjatiya, P., Gupta, S., Gupta, M. and Varma, V. 2017. Deep learning for hate speech detection in tweets. In: Proceedings of *Proceedings of the 26th International Conference on World Wide Web Companion*. 759-760.

Bahad, P., Saxena, P. and Kamal, R. 2019. Fake News Detection using Bi-directional LSTM-Recurrent Neural Network. *Procedia Computer Science*, 165: 74-82.

Bahdanau, D., Cho, K. and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Baktha, K. and Tripathy, B. 2017. Investigation of recurrent neural networks in the field of sentiment analysis. In: Proceedings of *2017 International Conference on Communication and Signal Processing (ICCSP)*. IEEE, 2047-2050.

Banks, J. 2010. Regulating hate speech online. *International Review of Law, Computers & Technology*, 24 (3): 233-239.

Baruah, A., Barbhuiya, F. and Dey, K. 2019. ABARUAH at SemEval-2019 Task 5: Bi-directional LSTM for Hate Speech Detection. In: Proceedings of *Proceedings of the 13th International Workshop on Semantic Evaluation*. 371-376.

Bengio, Y., Goodfellow, I. and Courville, A. 2017. *Deep learning*. Available: https://www.researchgate.net/publication/320703571_Ian_Goodfellow_Yoshua_Bengio_and_Aaron_Courville_Deep_learning_The_MIT_Press_2016_800_pp_ISBN_0262035618 (Accessed 21 January 2021).

Bleich, E. 2011. The rise of hate speech and hate crime laws in liberal democracies. *Journal of Ethnic and Migration Studies*, 37 (6): 917-934.

Brown, A. 2018. What is so special about online (as compared to offline) hate speech? *Ethnicities*, 18 (3): 297-326.

Brownlee, J. 2018. What is the difference between a batch and an epoch in a neural network? *Machine Learning Mastery* (Blog). Available: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/#:~:text=batches%20and%20epochs.-,What%20Is%20the%20Difference%20Between%20Batch%20and%20Epoch%3F,passes%20through%20the%20training%20dataset. (Accessed 18 December 2020).

Brownlee, J. 2021. How to use Learning Curves to Diagnose Machine Learning Model Performance. Available: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/ (Accessed 22 January 2021).

Büyüköz, B., Hürriyetoğlu, A. and Özgür, A. 2020. Analyzing ELMo and DistilBERT on Socio-political News Classification. In: Proceedings of *Proceedings of the Workshop on Automated Extraction of Socio-political Events from News 2020*. 9-18.

Cambria, E., Poria, S., Gelbukh, A. and Thelwall, M. 2017. Sentiment analysis is a big suitcase. *IEEE Intelligent Systems*, 32 (6): 74-80.

Charitidis, P., Doropoulos, S., Vologiannidis, S., Papastergiou, I. and Karakeva, S. 2020. Towards countering hate speech against journalists on social media. *Online Social Networks and Media*, 17: 100071.

Chen, G., Ye, D., Xing, Z., Chen, J. and Cambria, E. 2017. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In: Proceedings of *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2377-2383.

Cheng, J., Li, P., Zhang, X., Ding, Z. and Wang, H. 2017. CNN-based sequence labeling for fine-grained opinion mining of microblogs. In: Proceedings of *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 94-103.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. and Kuksa, P. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12 (Aug): 2493-2537.

Davidson, T., Warmsley, D., Macy, M. and Weber, I. 2017. Automated hate speech detection and the problem of offensive language. In: Proceedings of *Eleventh international AAAI conference on web and social media*.

De la Pena Sarracén, G. L., Pons, R. G., Cuza, C. E. M. and Rosso, P. 2018. Hate speech detection using attention-based LSTM. *EVALITA Evaluation of NLP and Speech Tools for Italian*, 12: 235.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dey, R. and Salemt, F. M. 2017. Gate-variants of gated recurrent unit (GRU) neural networks. In: Proceedings of *2017 IEEE 60th International Midwest Symposium On Circuits and Systems (MWSCAS)*. IEEE, 1597-1600.

