Detecting Hate Speech using Machine Learning and Sampling Techniques

Angela Yeukai Chikova

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Technology
in
Information Technology

Eastern Mediterranean University
September 2021
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Technology in Information Technology.

Assoc. Prof. Dr. Nazife Dimililer
Director, School of Computing and Technology

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Technology in Information Technology.

Assoc. Prof. Dr. Nazife Dimililer
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Nazife Dimililer

2. Asst. Prof. Dr. Mustafa Tanel Babagil

3. Asst. Prof. Dr. Fatma Tansu Hocanın

ABSTRACT

The spread of hate speech on social media platforms is a problem that is constantly becoming more imminent as the access to related technologies gets easier. This study focuses on detecting hate speech on an imbalanced multiclass twitter dataset using Machine Learning (ML) algorithms. The most commonly used ML algorithms namely, Logistic Regression, Support Vector Machines (SVM) and deep learning systems Gated Recurrent Unit (GRU), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Bi-directional Long Short-Term Memory (BiLSTM) and a hybrid model CNNBiLSTM have been used for hate speech detection. In order to overcome the problems that arise from using an imbalanced dataset several techniques are used to balance the dataset, Synthetic Minority Oversampling Technique (SMOTE), SMOTETomek, SMOTEENN, Adaptive Synthetic (ADASYN), class weights and the proposed method. Each classifier was trained with all data balancing techniques and their performances were compared in order to find the best classifier for classifying hate speech in the dataset. The best classifier was CNN using the proposed method and it had an F1-score of 0.96 with a Cohen Kappa score of 0.94 and an overall Recall and Precision score of 0.96. For the best system, the recall and precision scores for the hate class was 1.00 and 0.94 respectively.

Keywords: hate speech, multiclass imbalanced dataset, SMOTE, SMOTETomek, SMOTEENN, ADASYN, class weights, proposed method, machine learning, neural networks.

ÖZ

Nefret söyleminin sosyal medya platformlarında yayılması, ilgili teknolojilere erişim kolaylaştıkça sürekli artan bir sorundur. Bu çalışma, Makine Öğrenimi (ML) algoritmalarını kullanarak dengesiz çok sınıflı bir Twitter veri kümesinde nefret söylemini tespit etmeye odaklanmaktadır. En yaygın olarak kullanılan ML algoritmaları Lojistik Regresyon, Destek Vektör Makineleri (SVM) ve Kapılı Tekrarlayan Birim (GRU), Evrişimsel Sinir Ağı (CNN), Uzun Kısa Süreli Bellek (LSTM), Çift Yönlü Uzun Kısa- Nefret söyleminin tespiti için Term Memory (BiLSTM) ve bir hibrit model CNNBiLSTM gibi derin öğrenme sistemleri kullanılmıştır. Dengesiz bir veri kümesinin kullanılmasından kaynaklanan sorunların üstesinden gelmek için, veri kümesini dengelemek için çeşitli teknikler, Sentetik Azınlık Aşırı Örnekleme Tekniği (SMOTE), SMOTETomek, SMOTEENN, Uyarlanabilir Sentetik (ADASYN), sınıf ağırlıkları ve önerilen yöntem kullanılmıştır. Veri setinde nefret söylemini sınıflandırmak için en iyi sınıflandırıcıyı bulmak için her sınıflandırıcı her bir veri dengeleme tekniği ile eğitilmiş ve performansları karşılaştırılmıştır. Önerilen yöntemi kullanan en iyi sınıflandırıcı olarak 0.96'luk bir F1-puanına, 0.94'lik bir Cohen Kappa puanına ve 0.96'lik bir genel Geri Çağırma ve Kesinlik puanına sahip olan CNN algoritması belirlenmiştir. En iyi sınıflandırıcının nefret sınıfı için hatırlama ve kesinlik puanları sırasıyla 1.00 ve 0.94'tür.

Keywords: nefret söylemi, çok sınıflı dengesiz veri kümesi, SMOTE, SMOTETomek, SMOTEENN, ADASYN, sınıf ağırlıkları, önerilen yöntem, makine öğrenmesi, sinir ağları.

DEDICATION

TO GOD ALMIGHTY

&

MY FAMILY

ACKNOWLEDGMENT

I would like to thank my supervisor Assoc. Prof. Dr. Nazife Dimililer for her guidance, patience, encouragement and constructive suggestions during this course of this thesis. Her continued support gave me the inspiration that l needed to complete my thesis and her willingness to give her time to help me is something I will forever be grateful for.

I also want thank my family and as well as my friends for their encouragement and support during the course of my study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADASYN	Adaptive Synthetic Sampling Approach
AUROC	Area Under the Receiver Operating Characteristics
BERT	Bidirectional Encoder Representation from Transformers
BiLSTM	Bidirectional Long Short-Term Memory
BROUT	Balanced Random Oversampling Undersampling Technique
CNN	Convolutional Neural Network
CNNBiLSTM	Convolutional Neural Network and Bidirectional Long Short-Term Memory
DL	Deep Learning
DT	Decision Tree
ENN	Edited Nearest Neighbor
FN	False Negative
FP	False Positive
GRU	Gated Recurrent Unit
JSON	JavaScript Object Notation
KNN	K Nearest Neighbor
LR	Logistic Regression
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multi-Level Perceptron
NB	Naïve Bayes
NLP	Natural Language Processing
NN	Neural Network

RF	Random Forest
RNN	Recurrent Neural Network
ROS	Random Over Sampling
RUS	Random Under Sampling
SMOTE	Synthetic Minority Oversampling Technique
SMOTEENN	Synthetic Minority Oversampling Technique and Edited Nearest Neighbor
SMOTETomek	Synthetic Minority Oversampling Technique and Tomek links
SS	Sentiment Scores
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
URL	Uniform Resource Locator
USE	Universal Sentence Encoder

Chapter 1

INTRODUCTION

1.1 Introduction

The usage of social media as a means of communication has been increasing over the years as technology has been advancing. In the first quarter of 2021 it was recorded that there were 199 million users who were active on twitter per day with 500 million tweets posted per day[1] and Facebook recorded 1.88 billion daily active users[2]. As the number of users keeps on growing and people from different cultures and backgrounds express their views daily, it has become a difficult task to control the things that are posted on these platforms and one of the biggest problems that has arisen is the spread and detection of hate related content.

1.2 Challenges with detecting hate speech

There are different obstacles that are faced in the detection of hate speech and one of the major problems faced is the definition of hate speech itself because what some consider to be hate speech others do not. MacAvaney et al. expressed that not having a clear and uniform definition of hate speech poses a problem when trying to evaluate hate speech detection systems because the existing hate speech datasets were compiled using different definitions and that leads to datasets which have different information identified as hate making it difficult to identify which aspect was identified as hate[3]. Some of the different definitions of hate speech are as follows:

i. Davidson et al.: defined hate speech as the language used to express hate towards a targeted group or is intended to be derogatory in order to humiliate or to insult the members of a group.[4]

ii. Fortuna et al.: defined hate speech as language that is used to attack or diminish. Furthermore, they expressed that it incites violence or hate against groups based on specific characteristics such as religion, physical appearance, descent, ethnic or national origin, sexual orientation, gender identity and it can occur with different linguistic styles, even in subtle forms or when humor is used.[5]

Other challenges faced especially in social media stem from the use of characters such as numbers or other symbols inside or instead of words. For instance, for the word "TWEET" a user can substitute the letter "e" with "3" which make the word "TW33T", and that will create a new word which might be unknown and cannot be defined. Another problem in text classification can be the use of emoticons, emojis or memes that are being used together with texts when people tweet or post. If these symbols are used to convey hate and if they are removed during preprocessing or if they are not properly detected, the posted content can lose its actual meaning and a hate post can be mistakenly classified as non-hate.

1.3 Challenges of imbalanced data

Imbalanced data arises when the data in some classes is overly underrepresented compared to other classes[6]. This means that the majority class will have a higher number of samples compared to the minority classes. This is a problem during classification because classifiers will be biased towards the majority class. For instance, if a hate dataset contains two classes and majority class has 99% of normal

tweets and the minority class has 1% of hate tweets then during classification bias towards the majority class can occur and an accuracy of 99% can be obtained. If the aim is to identify the minority class, then it means the important information in the 1% will be lost as it will be misclassified as normal tweets. An example that shows a better picture is in trying to catch a fatal disease in medicine. If a misclassification of a non-fatal disease occurs then that means more tests will be done but if a misclassification of a fatal disease is done then that will pose serious health risks. These are some of the problems that can arise when dealing with imbalanced data and in almost all cases the majority class is less likely be misclassified. The minority class is usually misclassified and that leads to misclassification cost, time and risk evaluation[7].

1.4 Purpose of study

The purpose of this research is to find ways to improve hate detection on an imbalanced multiclass twitter dataset using Machine Learning (ML) algorithms and Neural Networks (NN). Since the dataset is imbalanced this research focuses on using different existing resampling techniques together with a proposed technique to find the best data balancing technique that leads to the improved performance of the classification models during hate speech detection. The performances of the classification models are evaluated in order to find the best classifier for hate speech detection.

Chapter 2

BACKGROUND MATERIALS

2.1 Introduction

The process of hate speech detection can be categorized as text classification which is the defined as the process of grouping text into different categories. Natural Language Processing (NLP)[8] is used by text classifiers to enable them to make analysis of data and to sort it by topic or sentiment. The classifiers can be ML algorithms or NN and they can work with labeled data or unlabeled data and their performances can be evaluated using different metrics. This chapter contains the information that is needed to understand the concepts that are going to be used in the following chapters. Section 2.2 contains the information about the classifiers used, Section 2.3 contains information about the features used on the dataset, Section 2.4 contains information about the imbalanced dataset used in this research and Section 2.5 contains the detailed information about the resampling techniques used to balance the dataset. The final section, Section 2.6 contains information about the different metrics that were used to evaluate the performance of the classifiers.

2.2 Machine Learning

ML is a subfield of artificial intelligence and it gives systems the ability to be able to automatically learn and to also improve from gained experience. ML is divided into three categories[9], [10]:

- Supervised learning: the algorithms are trained using a labeled dataset to be able to predict future events. The algorithm analyzes the training set from the

dataset and a function is produced to map input to output values. The algorithm is also able to make comparisons of the predicted output with the actual output in order to check how well the algorithm is performing. After training the algorithm will be able to predict the output from an unseen test set.

- Unsupervised learning: in this approach the data provided for training is not labelled. This is done to be able to find potential patterns from the dataset and that is generally called clustering.
- Reinforcement learning: a technique that enables an agent to learn by trial and error in an interactive environment using feedback from its own experiences and actions.

Semi supervised learning is also a category that is used in ML and it's a mixture of both supervised and unsupervised learning where both the labeled and the unlabeled data is used for training the model.

2.2.1 Machine Learning algorithms used in hate speech detection

2.2.1.1 Logistic regression

Logistic regression (LR) a ML statistical model that is used to predict the probability of a target variable. This is done by estimating the relationship between the (target) dependent variable and one or more of the independent variables. The formula is shown below:

$$logit(p) = \ln\left(\frac{p}{1-p}\right) = b0 + b1x1 + b2x2 \ldots + bkxk \qquad (1)$$

where;

p= the probability of the feature occurring

x1, x2…xk = set of input features of x

b1, b2…bk = the parameter value that will be estimated in the formula

The sigmoid function is used by LR because it maps the predictions that were predicted to the probabilities. A multinomial LR model was used in this research since that model is used to classify variables into multiple classes.

2.2.1.2 Decision Tree

The Decision Tree (DT) algorithm is used for creating classifiers that are able to predict the class of a target variable by using the decision rules that are learned from the training data. Multiple algorithms are used by DT to decide to split a node into more sub-nodes. The features of the dataset are represented by the internal nodes, the decision rules are represented by the branches and the outcome are represented by the terminal node. The representation of DT is shown in Figure 1.

