Racism Detection by Analyzing Opinions Through Sentiment Analysis of Tweets from Different Immigration Crisis

Heba Lubbad

MSc in Data Science
The University of Bath
2021-2022

Racism Detection by Analyzing Opinions Through Sentiment Analysis of Tweets from Different Immigration Crisis

Submitted by: Heba Lubbad

Copyright

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Master of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Abstract

As Twitter continues to dominate in the socio-political landscape, misuse of the platform and emerging racism and discrimination is seen in various forms. As seen through the recent racism and discrimination witnessed during the Russian invasion of Ukraine, tweets containing racist opinions and remarks can thrive based on origin, colour, religion, and culture and have been regarded to induce social instability and violence. Following comments on racism towards discrimination against Ukraine immigrants of Arab or Black descent and dehumanizing comments comparing Afghanistan and Ukraine immigrants, outrage and social instability resulted. Therefore, as demonstrated by the heavily funded research on hate speech detection, detecting and monitoring discrimination and racist opinions and remarks on Twitter is essential. This research implements and compares various models: SVM, Bidirectional Encoder Representations from Transformers (Bert) BertBase with Multi-Layer Perceptron (MLP) and convolutional neural networks (CNN) models, and a Gated Convolutional Recurrent- Neural Networks (GCR-NN). Several experiments are conducted to explore the best architecture and parameters concluding the BERT's superior performance in detecting racism. The GCR-NN also shows excellent performance when applied to external datasets, higher generalizability, and easier finetuning. Findings also indicate that the public perceives Afghanistan immigrants more negatively and differently from Ukrainian immigrants.

Contents

1 Introduction **1**
- 1.1 Background and Description . . . 1
- 1.2 Research Questions and Objectives . . . 2

2 Literature and Technology Survey **3**

3 Methodology **7**
- 3.1 Short Section Title . . . 7
  - 3.1.1 Primary Dataset . . . 7
  - 3.1.2 Different Immigration Crises Datasets . . . 7
- 3.2 Dataset Manipulation . . . 9
- 3.3 Pre-Processing . . . 9
- 3.4 Sentiment Annotation . . . 10
- 3.5 Machine Learning Models . . . 11
  - 3.5.1 SVM . . . 11
  - 3.5.2 Bert Base . . . 11
  - 3.5.3 Bert + MLP . . . 11
  - 3.5.4 Bert + CNN model . . . 11
  - 3.5.5 Bert + CNN combined output layers . . . 11
  - 3.5.6 Stacked Ensemble GCR-NN Model . . . 12

4 Implementation and Testing **13**
- 4.1 SVM . . . 13
- 4.2 Bert . . . 13
- 4.3 Bert+MLP . . . 13
- 4.4 Bert+CNN . . . 14
  - 4.4.1 Model Architecture . . . 14
- 4.5 BERT- Data Preparation . . . 15
  - 4.5.1 Splitting Data and Creating Tensors . . . 16
  - 4.5.2 preprocessor . . . 16
- 4.6 GCR-NN . . . 16
- 4.7 Loss and Optimizer . . . 17
- 4.8 Batch Size . . . 17
- 4.9 Epochs . . . 17
- 4.10 Dropout Layer . . . 18
- 4.11 Metrics . . . 18

5 Results **20**

5.1 Baseline Results 20
5.1.1 SVM 20
5.1.2 Bertbase MLP model 20
5.2 Bertbase CNN Model Experimentation and Results 22
5.2.1 Comparison of the 3 Key Layers 23
5.3 GCR-NN Model Experimentation and Results 24
5.4 Models Performance and Generalizability on Ukraine and Afghanistan Datasets 25
5.4.1 Further Analysis 28

6 Conclusions **30**
6.0.1 Answering the Research Questions 30
6.0.2 Limitations and Further Research 31

Bibliography **32**

A Design Diagrams **35**

List of Figures

2.1 General structure of hate-speech detection model 4
2.2 Strucutre of proposed GRNN 6

3.1 Dataset distribution where label 0 represents racist tweets 8
3.2 Sample from Afghanistan Dataset 8
3.3 Results of using Preprocessing Steps 10
3.4 Combined Outputs Architecture 12

4.1 Bert Model Preprocessing 14
4.2 Relu Activation function 15
4.3 Sigmoid and Softmax Activation function 15

5.1 Corresponding Confusion matrix 21
5.2 BertBase-MLP training plots 21
5.3 Hyperparameter Tuning Results 22
5.4 Example of Results Before and After Tuning 23
5.5 BertCNN results with different layers 24
5.6 Results of GCR-NN hyperparameter tuning 25
5.7 Results on Ukraine and Afghanistan Corpa 26
5.8 GCRNN Afghan Vs Ukraine 27
5.9 BertBase Afghan Vs Ukraine 27
5.10 Tweets from Afghanistan Dataset containing racist sentiment 28
5.11 WordCloud of Correctly Predicted Afghanistan tweets by GCR-NN 29
5.12 WordCloud of Correctly Predicted Ukraine tweets by GCR-NN 29

A.1 Architecture of GCR-NN 35
A.2 Architecture of BERTCNN pooled output 36

List of Tables

5.1 SVM Classification Scores 20
5.2 GCR-NN Classification Scores 25
5.3 Final Model Scores 25

Acknowledgements

This study would not have been complete without the supervision of Dr. Guy McCusker and Dr. Alex Taylor who have supported and guided me throughout and ensured a smooth and efficient journey. The supervisors went out of their way to check in on the progress of this work and to ensure the well-being of their graduates. Moreover, the guidance of the teaching assistants throughout the postgraduate studies such as Dr. Jordan Taylor has shaped the results of this work and it is important to acknowledge their contribution.

Chapter 1

Introduction

1.1 Background and Description

Twitter has become a dominating socio-political platform allowing live news updates that outpace news outlets by allowing witnesses and participants to share the events on the platform quickly. Twitter also allows people worldwide to voice their opinions on current crises. Amongst these voices, misuse of the platform has seen emerging racism and discrimination in various forms. As a matter of fact, two of the most used hashtags around social causes in Twitter history focus on race and criminal justice: #Ferguson and #BlackLivesMatter (Anderson,2016). This ranges from memes to openly racist remarks that incite social instability and violence. The recent Ukraine immigration crisis shows that Twitter users have used the platform to voice their opinions. Unfortunately, it has also seen various racist tweets from individuals to media outlets. Online Social Networks (OSNs) have noticeably played a significant role in such crises and have proven to be a reliable and fruitful source of data for analysis and studying political discourse. Social media has provided a new platform for hate speech thus information dissemination should be monitored, and racist remarks should be detected and blocked in a timely manner. Sentiment analysis provides a powerful tool to mine such data to analyze emotions and opinions. Moreover, the different type of discussions on racism illustrates geographical variability in racial attitudes and sentiment. Therefore, analyzing how people, events, and circumstances are represented provides insights into user communication and problems surrounding racism can be investigated. Automatic hate speech detection using machine learning algorithms is still new and requires extensive research efforts from both industry and Academia. It has been estimated that on a yearly basis there are hundreds of millions of euros are being invested to combat hate speech (Zhang et al. 2020). While social media platforms have implemented strict integrity and hateful contact policies, the problem remains complex because it involves several layers of complexity: computational complexity, given the large volume of content, as well as the subtleties and cultural aspects of each language, the problem of low resource languages and natural language's intrinsic ambiguity (Goutsos et al.,2021). This research aims to perform sentiment analysis on racist tweets by implementing various machine learning models to detect racism in tweets. The research will perform predictions on different immigration crises using tweets that contain hateful and racist speech aimed at refugees and migrants. Therefore, it will try to compare and improve on existing models throughout this paper and draw insights on sentiments and opinions shown in the Ukraine and Afghanistan crises.

1.2 Research Questions and Objectives

Throughout this study, the following questions will be explored and answered using the study's findings:

Research Question 1: Has the Bert model improved existing techniques on hate speech detection, specifically racism and to what extent?

Research Question 2: Which model architecture works best for racism classification, and how generalizable are they?

Research Question 3: Is more discrimination towards non-European immigrants observed through the Afghanistan and Ukraine immigration crises?

Therefore, the main objectives of this research are those shown below:

- Obtain a labelled twitter dataset on racism that will be publically available to encourage further research.
- Apply different state-of-the-art deep learning models and determine which of them has the highest accuracy and best classification scores.
- Obtain twitter datasets that are representative of the Ukraine and Afghanistan immigration crises that contain immigration keywords and label them as racist or non-racist using sentiment analysis tools.
- Use the best performing models to confirm or reject the hypothesis and the public opinion that negative and more racist sentiment is used when describing non-European immigration.
- Reflect on the proposed architectures' performance, generalizability, and limitations.

Chapter 2

Literature and Technology Survey

As more users join social media, the spread of hate speech has increased, thus pushing for urgent intervention from governments, enterprises, and academia. Therefore, there has been a rise in emerging research and publications on hate speech detection in different issues and languages. This paper will focus mainly on hate speech detection efforts and publications of anti-immigration speech.

A. General Hate Speech Detection

Parihar et al. (2021) provide a discussion on various relevant research in the field of hate speech detection. The paper generalises the structure of hate speech (Fig 3.) and argues that most publications utilise social media-based datasets with various hate speech labelling. For example, Malmalsi et al. (2017) annotated the dataset into three different labels: hate, offensive, and OK. Meanwhile, Cao et al. (2020) proposed a DeepHate architecture that classified tweets as either inappropriate or normal, achieving precision as high as 92.48%. The next step in building the hate speech detection model is pre-processing and feature extraction. Zhang, Robinson, and Tepper (2018) categorise feature extraction methods into classical and deep learning methods. The classical methods are manual feature engineering often used by machine learning algorithms such as Naive Bayes, Logistic Regression, and SVMs. The most commonly used and shown to be highly effective classical method is the surface features such as "bag of words, word and character n-grams" (p.747). According to Rustam et al.(2022), BOW was described as the most convenient and adaptable feature extraction technique. Moreover, a recent study by Lee et al. (2022) investigates which feature extraction technique works best with machine learning models. On average, their results have shown that BOW yields higher accuracy and lower miss classification rates. N-grams are defined as a sequence of consecutive words. Waseem and Hovy (2016) provide findings that n-grams are an effective option for hate speech detection by analysing the impact of using several extra-linguistic features in conjunction with character n-grams for hate speech detection. Another common method uses sentiment analysis to detect the polarity expressed in a tweet. For example, Jiang and Suzuki (2019) used sentiment analysis to detect hate speech from tweets.

Moreover, Gitari et al. (2015) present a lexicon-based approach for hate speech detection where they use subjectivity and semantic features to create a lexicon used by a hate speech detection classifier. However, the proposed work of Cao et al. (2020) has shown

that using multifaceted representations of the text such as "word embeddings, sentiments, and topical information" outperforms methods relying on single textual features. The next essential step in building a hate speech detection model is to choose the appropriate classification algorithm. Waseem and Hovy's work (2016) served as a foundation for several other studies that looked at predictive factors for hate speech detection. They provided access to their massive 16K twitter corpus, which is dedicated to hate speech detection in the English language. Initially, machine learning algorithms can be divided into supervised, semi-supervised, and unsupervised. Much of the research on hate speech detection has been supervised, requiring manual labelling of the dataset. While the most popular classification algorithm is Support Vector Machines (SVM), others were used such as Naïve Bayes, Random Forests, Logistic Regression, and Decision Trees. There is no evidence that one algorithm generally outperforms the rest as the choice of algorithm depends on the task and features extracted. For instance, Burnap and Williams (2014) tested different supervised algorithms for hate speech detection, and all the classifiers performed the same but varied accuracies when different features were tested.

