HATE SPEECH DETECTION IN SOCIAL MEDIA POSTS USING NATURAL LANGUAGE PROCESSING METHODS

A dissertation submitted to

the Faculty of Engineering and Natural Sciences of

International University of Sarajevo in partial

fulfillment of the requirements for the

degree of Bachelor of Science

in Software Engineering

by

Šejla Burnić

June, 2022

Dissertation written by

Šejla Burnić

Bachelor Thesis Committee

Prof. Dr. Khaldoun al Khalidi	Associate Professor, BiH, Chair
Prof. Dr. Kanita Karađuzović-Hadžiabdić	Associate Professor, BiH, Supervisor

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

ŠEJLA BURNIĆ

INTERNATIONAL UNIVERSITY OF SARAJEVO

DECLARATION OF COPYRIGHT AND AFFIRMATION

OF FAIR USE OF UNPUBLISHED WORK

DETECTION OF HATE SPEECH IN SOCIAL MEDIA POSTS USING NATURAL LANGUAGE PROCESSING METHODS

Affirmed by Šejla Burnić

Signature Date 14.6.2022.

ABSTRACT

In the ever-growing space of the internet, social media has become one of the main attractions for people worldwide. The emergence of social media has given us the possibility to reach millions of people in an instant. However, not all content shared on social media is posted with good intentions. One type of such content is hate speech – communication that is intended to attack another person or group of people because of identifiers such as race, religion, gender, and the like. Social media sites are paying efforts to address this problem however the abundance of content that is produced each day makes manual monitoring inefficient. This thesis explores a possible solution to this problem by using natural language processing (NLP). For the implementation of NLP methods machine learning was used – namely Support Vector Machines (SVM), Long Short-Term Memory (LSTM), and Bidirectional Encoder Representation from Transformers (BERT) models. In addition to the training machine learning models, this thesis explains the various preprocessing steps done in NLP. The results of the work are promising as they outperformed previous work done on the same dataset – the Twitter hate speech dataset. The best performing model was the SVM model which achieved a balanced accuracy of 87.61% with a sensitivity of 77.97% for hate speech tweets. The second best-performing model was the BERT model with a balanced accuracy of 73.22% and sensitivity for hate speech of 31.47%, followed by the LSTM model with 71.17% balanced accuracy and 25.65% sensitivity for hate speech.

Keywords: hate speech detection, natural language processing, support vector machines, recurrent neural networks, BERT

DEDICATION

This thesis, as the pinnacle of the last seventeen years of my education, is dedicated to my parents. I will forever be grateful for the support and love that I received from you. Even though I know I will never be able to repay you for the sacrifices that you have made I will try to honor them. I will honor them by being the best I can be at my work, but more importantly by showing integrity, compassion, and understanding towards others – qualities that you have always highlighted during my upbringing. Thank you and may Allah bless you with the highest ranks of Jannah.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

BERT	Bidirectional Encoder Representations from Transformers
BOW	Bag of Words
CNN	Convolutional Neural Network
FCM	Feature Concatenation Model
LSTM	Long Short-Term Memory
MCD	Monte Carlo Dropout
ML	Machine Learning
MLM	Masked Language Modeling
NLP	Natural Language Processing
NSP	Next Sentence Prediction
POS	Part-of-speech
RNN	Recurrent Neural Network
SCM	Spatial Concatenation Model
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machines
TF-IDF	Term Frequency – Inverse Document frequency
TKM	Textual Kernels Model

ACKNOWLEDGEMENTS

First of all, I want to thank Allah for guiding me and giving me patience through what was possibly the most difficult time in my life, and giving me the strength to focus on the writing of this thesis.

Second, I want to thank my family and friends who have supported and encouraged me, not just during the writing of this thesis but the last four years.

Third, I want to thank the Source of Hope Foundation for being a home away from home during my studies, for all the opportunities and projects that we brought to life, and for all the, hopefully lifelong, friendships made.

Lastly, I want to thank my mentor Assist. Prof. Dr. Kanita Karađuzović-Hadžiabdić for her continuous and hard work without whom this thesis would have not been possible.

Šejla Burnić

14.6.2021., Sarajevo, Bosnia and Herzegovina

CHAPTER ONE

INTRODUCTION

1.1. Overview

On June 2015, Dylann Roof, a 22-year-old American, committed an atrocious crime by killing nine black worshippers at the Emanuel African Methodists Episcopal Church. After his arrest, a thorough investigation concluded that his viewpoints were from "*self-taught from material found online and elsewhere*" [1]. On another occasion, Robert Bowers opened fire inside a Pittsburgh, Kansas, USA synagogue in October 2018, taking 11 lives. Investigation of the incident revealed that prior to the shooting Robert was active on the online platform Gab. This social media network was known to be popular among white nationalists and supremacists. Here, Bowers loudly and hatefully spread anti-immigrant and antisemitic content calling for the killing of said ethnic groups [2]. Further, K. Müller et al. [3] highlight the correlation between hateful content online and real-life hate crimes. The paper suggested that the wide availability of internet access has led to social media becoming fertile soil for the creation and spread of hateful ideas and content online. Further, it was suggested that access to such content fuels hatred and serves as a motivator to take real-life action in the form of hate crimes against minority groups.

It is evident from the cases mentioned above that online hate has become an issue that has transcendent the online world and crossed over into the real world. The availability of the internet and the uncensored nature of social media have laid a foundation for the creation and spread of hate online. More and more, the internet is becoming a platform for hate groups to not only spread their message to millions of people but to encourage them to do the same. Despite the overwhelming presence of hateful content online its' existence is a popular topic

among the advocates of free speech. They argue that hate speech is part of free speech and should be treated as such – have no legal consequences. This is, also, the official stance of the government of the United States of America as they argue that limiting speech, even if it is hateful, is a direct violation of the First Amendment of the Constitution of the U.S. [4] On the other hand, many governmental and non-governmental institutions recognize the existence of hate speech and have mechanisms how to fight and prevent it – online and offline [5]–[7].