Do, H. T.-T., Huynh, H. D., Van Nguyen, K., Nguyen, N. L.-T. and Nguyen, A. G.-T. 2019. Hate speech detection on vietnamese social media text using the bidirectional-lstm model. *arXiv preprint arXiv:1911.03648*.

Duncan, B. and Zhang, Y. 2015. Neural networks for sentiment analysis on Twitter. In: Proceedings of *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*. IEEE, 275-278.

Elouali, A., Elberrichi, Z. and Elouali, N. 2020. Hate Speech Detection on Multilingual Twitter Using Convolutional Neural Networks. *Revue d'Intelligence Artificielle*, 34 (1): 81-88.

Ethnologue Languages of the world. 2021. Available: https://www.ethnologue.com/ (Accessed 10/01/21).

Fortuna, P., da Silva, J. R., Wanner, L. and Nunes, S. 2019. A hierarchically-labeled portuguese hate speech dataset. In: Proceedings of *Proceedings of the Third Workshop on Abusive Language Online*. 94-104.

Fortuna, P. and Nunes, S. 2018a. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51 (4): 85.

Fortuna, P. and Nunes, S. 2018b. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51 (4): 1-30.

Gambäck, B. and Sikdar, U. K. 2017. Using convolutional neural networks to classify hate-speech. In: Proceedings of *Proceedings of the first workshop on Abusive Language Online*. 85-90.

Gao, L. and Huang, R. 2017. Detecting online hate speech using context aware models. *arXiv preprint arXiv:1710.07395*.

Gaumont, N., Panahi, M. and Chavalarias, D. 2018. Reconstruction of the socio-semantic dynamics of political activist Twitter networks—Method and application to the 2017 French presidential election. *PLoS One*, 13 (9).

Grandini, M., Bagli, E. and Visani, G. 2020. Metrics for Multi-Class Classification: An Overview. *arXiv preprint arXiv:2008.05756*.

Gu, Q., Zhu, L. and Cai, Z. 2009. Evaluation measures of the classification performance of imbalanced data sets. In: Proceedings of *International Symposium on Intelligence Computation and Applications*. Springer, 461-471.

Hochreiter, S. and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9 (8): 1735-1780.

Holmes, D. E. and Jain, L. C. 2006. *Innovations in machine learning*. Springer.

Hu, Y.-C. 2010. Pattern classification by multi-layer perceptron using fuzzy integral-based activation function. *Applied Soft Computing*, 10 (3): 813-819.

Huang, P., Xie, X. and Sun, S. 2019. Multi-view Opinion Mining with Deep Learning. *Neural Processing Letters*, 50 (2): 1451-1463.

Jain, G., Sharma, M. and Agarwal, B. 2019. Optimizing semantic LSTM for spam detection. *International Journal of Information Technology*, 11 (2): 239-250.

James, G., Witten, D., Hastie, T. and Tibshirani, R. 2013. *An introduction to statistical learning*. Springer.

Kadhim, A. I. 2018. An Evaluation of Preprocessing Techniques for Text Classification. *International Journal of Computer Science and Information Security (IJCSIS)*, 16 (6).

Karlik, B. and Olgac, A. V. 2011. Performance analysis of various activation functions in generalized MLP architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1 (4): 111-122.

Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kowsari, K., Brown, D. E., Heidarysafa, M., Meimandi, K. J., Gerber, M. S. and Barnes, L. E. 2017. Hdltex: Hierarchical deep learning for text classification. In: Proceedings of *2017 16th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 364-371.

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L. and Brown, D. 2019. Text classification algorithms: A survey. *Information*, 10 (4): 150.

Krueger, D., Maharaj, T., Kramár, J., Pezeshki, M., Ballas, N., Ke, N. R., Goyal, A., Bengio, Y., Courville, A. and Pal, C. 2016. Zoneout: Regularizing rnns by randomly preserving hidden activations. *arXiv preprint arXiv:1606.01305*.

Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., Zhong, V., Paulus, R. and Socher, R. 2016. Ask me anything: Dynamic memory networks for natural language processing. In: Proceedings of *International conference on machine learning*. 1378-1387.

Kursuncu, U., Gaur, M., Lokala, U., Thirunarayan, K., Sheth, A. and Arpinar, I. B. 2019. Predictive analysis on Twitter: Techniques and applications. In: *Emerging research challenges and opportunities in computational social network analysis and mining*. Springer, 67-104.