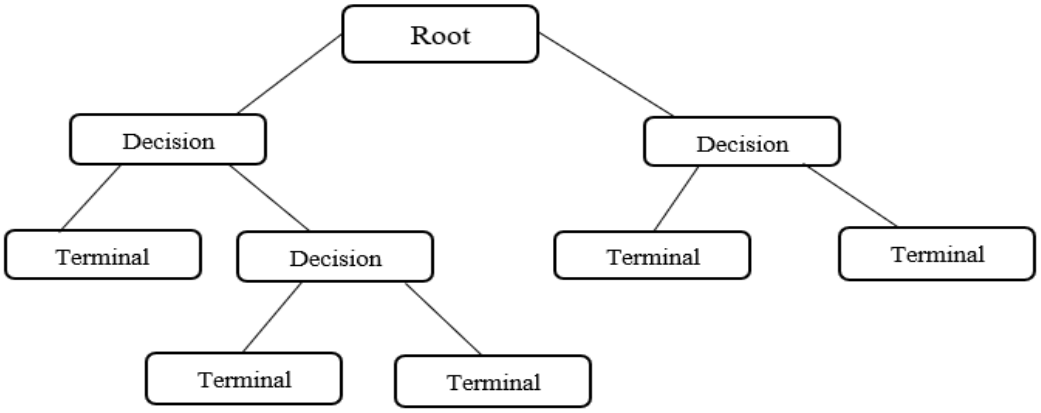

Figure 1: Decision Tree representation

2.2.1.3 Random Forest

Random Forest (RF) is an algorithm that makes use of ensemble methods for classification and regression problems. RF works by creating multiple individual DT randomly during training. Each individual tree provides a class prediction and the mean of the predictions made by the individual DT will be the output of RF.

2.2.1.4 Support Vector Machine

Support Vector Machine (SVM) is a linear model that is used for regression and classification problems. The algorithm works by creating a hyperplane that separates data into classes as shown in Figure 2. Since there are infinite lines that can separate two classes, SVM finds points that are closest to the hyperplane and those points are called support vectors. The margin which is the distance between the support vectors and the hyperplane is calculated. The hyperplane with the largest margin will be the best hyperplane used.

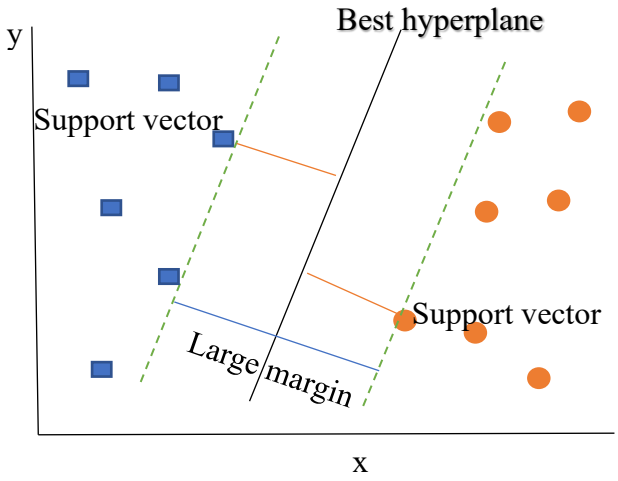

Figure 2: Visual representation of how SVM works

2.2.2 Neural Networks used in hate speech detection

Deep learning (DL) is a subset of ML that is inspired by how the human brain works. The DL algorithms work by analyzing data continually with a local structure. NN are used in DL and they are algorithms with a multi layered structure[11]. The algorithms pass data through several layers and each layer is able to progressively extract features and pass them to the next layer. NN work on different tasks like clustering, classification or regression. Unlabeled data can be grouped together based on the similarities that are found amongst the samples that are in the data using neural

networks. Furthermore, NN work on labeled data to classify the samples found in the dataset into different categories.

2.2.2.1 Recurrent Neural Networks

Recurrent Neural Networks (RNN) have a problem of suffering from short term memory. If a given sequence used is long, RNN will have a hard time carrying all the information in the sequence from earlier time steps to the ones later. This means that if you are trying to make predictions on a paragraph of tweets, RNN may leave out important information from the beginning of the tweets. This shows that RNN's suffer from the vanishing gradient which means that gradients shrink and this happens during back propagation. Gradients are values that are used to update the weights of neural networks and if they become too small the earlier layers in the neural network will not be able to learn. In order to solve this problem LSTM and GRU were created.

GRU is a type of RNN that uses a gated process to be able control and also manage how information flows between the neural network cells. Cho et al. explained that a GRU is able to help facilitate capturing of dependencies from a large amount of data without excluding any information from the prior potion of the series of data[12]. This performed by the gated units that solve the vanishing gradient problem. GRU has two gates The update gate helps the model to decide how much of the information from earlier requires moving along to the future steps. The gate can also be used to copy all of the details from the previous steps. The reset gate decides how much of the information from the past should be disregarded. It classifies unrelated data and informs the model which data to forget and move forward without it.

LSTM is a type of RNN that is capable of learning from history to process, classify and predict time series when long and unknown time gaps exist between important events[13]. It has three gates; input, output and forget gate The input gate is responsible for checking which information is relevant and should be added to the current step. The forget gate checks which information should be kept or thrown away and the output gate decides what the next hidden state should be.

A BiLSTM is a processing model that has of two LSTM. One accepts the input in a forward direction and the other takes it in a backward direction. The model effectively increases the amount of information that is available to the network and that improves the context that is available to the algorithm. The difference between LSTM and BiLSTM is that for LSTM the information just flows from backward to forward.

2.2.2.2 Convolutional Neural Network (CNN)

CNN is comprised of a class of deep feed forward artificial NN that uses a variation of the multilayer perceptron's designs that require minimal processing and they are inspired by the visual cortex of animals[14], [15]. CNN is usually used in image recognition but can now be used for text classification. In order to be used for text classification, tweets are represented as a matrix where a row is a vector that represent word. Different embeddings can be used to perform that task. Example, if a tweet has 10 words and a 200-dimension embedding is being used, you will have a 10x200 matrix an input and that would be the "image". Filters are then added over the input to find convolutions. Convolutions are mathematical combinations of two relationships that produce a third. These are then further reduced dimensionally by the max pooling layer so as to reduce the complexity and computation. Lastly the fully connected layers and the activation function on the output will provide values for each class.

2.3 Features used in hate speech detection

This section contains the different features that were used to train the machine learning algorithms. Term frequency inverse document frequency (TF-IDF) features where chosen because of their popularity for producing considerable results, sentiment scores from sentiment analysis were used to try and see if knowing the sentiment of the tweets increases the performance of the classifiers and doc2vec features were used because they find related sentences. FastText embeddings that were used for training the neural networks are also explained.

2.3.1 TF-IDF

TF-IDF can represent a document based on its words. These features are created based on the Term Frequency (TF) and also the Inverse Document Frequency (IDF). TF calculates the number of times a word appears in document divide by the total of words in that document and IDF calculates logarithm of the number of documents in the corpus divided by the number of documents where the specific word is used. Using TF-IDF tweets are assigned weights based on their importance and not their frequency.

2.3.2 Sentiment scores

The sentiment scores for the tweets were calculated using Valence Aware Dictionary and sEntiment Reasoner (VADER). VADER is a lexicon and rule-based tool for sentiment analysis that determines whether text is positive, negative or neutral. Furthermore, it can also tell how positive or negative the text is. VADER can easily detect slang, acronyms and emojis so and that makes it good for social media data.

2.3.3 Doc2Vec

Doc2vec is an unsupervised algorithm that is capable of learning fixed length feature vectors for documents or paragraphs. The doc2vec features are the numerical representation of the document.

2.3.4 FastText embeddings

FastText embeddings were created by Facebook to solve the problem that word2vec and Glove embeddings have of not being able to encode unknown words or words that are not in the vocabulary. FastText is an extension to Word2Vec and it follows the same Skip-gram and CBOW model. The difference between Word2Vec and FastText is that Word2Vec provides complete words that are whole into the neural network and FastText breaks the words into several sub-words (or n-grams) first and then feeds them into the neural network. For example, if the value for *n* is 3 and the word is '*phone*' then tri-gram will be ['<ph', 'pho', 'hon', 'one', 'ne>'] and the word embedding for the word will be sum of vector representation of the tri-grams. The hyper-parameters '*minn*' and '*maxn*' will be considered as 3 and the characters '<' and '>' represents start and end of the word. When this method is used words that are unknown can be represented in vector form as they will have a high probability that their n-grams are also present in other words.

2.4 Dataset used in hate speech detection

The dataset that was used is an imbalanced dataset for multiclass classification. It was created by Davidson et al. and it contains 24783 tweets that are classified into 3 classes or categories; hate tweets, offensive language tweets and neither hate nor offensive speech[4]. To begin the collection of the tweets, the researchers used a lexicon that was compiled by hatebase.org . The lexicon contained words and phrases that are identified as hate speech by internet users. They used the Twitter API to search for tweets using terms found in the lexicon and they gathered tweets from 33 458 users. The timeline of each twitter user was extracted resulting in a set of 85.4 million tweets. They extracted a random sample of 25k tweets that contained words found in the lexicon and they were manually coded by CrowdFlower (CF) workers. CF collects,

cleans and labels data. The CF workers labeled the tweets into three categories Hate, Offensive and Neither offensive nor hate speech and they did this using the definitions and further information provided by the researchers for each category. The labeled dataset was imbalanced and the distribution of tweets in the dataset is shown in Figure 3.

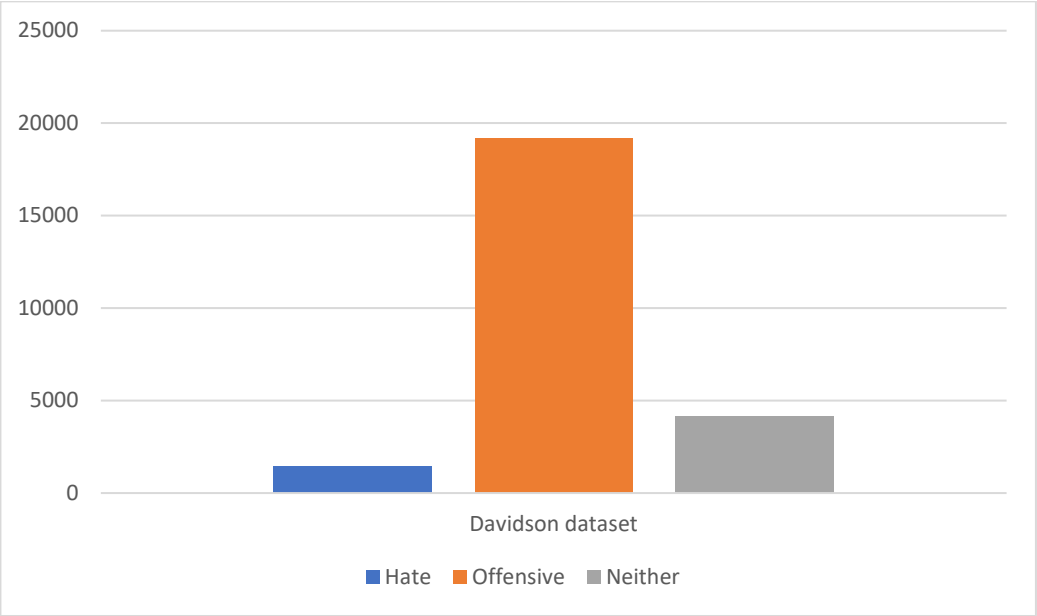

Figure 3: Distribution of tweets in the imbalanced Davidson dataset

2.5 Techniques to balance an imbalanced dataset

Different techniques were used to balance the dataset since dataset used was imbalanced. The techniques used were oversampling, hybrid oversampling and under sampling, class weights using sklearn and the proposed method discussed in Chapter 4. The techniques used except for the proposed method are explained in this section.

2.5.1 SMOTE

Resampling data is an approach that is used to deal with imbalanced datasets[16]. There is oversampling and under sampling. In this research the oversampling technique SMOTE was used[17]. SMOTE is a technique where the synthetic samples

are generated for the minority class. It is used to solve the issues of overfitting that can be caused by random oversampling. The way SMOTE works is that it chooses examples in the feature space that are close together. This is done when a line is drawn between the examples that are in the feature space and also a new sample is drawn at a point along that line. The steps of how the process works are shown below,

- Firstly, N is set up which is the total number of oversampling observations.
- Secondly, when the iteration starts, the first thing that happens is that a positive class instance is selected at random.
- Thirdly, the k-Nearest Neighbors (KNN) for that instance is obtained. KNN is 5 by default.
- Finally, N of the K instances is chosen to interpolate new synthetic instances. Any distance metric can be used and the difference in distance between the feature vector and its neighbors will be calculated. The obtained difference is then multiplied by any random value in (0,1] and will be added to the previous feature vector.