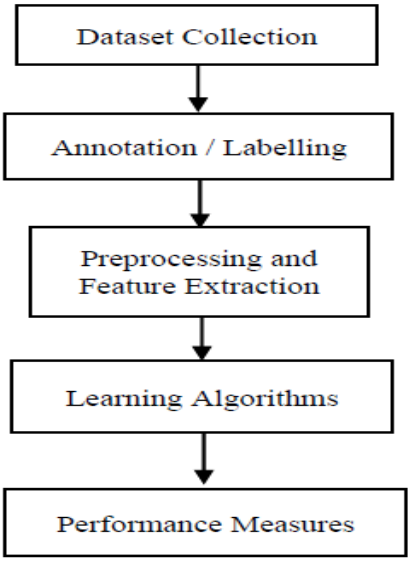


Figure 2.1: General structure of hate-speech detection model

Furthermore, Putri et al. (2020) conclude that Multinomial Naïve Bayes is the model for hate speech detection as it outperforms the others with an accuracy of 71%; while Chen et al. (2012) found that SVM is more accurate than Naïve Bayes at detecting offensiveness. As the study of HS detection has been a complex task, some research has attempted a different approach, such as transfer learning. For example, Rizoiu et al.(2019) have used transfer learning from an LSTM network coupled with ELMo embeddings and found that the model trained on multiple generic abusive language datasets will produce more robust predictions. Moreover, another approach done by Swamy et al. (2019) and Mozafari et al. (2019) is applying a Bidirectional Encoder Representations from Transformers (BERT) based transfer learning approach. The findings demonstrate the model's capacity to detect some biases in the collection or annotation of datasets. However, the biggest challenge in classifying abusive language across domains is dataset limitations and biases. Another gap remaining in hate speech detection research is the lack of research in languages aside from English. Nonetheless, Ranasinghe et al. (2020) used a social media dataset with multiple languages like English,

German and Hindi and proposed a BERT-based architecture. There has also been very little research done on hate speech detection in Arabic (Aljarah et al., 2021).

B. Deep Neural Network Methods More recently, hate speech detection has shifted towards exploiting deep neural network methods such as LSTMs, GRU, and CNNs combined with word embedding models such as ELMo and word2vec. Word embedding alleviated the data sparsity problem by introducing an additional semantic feature as it constructed distributed representations that introduce word dependency (Al-Hassan and Al-Dossari, 2019). One of the first attempts of Deep artificial Neural Networks in HS detection used paragraph2vec embeddings in a two-step classifier (Djuric et al., 2015). Moreover, according to Lilleberg et al. (2015), word2vec has piqued the curiosity of experts in the field and has performed well in detecting prejudice in social media posts. On the other hand, Pitsilis et al. (2018) employed an RNN model with word frequency vectorization instead of word embedding to build features for hate speech detection, breaking the barrier of language reliance in the word embedding technique. For hate speech identification, their results exceeded existing state-of-the-art deep learning algorithms.

   The use of deep learning has become more popular due to the advancement in graphic processing units and their ability to detect and extract features automatically. Deep learning methods have generally outperformed classical machine learning methods. For example, Zhang and Luo (2018) compared a deep neural network model to several classical machine learning methods and found by comparing the f1 scores that the neural network performed the best. Moreover, Ali, El Hammid, and Youssif's (2019) research compared the results of different deep neural network methods to SVM and Naïve Bayes for a classification sentiment analysis task and found that DNN methods outperformed greatly. However, this does not offer a golden rule where DNN methods always outperform classical methods as it greatly depends on the complexity of the hidden layers and the correct choice of algorithm and feature representation technique. Al-Smadi et al. (2018) work to support this statement as it compares the performance of RNNs and SVMs and shows that for a specific set of features the SVM greatly outperforms the RNN. Based on the literature, CNNs are well-known for acting as 'feature extractors' whereas RNN is well-suited to modelling ordered sequence learning issues. In the context of hate speech categorization, RNN learns word or character dependencies in tweets, whereas CNN extracts word or character combinations such as phrases and n-grams (Zhang Et al., 2018).

   The most powerful combination that has been recently investigated of CNN+RNN is in theory established to be a powerful structure for capturing order information between CNN features. This combination has also performed well in practice on different tasks such as activity recognition. Empirical research has also shown that building a DNN with (GRU) instead of LSTM achieved comparably better results. This is due to their simpler structure thus making them easier to train and generalize on smaller datasets. Due to improvement in deep learning's performance, researchers have used a stacked ensemble deep learning model which is created by merging gated recurrent units (GRU), convolutional neural networks (CNN), and recurrent neural networks RNN. This is referred to as Gated Convolutional Recurrent-Neural Networks (GCR-NN). CNN extracts crucial characteristics for RNNs to produce good predictions, whereas GRU is at the top of the GCR-NN model for extracting significant features from raw text. For example, Alshalan and H. Al-Khalifa (2020) use several deep learning algorithms to explore the

issue of hate speech on Saudi Twitter. BERT, CNN, GRU, and the ensemble of CNN and GRU (CNN+GRU) are used in a series of tests on two datasets. The CNN model scores an F1 score of 0.79. Moreover, Lee et al. (2022) purpose a GCR-NN structure (Figure 2.2.) that was able to detect 97% of the tweets containing racist comments. In relation to research focused on racism and refugees, Zhang et al. (2018) presented a deep neural network that integrated convolutional and gated recurrent networks targeting hate speech detection that achieved very promising results.

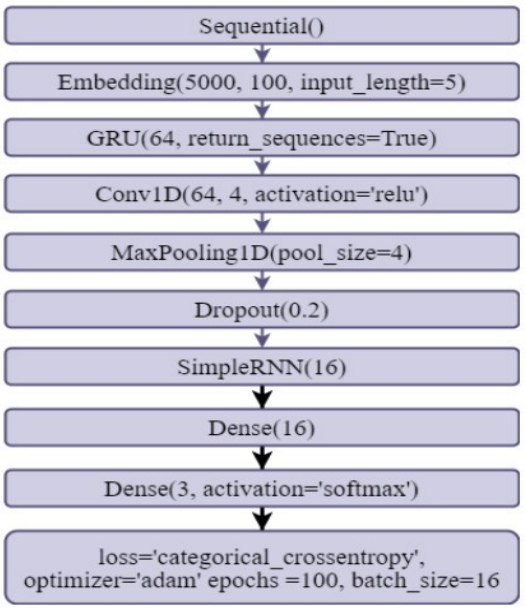


Figure 2.2: Strucutre of proposed GRNN

Chapter 3

Methodology

3.1 Datasets Description

3.1.1 Primary Dataset

This study refers to multiple datasets referenced by various research papers. The final primary racism dataset that will be used in the majority of this research to train and test different ML architectures is a combination of 3 different existing models. This research will discuss each dataset that was used to create the primary dataset and the data manipulation involved. The first dataset is a hate speech dataset which was based on the research paper "Automated Hate Speech Detection, and the Problem of Offensive Language" by Davidson et al. (2017) compiled a total of 85.4 million tweets from 33,458 Twitter users that contained hate speech words and phrases and randomly chose a corpus of 25k tweets for manual labelling by the CrowdFlower workers. The tweets were labelled either hateful, offensive, or neither. The Second dataset was provided by Analytics Vidhya, where tweets considered racist were labelled as 1, and non-racist tweets were labelled as 0. Another dataset annotated for racist tweets was taken from the research paper "Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter" by Zeerak and Dirk. However, they only publicly provided the tweets' IDs and their Labels. The original tweets were collected using python's Tweepy library and Twitter's API. The elevated access was obtained to increase the limit on the number of tweets scraped and allow tweet lookup by ID. 1,972 racist tweets were extracted from the provided dataset, and 10,000 random non-hateful tweets were chosen from this dataset. After some data manipulation that will be discussed later, the final dataset acquired is a balanced dataset containing 68,721 tweets, where 35,859 represent racist tweets and 32,862 represent non-hateful tweets, as shown in 3.1

3.1.2 Different Immigration Crises Datasets

Afghanistan immigration crisis Dataset

Due to the Twitter API's limitations, the Afghanistan dataset is collected from Twitter using SNScraper instead of Twitter's API. Twitter's API search index has a 7-day limit and will not be applicable when scraping historical data. Tweets concerning the Taliban takeover in Afghanistan were collected over the period between August 10-21. This specific period was chosen in previous research (Lee et al.,2022), representing the peak of the Taliban Takeover

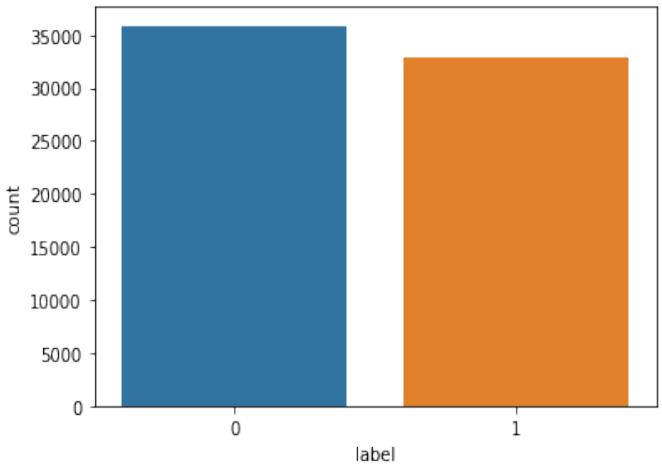


Figure 3.1: Dataset distribution where label 0 represents racist tweets

Date	Text
2021-09-04 23:52:31+00:00	@Tequila06209973 And they bring these immigrants in \newline from Afghanistan and from across the southern border and parade them \newline around all these military bases https://t.co/dJ6Zb07OEP\\ 2021-09-04 23:50:10+00:00
2021-09-04 23:50:10+00:00	@Brexit011 @sarahisitme @JonnieIlly @EU_NO_MORE It is all very well, but Boris is currently spending billions on Afghanistan and illegal immigrants. I do not agree with raising taxes and cutting services/pensions when we are paying over the odds flooding the country. Quite honestly the p*ss is being taken out of us
2021-09-04 23:45:03+00:00	If you haven't yet, please see our resource page on the @USCCBJFI website for the ongoing Afghanistan Resettlement Operation. Take a look today to find out how you can help, see our statements on the operation and more. Please visit justiceforimmigrants.org/afghanistan today! https://t.co/IhthVpMkrF

Figure 3.2: Sample from Afghanistan Dataset

events. Specific Keywords were used to filter the scraping process, such as "#saveafganistan", "Afghanistan immigrants", "#refugeesnotwelcome", etc. A total of 10,000 tweets were collected and provided by this research as a novel dataset, a sample is displayed in Figure 3.2.

Ukraine immigration Crisis Dataset

The last dataset used in this study refers to the most recent crisis in Ukraine after the Russian invasion. The dataset is called "Daily datasets of tweets about the ongoing Ukraine Russia Conflict" can be found on Kaggle and is updated daily containing around 51.01M tweets (2022). Tweets containing racist keywords will be extracted from this dataset starting from February 24th till the end of March. According to Statista, after Russia's invasion on February 24th, around 8 million Russian refugees have left their homes just by May 3rd. Therefore, the period chosen to be analyzed tries to represent the peak of the Ukraine immigration crisis.