Kwok, I. and Wang, Y. 2013. Locate the hate: Detecting tweets against blacks. In: Proceedings of *Twenty-Seventh AAAI Conference On Artificial Intelligence*.

Lai, S., Xu, L., Liu, K. and Zhao, J. 2015. Recurrent convolutional neural networks for text classification. In: Proceedings of *Twenty-ninth AAAI Conference on Artificial Intelligence*.

Le, Q. and Mikolov, T. 2014. Distributed representations of sentences and documents. In: Proceedings of *International conference on machine learning*. 1188-1196.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Van Der Laak, J. A., Van Ginneken, B. and Sánchez, C. I. 2017. A survey on deep learning in medical image analysis. *Medical image analysis*, 42: 60-88.

Liu, G. and Guo, J. 2019. Bidirectional LSTM with attention mechanism and convolutional layer for text classification. *Neurocomputing*, 337: 325-338.

Liu, P., Li, W. and Zou, L. 2019. NULI at SemEval-2019 Task 6: transfer learning for offensive language detection using bidirectional transformers. In: Proceedings of *Proceedings of the 13th International Workshop on Semantic Evaluation*. 87-91.

Liu, Q., Zhou, F., Hang, R. and Yuan, X. 2017. Bidirectional-convolutional LSTM based spectral-spatial feature learning for hyperspectral image classification. *Remote Sensing*, 9 (12): 1330.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

MacAvaney, S., Yao, H.-R., Yang, E., Russell, K., Goharian, N. and Frieder, O. 2019. Hate speech detection: Challenges and solutions. *PLoS One*, 14 (8).

McClelland, J. L., Rumelhart, D. E. and Group, P. R. 1986. Parallel distributed processing. *Explorations in the Microstructure of Cognition*, 2: 216-271.

Mehdad, Y. and Tetreault, J. 2016. Do characters abuse more than words? In: Proceedings of *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 299-303.

Melamud, O., McClosky, D., Patwardhan, S. and Bansal, M. 2016. The role of context types and dimensionality in learning word embeddings. *arXiv preprint arXiv:1601.00893*.

Melis, G., Dyer, C. and Blunsom, P. 2017. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*.

Merity, S., Keskar, N. S. and Socher, R. 2017. Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*.

Mhammedi, Z., Hellicar, A., Rahman, A. and Bailey, J. 2017. Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. In: Proceedings of *International Conference on Machine Learning*. PMLR, 2401-2409.

Mikolov, T., Chen, K., Corrado, G. and Dean, J. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Dean, J., Le, Q., Strohmann, T. and Baecchi, C. 2013b. Learning representations of text using neural networks. In: Proceedings of *NIPS Deep Learning Workshop*. 1-31.

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M. and Gao, J. 2020. Deep learning based text classification: A comprehensive review. *arXiv preprint arXiv:2004.03705*.

Miok, K., Skrlj, B., Zaharie, D. and Robnik-Sikonja, M. 2020. To ban or not to ban: Bayesian attention networks for reliable hate speech detection. *arXiv preprint arXiv:2007.05304*.

Mishev, K., Gjorgjevikj, A., Vodenska, I., Chitkushev, L. T. and Trajanov, D. 2020. Evaluation of Sentiment Analysis in Finance: From Lexicons to Transformers. *IEEE Access*, 8: 131662-131682.

Mondal, M., Silva, L. A. and Benevenuto, F. 2017. A measurement study of hate speech in social media. In: Proceedings of *Proceedings of the 28th ACM conference on Hypertext and Social Media*. 85-94.

Mossie, Z. and Wang, J.-H. 2020. Vulnerable community identification using hate speech detection on social media. *Information Processing & Management*, 57 (3): 102087.

Mozafari, J., Nematbakhsh, M. and Fatemi, A. 2019. Attention-based Pairwise Multi-Perspective Convolutional Neural Network for Answer Selection in Question Answering. *arXiv preprint arXiv:1909.01059*.