2.5.2 SMOTETomek

Under sampling and over sampling techniques can be combined to form a technique called hybridization. SMOTETomek is a hybrid technique of SMOTE and Tomek Links that targets to clean the data points that are overlapping for each of the classes that are distributed in sample space. After SMOTE oversamples the data, class clusters may be invading each other's space and that results in a classifier that will be overfitting. Tomek Links are class paired samples that opposite each other and are also the closest neighbors[18]. Most of the class observations are removed from the links because it is thought that they increase class separation that is close to the decision boundaries. Tomek Links are applied to the oversampled minority class samples done

by SMOTE in order to get better class clusters. Therefore, instead of only removing the observations from the majority class, both class observations are removed from the Tomek Links.

2.5.3 SMOTEENN

SMOTEENN is also a hybrid technique where a greater number of observations are removed from the sample space. Edited Nearest Neighbor (ENN) is an under-sampling technique where the nearest neighbors are estimated for each of the majority class[19]. If a particular instance of the majority class is misclassified by the nearest neighbor that instance will get deleted. Integrating ENN with the oversampled data obtained from using SMOTE helps in performing extensive data cleaning. The misclassification by ENN samples from both the classes will be removed and that will result in a clear and concise class separation.

2.5.4 ADASYN

ADASYN is a form of the SMOTE algorithm that is generalized[20]. It targets to oversample the minority class by generating synthetic instances for it. The main difference between the two is ADASYN takes into consideration the density distribution that determines the number of synthetic instances that are generated for samples which are difficult to learn. Because of this difference ADASYN will be able to help in adaptively changing the decision boundaries that are based on the samples that are difficult to learn.

2.5.5 Class weights

To remove bias from machine learning algorithms when dealing with an imbalanced dataset, class weights can be added. Class weights are used by modifying the training algorithm to be able take into consideration the skewed distribution of classes. This is done by giving weights to the majority and the minority classes. In the cost function

of the algorithm during training more weight is given to the minority class so that it could provide a penalty that is higher to the minority class and then the algorithm can focus on reducing errors in the minority class. An in-built parameter class-weight from sklearn was used in this research to help optimize the scoring of the minority class. Class weight='balanced' was used and during this assignment the model automatically assigns class weights that are inversely proportional to their respective frequencies. The formula to calculate this is written below:

$$wj = n\_samples \; / \; (n\_samples * n\_samples) \quad (2)$$

where;

- wj represents the weight for each class (j signifies the class)
- n_samples represent the total number of samples or rows in the dataset
- n_classes represent the total number of unique classes in the target
- n_samplesj represents the total number of rows of the respective class

2.6 Evaluation metrics used in hate speech detection

The performance of each classifier is evaluated using the metrics that were deemed fitting for multiclass classification by Grandini et al.[21]. The metrics that are used are Macro F1-score, Accuracy, Macro Recall, Macro Precision and the Cohen Kappa score. The metrics are calculated based on the confusion matrices and an illustration of confusion matrix for multiclass classification used in the study is shown in Figure 4.

True class	Predicted class			
		Hate	Offensive	Neither
	Hate	TP_{Hate}	$E_{OffensiveHate}$	$E_{NeitherHate}$
	Offensive	$E_{HateOffensive}$	$TP_{Offensive}$	$E_{NeitherOffensive}$
	Neither	$E_{HateNeither}$	$E_{OffensiveNeither}$	$TP_{Neither}$

Figure 4: Multiclass confusion matrix

TP represents the number of true positives in a class which means the number of samples that were classified correctly in that class. For example, TP_{Hate} shows the tweets that were classified correctly in the hate class. False negatives are samples from a class that were incorrectly classified as another class i.e., $E_{HateOffensive}$ are hate tweets that were misclassified as offensive and E_{HN} are hate tweets that were misclassified as neither so the false negatives for the hate class will be calculated as follows: $FN_{Hate} = E_{HateOffensive} + E_{HateNeither}$. The false positives for the hate class are calculated using the formula:

$$FP_{Hate} = E_{OffensiveHate} + E_{NeitherHate} \tag{3}$$

Macro averages of f1, recall and precision score was used. Using the information from the confusion matrix the other metrics can be defined below where k =class.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4}$$

$$Precision_k = \frac{TP_k}{TP_k + FP_k} \tag{5}$$

$$Recall_k = \frac{TP_k}{TP_k + FN_k} \tag{6}$$

Macro averages for precision and recall are calculated using the arithmetic mean of the metrics for individual classes and macro F1-score is the harmonic mean of macro precision and macro recall.

$$Macro\ Precision = \frac{\sum_{k=1}^{K} Precision_k}{K} \quad (7)$$

$$Macro\ Recall = \frac{\sum_{k=1}^{K} Recall_k}{K} \quad (8)$$

$$Macro\ F1 = 2 * (\frac{Macro\ Precision * Macro\ Recall}{Macro\ Precision + Macro\ Recall}) \quad (9)$$

The Cohen's Kappa statistics is a metrics that is used for multiclass classification problems to show the performance of the classifier. The metrics shows how your classifier is actually performing over the performance of the classifier that guesses at depending on the frequency of the class. This metrics will be used to compare the performance of the different classifiers used in this research in order to find the best one for hate speech detection. The score is calculated as follows:

$$K = \frac{c * s - \sum_{k}^{K} p_k * t_k}{s^2 - \sum_{k}^{K} p_k * t_k} \quad (10)$$

Where:

- c=$\sum_{k}^{K} C_{kk}$ is the total number elements predicted correctly
- s= $\sum_{i}^{K} \sum_{j}^{K} C_{ij}$ is the total number of the elements
- $p_k = \sum_{i}^{K} C_{ki}$ is the number of times the class k was predicted (total of column)
- $t_k = \sum_{i}^{K} C_{ik}$ is the number of times the class k truly occurs (total of row)

The Cohen kappa scores are interpreted as follows:

- If Value ≤ 0 there is no way to interpret any results so the classifier is useless.
- If 0> Value ≤0.20 it means there is slight agreement
- If 0.21≥Value≤0.40 it means fair
- If 0.41≥Value≤0.60 it means moderate
- If 0.61≥Value≤0.80 it means substantial
- If 0.81≥Value≤1 it means almost perfect agreement.

Chapter 3

LITERATURE REVIEW

3.1 Introduction

With the increase of the use of hate speech, many researchers have been interested in this field in order to help find solutions to this problem. This section will present recent findings in hate speech detection in two sections, hate speech detection using ML algorithms in Section 3.2 and hate speech detection using NN in Section 3.3. The dataset information about the papers reviewed and the datasets used in this research is presented in Table 1 and Table 2.

Table 1: Information about the total number of tweets in hate speech datasets

Dataset	**Total number of tweets**
Davidson	24783
First Dataset	14000
Second Dataset	16914
Watanabe et al. tertiary dataset	25020
Watanabe et al. binary dataset	25020
Oriola and Kotze	15707
Putri et al.	4002
Kaggle hate speech training dataset	31962
Kaggle hate speech test dataset	45159
Putra and Nurjanah	13194

Table 2: Information about the total number of tweets in the classes of hate speech datasets

Dataset	**Number of tweets in each class**
Davidson	Hate: 1430
	Offensive: 19190
	Neither: 4163
Second Dataset	Sexist: 3383
	Racist: 1972
	Neither: 11559
Watanabe et al. tertiary dataset	Offensive: 8340
	Clean: 8340
	Hateful: 8340
Watanabe et al. binary dataset	Offensive: 16680
	Clean: 8340
Putri et al.	Hate: 2776
	Not hate: 1226
Kaggle hate speech training dataset	Hate: 2242
	Non-hate: 29720
Putra and Nurjanah	Hate text: 1018
	Not Hate text: 12176

3.2 Detecting hate speech using machine learning algorithms

Davidson et al. compiled a dataset for hate speech detection and trained five algorithms LR, Naïve Bayes (NB), Decision Tree (DT), Random Forest (RF) and SVM using 5-fold cross validation[4]. The researchers created the dataset that is known as the Davidson dataset and the information about the dataset is presented in Table 1 and Table 2. The dataset is imbalanced and it contains three classes; hate, offensive and neither hate nor offensive speech. Preprocessing was performed on the tweets as each tweet was converted to lowercase and all words were stemmed. TF-IDF features and Sentiment Scores were used during the training of the models and the number of retweets, mention tags, URL's, hashtags and also features for the number of characters and words were included for each tweet. When the classifiers were trained and tested their results showed that the LR and the SVM models performed better than the rest.

The researchers proposed a final model based on a LR model that used L2 regulations addressing over fitting and feature selection. Their model had an F1-score and Recall score of 0.90 and a Precision score of 0.91. The model misclassified 40% of hate speech tweets.

An approach based on using unigrams, patterns, semantic and sentiment features combined together was used by Watanabe et al. to perform binary and tertiary classification to detect hate speech[22]. Three datasets were combined together to create a bigger balanced dataset that can be used for tertiary classification. The datasets used were the Davidson dataset, the First Dataset and the Second Dataset presented in Table 1 and the tweets were classified into three categories hateful, offensive and clean. The number of tweets in each class of the Watanabe et al. tertiary dataset are presented in Table 2 . In their tertiary dataset 21000 tweets were used for training with 7000 tweets in each class and the test and validation sets had 2010 tweets with 670 tweets in each class. The categories in their binary dataset were offensive and clean and the number of tweets in each class are shown in Table 2. Using their dataset for binary classification, 14000 from the offensive class and 7000 tweets from the clean class were used during training. The test set had 2680 tweets in the offensive class and 1340 tweets in the clean class. In order to perform classification the Weka toolkit was used[23]. Three classifiers namely SVM, RF and the J48graft[24] were employed. The J48graft performed better than the rest with 87.4% accuracy for binary classification and 78.4% accuracy score for tertiary classification.

Oriola and Kotze evaluated four machine learning techniques, namely LR, SVM, RF and Gradient Boosting, for hate and offensive speech detection in South African

tweets[25]. The dataset employed by the researchers was a multilingual dataset that is presented in Table 1. The dataset was split into three parts. The first part had 7100 tweets which were in English and English with Afrikaans, the second part had 4500 English and English with IsiZulu tweets and the last part had 4102 English and English with Sesotho tweets. The features used for training were word n-grams, char n-grams, syntactic based features and negative sentiment-based features. To improve the machine learning models the researchers applied hyper parameter optimization, ensemble and multi-tier learning on the classifiers. They split the dataset into 75% for training and 25% for testing. The classifiers were trained using10-fold cross validation. Since the dataset was imbalanced, synthetic minority oversampling technique (SMOTE) was employed[17] to balance the dataset. The results showed that for hate speech detection, the optimized SVM algorithm trained with Character n-grams features recorded the best TP rate of 0.894 for hate speech with an Accuracy score of 0.646.

A comparison of the performance of classification algorithms in hate speech detection was done by Putri et al.[26] using an imbalanced dataset presented in Table 1 and Table 2. The algorithms compared were NB, DT, AdaBoost, SVM and Multi Level Perceptron (MLP) and unigrams were used as features during training. In order to combat the problems that arise using an imbalanced dataset they used the oversampling technique SMOTE in order to balance their dataset. During their experiments, they made comparisons of the performance of the algorithms with SMOTE and without SMOTE and they found out that they were getting a higher accuracy score using SMOTE and a higher recall score without SMOTE. Their best algorithm using SMOTE technique was MLP which had an Accuracy of 83.4% and a Recall of 75.9%

and their best algorithm without SMOTE was NB which had an Accuracy of 72.1% and a Recall score of 93.2%. Putri et al. preferred to evaluate classification models using their recall score so based on that their best classification model was NB without SMOTE which had a Recall score of 93.2%.