3.2 Dataset Manipulation

The dataset used to train and validate the models for this research consisted of 3 separate datasets detailed above. The dataset by Davidson et al. (2017) contained hate speech and offensive labels where hate speech labels were identified based on tweets containing racist or sexist language, and the offensive labels generally represented offensive tweets. Therefore, the tweets labelled offensive were removed from the corpus. Unnecessary columns such as language, id, and location were dropped as they don't serve the purpose of this study. The tweets labelled based on racist language and sentiment were combined from the three datasets and labelled as 1, and the non-racist tweets were labelled as 0. The dataset was checked for null values and retweets.

3.3 Pre-Processing

The finalized dataset largely requires pre-processing as Twitter data contains many words and characters that can hinder the performance of a machine learning model.

- Tokenization: This refers to the process of splitting the raw text into smaller chunks allowing the algorithm to understand each word separately as well as within the tweet.
- Punctuation and number removal: Punctuation is essential for creating meaning within a sentence. However, for a machine learning model, this will distort the learning process by creating unnecessary features that will increase the feature vector size. Additionally, for the hate speech detection task numbers create an additional meaningless feature. Therefore, both punctuation and numbers have been removed from the tweets.
- Lowercasing: ML algorithms are case-sensitive such as where "Racism" and "racism" are considered completely different. This would decrease efficiency and increase vector space size. Therefore, each character in the tweets in the corpus is converted to lowercase.
- Replacing URLs and user mentions: Tweets containing URLs and the @ symbol for user mentions do not contribute to the sentiment of the tweet thus they were parsed and replaced.
- Stemming and Lemmatization: This is an essential step in data processing for NLP research. Stemming further reduces the feature vector size by combining different forms of the same word into one. For example, the words "go", "going", and "gone" are reduced to the root "go". This will be performed using the Stemmer porter algorithm. Similarly, lemmatization reduces the feature vector space by reducing a word to its root word, but in this case, a word such as "better" would also be reduced to "good" as they contribute to the same meaning. Therefore, using stemming and lemmatization unifies and reduces the feature vector space while retaining the meaning and sentiment of the tweet.
- Stopword Removal: Stopwords such as "I, am, the" also don't contribute to the sentiment of a tweet and create distortion and are removed from the tweets.
- Noise removal: Additional special characters, hashtags, and emojis are also identified as noise and removed from the corpus.

Raw Text	Clean Text
Fu-Gee-La. Oh, so NOW we all love refugees!! I love seeing Ukrainian refugees embraced with open arms. It's a shame that same warm welcome isn't given to refugees who don't look like me. #ukraine #russia #refugees #syria #middleeast #africa #skittles #immigrants #sotu #wpmot https://t.co/k45UBx0ViT	love refuge love see ukrainian refuge embrac open arm shame warm welcom isnt given refuge dont look ukrain russia refuge syria middleeast africa skittl immigr sotu wpmot
Putin's War on #Ukraine amplifies the need for Scotland to become an independent country: 1/ We'll be able to set our own foreign policy, 2/ We'll be able to set our own law on immigration and welcome refugees, 3/ We'll be able to have a Nuclear Free Scotland. #SelfDetermination	putin ukrain amplifi need scotland becom independ countri abl foreign polici abl immigr welcom refuge abl nuclear free scotland selfdetermin
@Brexit011 @sarahisitme @JonnieIlly @EU_NO_MORE It is all very well, but Boris is currently spending billions on Afghanistan and illegal immigrants. I do not agree with raising taxes and cutting services/pensions when we are paying over the odds flooding the country. Quite honestly the p*ss is being taken out of us	well bori current spend billion afghanistan illeg immigr agre rais tax cut servic pension pay odd flood countri quit honestli taken

Figure 3.3: Results of using Preprocessing Steps

3.4 Sentiment Annotation

This section will discuss the tool used for labelling the Ukraine and Afghanistan Datasets. The scraped dataset are based on immigration keywords but are not labelled. Therefore, the dataset will be labelled based on the sentiment of the tweet concerning the topic. Strong negative sentiment will indicate anti-immigration or racist opinions and will be labelled as 1 indicating the presence of racism or discrimination. Meanwhile, positive or neutral sentiment will be labelled as 0 indicating harmless and non-hateful tweets.

VADER Sentiment Analysis Tool

VADER (Valence Aware Dictionary for sEntiment Reasoning) is a model used for text sentiment analysis and built explicitly for detecting sentiment in social media data. In the research paper by Lee et al.(2022) discussed in detail in the literature review, they have labelled their dataset using the TextBlob library. The collected tweets contained racist language, and a negative sentiment was labelled as a racist tweet, while a positive or neutral sentiment was labelled as non-racist which is also adopted in this research. However, research has shown that VADER performs better at detecting negative data than the TextBlob library, and VADER also measures sentiment intensity. VADER is built based on a human-centric approach as human ratings and crowdsourcing empirically validate it. The VADER model also considers several contextual elements, such as punctuation, capitalization, and modifiers, influencing the text's emotion. Therefore, the VADER tool was preferred over the TextBlob tool.

3.5 Machine Learning Models

As stated previously, the purpose of this research is to implement and compare various state-of-the-art models. This section will explain the choice and design of these models.

3.5.1 SVM

SVM is a well-known machine learning algorithm that has been a popular choice for binary classification models (Leet et al.). The primary purpose of the SVM is to best locate the optimal hyperplane based on a set of features to correctly classify input data points (Mujahid et al., 2021). The SVM model was chosen as a base model for performance comparison with the target models. As previously discussed SVM model has been used in various previous research and has been referred to several times as the best-performing machine learning model for hate speech detection. In order to best assess the performance of the models of interest, combining the BOW feature extraction and the SVM was the best option, as research has reported high performance for such a combination.

3.5.2 Bert Base

Devlin and his colleagues at Google initially introduced BERT in 2019, revolutionizing natural language processing research with a "simple yet empirically powerful" (Delvin et al.,2018) transformer model. Bert is a multi-layer bidirectional transformer trained on a large corpus of over 3000M tokens. Devlin et al.(2018) produced two models $Bert_{base}$ and $Bert_{large}$. The $Bert_{base}$ contains an encoder with 12 layers and accepts an input of a sequence of tokens with a maximum of 512 tokens. The final embedding extracted is a 768-dimensional vector. Since the Bert model's emergence, NLP research has generally used the BERT model for prediction tasks setting record performance compared to existing hate speech models. The Bert models will be compared to the GCR-NN model considering they are thus far the best-proposed architectures.

3.5.3 Bert + MLP

A multilayer perceptron is a fully connected neural network comprised of at least one layer of neurons. They are known to be very flexible due to their simplicity and to perform well for simple binary classification problems. The Bert and MLP architecture is set up in this study as a baseline model to test the improved advantage of a CNN model.

3.5.4 Bert + CNN model

A Convolutional Neural network is a type of neural network that is widely used in image processing and classification tasks. Due to their effectiveness in the field of computer vision, CNNs were also adopted in the field of NLP. CNNs have proven to be successful in extracting higher-level features from sentences as they are able to detect hateful vocabulary at a phrase-level.

3.5.5 Bert + CNN combined output layers

This model was inspired by the architecture suggested by Mozafari et al. (2019) in their paper "A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social

Media". As shown in 3.4, the proposed architecture uses a combination of the outputs of all the transformer encoders instead of the output of the latest transformer encoder (Mozfari et al.,2019). Moreover, a maximum value for each transformer encoder is generated by applying a max pooling on the convolution output.

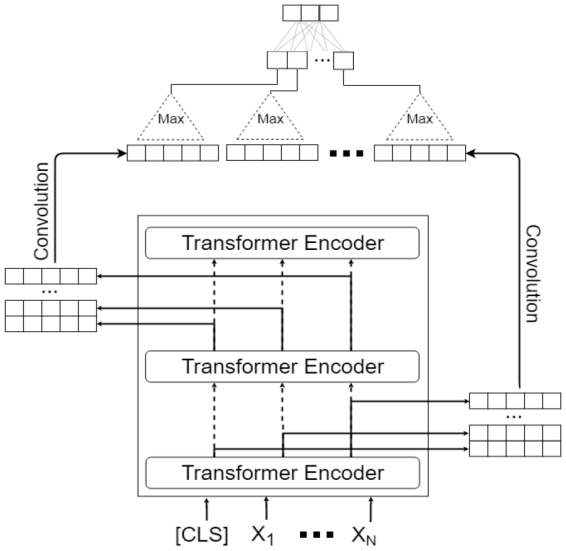


Figure 3.4: Combined Outputs Architecture

The Bertbase CNN implementation was adjusted from the proposed architecture and a different variation is proposed.

3.5.6 Stacked Ensemble GCR-NN Model

As previously discussed, the GCR-NN combines GRU, CNN, and RNN. The model is initialized by an embedding layer of an input size of 5000. The GRU model processes the embedding to extract features for the following layers. GRU architecture contains 64 units followed by a CNN layer with a kernel of size 4 x 4 with a Relu activation layer. This is followed by a max-pooling layer with a pooling size of 4, but for our purposes, this was changed to 2. The model uses a dropout layer of 0.2 to reduce model complexity and over-fitting. At the end of the model, an RNN of 16 units is attached at the end.

Chapter 4

Implementation and Testing

This section will explain the implementation behind each model, the important factors considered, and the evaluation metrics that were used in this study.

4.1 SVM

The SVM model was relatively simple and direct to set up. In order to construct a bag-of-words representation based on the tweets in the corpus, the CountVectorizer() function implemented in scikit-learn is used. The dataset is split using the scikit-learn function train_test_split() where the training Data contains 80% of the corpus and test data contains the remaining 20% as the parameter test_size has been set to 0.2. It is then used to fit the BOW model on the corpus and transform the tweets train and test datasets to vectorized data. The SVM model was used with a linear kernel and default settings and was fit on the vectorized training data.

4.2 Bert

As the literature review suggests, the appropriate BERT model for such a classification task would be the chosen "bert-en-uncased". This model only processes English language vocabulary; text inputs are normalized to lowercase within the model, and any accent markers have been removed. Input masking will be applied randomly to word pieces. Moreover, the BERT model will require the tweets to be given as a single sequence. As shown in figure 4.1, the model, the sequence is always tagged by a special classification token ([CLS]), and the ending of a sequence is identified by a special token ([SEP]). The input embedding is created by summing token, segment, and position embeddings. The last hidden state of BERT corresponding to this token (h[CLS]) will be used for classification tasks.

4.3 Bert+MLP

The Bertbase MLP implementation was inspired by the Kaggle implementation by Giovani Machado with a few variations.

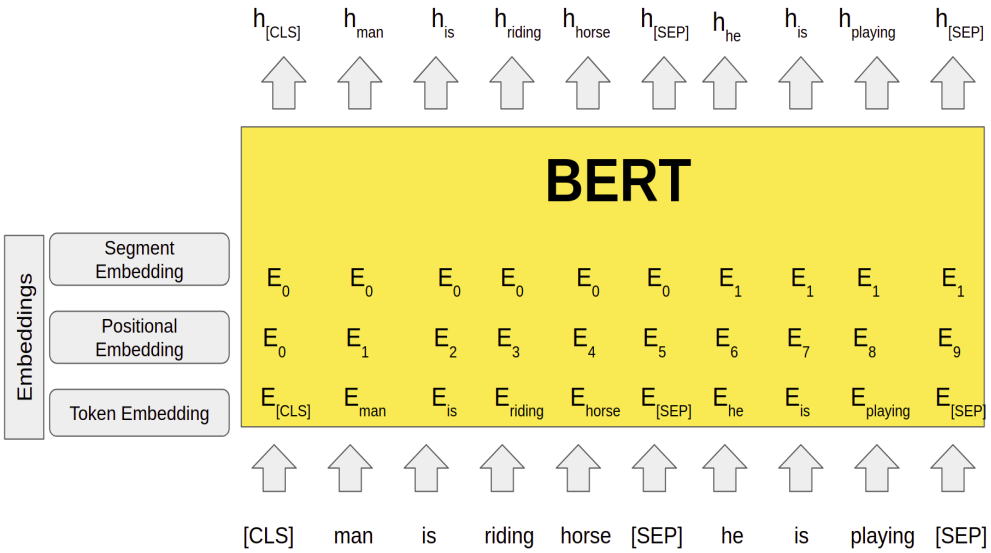


Figure 4.1: Bert Model Preprocessing

4.4 Bert+CNN

When fine-tuning a BERT model to stack a CNN, understanding the outputs of the model is important. There are 4 notable keys that can be used in the fine-tuning process:pooled_output, sequence_output, and encoder_outputs.