Mutanga, R., Tapiwa, Naicker, N. and Olugbara, O. 2020. Hate Speech detection using Transformer Methods. *International Journal of Advanced Computer Science and Applications*, 11 (9).

Nair, V. and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In: Proceedings of *ICML*.

Nasir, A. A., Mashor, M. Y. and Hassan, R. 2013. Classification of acute leukaemia cells using multilayer perceptron and simplified fuzzy ARTMAP neural networks. *The International Arab Journal of Information Technology*, 10 (4).

Nowak, J., Taspinar, A. and Scherer, R. 2017. LSTM recurrent neural networks for short text and sentiment classification. In: Proceedings of *International Conference on Artificial Intelligence and Soft Computing*. Springer, 553-562.

Ordóñez, F. J. and Roggen, D. 2016. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16 (1): 115.

Pálmadóttir, J. and Kalenikova, I. 2018. Hate speech an overview and recommendations for combating it. *Icelandic Human Rights Centre*: 1-27.

Park, J. H. and Fung, P. 2017. One-step and two-step classification for abusive language detection on twitter. *arXiv preprint arXiv:1706.01206*.

Pascanu, R., Mikolov, T. and Bengio, Y. 2012. Understanding the exploding gradient problem. *CoRR, abs/1211.5063*, 2: 417.

Patihullah, J. and Winarko, E. 2019. Hate speech detection for indonesia tweets using word embedding and gated recurrent unit. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, 13 (1): 43-52.

Pereira-Kohatsu, J. C., Quijano-Sánchez, L., Liberatore, F. and Camacho-Collados, M. 2019. Detecting and Monitoring Hate Speech in Twitter. *Sensors*, 19 (21): 4654.

Pitsilis, G. K., Ramampiaro, H. and Langseth, H. 2018. Effective hate-speech detection in Twitter data using recurrent neural networks. *Applied Intelligence*, 48 (12): 4730-4742.

Poomka, P., Pongsena, W., Kerdprasop, N. and Kerdprasop, K. 2019. SMS Spam Detection Based on Long Short-Term Memory and Gated Recurrent Unit. *International Journal of Future Computer and Communication*, 8 (1).

Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., Shyu, M.-L., Chen, S.-C. and Iyengar, S. 2018. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51 (5): 1-36.

Powers, D. M. 2020. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*.

Putri, T., Sriadhi, S., Sari, R., Rahmadani, R. and Hutahaean, H. 2020. A comparison of classification algorithms for hate speech detection. In: Proceedings of *IOP Conference Series: Materials Science and Engineering*. IOP Publishing, 032006.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1 (8): 9.

Reimers, N. and Gurevych, I. 2017. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*.

Ren, H., Wan, J. and Ren, Y. 2018. Emotion detection in cross-lingual text based on bidirectional LSTM. In: Proceedings of *International Conference on Security with Intelligent Computing and Big-data Services*. Springer, 838-845.

Ren, Y. and Ji, D. 2017. Neural networks for deceptive opinion spam detection: An empirical study. *Information Sciences*, 385: 213-224.

Ross, B., Rist, M., Carbonell, G., Cabrera, B., Kurowsky, N. and Wojatzki, M. 2017. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *arXiv preprint arXiv:1701.08118*.

Roy, P. K., Singh, J. P. and Banerjee, S. 2020. Deep learning to filter SMS Spam. *Future Generation Computer Systems*, 102: 524-533.

Ruder, S. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Ruder, S., Ghaffari, P. and Breslin, J. G. 2016. A hierarchical model of reviews for aspect-based sentiment analysis. *arXiv preprint arXiv:1609.02745*.

Sajjad, H., Dalvi, F., Durrani, N. and Nakov, P. 2020. Poor Man's BERT: Smaller and Faster Transformer Models. *arXiv preprint arXiv:2004.03844*.

Sak, H., Senior, A. and Beaufays, F. 2014. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*.

Saksesi, A. S., Nasrun, M. and Setianingsih, C. 2018. Analysis Text of Hate Speech Detection Using Recurrent Neural Network. In: Proceedings of *2018 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*. IEEE, 242-248.

Sanh, V., Debut, L., Chaumond, J. and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Santosh, T. and Aravind, K. 2019. Hate speech detection in hindi-english code-mixed social media text. In: Proceedings *of the ACM India Joint International Conference on Data Science and Management of Data*. 310-313.