Martins et al. used emotional analysis for hate speech detection where lexicon based and machine learning approaches were combined to predict hate speech through semantic analysis[27]. They applied a hateful emotional model on the Davidson dataset[4] presented in Table 1 and Table 2 to identify the emotions in tweets. The next step they did was to create a new dataset from the preprocessed Davidson dataset that contained 975 tweets extracted from each category of the dataset hate, offensive and neither. The Weka toolkit[23] was used to perform text classification and the algorithms used were SVM, RF and NB. The classifiers were trained using 10-fold cross validation and default parameters. Martin et al. used the results that Davidson et al. obtained to compare the performance of their classifier with the Davidson et al. classifier. The Davidson et al. classifier had a Recall score of 0.61 with a Precision score of 0.41 for the hate class[4]. Their results showed that their RF classifier had a better Precision score compared to Davidson et al. classifier and their SVM had the highest Accuracy score of 80.56%. Comparing their results, they noticed that there was growth in the Precision rate for the hate class, the Precision rate increased from 41% in Davidson et al. research to 80.64%.

A logistic regression classifier was used in[28] to classify tweets in two categories: hateful or not hateful. The dataset used was hate speech twitter dataset from Kaggle and it was split into training and testing sets. The information about the datasets used

is presented in Table 1. The preprocessing steps performed were the removal of twitter handles, punctuation, numbers and special characters. Tokenization and stemming were also performed. Bag of words and TF-IDF features were extracted after preprocessing and they were used to a LR classifier. Bag of words is the representation of text that shows the frequency of words in a document. It just keeps track of word count and it doesn't consider the order of words or grammar. They used TF-IDF features because they consider the importance of words. When the model was trained and tested using TF-IDF features an Accuracy score of 94.62% was obtained and an Accuracy of 94.11% was obtained using bag of words features.

Rathpisey and Adji conducted research on how to handle the imbalance issue in the classification of hate speech using sampling based methods[29] on the Davidson dataset presented in Table 1 and Table 2. They performed preprocessing on the tweets and the steps they took were removing unnecessary special characters, stemming, lower casing, tokenization and normalization. The sampling methods they used on the imbalanced dataset were Random Oversampling (ROS), Random Under Sampling (RUS), SMOTE and ADASYN. The classification algorithms used were LR, SVM and NB. All algorithms were trained using 5-fold cross validation and the best results were obtained using LR with ROS which gave an Accuracy of 91% and a F1-score of 0.95. They also observed that the Accuracy of all algorithms improved when all the oversampling techniques were used but the Accuracy of NB is the only one that improved when RUS was used.

3.3 Detecting hate speech using neural networks

Six models were used by Raj et al. to detect hate speech in three languages: English, German and Hindi[30]. The models used were a one layered and a two layered CNN

model, a one and a two layered BiLSTM and a hybrid model of CNN and LSTM: CNN+BiLSTM. Two types of embeddings, Glove and FastText were used to convert the datasets into vectors of numbers to be used for training. During the preprocessing step, text was converted to lowercase, URL, punctuation, retweet symbols and stop words were removed. The twitter retweet symbol (RT) was also removed together with stop words. Tokenization was performed and a vocabulary of tokens was created followed by encoding. In order to ensure that all the preprocessed tweets had the same they used pad sequencing and assigned a fixed length of 100. The datasets they used for all the languages were highly imbalanced so in order to balance the datasets they used the SMOTE and ADASYN technique. They split their experiments into sub tasks where "sub-task A" showed the results without a data balancing technique and "sub-task B" showed the results with a data balancing technique implemented. For the English dataset the best results for "sub-task A" was a macro F1-score of 0.86 and the model used was CNN+GloVe embeddings and "sub-task B" the best results was a macro F1-score of 0.54 obtained using CNN+GloVe+ADASYN.

For the German dataset the best results for "sub-task A" was a macro F1-score of 0.75 obtained using CNN+Fasttext embeddings and "sub-task B" had a macro F1-score of 0.45 obtained using CNN+FastText+ADASYN. For the Hindi dataset the best results for "sub-task A" were obtained using BiLSTM+FastText with a macro F1-score of 0.69 and a macro F1-score of 0.36 for "sub-task B" using CNN+FastText+ADASYN. Their results showed that adding a data balancing technique reduced the F1-score for all the datasets but those results were still higher than using the models with the imbalanced dataset.

Paul and Bora implemented LSTM and BiLSTM for the detection of hate speech on a twitter dataset[31]. They used the Kaggle hate speech training dataset which is presented in Table 1 and Table 2. During preprocessing special characters were removed, the text was converted to lower case and stemming was performed. Moreover, padding was performed to make the sentences the same length. Since the labels used for prediction hate or non-hate are string values, they used one hot encoding to assign the labels binary values. The data was split at a ratio of 67:33 for training and testing and word embeddings were used during training. Since the dataset was highly imbalanced, up-sampling was performed in order to make sure the hate class was equal to the non-hate class. This was done by randomly selecting from the hate class and adding back to the dataset and this ensured that the researchers had a balanced dataset. When the performance of the classifiers was tested the LSTM performed better that the BiLSTM and it had a Precision score of 0.9508, Recall score of 0.9986 and a F1-score of 0.9785.

Alshalan and Al-Khalifa investigated 3 NN models namely GRU, CNN and a hybrid model CNN+GRU that was based on both neural networks to investigate hate speech in Arabic tweets[32]. Furthermore, they investigated the transformer model BERT. They created a dataset with 9316 tweets that had three classes hateful, abusive and normal. Since they were performing binary classification which consists of 2 classes, they used only the tweet classified as hateful and abusive. Therefore, the tweets they used were 8964 which comprised of hate tweets and abusive tweets. Preprocessing was performed on the dataset and the dataset was split into 75% for training and 25% for testing. They did hyper parameter tuning and used cross validation on the training data. A pretrained word2vec model that uses continuous bag of words was used for

training. Their results revealed that the CNN model performed the best with an F1-score of 0.79 and an Area Under the Receiver Operating Characteristics (AUROC) of 0.89. In order to check the effectiveness of their models they trained their models using the complete dataset and tested the models using a different dataset that contained a total of 600 tweets. When these tests were done, they found that the CNN model was still the performing better the rest with an F1-score 0.69 and a AUROC of 0.79.

Research was conducted by Putra and Nurjanah on how to detect hate speech in Instagram comments using a TextCNN model that was modified and they used word2vec with skip-gram models[33]. This research was conducted on Indonesian comments. The dataset they used was created by crawling using hashtags that contained hate speech and they also used an Instagram scrapper. They stored the dataset in JSON file and used an algorithm to transfer the dataset to tables. After transferring the dataset, they performed preprocessing in which they removed all punctuation, numbers, emojis, symbols, duplicate comments and letters that are not found in the alphabet. They split their dataset into 80% training data and 20% testing data. The training data was labeled by expert annotators into two classes hate and not hate. The information about their dataset is presented in Table 1 and Table 2. In order to combat the problem of data imbalances they used two techniques to balance the dataset, ROS and RUS. Putra and Nurjanah used a word2vec together with skip-gram models so that they could represent the words that are rarely found in documents and to also get the context of the sentences.

They also hypothesized that accuracy can be affected by word2vec windows and applied three word2vec windows with sizes 5, 10 and 15 and observed the results.

They ran five experiments on the different word2vec window sizes, TextCNN using imbalanced data, TextCNN using 10-fold cross validation, TextCNN using ROS, TextCNN using RUS and TextCNN using class weights. The TextCNN model used in their experiments was their own modified version. They obtained their best results using the modified TextCNN with the word2vec skip-gram model with a window size of 15 the data balancing technique ROS which gave them an F1-score of 93.70%.

Chapter 4

PROPOSED ARCHITECTURE

4.1 Introduction

In this chapter the process of how the dataset was balanced using the proposed technique named: Balanced Random Oversampling Undersampling Technique (BROUT) is explained in Section 4.2. All the steps that were taken to perform hate speech detection and get the results are explained in section. Furthermore, the information about the number of tweets that were used during training for all classifiers after the data balancing techniques explained in Chapter 2 and BROUT were implemented is explained in section.

4.2 Proposed method for balancing the dataset

Table 3: Comparison of the number of tweets in imbalanced Davidson dataset and the BROUT dataset

Dataset	**Total number of tweets**	**Number of tweets in Hate class**	**Number of tweets in Offensive class**	**Number of tweets in Neither class**
Imbalanced Davidson dataset	24783	1430	19190	4163
BROUT dataset	35929	11440	12000	12489

The proposed method used to balance the imbalanced dataset BROUT works by creating a new dataset by resampling the imbalanced dataset. The resampling was performed by applying both over sampling and under sampling on the imbalanced

dataset. The minority classes which are the Hate class and the Neither class were over sampled and this was done by duplicating the tweets in the Hate class eight times and duplicating the tweets in the Neither class three times. Since the Offensive class was the majority class, under sampling was performed and 12000 tweets were extracted from the Offensive class in order to get the number of tweets in that class closer to the number of tweets in the other classes. The number of tweets found in the Hate class, Offensive class and Neither class of the balanced dataset using BROUT are recorded in Table 3. After balancing the tweets in each class, the tweets were then combined to create the proposed dataset that had a total of 35929 tweets. Random shuffling was then performed prior to the use of the dataset resampled using the proposed technique which will be referred to as the BROUT dataset. The visual distribution of tweets in the imbalanced Davidson dataset and the BROUT dataset is shown in Figure 5.

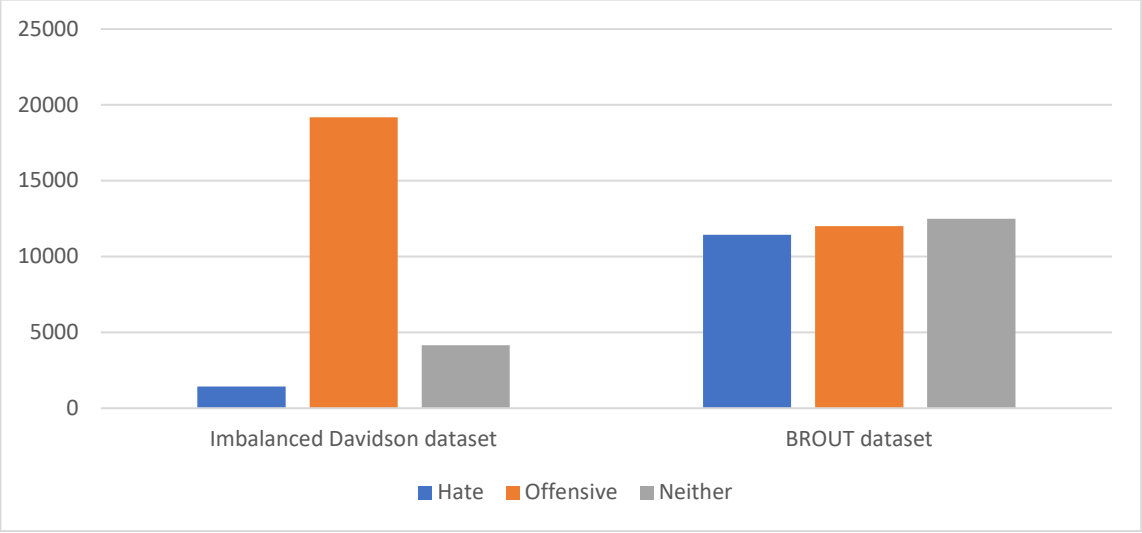

Figure 5: Tweets in imbalanced Davidson dataset vs the BROUT dataset

The procedure for how BROUT was implemented is shown in Figure 6.