One of the problems of fighting hate speech is the non-existence of a uniform definition of hate speech and therefore its' legal definition, if one exists, varies from country to country. According to the Cambridge Dictionary hate speech is "*public speech that expresses hate or encourages violence towards a person or group based on something such as race, religion, sex, or sexual orientation*" [8]. The United Nations defines hate speech as "*any kind of communication in speech, writing or behaviour, that attacks or uses pejorative or discriminatory language with reference to a person or a group on the basis of who they are, in other words, based on their religion, ethnicity, nationality, race, colour, descent, gender or other identity factor*" [7].

As online hate speech is mostly present on and propagates through social media it is in the interest of legislators and social media networks themselves to take appropriate action to identify and counter such behavior. One of the challenges of monitoring online content is the large amount of social media content that is created every day. Hence, this is cumbersome to do manually and other, more sophisticated, methods must be used. One of the methods that make it possible to identify hate speech is natural language processing. Natural language processing (NLP) is a branch of computer science that strives to give machines the ability to understand text and spoken words in the same way that humans do [9]. NLP uses statistical,

machine learning, and deep learning models in combination with computational linguistics to process human language and to give computers the ability to ‘understand’ its meaning.

1.2. Problem statement

As mentioned in the overview section, hate speech on social media has become a serious topic in recent years. The main reason for this is the correlation between hate speech online and real-life hate crimes. One of the main challenges in the identification of hate speech online is the amount of social media content that needs to be reviewed. Manual review is inefficient and time-consuming due to the large amount of user-generated content. However, by implementing NLP methods social media content can be efficiently and effectively classified and appropriate actions can be taken with little to no human input.

In this thesis, machine learning models were used for classification of social media posts on a Twitter hate speech dataset in the English language. The algorithms used to train the models were support vector machines (SVM), long short-term memory (LSTM), and bidirectional encoder representations from transformers (BERT).

1.3. Objectives

The first aim of this thesis is to give an overview of the preprocessing methods that are used in NLP tasks. The second objective is to create a machine learning model that will as accurately as possible detect hate speech in textual social media posts on Twitter. The last objective is to identify possible drawbacks and limitations of this research thesis and offer suggestions for future work that could potentially improve upon the achieved results.

1.4. Organization of thesis

The first chapter of the theses provided a brief overview on the topic of hate speech and highlighted the importance of identifying such speech. The second chapter examines previous

work done on hate speech detection on social media platforms. Chapter three goes into the details of feature engineering specific to the NLP field as well as the methods used for model training mentioned in the problem statement. Further, chapter three also briefly explains the data collection process done by the authors of the Twitter hate speech dataset. Lastly, chapter three provides a detailed explanation of the preprocessing and model implementations. Chapter four presents the results of the model evaluation phase as well as a discussion of the obtained results. Chapter five concludes the thesis by summarizing the work done and presented in previous chapters.

CHAPTER TWO

RELATED WORKS

2.1. Introduction

This chapter lists and briefly explains some of the previously done work on hate speech detection in social media content.

2.2. Literature review

I. Kwok and Y. Wang [10] proposed a method for racism detection against Black people in Twitter posts. The used dataset contained Tweets that were labeled as racist or non-racist. Bag-of-words (BOW) text representation was used as the feature transformation technique. One-word sequences, unigrams, were used to construct a lexicon from the Twitter posts corpora. The preprocessing on the corpora included setting all document to lowercase, deleting mentions, links, and stop-words from the text, and fixing spelling of slurs contained in the document. Then, the preprocessed Twitter posts were put into the BOW model to create their vector representation. Lastly, the preprocessed data was used to train a Naïve Bayes classifier model using 10-fold cross validation. An accuracy of 76% was obtained with an error rate of 24%. Some drawbacks of this methodology, as the authors mention, was that it did not catch the relationship between words – since unigram features were used. Moreover, it is mentioned that racism does not necessarily include use of racial slurs and can be expressed more subtly. It is also mentioned that to correctly detect racism, the race of the author of a text has to be taken into consideration – which is difficult to determine given the relative anonymity of online users.

Davidson et al. [11] also created a model for detection of hate speech in Twitter posts. However, their definition of hate speech was wider as it included other types of hate speech – such as racism, nationalism, homophobia, and the like. For model training the authors created

their own dataset. Twitter API was used to collect tweets, and crowd sourcing was used to annotate each tweet as non-hate, offensive, and hate speech. Then, the text in the corpora was set to lowercase; and mentions, links, hashtags, emoticons, and punctuation marks were removed from the text. Next, syntactic, Part-of-Speech (POS) tagging, and semantic features were extracted. Lastly, a logistic regression model was trained with 5-fold cross validation. The proposed method achieved 91% overall accuracy; however, it was found that almost 40% of hate speech text was misclassified. It was stated that, to improve results, the social context and the context of the conversation need to be taken into consideration. Further, it was suggested that other factors – like race, gender, and age of the author of a tweet – are also important and should be considered.

Another research done on hate speech detection in Twitter posts was done by H. Watanabe et al. [12]. The paper used the dataset created by the authors of [11]. Identically to [11], the raw Twitter text was set to lowercase; and tags, mentions, and hashtags were removed. The preprocessing also included decomposition of hashtags – e.g., "#Fu*kMuslims" would be decomposed into the words "Fu*k" and "Muslims". In addition to semantic- and sentiment-based features, unigram features were extracted to detect the presence of certain words, and pattern features – relations between words. The research included training of three different models using the Random Forest, Support Vector Machines (SVM), and C4.5 decision tree algorithms. For every algorithm, two types of classification models were trained – a binary, combining hate and offensive category into one category, and the original ternary classification model. For both, binary and ternary classification, it was found that the model trained with the C4.5 algorithm performed the best with an accuracy of 87% and 78%, respectively.