- pooled_output represents each input sequence as a whole. The shape is [batch_size, H]. It could be thought of as an embedding of the tweet.
- sequence_output that represents each input token in the context. The shape is [batch_size, seq_length, H]. It can be thought of as a contextual embedding for every token in the tweet.
- encoder_outputs are the intermediate activations of the L Transformer blocks. outputs["encoder_outputs"][i] is a Tensor of shape [batch_size, seq_length, 1024] with the outputs of the i-th Transformer block, for 0 <= i < L. The last value of the list is equal to sequence_output.
- the default output which is equal to the pooled_output

Different variations of the outputs were tested to check for the best performance. The structure of the BERT+CNN model is largely based o the TensorFlow tutorial: *Classify text with BERT* and a Kaggle notebook titled *Hate Speech - BERT+CNN and BERT+MLP in Tensorflow.* Extracting the sequence and pooled output was straightforward. However, the more complex implementation was the one following the work of Safaya et al. (2020) where the output of the last four hidden layers of the encoder outputs was concatenated using the keras.layers.concatenate class to get vector representations of size 4x128x512 as shown previously in Figure 3.1.

4.4.1 Model Architecture

Initially, the embeddings were directly passed to a Keras Conv1D layer with a 32 filter size, a window of size 2 and a hidden size of BERT which is 768 in BERTbase. The model is followed

by a relu activation which is a piece-wise linear function that will output the input directly if it is greater than 0, otherwise, it will output zero shown in figure 4.2. A GlobalMaxPool1D() layer is then applied for maximum value output. The original final dense layer was passed with 512 units but this resulted in massive overfitting which is why this model was passed 128 units instead. A sigmoid function would typically be suggested over the softmax function for binary classification but in this study the softmax was recommended in both notebooks (Figure 4.3).

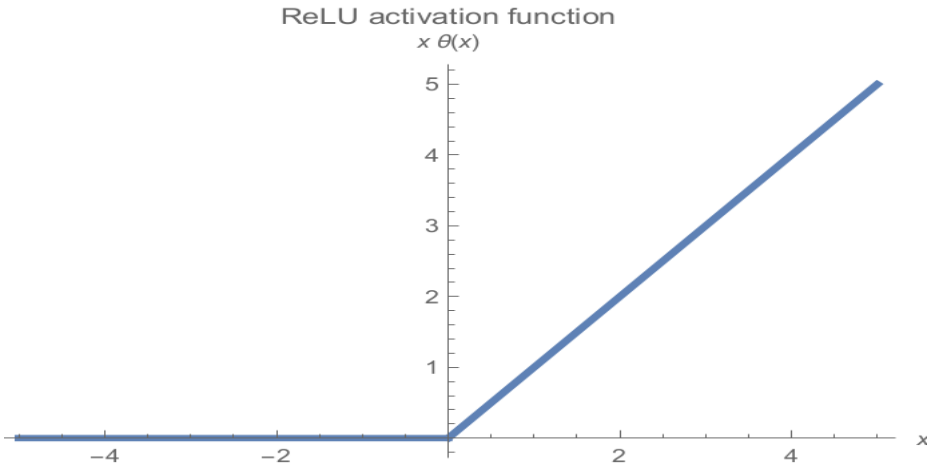
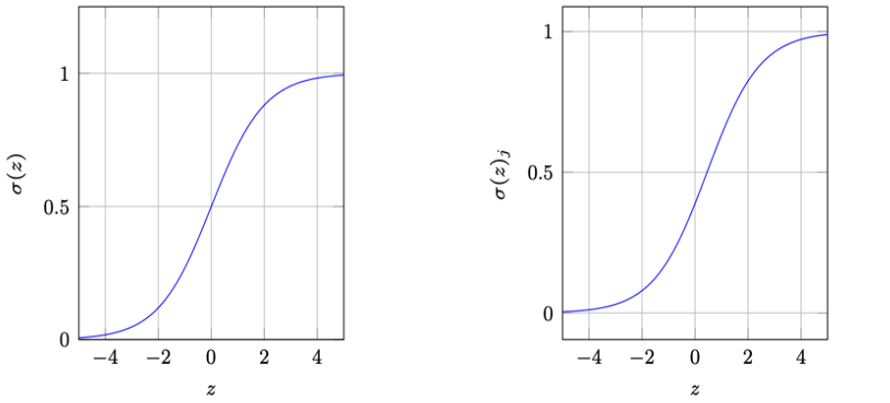


Figure 4.2: Relu Activation function



(a) Sigmoid activation function. (b) Softmax activation function.

Figure 4.3: Sigmoid and Softmax Activation function

4.5 BERT- Data Preparation

As previously defined, the tokenization step is important and will be done using the BERT Tokenizer. The Bert-base uncased transformer contains a vocabulary of 30,522 words. A unique feature of Bert tokenization is that it deals with unfamiliar words using a BPE-based WordPiece tokenisation technique. This approach process the unfamiliar word by breaking it down into sub-words that the tokenizer is familiar with, which can represent the word.

4.5.1 Splitting Data and Creating Tensors

The dataset was split into train, validation, and test dataset with a split of 0.6,0.2, and 0.2, respectively. The training dataset will fit and train the data, while the test data will provide an unbiased evaluation of the model's performance. The validation dataset will be used for hyperparameter tuning and to assess the model's performance. The data cleaning function was applied to each dataset separately. Typically during Keras and Tensorflow model training, the ImageDataGenerator object is used. However, the tf.data module is 38x faster and provides complex and highly efficient data processing pipelines (Rosebrock,2021). Using from_tensor_slices, a dataset is created with a separate element for each row of the input tensor. The dataset is then shuffled, and a batch size of 32 is given. The inputs of the model are then of the form:
[CLS] Sentence A [SEP] Sentence B [SEP]

4.5.2 preprocessor

Each Bert model is mapped to a corresponding encoder and preprocessor. There is a preprocessing model for each BERT encoder. Using TensorFlow operators from the TF.text package, the Bert encoder converts raw text from numeric input tensors. An advantage to the Bert model is that it does not follow and require traditional preprocessing techniques as they are incorporated into the model. Each corresponding TF Hub preprocessing model requires little to no extra configuration as it already comes preconfigured with representative vocabulary and text normalization logic. This process reduces code and preprocessing effort, "accelerates the computation, is less error-prone, and enables the serialization of the full text-to-outputs model" (). Thus, such a feature and its advantages make BERT more effortless to serve in production.

```
bert_model_name = "bert_en_uncased_L-12_H-768_A-12"
preprocessor =
    hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3")
encoder = hub.KerasLayer("https://tfhub.dev/tensorflow/
bert_en_uncased_L-12_H-768_A_12/4",trainable=True)
bert_preprocess_model = hub.KerasLayer(preprocessor)
```

Calling preprocess() transforms raw text inputs into a fixed-length input sequence for the BERT encoder. It consists of a tensor input_word_ids with numerical ids for each tokenized input, including start, end and padding tokens, plus two auxiliary tensors: an input_mask (that tells non-padding from padding tokens) and input_type_ids for each token.

4.6 GCR-NN

The same optimizer and loss function as the Bert model but required a different tokenizer. This research uses TensorFlow and Keras for modelling. Keras has a pre-processing module for text which is the tf.keras.pre-processing.text.Tokenizer() class and is initialized as shown below. One of the methods in this class is the fit_on_texts() method. Using a list of texts that is passed to this function, the internal vocabulary is updated accordingly. The vocabulary index is created based on word frequency that is stored in a dictionary where every word gets a unique integer value. This method will be used to fit the training corpus and the following method will be used to the text into a sequence of integers. The texts_to_sequences() method will

use the index dictionary to replace the text value with its integer value. The inputs need to be passed with the same size which is achieved through the pad_sequences() function. The function returns a sequence of a constant size, which can be passed as a parameter.

```
max_words = 5000
max_len = 5
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(x_train)
sequences = tokenizer.texts_to_sequences(x_train)
sequences_matrix_xtrain = pad_sequences(sequences,maxlen=max_len)

sequences_xval = tokenizer.texts_to_sequences(x_val)
sequences_matrix_xval = pad_sequences(sequences_xval,maxlen=max_len)
```

4.7 Loss and Optimizer

For all variations of the BERTbased models, the same loss and custom optimizer were used. The Sparse Categorical Cross Entropy is chosen and the from_logits argument is set True which expects the y_pred to be a logits tensor. This study uses the same optimizer Bert was trained within Delvin et al.'s (2018) original paper: the "Adaptive Moments" (Adam). This optimizer minimizes the prediction loss and does regularization by weight decay (not using moments), which is also known as AdamW. For the learning rate (init_lr), you will use the same schedule as BERT pre-training: linear decay of a notional initial learning rate, prefixed with a linear warm-up phase over the first 10% of training steps (num_warmup_steps). In line with the BERT paper, the initial learning rate is smaller for fine-tuning (best of 5e-5, 3e-5, 2e-5).

4.8 Batch Size

The batch size is a hyperparameter that plays an important role in allocating the number of samples to work through before updating the internal model parameters. Batch size is usually set between 1 and the size of the training set and this learning algorithm is called mini-batch gradient descent. In most research popular batch sizes are 16,32,64,128. Using a smaller batch size has been empirically shown to have faster convergence towards a good solution which is smaller batch sizes allow the model to "start learning before having to see all the data." Meanwhile increasing the batch size has empirically shown better generalization. However, this is at the cost of slower, empirical convergence to that optima. A batch size of 32 was used in this study which was decided during hyperparameter tuning.

4.9 Epochs

An epoch is when one pass of the entire dataset is made over the neural network. An epoch is comprised of one or more batches (Shen,2018). With each epoch, the dataset's internal model parameters are updated. Therefore, generally with more epochs, the model has more time to improve. For the BertBase models, 80 epochs were initially set but later reduced due to the model's performance stabilizing much earlier than 80 epochs thus reducing run-time.

4.10 Dropout Layer

A recent study by Anigri et al. (2021) investigates the impact and importance of tweaking the dropout hyper-parameters on the BERT classification performance. The study concludes that combining the dropout and hidden layers significantly reduced overfitting and improved the model's performance. This study will further investigate such findings, which will be discussed in the results and experimentation in the next chapter.

4.11 Metrics

To evaluate the results obtained in each of the models developed, four different metrics have been used: Accuracy (A), Precision (P), Recall (R) and F1-score (F1). Moreover, the training and validation accuracy and loss plots as well as the use of the confusion matrix aided the evaluation in the next section.