BROUT PROCEDURE

1: D_T = the number total number that the minority class should not exceed

2: B_D = Balanced Dataset

3: **FOR** each class in dataset **do**

4: Check the number of tweets in each class

5: **IF** $Class_1 > Class_2$ **then**

6: Perform undersampling by extracting desired number of tweets from $Class_1$

7: **REPEAT** {
Perform oversampling by Duplicating $Class_2$ }
UNTIL $(Class_1 - D_T \geq Class_2 <= Class_1 + D_T)$

8: $B_D = Class_1 \cup Class_2$

9: Shuffle B_D

10: **END IF**

11: **END FOR**

Figure 6: BROUT algorithm

4.3 Method for performing classification

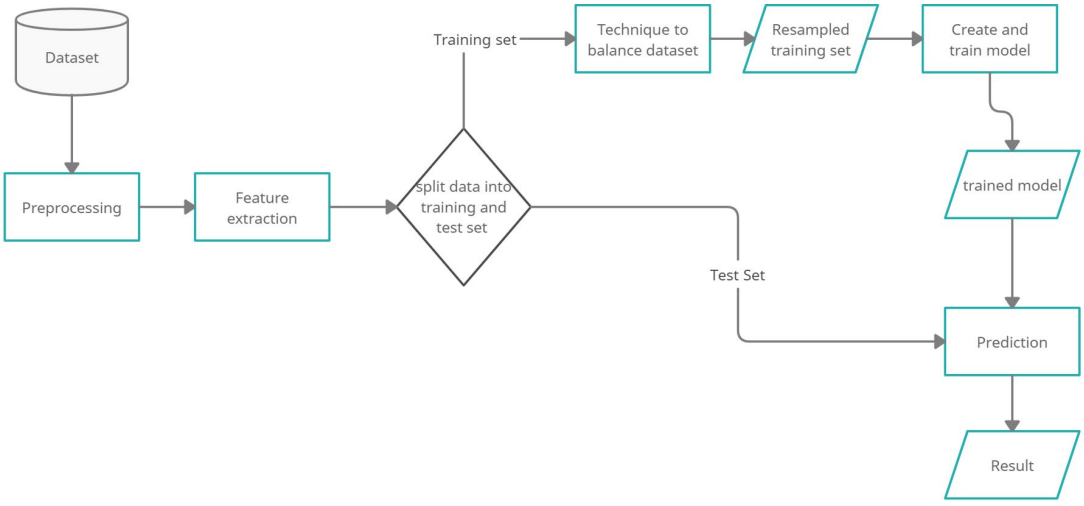

Figure 7: Experimental pipeline used for hate speech detection

All the steps that were taken to perform classification are shown in Figure 7. The first step that was performed after the dataset has been collected is preprocessing.

Preprocessing refers to the manipulation of data that is done before the dataset is used in order to increase the efficiency of the classification models by removing noise and making sure the dataset is in a format that the models will be able to use. Examples of what the tweets look like before preprocessing is performed are given in Figure 8.

Example 1: !!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. & as a man you should always take the trash out...

Example 2: you's a muthaf***in lie “@LifeAsKing: @20_Pearls @corey_emanuel right! His TL is trash …. Now, mine? Bible scriptures and hymns”

Figure 8: Examples of tweets before preprocessing

In order to remove hashtags, handles, URLs and numbers from the tweets regular expressions (Regex) was used. Regex is a sequence of characters that is used for manipulating text. All the tweets were converted to lowercase using the Python String lower () method and punctuation was removed from the tweets using the Python String translate () method. In order to split the tweets into tokens the word_tokenize function from the NLTK library was used. The tokens are used to help understand the context in developing models in natural language processing. Tokenization also helps to interpret the meaning of text by analyzing the words sequence.

Stop words are words that are used frequently and do not add meaning to the sentence. Removing stop words usually results in to reduced noise and the dimension of feature sets in the vocabulary. In order to remove the Retweet (RT) from the tweets, “rt” was added as a stop word to the NLTK stop words corpus and it was removed together with the other stop words. Lemmatization which is the process of converting words to their

meaningful base forms was also performed on the tweets. Figure 9 shows examples of what the tweets look like after preprocessing has been performed.

Example 1: alf legit retard

Example 2: people live thenetherlands unwashed trash

Figure 9: Example of tweets after preprocessing

The second step feature extraction, was performed and the features used for machine learning algorithms were TF-IDF features, Sentiment scores and Doc2vec scores. FastText embeddings were used for neural networks. The dataset is then split into 70% for training and 30% for testing. The techniques to balance the dataset are then applied on the training set only and the training set will be resampled. Table 4 shows the number of tweets in each class after each technique to balance the dataset has been applied to the training set. The number of tweets in each class before any technique is used are also recorded in Table 4 together with the number of tweets in each class during training when using the proposed method.

Table 4: Distribution of tweets after applying data balancing technique

Technique	**Class 0 (hate)**	**Class 1 (offensive)**	**Class 2 (neither)**
Original	1003	13443	2902
Class weights	1003	13443	2902
SMOTE	13443	13443	13443
SMOTETomek	13214	12944	13097
SMOTEENN	11554	5108	10687
ADASYN	13652	13443	14007
BROUT	7977	8395	8706

After the training dataset has been resampled, the classification models are then created and they are trained using the resampled training set. The test set is then used the trained model to make predictions and give the classification results.

4.4 Experimental setup

Python programming language was used on the interactive programming environment Google Colaboratory. The python version used is Python 3.7. Python machine learning libraries were used for the traditional machine learning algorithms and Python neural network libraries were used for all neural networks. In order to extract features for machine learning algorithms inputs Sckit-Learn[34] was used. The training of neural networks was performed using Keras library[35] with TensorFlow[36].

4.4.1 Basic parameters for neural networks

Table 5: Neural Networks basic parameters

Parameter	Result
Embedding size	300
Number of unique words (max_features)	19479
Maximum number of words in each tweet (max_len)	55
Batch size	128
Number of epochs	20
Number of K-fold splits	5
Dense units' activation	Rectified Linear Unit (ReLU)
Output neuron activation	Softmax
Optimizer	Adam
loss	Categorical cross entropy
Learning rate	0.001
Test set	20%
Validation set	10%
Call backs	Early stopping • Monitor= validation loss • Patience= 3 • Min_delta= 0.0001

4.4.2 Model configurations

Different hyperparameters were used for the classifiers and the configuration that brought about the best performance was chosen for each of the classifiers. For LR, the

one vs rest scheme was used with a *saga* solver and *l2* regularization. *One vs rest* was also used for SVM with *l2* regularization and a maximum iteration, *max_iter* of 100. The criterion *gini* was used for RF with *n_estimator* of 500 and *max_features* set to auto. Default parameters were used for DT.

For CNN, an embedding layer was added followed by a spatial dropout layer of 0.2. A conv1D layer with 100 filters with a kernel size of 4 was added. Batch normalization, global max pooling and a dropout layer of 0.2 were added respectively. A dense layer with 50 units was added. An embedding layer was first added for GRU followed by a GRU layer with 128 units with a dropout of 0.2. A dense layer with 64 units was added. The embedding layer was also added for LSTM and BiLSTM followed by a spatial dropout layer of 0.5. For LSTM, a LSTM layer with 200 units with a dropout of 0.5 was added and in BiLSTM a bidirectional layer with the same units was added. A final dense layer of 64 units was added for both LSTM and BiLSTM.

For CNNBiLSTM, an embedding layer was added followed by a spatial dropout layer of 0.5. A building block was constructed which was used twice. The building block included a conv1D layer with 100 filters with a kernel size of 4 followed by a leaky Relu layer with an alpha of 0.2. A max pool layer was added followed by a bidirectional layer with 200 LSTM units. In depth explanations for the layers used in for RNN is provided in[37] and CNN layers information is provided in [38], [39].

Chapter 5

RESULTS AND DISCUSSION

5.1 Introduction

This chapter shows the results of the classifiers trained using the data balancing techniques explained in Chapter 2 and BROUT explained in Chapter 4. The results for hate speech detection performed using ML algorithms LR, DT, RF and SVM are presented in Section 5.2 and the comparison of the performances of the ML algorithms is in Section 5.3. The results for the NN used GRU, LSTM, BiLSTM, CNN and CNNBiLSTM are recorded in Table 27. The analysis of the NN results is done in two parts.

The first part Section 5.4.1 gives a detailed analysis of how the data balancing techniques explained in Chapter 2 affected the performance of the classifiers for hate speech detection. The second part Section 5.4.2 provides a detailed analysis of the performance of the NN trained using BROUT. A comparison of the best performing classifier in this research was made against a state of art in Section 5.5 to be able to check if the best classifier trained using BROUT improved the process of hate speech detection. Finally, a comparison of the performances of ML algorithms against NN is performed in Section 5.6.

5.2 Classification Results for ML algorithm

5.2.1 Classification results for LR

Table 6: LR results using no data balancing technique

Features	Accuracy	F1-score	Recall	Precision	Cohen Kappa
TF-IDF	0.88	0.68	0.66	0.70	0.66
SS	0.77	0.36	0.37	0.45	0.12
Doc2vec	0.78	0.33	0.35	0.45	0.00
TF-IDF + Sentiment Scores	0.85	0.54	0.52	0.68	0.50
TF-IDF+ Doc2vec	0.88	0.67	0.66	0.69	0.66
Sentiment Scores+ Doc2vec	0.79	0.41	0.40	0.48	0.23
ALL features	0.85	0.55	0.53	0.69	0.57

Table 7: LR results using class weights

Features	Accuracy	F1-score	Recall	Precision	Cohen Kappa
TF-IDF	0.81	0.67	0.75	0.75	0.64
Sentiment Scores	0.77	0.36	0.37	0.45	0.12
Doc2vec	0.78	0.33	0.35	0.45	0.07
TF-IDF + Sentiment Scores	0.80	0.67	0.77	0.63	0.57
TF-IDF+ Doc2vec	0.85	0.68	0.71	0.66	0.63
Sentiment Scores + Doc2vec	0.48	0.40	0.53	0.44	0.19
ALL features	0.80	0.67	0.77	0.63	0.57

Table 8: LR results using SMOTE, SMOTETomek, SMOTEENN and ADASYN

Features	Accuracy	F1-score	Recall	Precision	Cohen Kappa
TF-IDF	0.88	0.68	0.66	0.70	0.66
Sentiment Scores	0.77	0.36	0.37	0.45	0.12
Doc2vec	0.78	0.33	0.35	0.46	0.00
TF-IDF + Sentiment Scores	0.85	0.54	0.52	0.68	0.50
TF-IDF+ Doc2vec	0.88	0.67	0.66	0.70	0.66

Table 9: LR results using SMOTE, SMOTETomek, SMOTEENN and ADASYN (cont.)

Features	Accuracy	F1-score	Recall	Precision	Cohen Kappa
Sentiment Scores + Doc2vec	0.79	0.41	0.41	0.48	0.23
ALL features	0.86	0.55	0.53	0.69	0.57

Table 10: LR results using BROUT

Features	Accuracy	F1-score	Recall	Precision	Cohen Kappa
TF-IDF	0.95	0.95	0.95	0.95	0.93
Sentiment Scores	0.54	0.52	0.53	0.53	0.30
Doc2vec	0.47	0.47	0.47	0.47	0.21
TF-IDF + Sentiment Scores	0.80	0.79	0,79	0.80	0.70
TF-IDF + Doc2vec	0.95	0.95	0.95	0.95	0.93
Sentiment Scores + Doc2vec	0.56	0.54	0.53	0.55	0.33
All Features	0.80	0.80	0.80	0.80	0.71

Comparing the results in Table 5, Table 6, Table 7 and Table8 shows that LR had the performance when the dataset was balanced using BROUT and the TF-IDF and TF-IDF+ Doc2vec features. SMOTE, SMOTETomek, SMOTEENN and ADASYN all produced the same results when they were used to balance the training data and their results are recorded in Table 7. Using class weights to balance the dataset proved to be a better technique compared to the resampling techniques SMOTE, SMOTETomek, SMOTEENN and ADASYN. The Cohen Kappa score of less than 0.2 recorded when sentiment scores and Doc2vec features were used indicated that LR was unable to learn during training which led to it not being able to classify the tweets.