Unlike the previously mentioned research papers, the research done in [13] focused on hate speech detection in Facebook posts in the Italian language. The steps done during

preprocessing of the texts included morpho-syntactical tagging of words, POS tagging, extraction of sentiment features, and extraction of features of the raw text – like number of word tokens. The dataset in question had three class attributes: strong hate, weak hate, and no hate. Similarly to [12], two different classification models were created – a binary classification model, combining strong hate and weak hate classes into one class; and a ternary classification model. The models were trained with the SVM and long short-term memory (LSTM) algorithms. The LSTM model is a recurring neural network model that, unlike the vanilla RNN, can learn long-term dependencies. The SVM model achieved the highest accuracy – 66% and 77% for ternary and binary classification, respectively.

[14] is one of the more recent efforts done on automatic hate speech detection. This work, like previous works, strived to implement a machine learning model that would detect hate speech in Twitter posts, however, in the Slovenian language. The dataset was collected from Twitter over a 3-year period. Over 13 million tweets were collected; however, a subset of 50k and 10k randomly selected tweets were selected for the training and evaluation sets, respectively. Next, the individual tweets were manually annotated as belonging to one of four classes: acceptable, inappropriate, offensive, or violent. Models were trained using the Naïve Bayes, Logistic regression, SVM, Bidirectional Encoder Representations from Transformers (BERT), and cseBERT machine learning algorithms. The BERT and cseBERT models are transformer-based, state-of-the-art machine learning models used for NLP tasks. The BERT model is a pre-trained NLP model trained on a large corpus that can be fine-tuned to the requirements of a particular task. While the BERT model is trained on a corpus for the English language sceBERT is trained on a corpus of the Slovenian-Croatian-English language. The paper did not mention the preprocessing done on the collected Twitter data. After model training and evaluation, it was found that the best performing model was the fine-tuned

cseBERT model with an overall accuracy of 80%, in both the training and evaluation datasets. Another contribution made by this research was the analysis and creation of retweet networks – network of people with similar political and/or social views that share similar content online. It was argued that, as stated in previously mentioned papers, for effective hate speech detection it is not enough to look at the raw content of a tweet stripped from the context in which it was posted. The use of deep learning methods was suggested as a technique for identifying online communities that act as echo-chambers for hate speech.

Another effort at automatic hate speech detection was done in [15]. This research focused on detection of hate speech that attacks a person based on their race, nationality, and/or religion. The dataset used in this research was compiled by crawling two online forums. The obtained blog posts were then annotated as strongly hateful, weakly hateful, or non-hateful on a paragraph basis. Then, the individual sentences in the classified paragraphs were labeled as either objective or subjective. This was done using a rule-based approach while relying on a lexicon of well-established clues for subjectivity, both negative and positive polarity. Next part of the research included constructing a lexicon for hate speech from the features present in the corpus. This was done by identifying opinionated words and nouns in the subjective sentences that have a negative semantic meaning. Next step was to include verbs that were present in the complete dataset but are often used in hate speech – like discriminate, loot, or riot. The last step was to extract grammatical patterns that depict hateful intent. Unlike the research mentioned before, a rule-based classifier using the Java programming language was created. This was done by constructing rules based on which sentences were classified as one of the three previously mentioned classes. The classifier was created by taking into consideration different combinations on the three types of features that were extracted. The best results were achieved by including all three types of features with an overall accuracy of 72%.

R. Mutanga et al. [16] is another attempt at creating a hate speech detection model from the dataset created in [11]. The proposed solution uses deep learning models based on attention and transformers. The preprocessing consisted of removal of punctuation marks, normalization of hashtags, lowercasing text, stemming word tokens, and removing infrequent tokens. The models were trained using the LSTM and distilBERT architectures. DistilBERT, on the other hand, is a pre-trained model like the BERT base model however with less parameters and a faster training time. The best performing model was distilBERT with an average accuracy of 92% across all three classes. The LSTM model achieved an average accuracy of 66%.

The research done in K. Miok et al. [17] proposed a different method for solving the problem of hate speech. It was put forward that NLP tasks are usually carried out by deep learning methods which are black box methods and consequently do not offer information regarding the reliability of the classification result. To solve this problem, it was proposed to use a Bayesian method that uses Monte Carlo Dropout (MCD) in the attention layers of the transformer models that, as a result, would provide the necessary data to calculate reliability scores for the predictions. Monte Carlo Dropout was used to create a more robust model and to prevent overfitting. The only difference that MCD made was that the dropout layers were active during both – the training and prediction phase. This resulted in multiple, possibly different, predictions during testing phase for one data point. The predictions were then averaged to get the final prediction for the data point. To test the proposed method, already available datasets in the English, Croatian, and Slovenian language were used. The BERT and MCD LSTM models were used as baseline models for model evaluation. Following training it was concluded that MCD BERT performed the best in all three languages. Lastly, the MCD BERT predictions were combined with sentiment features, obtained by using the SenticNet framework, of the tweets. The combined features were fed into an SVM model with 5-fold

cross validation. The conclusion was that additional information regarding the emotional content of the tweets did not improve the performance of the model.