1. Confusion Matrix In a binary classification problem, the matrix helps understand the model's performance on each class and in this case on how well the models perform on the racism (negative) class specifically. There are 4 metrics that are produced from the confusion matrix: TP: True positives, TN: True negatives, FP: False Positives, FN: False negatives. Using these 4 metrics, other metrics are calculated and used for evaluation.
2. Accuracy is a metric used to calculate the percentage of correct predictions of the total number of samples. For Binary classification accuracy is a metric that is used to describe the model's performance for both non-racist and racist classes.

$$\text{accuracy} = \frac{\text{correct predictions}}{\text{all predictions}}$$

3. Precision is a metric used to calculate what percentage of the positive samples have been correctly classified with respect to all positive predictions. The formula used to calculate precision is as follows:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

4. Recall is a metric used to calculate what percentage of the samples classified as positive have been correctly classified with respect to all the samples that are truly positive. The formula used to calculate recall is as follows:

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

5. F1-score is a metric used to calculate the effectiveness of a classifier by combining the precision and recall values. The F1 score is also really effective when the TN are high which for this study is the misclassification of racist tweets which is a helpful measure

as the study is focused on reflecting on which model performs best at classifying racist tweets. The formula used to calculate this metric is as follows:

$$F1 = 2. \frac{Precision \times Recall}{Precision + Recall}$$

6. Training and Validation accuracy and loss Plots: validation curves are used to evaluate hyperparameter performance on the models and detect any overfitting or underfitting. The learning curves are also a good indicator of the bias and variance of the estimators used in the model training.

Chapter 5

Results

5.1 Baseline Results

5.1.1 SVM

The base model for performance comparison is the SVM model, which showed promising accuracy of 94.13%. The base model showed promising results, was easy to implement and had fast runtime. The following models will be compared to the SVM model's performance. For better readability, the racism class will be referred to as the positive class of tweets containing racist language and the non-racism class as the negative class of non-hateful tweets. The SVM model's resulting classification report shows that the positive class's precision is 0.96 compared to 0.93 for the negative class. However, recall is much higher for the negative class with 0.97 compared to a recall of 0.9 for the positive class. Thus, higher positive precision shows that the SVM model is better at predicting racist tweets, but higher recall shows the model can detect non-racist tweets better.

Table 5.1: SVM Classification Scores

label	precision	recall	f1 score
0	0.93	0.97	0.95
1	0.96	0.90	0.93

5.1.2 Bertbase MLP model

The first model implemented after the SVM model was the Bertbase MLP model, which had a higher accuracy score of 95.18%. However, when plotting the loss and accuracy graphs of the MLP model, the graphs in figure 5.1 show a high degree of overfitting despite hyperparameter tuning efforts. Despite the original implementation suggesting using a random initializer, better results and reduced overfitting were observed by replacing the initializer with the HeNormal() initializer. Moreover, decreasing the neurons in the dense layer further reduced the overfitting of the model. The overfitting did not seem to impact the accuracy of the test dataset, but such a large gap between training and validation loss would hinder its generalization and suggests the implementation of further models.

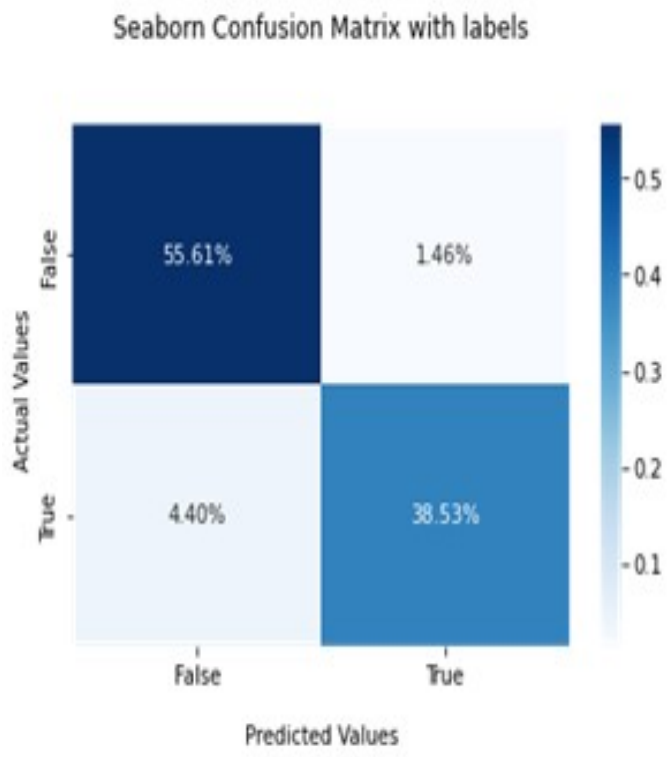


Figure 5.1: Corresponding Confusion matrix

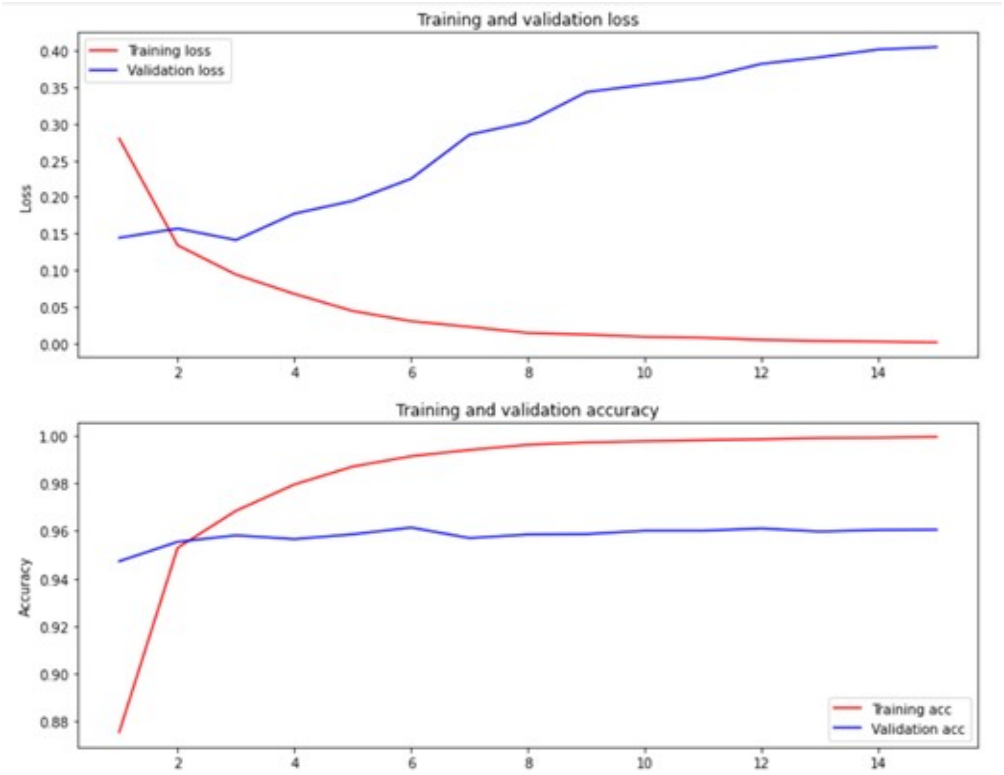


Figure 5.2: BertBase-MLP training plots

5.2 Bertbase CNN Model Experimentation and Results

The Bertbase CNN model required much more effort in hyperparameter tuning and experimenting with different architectures and layers to achieve satisfactory results. As previously discussed, the following layers were used to experiment with the best performance in detecting racism tweets: pooling, sequence, and concatenated encoder layers. The following configurations were essential for the BertCNN model improvement regardless of the layer tested: dropout layers, number of neurons chosen for the dropout layer, and number of neurons in the Dense Layer, with the results shown in figure 5.3. The optimal parameters were chosen based on accuracy and validation loss results. The model performs very well on the training set with a consistently high training accuracy and an ideal training loss close to zero. The originally inspired architecture with 512 neurons in the Dense layer and one dropout layer with a rate of 0.1 showed excellent accuracy but was largely overfitted with a validation loss of 0.42. Therefore, another dropout layer with a rate of 0.2 was added, slightly decreasing the loss. However, decreasing the number of neurons in the Dense layer significantly decreased the validation loss and narrowed the gap between the training and validation loss.

It was observed from the results in figure 5.3 that there is a noticeable trade-off between dropout layers and the number of neurons in the dense layer. The model's accuracy generally increased after adding dropout layers and increasing the rate. Moreover, adding a dropout layer between the outputs from the BERT model and the first convolutional layer in the dense layer decreased the validation loss and increased the validation accuracy. Thus implying a vital role in the model's architecture that was not previously suggested. Furthermore, decreasing the number of neurons in the dense layer significantly decreased the validation loss suggesting a less complex model is more suitable for this task. However, after increasing the dropout layer and the rate, the validation loss seems to increase but offers slightly higher accuracy. Therefore, the chosen hyperparameters were a balance of the trade-off and chose a low validation loss but preferred a higher accuracy model. The final model architecture that was applied to all the following variations of the BERTCNN model was: 3 dropout layers with a rate of 0.4 with a higher rate of 0.6 for the first dropout layer and 128 neurons for the Dense layer, which gave an accuracy of 96.3% and validation loss of 0.17.

Dropout Layers #	Dropout neurons	Dense Layer neurons	Test Accuracy	Validation Loss
1	0.1	512	95.43	0.42
2	0.2	512 256	95.46 95.15	0.36 0.194
3	0.2 0.4	256	95.2 95.8	0.22 0.185
3	0.2 0.4 0.6	128	95.8 96.3 96.4	0.21 0.17 0.24
4	0.4	128	94.9	0.26