5.2.2 Classification results for SVM

Table 11: SVM results using no data balancing technique

Features	Accuracy	F1-score	Recall	Precision	Cohen Kappa
TF-IDF	0.89	0.68	0.66	0.76	0.71
Sentiment Scores	0.77	0.29	0.33	0.25	0.00
Doc2vec	0.77	0.29	0.33	0.25	0.00
TF-IDF + Sentiment Scores	0.89	0.61	0.66	0.75	0.71
TF-IDF+ Doc2vec	0.89	0.68	0.66	0.66	0.71
Sentiment Scores + Doc2vec	0.75	0.30	0.33	0.33	0.00
ALL features	0.89	0.65	0.64	0.77	0.70

Table 12: SVM results using class weights

Features	Accuracy	F1-score	Recall	Precision	Cohen Kappa
TF-IDF	0.88	0.73	0.76	0.71	0.71
Sentiment Scores	0.71	0.43	0.49	0.41	0.31
Doc2vec	0.76	0.31	0.34	0.48	0.0
TF-IDF + Sentiment Scores	0.88	0.74	0.78	0.72	0.72
TF-IDF+ Doc2vec	0.89	0.74	0.78	0.72	0.72
Sentiment Scores + Doc2vec	0.73	0.44	0.49	0.42	0.33
ALL features	0.88	0.74	0.78	0.70	0.70

Table 13: SVM results using SMOTE, SMOTETomek, SMOTEENN and ADASYN

Features	Accuracy	F1-score	Recall	Precision	Cohen Kappa
TF-IDF	0.89	0.68	0.67	0.74	0.70
Sentiment Scores	0.77	0.29	0.33	0.25	0.00
Doc2vec	0.72	0.29	0.33	0.25	0.00
TF-IDF + Sentiment Scores	0.89	0.69	0.67	0.75	0.71
TF-IDF + Doc2vec	0.89	0.67	0.66	0.75	0.70
Sentiment Scores + Doc2vec	0.77	0.29	0.33	0.45	0.00
All Features	0.90	0.68	0.67	0.76	0.71

Table 14: SVM results using BROUT

Features	Accuracy	F1-score	Recall	Precision	Cohen Kappa
TF-IDF	0.93	0.93	0.93	0.93	0.89
Sentiment Scores	0.53	0.49	0.52	0.53	0.29
Doc2vec	0.44	0.40	0.43	0.44	0.16
TF-IDF + Sentiment Scores	0.93	0.92	0.93	0.93	0.88
TF-IDF + Doc2vec	0.93	0.93	0.97	0.99	0.84
Sentiment Scores + Doc2vec	0.56	0.53	0.56	0.56	0.34
All Features	0.92	0.92	0.92	0.92	0.88

The Results in Table 9, Table 10, Table 11, Table 12 show SVM had the best performance when the dataset was balanced using BROUT. It had the had the highest F1-score of 0.93 when it was trained using TF-IDF features and TF-IDF+Doc2vec features but the classifier was performing better using just TF-IDF features shown in Table 12, as the Cohen Kappa score of SVM using TF-IDF was 0.89 and the Cohen Kappa score of SVM using TF-IDF+Doc2vec features was 0.84.

5.2.3 Classification results for DT

Table 15: DT results using no data balancing technique

Features	Accuracy	F1-score	Recall	Precision	Cohen Kappa
TF-IDF	0.88	0.68	0.68	0.69	0.67
Sentiment Scores	0.71	0.41	0.41	0.42	0.13
Doc2vec	0.64	0.36	0.36	0.36	0.06
TF-IDF + Sentiment Scores	0.88	0.68	0.68	0.68	0.66
TF-IDF + Doc2vec	0.88	0.68	0.68	0.68	0.66
Sentiment Scores + Doc2vec	0.70	0.43	0.43	0.43	0.20
All Features	0.87	0.68	0.68	0.67	0.66

When DT was trained and tested using the imbalanced data as shown in Table 15 the same F1-scores were recorded when TF-IDF, and Scores and TF-IDF+Doc2vec features were used but the best performance of DT according to the Cohen Kappa score was when TF-IDF features were used.

Table 16: DT results using class weights

Features	**Accuracy**	**F1-score**	**Recall**	**Precision**	**Cohen Kappa**
TF-IDF	0.86	0.71	0.74	0.69	0.66
Sentiment Scores	0.66	0.44	0.47	0.43	0.22
Doc2vec	0.64	0.36	0.36	0.36	0.03
TF-IDF + Sentiment Scores	0.86	0.69	0.72	0.66	0.64
TF-IDF + Doc2vec	0.86	0.69	0.71	0.67	0.64
Sentiment Scores + Doc2vec	0.68	0.42	0.42	0.42	0.18
All Features	0.86	0.70	0.72	0.68	0.65

When DT was trained and tested using class weights as shown in Table 16, it performed best when TF-IDF features according to F1-scores and the Cohen Kappa score.

Table 17: DT results using SMOTE

Features	**Accuracy**	**F1-score**	**Recall**	**Precision**	**Cohen Kappa**
TF-IDF	0.88	0.69	0.69	0.69	0.68
Sentiment Scores	0.70	0.40	0.40	0.42	0.13
Doc2vec	0.63	0.36	0.36	0.36	0.05
TF-IDF + Sentiment Scores	0.86	0.69	0.72	0.67	0.64
TF-IDF + Doc2vec	0.87	0.67	0.67	0.68	0.66
Sentiment Scores + Doc2vec	0.70	0.43	0.43	0.43	0.19
All Features	0.86	0.69	0.72	0.68	0.64

According to the results in Table 17, the best performance of DT when SMOTE was used to balance the dataset was recorded when TF-IDF. This is shown by the Cohen kappa scores. TF-IDF, TF-IDF+ Sentiment Scores and all features used all had the same F1-score but the Cohen Kappa scores showed DT performed best when TF-IDF features where used.

Table 18: DT using SMOTETomek

Features	**Accuracy**	**F1-score**	**Recall**	**Precision**	**Cohen Kappa**
TF-IDF	0.88	0.69	0.69	0.70	0.68
Sentiment Scores	0.71	0.41	0.40	0.42	0.13
Doc2vec	0.64	0.36	0.36	0.36	0.06
TF-IDF + Sentiment Scores	0.87	0.68	0.68	0.68	0.66
TF-IDF + Doc2vec	0.87	0.68	0.68	0.68	0.66
Sentiment Scores + Doc2vec	0.69	0.42	0.43	0.42	0.19
All Features	0.87	0.68	0.68	0.67	0.66

When SMOTETomek was used to balance the dataset in Table 18, DT had the best performance when TF-IDF features were used according to the F1-scores and Cohen Kappa scores.

Table 19: DT using SMOTEENN

Features	**Accuracy**	**F1-score**	**Recall**	**Precision**	**Cohen Kappa**
TF-IDF	0.88	0.69	0.69	0.69	0.69
Sentiment Scores	0.70	0.41	0.40	0.42	0.13
Doc2vec	0.64	0.36	0.37	0.36	0.07
TF-IDF + SS	0.87	0.68	0.68	0.68	0.66
TF-IDF + Doc2vec	0.87	0.67	0.68	0.66	0.65
Sentiment Scores + Doc2vec	0.79	0.42	0.42	0.51	0.25
All Features	0.88	0.68	0.68	0.68	0.67

Table 19 shows that the highest performance for DT when SMOTEENN was used was recorded when TF-IDF features were used according to F1-scores and Cohen Kappa scores.

Table 20: DT results using ADASYN

Features	**Accuracy**	**F1-score**	**Recall**	**Precision**	**Cohen Kappa**
TF-IDF	0.88	0.68	0.67	0.68	0.67
Sentiment Scores	0.71	0.41	0.40	0.42	0.13
Doc2vec	0.64	0.36	0.36	0.36	0.07
TF-IDF + Sentiment Scores	0.86	0.66	0.66	0.66	0.63
TF-IDF + Doc2vec	0.86	0.66	0.66	0.65	0.64
Sentiment Scores + Doc2vec	0.79	0.42	0.42	0.51	0.25
All Features	0.85	0.65	0.65	0.65	0.60

When ADASYN was used to balance dataset the highest performance of DT was recorded when TF-IDF features were used according to Accuracy, F1-scores and Cohen Kappa score.

Table 21: DT using BROUT

Features	**Accuracy**	**F1-score**	**Recall**	**Precision**	**Cohen Kappa**
TF-IDF	0.95	0.95	0.95	0.95	0.92
Sentiment Scores	0.81	0.81	0.81	0.83	0.71
Doc2vec	0.46	0.46	0.47	0.46	0.20
TF-IDF + SS	0.95	0.95	0.95	0.96	0.92
TF-IDF + Doc2vec	0.94	0.94	0.95	0.95	0.91
Sentiment Scores + Doc2vec	0.71	0.71	0.71	0.70	0.56
All Features	0.95	0.94	0.95	0.95	0.92

When BROUT was used to balance the dataset, DT had the best performance when TF-IDF, TF-IDF+ SS and all features were used.

5.2.4 Classification results for RF

Table 22: RF using no data balancing technique

Features	Accuracy	F1-score	Recall	Precision	Cohen Kappa
TF-IDF	0.90	0.70	0.69	0.74	0.73
Sentiment Scores	0.76	0.39	0.39	0.46	0.15
Doc2vec	0.78	0.35	0.36	0.55	0.10
TF-IDF + Sentiment Scores	0.89	0.65	0.63	0.74	0.68
TF-IDF + Doc2vec	0.90	0.65	0.64	0.76	0.70
Sentiment Scores + Doc2vec	0.79	0.44	0.43	0.57	0.27
All Features	0.89	0.63	0.61	0.75	0.67

When RF was trained and tested on the imbalanced dataset, it had the best performance when TF-IDF features according to the Accuracy scores, F1-scores and Cohen Kappa scores as shown in Table 22.

Table 23: RF using class weights

Features	Accuracy	F1-score	Recall	Precision	Cohen Kappa
TF-IDF	0.90	0.74	0.75	0.74	0.74
Sentiment Scores	0.72	0.45	0.46	0.46	0.26
Doc2vec	0.78	0.34	0.36	0.54	0.08
TF-IDF + Sentiment Scores	0.89	0.66	0.65	0.74	0.69
TF-IDF + Doc2vec	0.90	0.66	0.65	0.76	0.71
Sentiment Scores + Doc2vec	0.79	0.43	0.42	0.53	0.25
All Features	0.89	0.63	0.62	0.75	0.68

When class weights were used to balance the dataset, RF had the best performance when TF-IDF features as shown in Table 23.

Table 24: RF results using SMOTE, SMOTETomek, SMOTEENN

Features	**Accuracy**	**F1-score**	**Recall**	**Precision**	**Cohen Kappa**
TF-IDF	0.90	0.74	0.75	0.73	0.73
Sentiment Scores	0.76	0.39	0.38	0.45	0.15
Doc2vec	0.78	0.34	0.36	0.54	0.08
TF-IDF + Sentiment Scores	0.89	0.65	0.64	0.75	0.68
TF-IDF + Doc2vec	0.90	0.66	0.65	0.76	0.70
Sentiment Scores + Doc2vec	0.79	0.44	0.43	0.57	0.27
All Features	0.89	0.62	0.61	0.73	0.66

When SMOTE, SMOTETomek and ADASYN were used to balance the dataset they produced the same results. The best performance of RF when those techniques were used was recorded when TF-IDF features were used as shown in Table 24.

Table 25: RF results using ADASYN

Features	**Accuracy**	**F1-score**	**Recall**	**Precision**	**Cohen Kappa**
TF-IDF	0.90	0.74	0.75	0.73	0.73
Sentiment Scores	0.76	0.39	0.39	0.46	0.15
Doc2vec	0.78	0.36	0.37	0.47	0.12
TF-IDF + Sentiment Scores	0.89	0.64	0.62	0.75	0.68
TF-IDF + Doc2vec	0.90	0.67	0.66	0.73	0.70
Sentiment Scores + Doc2vec	0.79	0.42	0.41	0.52	0.24
All Features	0.90	0.70	0.69	0.74	0.73

When RF was trained using ADASYN as shown in Table 25, the highest F1-scores were recorded when TF-IDF features were used. The Cohen Kappa scores show that when TF-IDF features and all features were used RF was performing the same as they had the same Cohen Kappa score.