Research in [18] takes a different approach to hate speech detection. It mentions that, in the ever-growing space of internet it was not enough to analyze text only. Hence, the problem was extended from a unimodal problem, using one type of medium for hate speech detection, to a multimodal problem, using multiple mediums. It was suggested that by leveraging from visual clues and combining them with the text of a post hate speech could be detected more reliably. Because datasets for hate speech detection containing both textual and visual data did not exist at that time one of the contributions of this research was the creation of such a dataset. The Twitter API was used to obtain tweets – their text and the accompanying images. Because a lot of noise data was present in the dataset, especially in the images, tweets containing three or less words and images that are screenshots of other tweets were discarded. Finally, the tweets were annotated with the help of crowd sourcing. The next step included extracting both textual and visual features from the tweets. To extract visual features, Google's Inception v3 pre-trained model was used. For the potential text contained in the image Google's Vision API Text Detection module was leveraged. The tweet text and the image text were then encoded by LSTM. Three different RNN and CNN models were trained with the preprocessed data: Feature Concatenation Model (FCM), Spatial Concatenation Model (SCM), and Textual Kernels Model (TKM). Besides using three different models, all possible combination of features (tweet text, tweet image, and tweet image text) were used for training. Upon model evaluation it was concluded that, when using all three features, all three models performed identically with an accuracy of 68%. However, despite the initial assumption that multimodal models were better at detecting hate speech, it was found that the unimodal model, which used only the tweet

text, performed equally as good. This result was attributed to the errors that occurred during data annotation and the presence of noisy data that failed to be removed.

2.3. Summary

Chapter two reviewed the previously done work on hate speech detection on social media. The most used methods in previous research were support vector machines (SVM) and deep learning methods like convolutional neural networks (CNNs) and recurrent neural networks (RNNs). More recent work highlights the rising popularity of pre-trained, transformer-based models – like the BERT model created by Google. Pretrained models were shown to achieve state-of-the-art results and are becoming the preferred method of choice in the field of NLP. Finally, research concludes that using purely text-based features is not enough for the successful detection of hate speech and that the context in which the text was written is essential.

CHAPTER THREE

MATERIALS AND METHODS

3.1. Introduction

This chapter provides a detailed explanation of the conducted experiments. First, the dataset that was used for model training and evaluation is presented. Next, feature engineering techniques used in NLP are described with provided examples. Then, the algorithms used during this research and their structure are detailed. Lastly, the preprocessing and training done for every model are explained.

3.2. Dataset

The dataset used in this paper was collected by the authors of [11] in 2017. The authors used Hatebase.org to create a hate speech lexicon from words and phrases that online users found to fall under the category of hate speech. Further, they used Twitter API to search for tweets containing terms from the compiled hate speech lexicon. After finding the tweets containing hate speech, they extracted the users that wrote those tweets and downloaded all tweets written by those users. This resulted in a dataset of 85.4 million tweets. They then randomly sampled 25k tweets that contain terms from the hate speech lexicon. Lastly, each tweet was manually annotated by CrowdFlower workers. The authors supplied the workers with definitions of hate speech after which the workers coded the tweets as belonging to one of three classes: hate speech, offensive language, or neither. The authors of the dataset defined hate speech to be "*language that is used to express hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group*". Speech that contained derogatory terms but was used in a non-derogatory or threatening manner was considered offensive. The resulting class proportions are shown in Figure 3.1.

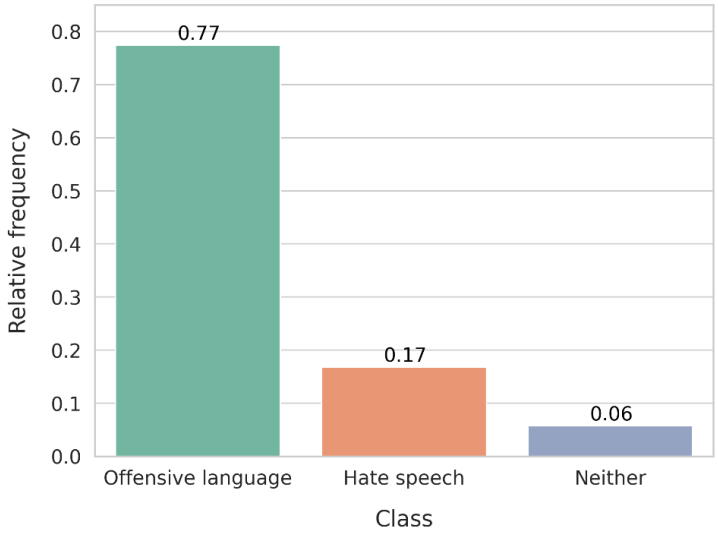

Figure 3.1 Twitter hate speech dataset class distribution

3.3. Feature engineering

Feature engineering is the activity of transforming raw data into an appropriate format for machine learning model training [19]. Data preprocessing done during this project depended on the type of algorithm used for model training, as different models require different formats for input data. Nevertheless, all feature engineering done during the making of this thesis can be categorized into one of the following groups:

1. Text cleaning
2. Tokenization
3. Lemmatization
4. Vectorization
5. Feature transformation
6. Feature selection

3.3.1. Text cleaning

Text cleaning is one of the most crucial steps in NLP tasks as it removes noise data from the text. Text cleaning, usually, involves the removal of certain characters like punctuation marks, non-ASCII characters, numbers, special characters, stop words, excess whitespaces, etc. Additional text cleaning criteria can be applied as well but they are, mostly, task-specific.

3.3.2. Tokenization

One of the first steps in NLP tasks is to extract features from the text which is to be analyzed. As text is unstructured, the text cannot be analyzed as a whole, as one feature, but must be broken into several features called tokens. The process of segmenting continuous text into tokens is called tokenization [20]. Tokenization can be done on a sentence, word, and character level [21]. Sentence level tokenization breaks up continuous text into individual sentences. Word level tokenization segments text into words. It is also possible to segment text into word n-grams – a continuous sequence of *n* words. Character tokenization is most often done as character n-grams – a continuous sequence of *n* characters. Examples of unigram word tokenization are shown in Figure 3.2.