Figure 5.3: Hyperparameter Tuning Results

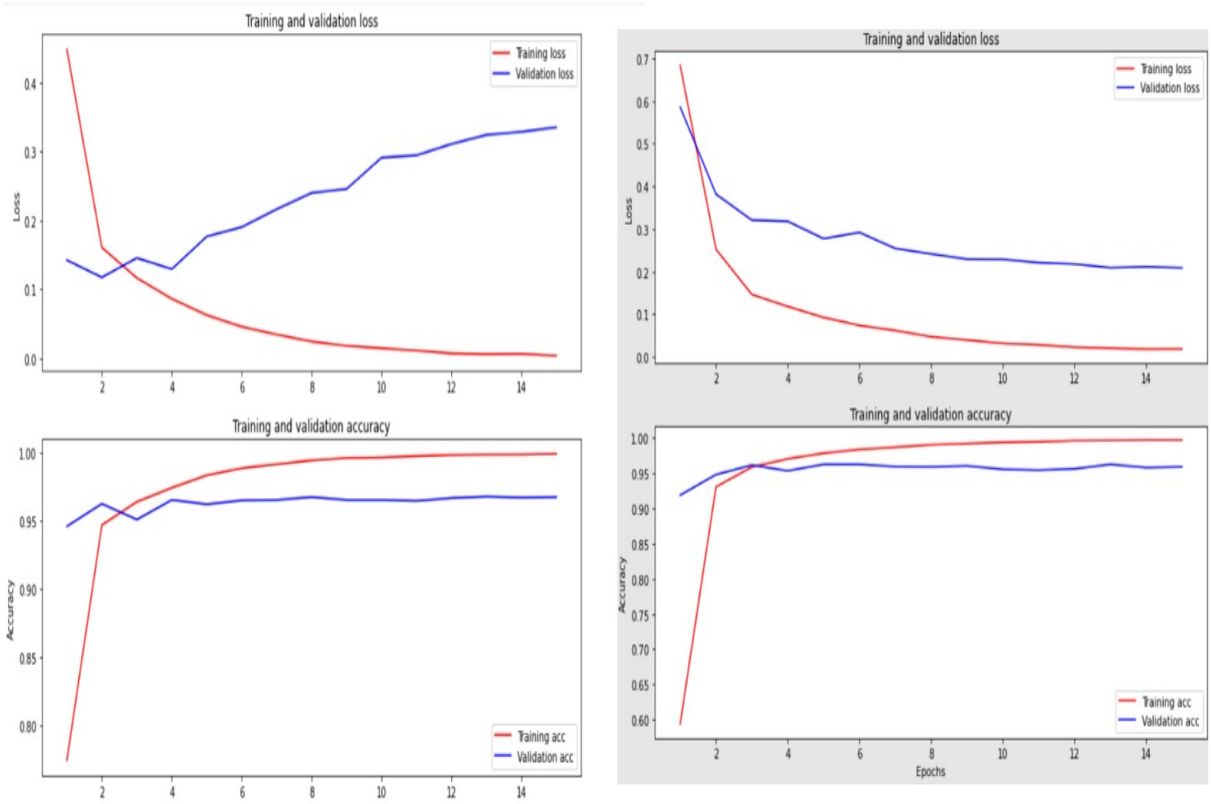


Figure 5.4: Example of Results Before and After Tuning

5.2.1 Comparison of the 3 Key Layers

The three layers showed comparable accuracies: the sequence layer showed the highest accuracy of 96.5%, followed by the pooling layer with 96.2%, and finally, the encoder layer with 96.0%. Therefore, classification scores were produced to analyse the model's variations further. Notably, the encoder layer scored much lower than the sequence and pooled outputs on recall. The encoder layer has a high precision of 0.94 at classifying the positive class but struggles to detect them with a recall of 0.88 compared to the pooled and sequence layers. The pooled layer only slightly shows better performance than the sequence layer with 1% higher precision, recall, and F1 scores. Another factor contributing to the decision to use the pooled output was the lower validation loss compared to the sequence layer observed while running the models.

Encoder layers

	precision	recall	f1-score
0	0.95	0.95	0.95
1	0.94	0.88	0.91

Sequence layer

	precision	recall	f1-score
0	0.97	0.95	0.96
1	0.93	0.93	0.93

Pooled output

	precision	recall	f1-score
0	0.97	0.95	0.97
1	0.95	0.94	0.94

Figure 5.5: BertCNN results with different layers

5.3 GCR-NN Model Experimentation and Results

The proposed GCR-NN model inspired by Lee et al. (2022) reported 97% accuracy results on their racism dataset and 95% on a dataset similar to the one used in this study. Initially, an accuracy of 90.56% was achieved, which was much lower than the reported results and even the baseline model. Therefore, a few adjustments were experimented with to test if the accuracy would increase. The following hyperparameters were tested: input_length, dense layer neurons, number of epochs, and batch size. As seen in table 5.6, The input length was changed to 10 as the proposed input length of 5 resulted in inadequate performance. Meanwhile, an input length of 10 and 20 decreased the model's performance significantly. The model's best performance tends to be reached within the first 40 epochs and a higher number of epochs has shown to decrease performance in some cases. A batch size of 16,32, and 64 were tested but gave very similar results. However, the batch size of 32 showed a slight increase in accuracy so it was chosen. Tuning the number of neurons in the dense layer also had a slight increase in the accuracy reaching the best accuracy of 94.18% with 128 neurons in the Dense layer. The most important parameter however that significantly changed the results was the input length of the words in the embedding layer.] The hyperparameter setting that resulted in 94.18% accuracy was chosen.

After the final hyperparameters were decided, a further analysis was done by producing the classification scores for the GCR-NN model. The GCR-NN model has the lowest precision score thus far but notably the highest recall score. The model can detect the positive class with an exceptional recall of 0.97.

Table 5.2: GCR-NN Classification Scores

label	precision	recall	f1 score
0	0.96	0.93	0.95
1	0.91	0.97	0.93

Input Length	epochs	Dense Layer neurons	Batch Size	Test Accuracy
5	100	16	16	90.56
10	60	16	16	0.78
	40		32	0.80
15	40	16	32	93.83
		32		94.07
		128		94.18
20	40	128	32	0.76

Figure 5.6: Results of GCR-NN hyperparameter tuning

Final Results of the ML models used

Table 5.3: Final Model Scores

Model	Test Accuracy	Precision	Recall	F1
SVM	94.13	0.96	0.90	0.95
Bertbase MLP	95.18	0.88	0.95	0.92
BERTbase CNN:Pooling Layer	96.3	0.95	0.94	0.94
BERTbase CNN:Sequence Layer	96.5	0.93	0.93	0.93
BERTbase CNN:Encoder Layer	96.0	0.94	0.88	0.91
GCR-NN	94.14	0.90	0.97	0.93

5.4 Models Performance and Generalizability on Ukraine and Afghanistan Datasets

The Ukraine and Afghanistan Twitter datasets contain text that is relevant to the time of the crisis and contains public opinions on immigration and racism as the events were ongoing. The

tweets were labelled using the VADER tool with negative, positive, and neutral. Notably, the Afghanistan tweets show a higher percentage of negative tweets, 53%, whereas nearly half of the tweets show negative sentiment concerning immigration. Meanwhile, the Ukraine tweets show a 39.1% negative sentiment when immigration is discussed. The GCR-NN and the best Bertbase CNN model were used to make predictions on both datasets. This study is interested in the models that can confirm the hypothesis that there is a more significant anti-immigration sentiment in the Afghanistan crisis tweets than in the Ukraine crisis by detecting a notably higher number of racist tweets. Moreover, the study is interested in how well these models perform on the external corpora, thus commenting on the generalizability of the models.

The BertBase CNN on the earlier dataset demonstrated an accuracy of 96.3% and a Recall of 0.94. Meanwhile, the GCR-NN model demonstrated an accuracy of 94.2% and a recall of 0.97. The Recall score is essential to this analysis because it will measure how well the model detects racist tweets. Predictions were run on both datasets, and a few measures were considered: percentage of racist tweets predicted, Recall, and F1 score. The GCR-NN model detected a similar percentage of racist tweets compared to the actual percentage of racist tweets. However, despite a representative prediction, the low Recall and F1 score a high misclassification rate. However, the GCR-NN shows a much higher recall for racist tweets in the Afghanistan dataset than in the Ukraine dataset, which will be discussed later. The BERTCNN performed much better on the Ukraine dataset than on the Afghanistan Dataset. The BERTCNN only predicted half of the actual percentage of racist tweets and predicted 8% more racist tweets in the Ukraine dataset. The GCR-NN model performed much better on the Afghanistan tweets, while the BertBase Model performed slightly better on the Ukraine tweets.

Dataset	Model	Actual Racist tweets	Predicted Racist tweets	Recall	F1 score
Ukraine	GRCNN	39.1%	30.8%	0.51	0.43
Afghanistan	GRCNN	53%	40 %	0.70	0.56
Ukraine	BERTCNN	39.1%	47%	0.57	0.53
Afghanistan	BERTCNN	53%	25.9%	0.3	0.40

Figure 5.7: Results on Ukraine and Afghanistan Corpa

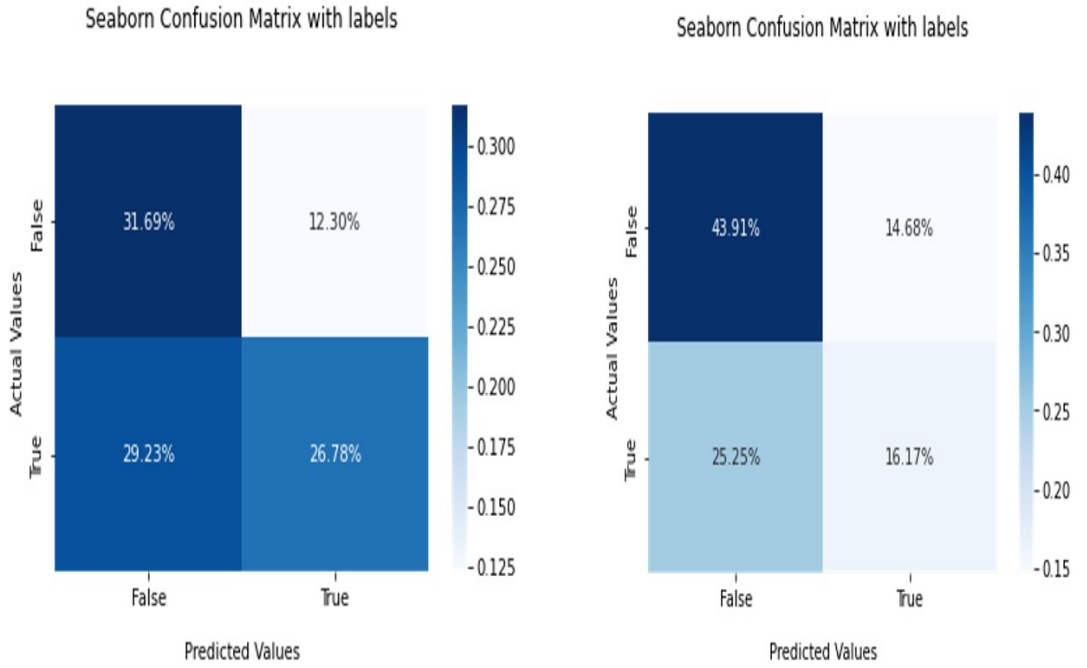


Figure 5.8: GCRNN Afghan Vs Ukraine

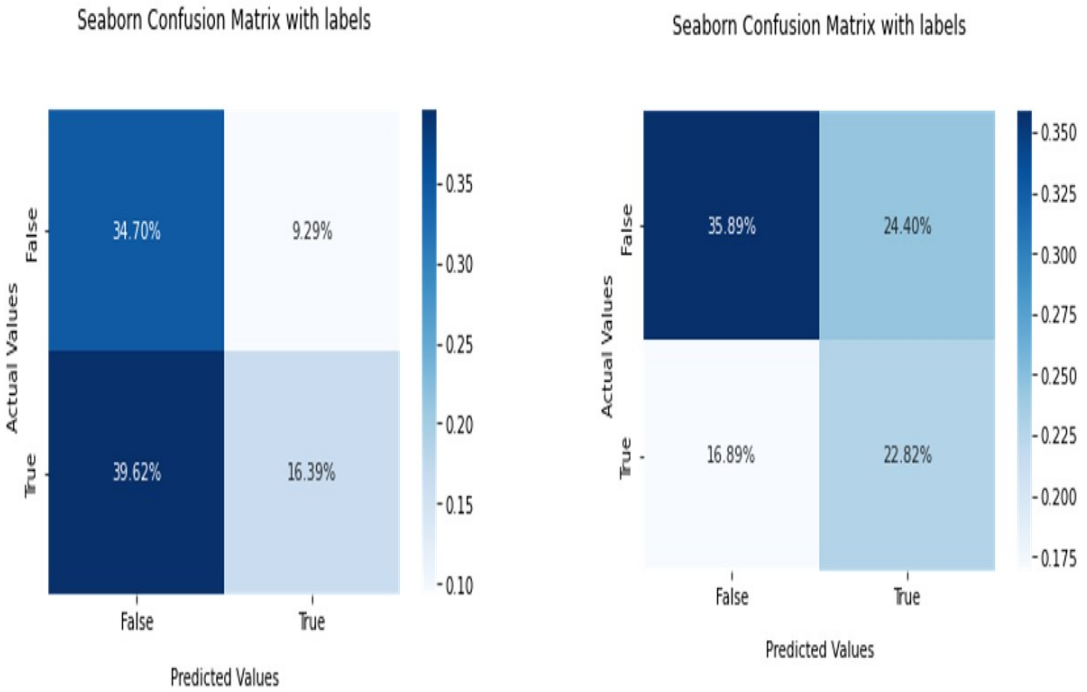


Figure 5.9: BertBase Afghan Vs Ukraine

5.4.1 Further Analysis

The BertCNN model showed a noteworthy poor performance on the Afghan dataset. Therefore, BertCNN model predictions were compared to the actual values and the VADER compound score, as shown in figure 5.10.