Table 26: RF results using BROUT

Features	**Accuracy**	**F1-score**	**Recall**	**Precision**	**Cohen Kappa**
TF-IDF	0.96	0.95	0.95	0.95	0.94
Sentiment Scores	0.83	0.83	0.83	0.85	0.74
Doc2vec	0.55	0.55	0.55	0.55	0.33
TF-IDF + Sentiment Scores	0.89	0.65	0.63	0.74	0.68
TF-IDF + Doc2vec	0.96	0.95	0.95	0.95	0.69
Sentiment Scores + Doc2vec	0.79	0.44	0.43	0.57	0.27
All Features	0.89	0.63	0.61	0.75	0.67

When BROUT was used to balance the dataset, the best performance of RF was recorded when TF-IDF features were used.

5.3 Comparison of the performances of ML algorithms

RF performed better than SVM, DT and LR when BROUT was used. It also outperformed the other ML algorithms when the imbalanced data was used and the also all the other techniques that were used to balance the dataset. When the different techniques to balance the dataset were used, the performance of all the classifiers decreased compared to when the imbalanced data was used. TF-IDF features produced the best results for all the classifiers. Combining the different features did not improve the performance of the ML algorithms.

5.4 Classification Results for neural networks

The results for all the NN using the data balancing techniques explained in in this section are presented in Table 27.

Table 27: Neural networks classification results

NN	Technique	Accuracy	F1 score	Recall	Precision	Cohen kappa
GRU	No technique	0.91	0.73	0.71	0.78	0.74
	Class weights	0.84	0.72	0.81	0.69	0.65
	SMOTE	0.82	0.68	0.77	0.65	0.61
	SMOTETomek	0.82	0.65	0.70	0.64	0.58
	SMOTEENN	0.77	0.64	0.72	0.62	0.52
	ADASYN	0.82	0.66	0.70	0.64	0.58
	BROUT	0.96	0.95	0.95	0.95	0.90
LSTM	No technique	0.90	0.74	0.73	0.76	0.73
	Class weight	0.82	0.70	0.82	0.67	0.61
	SMOTE	0.83	0.69	0.77	0.66	0.57
	SMOTETomek	0.81	0.66	0.74	0.66	0.57
	SMOTEENN	0.70	0.61	0.74	0.62	0.45
	ADASYN	0.84	0.69	0.73	0.67	0.50
	BROUT	0.88	0.88	0.88	0.89	0.83
BiLSTM	No Technique	0.89	0.69	0.76	0.75	0.70
	Class weights	0.81	0.70	0.82	0.68	0.61
	SMOTE	0.82	0.67	0.74	0.66	0.59
	SMOTETomek	0.81	0.67	0.74	0.66	0.58
	SMOTEENN	0.74	0.63	0.75	0.64	0.49
	ADASYN	0.80	0.66	0.73	0.67	0.56
	BROUT	0.91	0.91	0.91	0.91	0.87
CNN	No technique	0.90	0.73	0.71	0.76	0.72
	Class weights	0.89	0.74	0.79	0.71	0.69
	SMOTE	0.84	0.67	0.72	0.64	0.62
	SMOTETomek	0.83	0.66	0.71	0.64	0.60
	SMOTEENN	0.77	0.63	0.72	0.61	0.52
	ADASYN	0.83	0.65	0.70	0.63	0.60
	BROUT	0.96	0.96	0.96	0.96	0.94
CNN-BiLSTM	No technique	0.89	0.71	0.69	0.75	0.73
	Class weights	0.84	0.72	0.82	0.69	0.62
	SMOTE	0.83	0.67	0.73	0.67	0.62
	SMOTETomek	0.80	0.66	0.70	0.65	0.60
	SMOTEENN	0.80	0.66	0.73	0.65	0.57
	ADASYN	0.80	0.66	0.73	0.66	0.55
	BROUT	0.92	0.92	0.92	0.93	0.88

5.4.1 Analysis of the performance of the classifiers after data balancing techniques were implemented

The results in Table 27 show that adding the techniques Class weights, SMOTE, SMOTETomek, SMOTEENN and ADASYN for all classifiers lowers the performance of the classifiers as the Accuracy, Recall, Precision and Cohen Kappa scores all decreased when the classifiers were trained using the data balancing techniques compared to the results of the classifiers being trained with no technique. Except for the slight increase when class weights were used for BiLSTM, CNN and CNNBiLSTM, F1-score was also decreased

The classifiers had higher F1-scores, Accuracy, Recall, Precision and Cohen Kappa scores when Class weights were used which shows that adding Class weights leads to classifiers having a better performance than using the over sampling techniques and the hybrid sampling techniques for balancing a dataset. The technique that led to the lowest performance of the classifiers was SMOTEENN except when the hybrid model CNNBiLSTM was used where ADASYN lead to the lowest performance as it had the lowest F1- score and Cohen Kappa score as shown in Table 27.

5.4.2 Analysis of the performance of the classifiers using BROUT

The Accuracy, F1-score, Recall, Precision and Cohen kappa score all increased when the proposed technique, BROUT was used. The best performing classifier was CNN using the proposed technique which had an F1 score of 0.96 and a Cohen kappa score of 0.94. The confusion matrix for the best classifier and the learning curves that show how the model was learning over time are shown in Figure 10 and Figure 11 respectively.

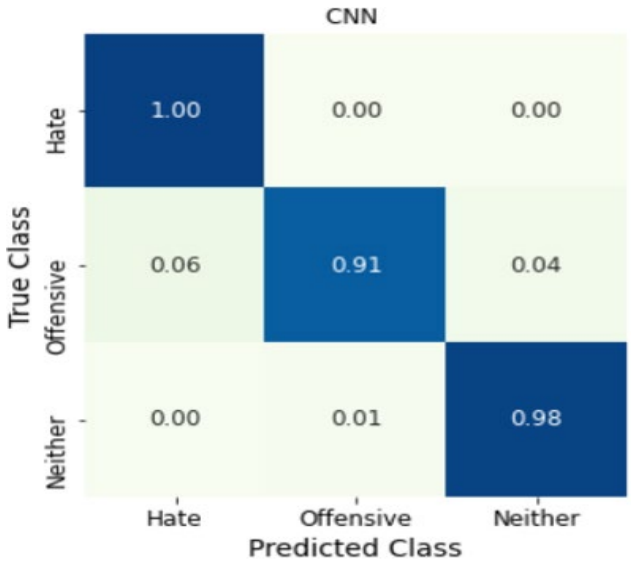

Figure 10: Confusion matrix of CNN, the best performing classifier

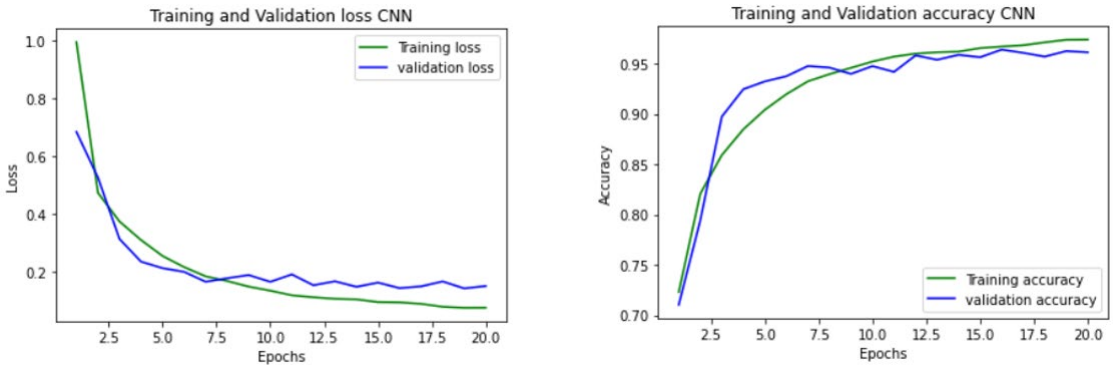

Figure 11: Learning curves of the best performing classifier CNN

The confusion matrices for NN using BROUT are shown in Figure 12.

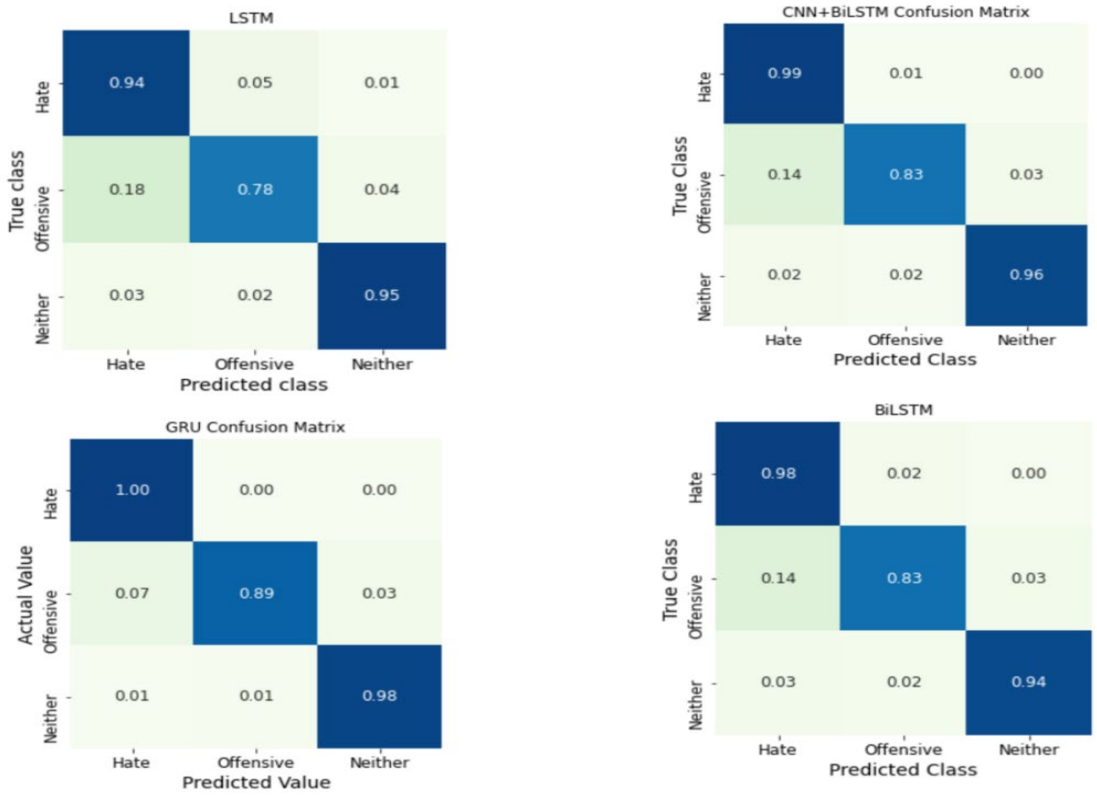

Figure 12: Confusion matrices for NN using BROUT

The LSTM classifier managed to classify 94% of hate tweets in the Hate class correctly and it misclassified 5% of hate tweets as offensive tweets and 1 % of hate tweets where misclassified as tweets in the Neither class. It also only managed to classify 78% of offensive tweets in the Offensive class correctly and 95% of the tweets in the Neither class. Compared to the performance of the other classifiers shown in Figure 12 and also the performance of the best performing CNN in Figure 10, it shows that LSTM performed the least compared to the other NN and this is further shown by the F1-scores and the Cohen Kappa scores in Table 27.

BiLSTM and CNN+BiLSTM performed almost the same with only a 0.01 difference in the F1-score, Recall, Precision and Cohen Kappa scores shown in Table 27. The two classifiers managed to classify the same number of tweets correctly in the

Offensive class as shown in Figure 12. The results from CNN+BiLSTM indicates that creating the hybrid model did not create a classifier that is better at hate speech detection as it performed almost the same with BiLSTM and it was out performed by CNN which is the best performing classifier of this research.

The GRU classifier had the second-best performance for classifying Hate tweets using BROUT with only a 0.01 difference in the F1-score with CNN having the higher F1-score as shown in Table 27. However, the Cohen Kappa scores showed that CNN was the best performing classifier as it had a higher Cohen Kappa score as shown in Table 27. According to the confusion matrices in Figure 12 and Figure 10, GRU managed to classify all the hate tweets in the Hate class like CNN and they classified the same number of tweets in the Neither class but CNN managed to classify 91% of Offensive tweets correctly whilst GRN classified 89% of the Offensive tweets correctly.