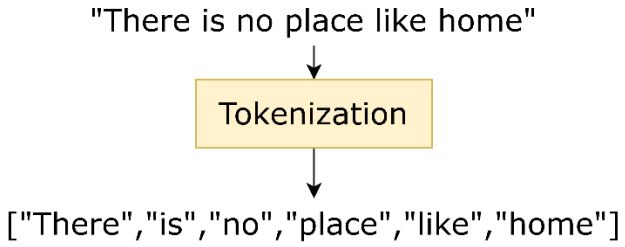

Figure 3.2 Tokenization example

The process of tokenization depends on the language of the text that is to be segmented into tokens. For instance, the Arabic language is morphologically complex and as a result the complexity of tokenization is high [22]. The tokenization of English text, on the other hand, is less complex. To generate word tokens, whitespaces are usually used as separators [23].

3.3.3. Lemmatization

The possible next step in text preprocessing is lemmatization. Lemmatization is the process of mapping all the different forms of a word to its' root, or lemma [24]. For instance, the words *did*, *done,* and *doing* are forms of the verb *do*. This step is done to reduce the number of words in the corpus. This, consequently, reduces the number of features. Lemmatization is a complex problem and unlike stemming, the removal of suffixes, demands more linguistic knowledge, especially in morphologically complex languages [25]. Due to this, the development of efficient lemmatizers remains an open problem in NLP research [26].

3.3.4. Text vectorization

An important part of every NLP task is how the text in a corpus is represented. What text representation is going to be used depends on the algorithm into which the corpus is fed into. One way of representing text is through vectors [27]. This can be done by mapping every word in the vocabulary (V) of a text corpus to a unique ID and then represent each sentence or document in the corpus as a V-dimensional vector.

Table 3.1 Text corpus for TF-IDF

Document ID	**Document text**
d1	The cat chases the mouse
d2	The mouse ran away
d3	The cat died

The text corpus in Table 3.1 consists of three documents and is comprised of seven words – the, cat, chases, mouse, ran, away, and died. Given these seven words every document in the corpus can be represented as a 7-dimensional vector. What the vector is going to look like depends on the approach taken.

One approach of text vector representation is TF-IDF [28]. TF-IDF stands for term frequency-inverse document frequency. It is the numerical representation of text that considers the importance of words. For instance, to represent word *w* in document d_i knowledge about its' frequency in document d_i and its' frequency in the whole corpus is needed. The importance of word *w* in d_i increases in proportion to its' frequency in d_i but decreases in proportion to its frequency in the whole corpus. This is represented using two quantities: TF and IDF. The two are combined to arrive at the final TF-IDF score.

TF (term frequency) measures how often a term *t* appears in document *d*. Because documents can vary in length a term may appear multiple times in a longer document than it appears in shorter documents. To account for this fact, the number of occurrences of the term *t* in document *d* is divided by the length of document *d*. The formula for TF is written in Equation 3.1.

$$TF(t,d) = \frac{Number\ of\ occurences\ of\ term\ t\ in\ document\ d}{Total\ number\ of\ terms\ in\ document\ d} \tag{3.1}$$

IDF (inverse document frequency) measures the importance of term *t* across the whole corpus. Some words, like stop words, are frequently used but are not important. To account for such cases, IDF is there to weigh down very common terms and increase the weight of rare terms. IDF of a term *t* is calculated according to Equation 3.2.

$$IDF(t) = \log_e \frac{Total\ number\ of\ documents\ in\ corpus}{Number\ of\ documents\ with\ term\ t\ in\ them} \quad (3.2)$$

Finally, to get the TF-IDF score, TF and IDF are combined as in Equation 3.3.

$$TF\ IDF = TF * IDF \quad (3.3)$$

The vector representations of the documents of the corpus in Table 3.1 are shown in Table 3.2.

Table 3.2 Example of TF-IDF calculation

Word	**TF**			**IDF**	**TF-IDF**		
	d1	**d2**	**d3**		**d1**	**d2**	**d3**
The	2/5	1/4	1/3	log(3/3) = 0	0	0	0
Cat	1/5	0	1/3	log(3/2) = 0.4	0.08	0	0.132
Chases	1/5	0	0	log(3/1) = 1	0.2	0	0
Mouse	1/5	1/4	0	log(3/2) = 0.4	0.08	0.1	0
Ran	0	1/4	0	log(3/1) = 1	0	0.25	0
Aways	0	1/4	0	log(3/1) = 1	0	0.25	0
Died	0	0	1/3	log(3/1) = 1	0	0	0.33

Another method for text vectorization is called word embeddings. Word embeddings are created by feeding text as an input to a pre-trained model – usually trained with help of deep learning methods – which then transforms the textual inputs into vector representations of each word. In word embeddings, words that have similar meanings are represented using similar representations [26].

3.3.5. Feature transformation

Feature transformation refers to the process of transforming features from one format into another, task suitable and convenient, format e.g., transforming numerical features into

categorical. Another type of feature transformation, which is applied to numerical features, is scaling. Scaling is done to ensure that all numerical features are in the same range [29]. For example, min-max normalization scales all values of a feature based on the minimum and maximum values of that feature to obtain a feature in the range [0,1]. On the other hand, z-score normalization uses the mean and standard deviation of the numerical feature to scale all values within the range [-3,3] indicating how many standard deviations a value is below or above the mean [30].

3.3.6. Feature selection

When processing text and transforming it into another form of representation, like a TF-IDF vector, the number of features resulting from that type of transformation can become very high depending on the length of the vocabulary of the corpus, length of the character and word n-grams considered, whether the words are stemmed and lemmatized before constructing the TF-IDF vector, etc. However, some of the resulting features might be redundant and add unneeded complexity to the model. Further, keeping certain features might lower the model accuracy. Therefore, to reduce the dimensionality of the input data and hence lower model complexity and training time one must choose the features of data that contribute most to predicting the outcome variable [31].