Schabas, W. A. 2000. Hate speech in Rwanda: The road to genocide. *McGill LJ*, 46: 141.

Schmidt, A. and Wiegand, M. 2017. A survey on hate speech detection using natural language processing. In: Proceedings of *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*. 1-10.

Shen, T., Zhou, T., Long, G., Jiang, J., Pan, S. and Zhang, C. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *arXiv preprint arXiv:1709.04696*.

Silva, A. and Roman, N. 2020. Hate Speech Detection in Portuguese with Naïve Bayes, SVM, MLP and Logistic Regression. In: Proceedings of *Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional*. SBC, 1-12.

Singh, P. K. and Husain, M. S. 2014. Methodological study of opinion mining and sentiment analysis techniques. *International Journal on Soft Computing*, 5 (1): 11.

Sohangir, S., Wang, D., Pomeranets, A. and Khoshgoftaar, T. M. 2018. Big Data: Deep Learning for financial sentiment analysis. *Journal of Big Data*, 5 (1): 3.

Sokolova, M., Japkowicz, N. and Szpakowicz, S. 2006. Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation. In: Proceedings of *Australasian joint conference on artificial intelligence*. Springer, 1015-1021.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15 (1): 1929-1958.

Sukhbaatar, S., Weston, J. and Fergus, R. 2015. End-to-end memory networks. In: Proceedings of *Advances in neural information processing systems*. 2440-2448.

Sung, D.-K. and Jeong, Y.-S. 2018. Political Opinion Mining from Article Comments using Deep Learning. *한국컴퓨터정보학회논문지*, 23 (1): 9-15.

Tang, D., Qin, B. and Liu, T. 2015. Document modeling with gated recurrent neural network for sentiment classification. In: Proceedings of *Proceedings of the 2015 Conference On Empirical Methods In Natural Language Processing*. 1422-1432.

Ullmann, S. and Tomalin, M. 2020. Quarantining online hate speech: technical and ethical perspectives. *Ethics and Information Technology*, 22 (1): 69-80.

Umar, A., Sulaimon, A. B., Muhammad, B. A., Olawale, S. A., Department of Computer Science, F. U. o. T. M. N. and Department of Cyber Security Science, F. U. o. T. M. N. 2019. Comparative Study Of Various Machine Learning Algorithms For Tweet Classification. *i-manager's Journal on Computer Science*, 6 (4): 12.

Uysal, A. K. and Gunal, S. 2014. The impact of preprocessing on text classification. *Information Processing & Management*, 50 (1): 104-112.

Van Huynh, T., Nguyen, V. D., Van Nguyen, K., Nguyen, N. L.-T. and Nguyen, A. G.-T. 2019. Hate Speech Detection on Vietnamese Social Media Text using the Bi-GRU-LSTM-CNN Model. *arXiv preprint arXiv:1911.03644*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. 2017. Attention is all you need. In: Proceedings of *Advances in neural information processing systems*. 5998-6008.

Vinyals, O., Kaiser, Ł., Koo, T., Petrov, S., Sutskever, I. and Hinton, G. 2015. Grammar as a foreign language. In: Proceedings of *Advances in neural information processing systems*. 2773-2781.

Wadawadagi, R. and Pagi, V. 2020. Sentiment analysis with deep neural networks: comparative study and performance assessment. *Artificial Intelligence Review*.

Wang, B., Ding, Y., Liu, S. and Zhou, X. 2019. YNU_Wb at HASOC 2019: Ordered Neurons LSTM with Attention for Identifying Hate Speech and Offensive Language. In: Proceedings of *FIRE (Working Notes)*. 191-198.

Wang, X., Li, Y. and Xu, P. 2018. A hybrid BLSTM-C neural network proposed for chinese text classification. In: Proceedings of *2018 Sixth International Conference on Advanced Cloud and Big Data (CBD)*. IEEE, 311-315.

Wankhede, S. B. 2014. Analytical study of neural network techniques: SOM, MLP and classifier-a survey. *IOSR Journal of Computer Engineering*, 16 (3): 86-92.