5.5 Comparison of the state of art results against the best performing classifier

This research was inspired by the work done by Davidson et al.[4]. In this section I will be comparing their results with the results from the best classifier.

Table 28: Comparison of state of art results against the best performing classifier, CNN

Classifier	**F1 score**	**Recall**	**Precision**
Davidson et al. classifier	0.90	0.90	0.91
Best performing classifier, CNN	0.96	0.96	0.96

The best model by Davidson et al. was a logistic regression model. Their best model misclassified 40% of hate tweets and the recall and precision for the hate class 0.61

and 0.41 respectively. The highest classification performed using BROUT was given by CNN as shown in Table 27. It managed to classify all the tweets in the hate class correctly with a recall score of 1.00 and precision score of 0.94. The confusion matrices for both results are shown in Figure 13. The confusion matrices also show that both models classified the same number of Offensive tweets and the best classifier classified more Neither tweets than the Davidson et al. model.

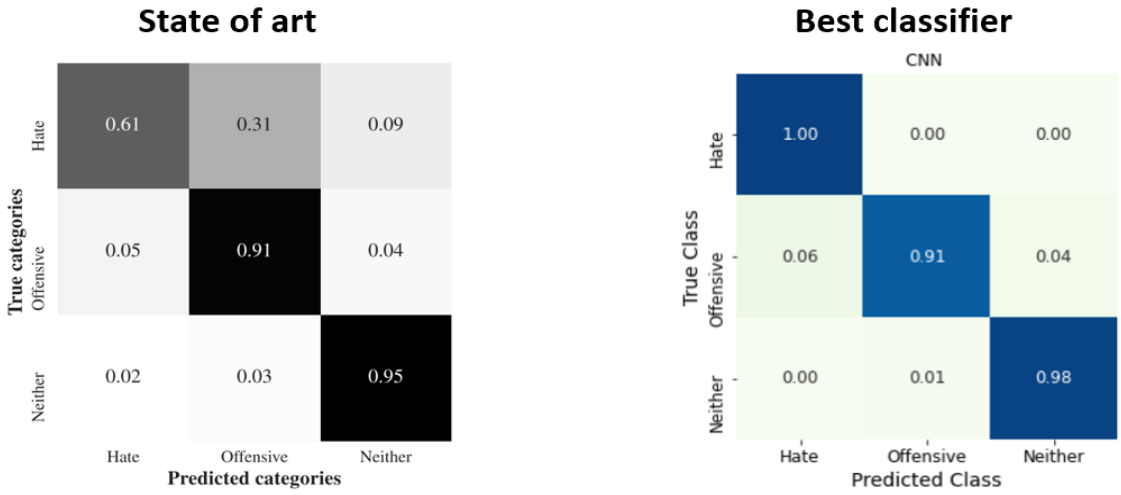

Figure 13: Confusion matrix of state of art vs best performing classifier, CNN

5.6 Comparison of the performances of ML algorithms and NN

The NN had overall better performance that the ML algorithms except when BROUT was used where the ML algorithms performed better that LSTM and BiLSTM. The resampling techniques SMOTE, SMOTETomek, SMOTEENN and ADASYN where better adapted when NN were used as it is shown in Table 27 how a clear comparison can be made between the techniques to see which technique was leading to a better performance compared to the other. When the resampling techniques were used with LR, SVM and RF they produced the same results which was not able to show which technique is better compared to when NN were used.

Chapter 6

CONCLUSION

In this research different data balancing techniques were used on a multiclass imbalanced twitter hate speech dataset to check if the performance of classifiers would improve in hate speech detection. A proposed technique to balance the dataset BROUT was introduced and the results gathered using BROUT were compared against the other data balancing techniques. To check how the classifiers were actually performing after each data balancing technique the Cohen Kappa score was used as a metrics together with F1-scores, Accuracy, Recall, and Precision.

Taking note of the Cohen kappa scores we can see that the performance of the classifiers decreased when data balancing techniques were implemented except when the proposed method was used. The best performance was recorded using the CNN classifier together with the proposed technique and it had an Accuracy score of 0.96, F1-score score of 0.96, a Cohen Kappa score of 0.94 and a Recall and Precision score of 0.96. The classifier also performed better than the state of art of model.

In future works I would like to work with other hate speech imbalanced dataset form other social media platforms and implement BROUT to see check how it would perform. I would also want to work with different embeddings like Universal Sentence Encoder (USE) and transformer models like BERT to see if they improve the performance of classifiers in hate speech detection.

REFERENCES

[1] "Twitter by the Numbers (2021): Stats, Demographics & Fun Facts." https://www.omnicoreagency.com/twitter-statistics/ (accessed Jul. 26, 2021).

[2] "Facebook by the Numbers (2021): Stats, Demographics & Fun Facts." https://www.omnicoreagency.com/facebook-statistics/ (accessed Jul. 26, 2021).

[3] S. MacAvaney, H. R. Yao, E. Yang, K. Russell, N. Goharian, and O. Frieder, "Hate speech detection: Challenges and solutions," *PLoS One*, vol. 14, no. 8, pp. 1–16, 2019, doi: 10.1371/journal.pone.0221152.

[4] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," *Proc. 11th Int. Conf. Web Soc. Media, ICWSM 2017*, pp. 512–515, 2017.

[5] P. Fortuna and S. Nunes, "A survey on automatic detection of hate speech in text," *ACM Comput. Surv.*, vol. 51, no. 4, 2018, doi: 10.1145/3232676.

[6] S. Wang and X. Yao, "Multiclass imbalance problems: Analysis and potential solutions," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 42, no. 4, pp. 1119–1130, 2012, doi: 10.1109/TSMCB.2012.2187280.

[7] H. Ali, M. N. M. Salleh, R. Saedudin, K. Hussain, and M. F. Mushtaq, "Imbalance class problems in data mining: A review," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 14, no. 3, pp. 1552–1563, Jun. 2019, doi:

10.11591/ijeecs.v14.i3.pp1552-1563.

[8] G. G. Chowdhury, “Natural language Processing.”

[9] S. K. Khanna, “Machine Learning v/s Deep Learning,” *Int. Res. J. Eng. Technol.*, p. 455, 2008, [Online]. Available: www.irjet.net.

[10] B. Eingereicht Im Rahmen Der Bachelorprüfung, B. Prüfer, O. Zukunft, S. Sarstedt, and C. Dohmen, “Carolin Dohmen Detecting Hate Speech in Social Media-A Machine Learning Approach Title of the paper Detecting Hate Speech in Social Media-A Machine Learning Approach.”

[11] A. Mathew, P. Amudha, and S. Sivakumari, “Deep learning techniques: an overview,” in *Advances in Intelligent Systems and Computing*, 2021, vol. 1141, pp. 599–608, doi: 10.1007/978-981-15-3383-9_54.

[12] K. Cho *et al.*, “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,” Jun. 2014, [Online]. Available: http://arxiv.org/abs/1406.1078.

[13] R. Rana, “Gated Recurrent Unit (GRU) for Emotion Classification from Noisy Speech,” Dec. 2016, [Online]. Available: http://arxiv.org/abs/1612.07778.

[14] M. Z. Amin and N. Nadeem, “Convolutional Neural Network: Text Classification Model for Open Domain Question Answering System.”

[15] “Report on Text Classification using CNN, RNN & HAN | by Akshat Maheshwari | Jatana | Medium.” https://medium.com/jatana/report-on-text-classification-using-cnn-rnn-han-f0e887214d5f (accessed Aug. 04, 2021).

[16] A. I. Marqués, V. García, and J. S. Sánchez, “On the suitability of resampling techniques for the class imbalance problem in credit scoring,” *J. Oper. Res. Soc.*, vol. 64, no. 7, pp. 1060–1070, 2013, doi: 10.1057/jors.2012.120.

[17] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” *J. Artif. Intell. Res.*, vol. 16, no. June, pp. 321–357, 2002, doi: 10.1613/jair.953.

[18] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, “A study of the behavior of several methods for balancing machine learning training data,” *ACM SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 20–29, Jun. 2004, doi: 10.1145/1007730.1007735.

[19] D. L. Wilson, “Asymptotic Properties of Nearest Neighbor Rules Using Edited Data,” *IEEE Trans. Syst. Man Cybern.*, vol. 2, no. 3, pp. 408–421, 1972, doi: 10.1109/TSMC.1972.4309137.

[20] H. He, Y. Bai, E. A. Garcia, and S. Li, “ADASYN: Adaptive synthetic sampling approach for imbalanced learning,” in *Proceedings of the International Joint Conference on Neural Networks*, 2008, pp. 1322–1328, doi: 10.1109/IJCNN.2008.4633969.

[21] M. Grandini, E. Bagli, and G. Visani, “Metrics for Multi-Class Classification: an Overview,” Aug. 2020, [Online]. Available: http://arxiv.org/abs/2008.05756.

[22] H. Watanabe, M. Bouazizi, and T. Ohtsuki, “Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection,” *IEEE Access*, vol. 6, pp. 13825–13835, Feb. 2018, doi: 10.1109/ACCESS.2018.2806394.

[23] M. Hall, G. Holmes, B. Pfahringer, P. Reutemann, E. Frank, and I. H. Witten, “The WEKA data mining software: An update Commercial data mining projects using the ADAMS platform View project The WEKA Data Mining Software: An Update,” 2014. [Online]. Available: https://www.researchgate.net/publication/221900777.

[24] G. I. Webb, “Decision Tree Grafting From the All-Tests-But-One Partition,” Morgan Kaufmann.

[25] O. Oriola and E. Kotze, “Evaluating Machine Learning Techniques for Detecting Offensive and Hate Speech in South African Tweets,” *IEEE Access*, vol. 8, pp. 21496–21509, 2020, doi: 10.1109/ACCESS.2020.2968173.

[26] T. T. A. Putri, S. Sriadhi, R. D. Sari, R. Rahmadani, and H. D. Hutahaean, “A comparison of classification algorithms for hate speech detection Recent citations A comparison of classification algorithms for hate speech detection,” doi: 10.1088/1757-899X/830/3/032006.

[27] R. Martins, M. Gomes, J. J. Almeida, P. Novais, and P. Henriques, “Hate speech classification in social media using emotional analysis,” in *Proceedings - 2018 Brazilian Conference on Intelligent Systems, BRACIS 2018*, Dec. 2018, pp. 61–66, doi: 10.1109/BRACIS.2018.00019.

[28] G. Koushik, K. Rajeswari, and S. K. Muthusamy, “Automated hate speech detection on Twitter,” Sep. 2019, doi: 10.1109/ICCUBEA47591.2019.9128428.

[29] IEEE Staff, *2019 5th International Conference on Science in Information Technology (ICSITech)*. IEEE, 2019.

[30] R. Raj, S. Srivastava, and S. Saumya, “NSIT & IIITDWD @ HASOC 2020: Deep learning model for hate-speech identification in Indo-European languages,” 2021. [Online]. Available: https://docs.cltk.org/en/latest/hindi.html.

[31] C. Paul and P. Bora, “Detecting Hate Speech using Deep Learning Techniques,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 2, pp. 619–623, 2021, doi: 10.14569/IJACSA.2021.0120278.

[32] 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, pp. 1–16, Dec. 2020, doi: 10.3390/app10238614.

[33] D. Nurjanah, I. Gede, and M. Putra, “Hate Speech Detection In Indonesian

Language Instagram.” [Online]. Available: https://www.pantip.com.

[34] “scikit-learn: machine learning in Python — scikit-learn 1.0 documentation.” https://scikit-learn.org/stable/ (accessed Sep. 25, 2021).

[35] “Keras: the Python deep learning API.” https://keras.io/ (accessed Sep. 25, 2021).

[36] “TensorFlow.” https://www.tensorflow.org/ (accessed Sep. 25, 2021).

[37] “Recurrent layers.” https://keras.io/api/layers/recurrent_layers/ (accessed Sep. 28, 2021).

[38] “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science.” https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (accessed Sep. 28, 2021).

[39] “Convolutional Neural Network. Learn Convolutional Neural Network from… | by dshahid380 | Towards Data Science.” https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529 (accessed Sep. 28, 2021).