There are different approaches for feature selection: embedded methods – feature selection is part of the machine learning algorithm itself and no prior dimensionality reduction needs to be done; filter methods – features are selected before the machine learning algorithm is run, usually based on some statistical metrics such as correlation; and wrapper methods – feature selection is done with help of target machine learning algorithm as a black box for finding the optimal subset of features [32].

3.4. Methods

The task of text classification is a very tedious one and choosing the right method for doing so is of utmost importance. For this research thesis, three algorithms with different underlying structures were used to train the predictive models – namely Support Vectors Machines (SVM), Long Short-Term Memory (LSTM), and Bidirectional Encoder Representations from Transformers (BERT). Referencing back to previously done research [1], [3]–[10], the mentioned methods were among the most popular and hence they have been chosen as models for this work. The details and underlying structure of the three algorithms are explained in the following subsections.

3.4.1. Support vector machines – SVM

A support vector machine (SVM) is a classification algorithm that works by creating a surface that represents a decision boundary between data points belonging to different classes [35]. The name SVM originated from the fact that the decision boundary is represented by using a subset of training examples, called support vectors. Data points are created from examples in the dataset by mapping them to a multidimensional space based on the feature values. The decision boundary that separates data points in different classes is called a hyperplane – an (*n-1*)-dimensional surface where *n* is the number of features that every example has. An example of hyperplanes can be seen in Figure 3.3.

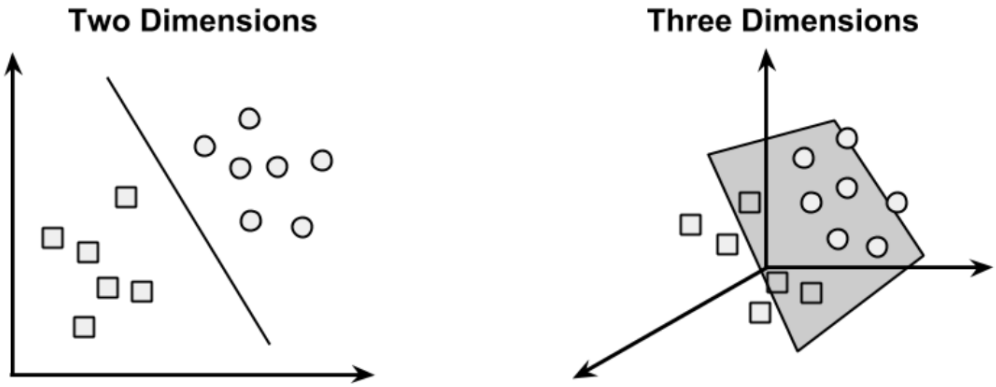

Figure 3.3 Example of hyperplanes. Adapted from [35]

A hyperplane can be constructed in many different ways, as seen in Figure 3.4; however, to ensure that the hyperplane will generalize better to unseen examples a maximal margin hyperplane has to be constructed. A maximal margin hyperplane (MMH) is a hyperplane such that the greatest separation between classes is achieved. For an example of an MMH refer to Figure 3.5.

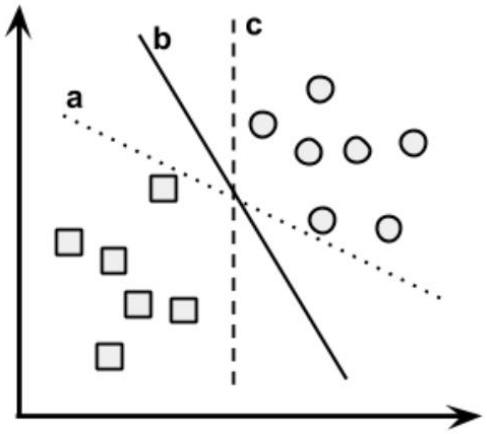

Figure 3.4 Hyperplane construction. Adapted from [35]

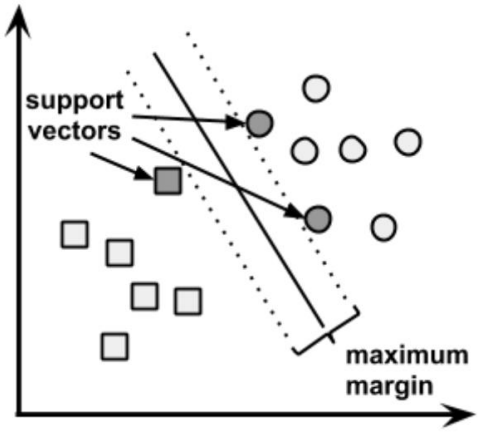

Figure 3.5 Maximal margin hyperplane. Adapted from [35]

This solution is optimal for linearly separable data as it was originally created for it. However, it is highly uncertain that the data will always be linearly separable. To ensure good

enough performance for these cases a slack variable is introduced. This slack variable creates a soft margin that allows data points to fall on the incorrect side of the decision boundary; however, these data points are penalized with a cost value. When this happens, the problem of MMH transforms into the problem of minimizing the cost associated with wrongly classified data points.

Another problem with SVMs is that they perform poorly on data where relationships between variables are non-linear [32]. The solution to this is a process called the kernel trick. This maps the problem into a higher dimension space that may result in non-linear relationships appearing linear.

3.4.2. Long short-term memory – LSTM

One of today's most prominent computational tools for language processing are neural networks [26]. Their name originates from the McCulloch-Pitts neural unit model that strived to depict a simplified version of the human neuron with help of propositional logic. The artificial neuron takes in a set of real-valued inputs, performs a computation on them, and outputs a result.