Warner, W. and Hirschberg, J. 2012. Detecting hate speech on the world wide web. In: Proceedings of *Proceedings of the second workshop on language in social media*. 19-26.

Waseem, Z. 2016. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In: Proceedings of *Proceedings of the first workshop on NLP and computational social science*. 138-142.

Waseem, Z. and Hovy, D. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In: Proceedings of *Proceedings of the NAACL student research workshop*. 88-93.

Whillock, R. K. and Slayden, D. 1995. *Hate speech*. 2455 Teller Road, Thousand Oaks, CA 91320: SAGE Publications.

Yang, W., Zuo, W. and Cui, B. 2019. Detecting malicious urls via a keyword-based convolutional gated-recurrent-unit neural network. *IEEE Access*, 7: 29891-29900.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R. and Le, Q. V. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In: Proceedings of *Advances in neural information processing systems*. 5753-5763.

Yasseri, T. and Vidgen, B. 2019. Detecting weak and strong Islamophobic hate speech on social media. *Journal of Information Technology and Politics*, 17 (1).

Yogarajan, V., Gouk, H., Smith, T., Mayo, M. and Pfahringer, B. 2020. Comparing High Dimensional Word Embeddings Trained on Medical Text to Bag-of-Words for Predicting Medical Codes. In: Proceedings of *Asian Conference on Intelligent Information and Database Systems*. Springer, 97-108.

Young, T., Hazarika, D., Poria, S. and Cambria, E. 2018. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13 (3): 55-75.

Zabidi, A., Khuan, L. Y., Mansor, W., Yassin, I. M. and Sahak, R. 2010. Classification of infant cries with asphyxia using multilayer perceptron neural network. In: Proceedings of

2010 Second International Conference on Computer Engineering and Applications. IEEE, 204-208.

Zhang, X., Chen, F. and Huang, R. 2018. A combination of RNN and CNN for attention-based relation classification. *Procedia computer science*, 131: 911-917.

Zhang, X., Zhao, J. and LeCun, Y. 2015. Character-level convolutional networks for text classification. In: Proceedings of *Advances in neural information processing systems*. 649-657.

Zhang, Z. and Luo, L. 2019. Hate speech detection: A solved problem? The challenging case of long tail on twitter. *Semantic Web*, 10 (5): 925-945.

Zhang, Z., Robinson, D. and Tepper, J. 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In: Proceedings of *European semantic web conference*. Springer, 745-760.

Zulqarnain, M., Ghazali, R., Hassim, Y. M. M. and Rehan, M. 2020. Text classification based on gated recurrent unit combines with support vector machine. *International Journal of Electrical & Computer Engineering (2088-8708)*, 10.

ANNEXURE A: COVER PAGE OF TURN IT IN REPORT

Final Dissertation

ORIGINALITY REPORT

15%	9%	8%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Raymond T Mutanga, Nalindren Naicker, Oludayo O. "Hate Speech Detection in Twitter using Transformer Methods", International Journal of Advanced Computer Science and Applications, 2020 Publication	1%
2	Submitted to Postgraduate Institute of Medicine Student Paper	1%
3	Submitted to Federal University of Technology Student Paper	1%
4	doctorpenguin.com Internet Source	1%
5	ukzn-dspace.ukzn.ac.za Internet Source	1%
6	www.aclweb.org Internet Source	1%
7	"Mining Intelligence and Knowledge Exploration", Springer Science and Business Media LLC, 2020	<1%

ANNEXURE B: LANGUAGE PROFICIENCY CERTIFICATE

THE WRITING STUDIO
Writing and Editing Practice

Certificate 1421

TO WHOM IT MAY CONCERN

5 April 2021

This dissertation, entitled **A Comparative Study of Deep Learning Algorithms for Hate Speech Detection on Twitter** by Raymond Mutanga, has been edited and reviewed to ensure technically accurate and contextually appropriate use of language for research at this level of study.

Yours sincerely

CM ISRAEL, BA Hons (UDW) MA (UND) MA (US) PhD(UNH)
LANGUAGE EDITOR AND WRITING CONSULTANT
Connieisrael90@gmail.com Mobile 082 4988166