index	renderedContent	clean_tweet	vadar compound	sentiment
3095	I would like to know how many immigrants have crossed the border since the beginning of the problems in Afghanistan. Anyone?	know many immigrants crossed border beginning problems afghanistan anyone	-0.4019	1
18014	@seanhannity So is fentanyl not the most dangerous drug in the world? At least this will help with population control which seems to be the second priority for the administration after the first which is to resettle immigrants from the southern border and Afghanistan into swing states.	fentanyl dangerous drug world least will help population control seems second priority administration first resettle immigrants southern border afghanistan swing states	-0.1027	1
48211	@free_rover @newdaynewlife11 @Covid19Critical Yet that same government is giving it to immigrants coming in from Afghanistan. Check out the CDC website and see for yourself before they take it down. They must be racist giving it to immigrants as "toxic" as it is. Wait - weren't they saying it's horse medicine? 🙄	yet government giving immigrants coming afghanistan check cdc website see take must racist giving immigrants toxic wait werent saying horse medicine	-0.0516	1
58504	@JoshMandelOhio Jesus & the Apostles were all Jewish, 3,000 miles west of Afghanistan. Can't figure out from who you learned your Christianity & Bible, but obviously it was from some racist & bigoted ppl. I live in city w/ many Middle-Eastern immigrants - they're good & decent ppl.	jesus amp apostles jewish miles west afghanistan cant figure learned christianity amp bible obviously racist amp bigoted ppl live city many middle eastern immigrants theyre good amp decent ppl	-0.2732	1

Figure 5.10: Tweets from Afghanistan Dataset containing racist sentiment

Limitations of the VADER library in this study:

The VADER library is tasked to allocate a negative score for negative sentiment found in the dataset. Therefore, many tweets are misclassified as racist due to the negative sentiment present in the tweet, such as the last tweet, where there are many negative words, but the tweet itself was not racist, which is highly misleading to the algorithm. Therefore, tweets were relabeled as negative only if they were associated with a high negative score. After observing more tweets, most of the tweets scoring above -0.5 were correctly labelled as racist. Consequently, this reduced the percentage of racist tweets in the corpus to 36Results after adjusting the VADER scoring: The adjustment to the dataset Afghanistan BERT model improved recall to 0.36 and F1 score to 0.41 and predicted a 25% of racist tweets. The new estimate is much closer to the actual percentage than the previous prediction of the previous implementation. Consequently, The same strict labelling was applied to the Ukraine dataset but with a slightly different implementation. Restricting the labelling also decreased the percentage of false positives from 39.62% to 23.50%. The Ukraine dataset negative tweets dropped to 21%, which would already impact the model's performance due to imbalance. Therefore, the neutral tweets were dropped for better analysis, which balanced the dataset between negative and positive tweets in the Ukraine dataset. There was a slight increase in recall from 0.51 to 0.58 and the F1 score from 0.43 to 0.54. As the GCRNN model showed the most promising results in both datasets, a further investigation of the model's sensitivity is also measured by the precision value better to understand the difference in performance in both datasets. The precision for the Ukraine dataset was much higher than the Afghanistan dataset, with 0.5 compared to 0.38. Therefore, a word cloud of the correctly predicted racist tweets was generated for both datasets. As shown in figures 5.11 and 5.12, the Afghanistan dataset word cloud for the correctly classified racist tweets contains many hateful words. Most Notably,

the words:" immigrant, Taliban, illegal, terrorist, and anti refugees" seem to be associated with the Afghanistan immigration tweets. On the other hand, the word cloud produced for the Ukraine tweets highly mentions some prominent words: "racism, racist, anti, black, and African". As the original dataset was trained on generally racist tweets, this could explain why the model had higher precision and sensitivity to the words in the Ukraine dataset rather than the Afghanistan dataset. The model's better precision on the Ukraine tweets could be correlated to the nature of the words detected and present in the Ukraine corpus, which is more relevant to racism. Meanwhile, the Afghanistan tweets, while also racist, use a more specific anti-immigration language which is not present in the general corpus.

Figure 5.11: WordCloud of Correctly Predicted Afghanistan tweets by GCR-NN

Figure 5.12: WordCloud of Correctly Predicted Ukraine tweets by GCR-NN

Chapter 6

Conclusions

This study performs an extensive comparison of state-of-the-art hate speech detection models on a newly collected and labelled racism dataset. The dataset will be publicly available on GitHub and Kaggle. The findings help understand each model's strengths and weaknesses in performance and application. The models were not only rated by their classification score but by their training and validation performance. Despite, the GCR-NN having less accuracy than the Bertbase models and being close to the baseline model, the model's training and validation plots anticipated its, superior generalizability to the other models. Whereas the Bertbase models seemed to overfit and required a lot of effort to reduce overfitting.

6.0.1 Answering the Research Questions

Research Question 1: The Bertbase model indeed increased the overall scores in hate speech detection and required less pre-processing due to its advanced built-in pre-processing methods. However, when applied to external datasets, its performance suffered, showing its poor generalizability and requirement for larger datasets and further fine-tuning for better performance. In tasks where small percentages in classification matter and data are available at large, the Bertmodel is recommended as its performance shows the potential to exceed existing models with high accuracy and precision. As shown in the experimentation, for the racism detection task, the BertBase Pooling layer is recommended as it demonstrated the best results of the three at detecting racist tweets. As explained previously, the pooled layer can be considered the embedding of the tweet, while the sequence layer can be the contextual embedding of every token of the tweet. The pooled output's better performance could be attributed to its more general and less complex nature. However, for simpler tasks, the SVM is recommended. The SVM is empirically proven to perform very well on hate speech detection tasks and is very simple to implement, also yielding the highest precision of the proposed architectures.

Research Question 2: Despite the excellent performance, the BERTbase CNN showed on the primary dataset, its poor performance on the corpora demonstrates its lack of generalizability. Meanwhile, the GCR-NN was able to correctly represent the racism difference between the Ukraine and Afghanistan datasets, with much better performance than the BERTbase model. The GCR-NN had the highest recall scores overall in the three datasets indicating that it is the best model at detecting racist tweets in a corpa. However, the GCR-NN model suffers from a generalization issue due to the input length parameter as it depends on the text input. The input length parameter significantly altered the performance of the model as seen by the

experimentation within the GCR-NN model.

Research Question 3: The VADER percentage results classified a higher negative sentiment towards Afghanistan immigration than Ukraine immigrants, which was confirmed by the GCR-NN predictions. The difference was also backed by the model's better sensitivity to words in the Ukraine dataset, as it contained distinct racist language. Common words that the ML model detected refer to the issue that became a prominent social issue during the Russian invasion: Black people were discriminated against at the border. The issue was examined in many articles, such as "The Russian invasion of Ukraine shows racism has no boundaries" by Rashawn Ray(2022). In contrast, the Afghanistan dataset contained more anti-immigration language, which was more challenging to detect as the primary dataset was not representative enough of anti-immigration vocabulary. Moreover, a statement that can be inferred from the results is that the public tends to perceive non-European immigrants as illegal, while this language was not used within the Ukraine corpus. However, it must not be overlooked that immigration and immigrants, regardless of origin, still receive a significant percentage of negative and racist sentiment from social media opinions. Such findings confirm the urgent need for social awareness concerning immigrants to improve public perception and push for efforts to target racist and hateful tweets online.

6.0.2 Limitations and Further Research

Throughout this study, it is evident for more generalizable predictions a much larger and more representative dataset is required especially for discrimination and anti-immigration language. The study encourages further research to expand on the provided dataset which helps train and build better models for detecting racist and anti-immigration tweets. Further research could also be done with different datasets to further support or contradict the findings made in this study concerning the GRC-NN's performance and potential. Moreover, the statements made on the discrimination made between origins have their limitations as they are based on 2 crises and require more examples and investigation to increase the certainty of these statements. The conclusions concerning final architectures can also be limited as they are based on the best architectures for the racism dataset provided which could be tested and explored on more datasets to confirm this study's findings. Finally, the study was capable to provide novel datasets, proposing variations of architectures, analysis of the impact of different hyperparameters, and insights into the ongoing immigration and social issues. An important goal of this study is to bring awareness to the treatment of immigrants and the negative perceptions that create racism and discrimination which Twitter and other social media platforms should put more effort into targetting and preventing.

Bibliography

Al-Hassan, A. and Al-Dossari, H., 2019, February. Detection of hate speech in social networks: a survey on multilingual corpus. In 6th international conference on computer science and information technology (Vol. 10).

Ali, N.M., Abd El Hamid, M.M. and Youssif, A., 2019. Sentiment analysis for movies reviews dataset using deep learning models. International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol, 9.

Cao, R., Lee, R.K.W. and Hoang, T.A., 2020, July. DeepHate: Hate speech detection via multi-faceted text representations. In 12th ACM conference on web science (pp. 11-20).

Chiril, P., Pamungkas, E.W., Benamara, F., Moriceau, V. and Patti, V., 2022. Emotionally informed hate speech detection: a multi-target perspective. Cognitive Computation, 14(1), pp.322-352.

Djuric, N., Zhou, J., Morris, R., et al.: Hate speech detection with comment embeddings. In: Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Companion, pp. 29–30. ACM, New York (2015). https://doi.org/10.1145/2740908.2742760

F. Rustam, M. Khalid, W. Aslam, V. Rupapara, A. Mehmood, and G. S. Choi, "A performance comparison of supervised machine learning models for COVID-19 tweets sentiment analysis," PLoS ONE, vol. 16,no. 2, Feb. 2021, Art. no. e0245909.

G. K. Pitsilis, H. Ramampiaro, and H. Langseth, "Effective hate-speech detection in Twitter data using recurrent neural networks," Appl. Intell., vol. 48, no. 12, pp. 4730–4742, Dec. 2018.

Gitari, N.D., Zuping, Z., Damien, H. and Long, J., 2015. A lexicon-based approach for hate speech detection. International Journal of Multimedia and Ubiquitous Engineering, 10(4), pp.215-230.

Goutsos, D. (2021). Multimodal hate speech detection in greek social media. Multimodal Technologies and Interaction, 5(7), 34. doi:http://dx.doi.org/10.3390/mti5070034

Haq, E.U., Tyson, G., Lee, L.H., Braud, T. and Hui, P., 2022. Twitter dataset for 2022 russo-ukrainian crisis. arXiv preprint arXiv:2203.02955.

J. Lilleberg, Y. Zhu, and Y. Zhang, "Support vector machines and Word2vec for text classification with semantic features," Proc. 2015 IEEE 14th Int. Conf. Cogn. Informatics Cogn. Comput. ICCI*CC 2015, pp. 136–140, 2015.

Ketsbaia, L., Issac, B. and Chen, X., 2020, December. Detection of Hate Tweets using Machine Learning and Deep Learning. In 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) (pp. 751-758). IEEE.