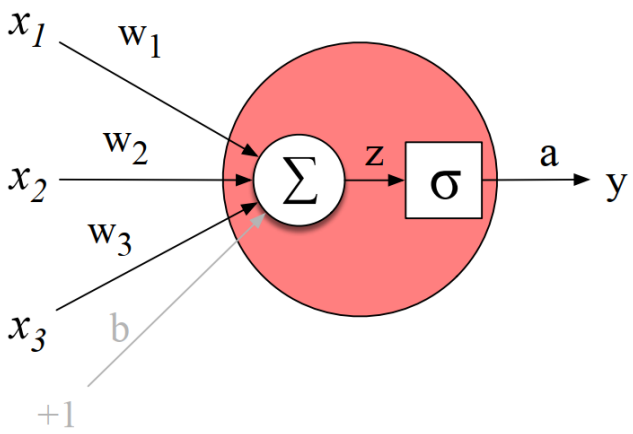

Figure 3.6 Neural unit. Adapted from [26]

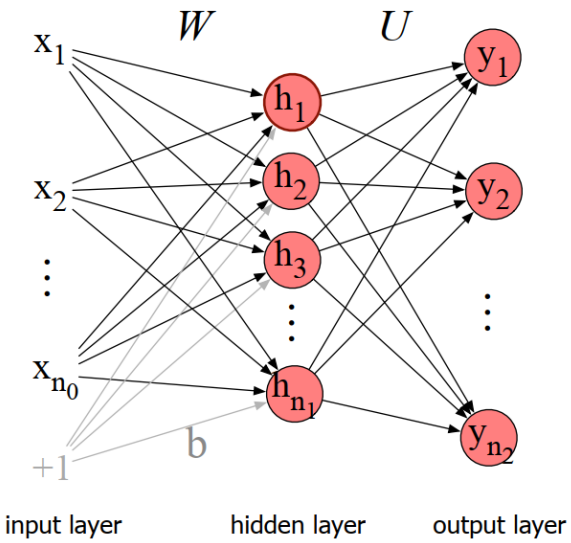

Figure 3.7 Neural network structure. Adapted from [26]

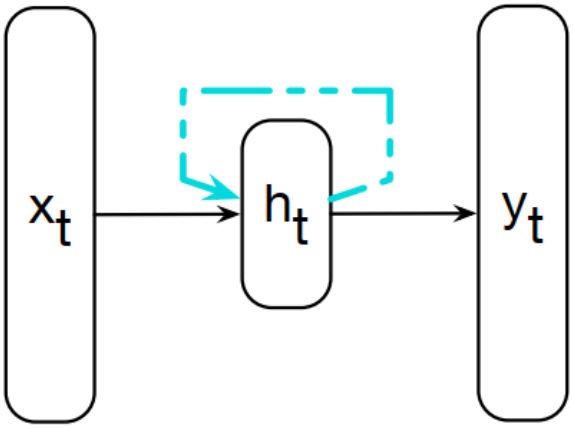

Figure 3.8 Recurrent neural network structure. Adapted from [26]

As shown in Figure 3.6, the neuron takes an input vector, *X*, with values *x1*, *x2*, *x3* and takes the weighted sum of these inputs. The weights of each input are determined by the weight vector, *W*, with weights *w1*, *w2*, *w3*. Then, the bias term, *b*, is added to the obtained sum. Lastly, the resulting sum, *z*, is put into an activation function (in this case it is the sigmoid activation function) which produces a result, *y*.

Neural networks consist of several neurons distributed into interconnected layers, as depicted in Figure 3.7. In the standard neural network, all units in a layer are connected to all other units in the next and previous layers. The results of one layer are propagated to the next layer where they are again processed and their output is fed into the next, higher, layer. This results in a neural network that can be depicted with an acyclic graph and is the simplest kind of neural network – a feedforward network. Another type of neural network where the information can pass in both directions, resulting in a network with cycles, is a recurrent neural network. A recurrent neural network structure is shown in Figure 3.8.

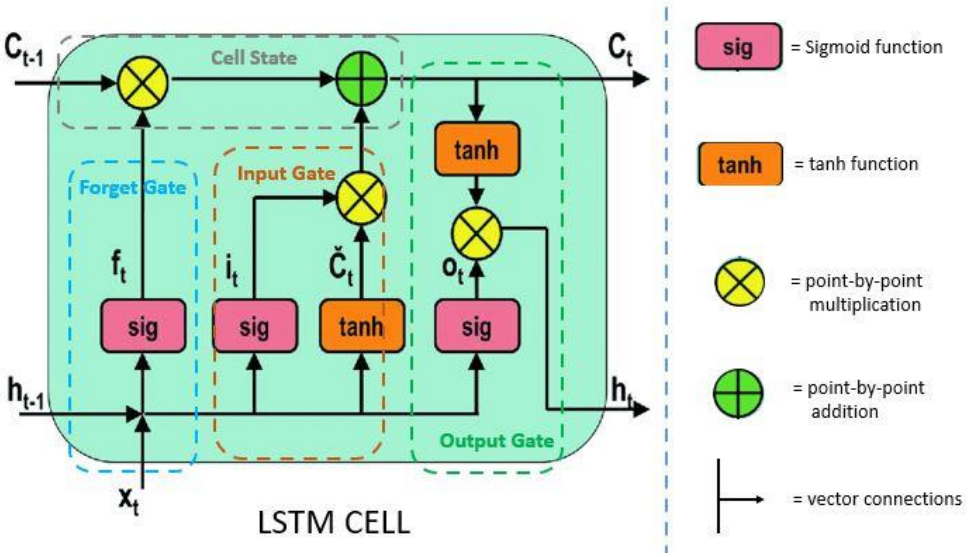

Figure 3.9 LSTM cell structure. Adapted from [36]

Despite the popular use of vanilla RNNs in NLP tasks, one problem with RNNs is that they are not suitable for tasks where they need to keep track of long dependencies as it is difficult for RNNs to carry forward critical information. The hidden layers and their weights need to perform two tasks simultaneously: provide information relevant to the current decision; and update and carry forward information relevant for future decisions. This is a difficult task given the RNN cell's structure. However, in 1997, an extension to the RNN unit was made. With the introduction of gates in the RNN units this problem was solved [37]. An LSTM unit is shown in Figure 3.9. The LSTM unit consists of gates: the forget gate, input gate, output gate, and cell state. The forget gate determines what information will be propagated and what information will be discarded. The input gate processes the current input, $X_{t,}$ and the hidden state, h_{t-1}, to create the current input vector i_t. Cell state computes the current cell state, C_t, given the previous cell state C_{t-1}, forget vector f_t, and input vector i_t. Finally, the output gate generates the new cell hidden state h_t.