L. Jiang and Y. Suzuki, "Detecting hate speech from tweets for sentiment analysis," 2019 6th International Conference on Systems and Informatics (ICSAI), 2019, pp. 671-676, doi: 10.1109/ICSAI48974.2019.9010578.

Lee, E., Rustam, F., Washington, P.B., El Barakaz, F., Aljedaani, W. and Ashraf, I., 2022. Racism Detection by Analyzing Differential Opinions Through Sentiment Analysis of Tweets using Stacked Ensemble GCR-NN Model. IEEE Access.

M. Al-Smadi, O. Qawasmeh, M. Al-Ayyoub, Y. Jararweh, and B. Gupta, "Deep Recurrent neural network vs. support vector machine for aspect-based sentiment analysis of Arabic hotels' reviews," J. Comput. Sci., vol. 27, pp. 386–393, 2018.

Medium. 2022. Effect of batch size on training dynamics. [online] Available at: <https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e> [Accessed 9 September 2022].

Mozafari, M., Farahbakhsh, R. and Crespi, N., 2019, December. A BERT-based transfer learning approach for hate speech detection in online social media. In International Conference on Complex Networks and Their Applications (pp. 928-940). Springer, Cham.

Parihar, A.S., Thapa, S. and Mishra, S., 2021, June. Hate Speech Detection Using Natural Language Processing: Applications and Challenges. In 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI) (pp. 1302-1308). IEEE.

Pitsilis GK, Ramampiaro H and Langseth H. Effective hate-speech detection in Twitter data using recurrent neural networks.Appl Intell 2018; 48(12):4730–4742.

Putri, T.T.A., Sriadhi, S., Sari, R.D., Rahmadani, R. and Hutahaean, H.D., 2020, April. A comparison of classification algorithms for hate speech detection. In Iop conference series: Materials science and engineering (Vol. 830, No. 3, p. 032006). IOP Publishing.

PyImageSearch. 2022. A gentle introduction to tf.data with TensorFlow - PyImageSearch.

[online] Available at: <https://pyimagesearch.com/2021/06/14/a-gentle-introduction-to-tf-data-with-tensorflow/> [Accessed 9 September 2022].

R. Alshalan and H. Al-Khalifa, "A deep learning approach for automatic hate speech detection in the Saudi Twittersphere," Appl. Sci., vol. 10, no. 23, p. 8614, Dec. 2020.

Ranasinghe, Tharindu & Zampieri, Marcos. (2020). Multilingual Offensive Language Identification with Cross-lingual Embeddings. 5838-5844. 10.18653/v1/2020.emnlp-main.470.

Rizoiu M, Wang T, Ferraro G, and Suominen H. Transfer Learning for Hate Speech Detection in Social Media. CoRR, abs/ 1906.

S. Malmasi and M. Zampieri, "Detecting hate speech in social media,"arXiv preprint arXiv:1712.06427, 2017.

Twitter. (n.d.). Tweet object | docs | twitter developer platform. Twitter. Retrieved April 5, 2022, from https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/tweet Waseem, Z. and Hovy, D., 2016, June. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In Proceedings of the NAACL student research workshop (pp. 88-93).

Zhang Z and Luo L. Hate speech detection: a solved problem? The challenging case of long tail on Twitter. arXiv preprint: arXiv:1803.03662, 2018.

Zhang, Z., Robinson, D. and Tepper, J., 2018, June. Detecting hate speech on twitter using a convolution-GRU based deep neural network. In European semantic web conference (pp. 745-760). Springer, Cham.

Rizoiu, M.A., Wang, T., Ferraro, G. and Suominen, H., 2019. Transfer learning for hate speech detection in social media. arXiv preprint arXiv:1906.03829.

Appendix A

Design Diagrams

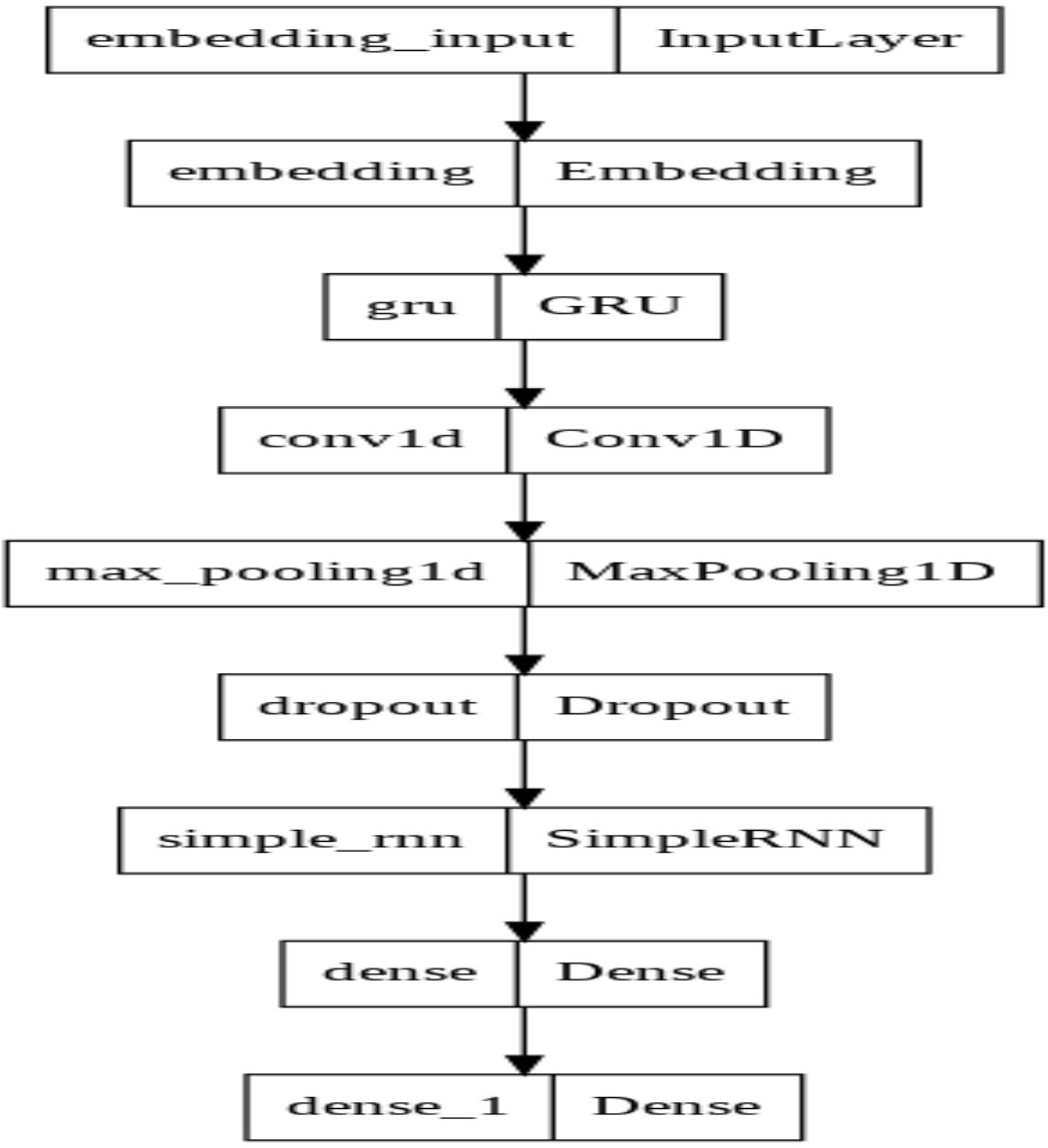


Figure A.1: Architecture of GCR-NN

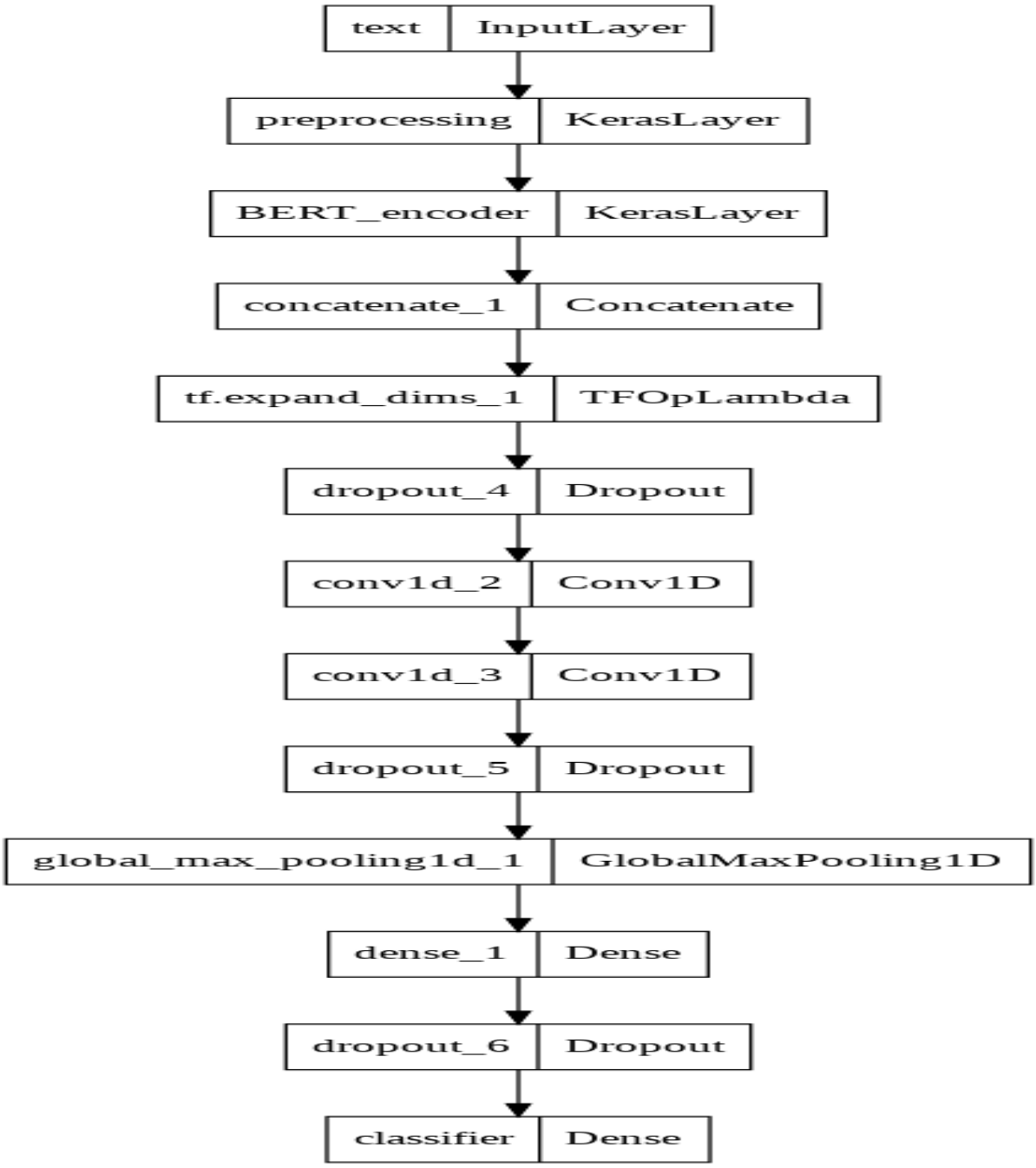


Figure A.2: Architecture of BERTCNN pooled output