3.4.3. BERT

The last machine learning model used for the task of hate speech detection was the BERT model. BERT stands for Bidirectional Encoder Representations from Transformers. BERT is a pre-trained model that, when given text as an input, outputs a vector representation of the input text. The first BERT model was proposed by Google in [38]. The model was trained on two tasks: MLM (Masked Language Modelling) where a part of tokens is "masked", and the model must predict what the masked tokens are; and NSP (Next Sentence Prediction) where the model must predict what sentence B follows the sentence A. The base BERT model was trained on a corpus that combines the BooksCorpus (800M words) and English Wikipedia (2,500M words). Using BERT for NLP tasks consists of two parts: pre-training (done by Google) and fine-tuning (done to accommodate the specific task at hand). Since its' emergence in 2019, multiple BERT models have been trained to support different languages – such as Dutch [39]; Romanian [40]; and Bosnian, Croatian, Montenegrin, and Serbian [41].

The process that BERT performs on text is called word embedding. However, BERT takes a different approach to word embedding. In classic word embeddings the same word is always represented using the same vector. For instance, consider the following two sentences:

"I hate Muslims"

"The hate that Muslims receive is unbearable"

The word *hate* would be represented using the same vector in both sentences even though the contexts in which the word is used are different. This makes it difficult for a machine to distinguish between different connotations of the same word. BERT solves this problem by creating contextual word embeddings – it looks to the right and left of a word and creates a word embedding that depends on the context in which a word is used.

3.5. Initial preprocessing

As mentioned in section 3.2, the text data used in this NLP task was obtained from Twitter. The Twitter text data – tweet – is linguistically unstructured data in the sense that every combination of characters – ASCII and non-ASCII – is allowed if it does not exceed 280 characters, including whitespaces. Twitter users can publish tweets without regard for spelling or grammar rules. Furthermore, Twitter, as a social media platform, has specific features such as tagging of people, usage of hashtags, and resharing, retweeting, of content. Hence the first step was to clean the text data. Data cleaning included the removal of hashtags, tags (of users), URLs, punctuation marks, non-ASCII characters, HTML entities (emojis), redundant white spaces, and lowercasing the documents in the corpus. The next step was to lemmatize the individual tokens in the cleaned text. Lastly, all stop words were removed from the lemmatized text. This concluded the first part of the feature engineering process. The next steps were method-specific and they are explained, together with the used machine learning methods, in the following section. The dataset obtained after initial preprocessing will be referred to as "preprocessed data" in the rest of the chapter.

3.6. Performed experiments

3.6.1. Experiment 1: SVM

The first experiment was conducted with help of the scikit-learn library. To train the model LinearSVC (Linear Support Vector Classification) algorithm was used.

For this experiment, three types of features were extracted from the preprocessed data: TF-IDF vector for the words, TF-IDF vector for POS (Part-of-speech) tags, and sentiment features. The TF-IDF vectors for individual documents were constructed based on the schema explained in section 3.3.4. The next type of features extracted were POS features. POS features

were obtained by determining the POS tag of all tokens in a document. This resulted in a new text feature consisting of POS tags for all tokens in a document in the order that they appear. Then, the obtained text feature was vectorized using TF-IDF vectorization to obtain the TF-IDF vector of POS tags. The last features extracted were sentiment features. Sentiment features give information on whether a document has positive, negative, or neutral sentiment. Feature extraction resulted in a sparse matrix with 11.3k numerical features. To counter this, all features in the matrix were scaled using StandardScaler() from scikit-learn library which performed z-score normalization. Lastly, to reduce the dimensionality of the input data, LinearSVC algorithm was used as a black box to determine and extract the most important features. Following this, the number of features was reduced to 5.3k.

The last step of the first experiment was model training. Model training was done with hyperparameters set to their default values or set to the value recommended in the scikit-library for the specific task.

3.6.2. Experiment 2: LSTM

The second experiment was done using the Keras library. The recurrent neural network that was trained in this experiment consisted of six layers: input layer – which took the preprocessed data as input; vectorization layer – which constructed a vocabulary where every word or n-gram of words has a unique integer index and the indexes are used to transform text input into a vector of integers (the dimension of the resulting vector was restricted to 7000 features); embedding layer – turns positive integers into dense vectors of fixed size (128 for this experiment); two bidirectional LSTM layers – LSTM network layer that allows information to flow in both directions – forward and backward; and dense layer – layer that is densely connected to every neuron in the preceding layer and serves as the final layer of the neural network that uses the soft-max function to do the final prediction.

The model was trained for five epochs – the number of times the model is retrained on the train data – whilst keeping track of metrics for train and validation data.

3.6.3. Experiment 3: BERT

The last experiment was conducted using TensorFlow library. The small-BERT model was used for training as the resources were limited for this research and this was the suggested model for such situations. This model, like all BERT models, has a predefined preprocessor that transforms text input into the appropriate format for input into the BERT encoder model. The model that resulted from this experiment had five layers: input layer; preprocessing layer – BERT model-specific preprocessor; encoding layer – BERT encoder layer; dropout layer – sets weights of randomly selected neurons to 0 to prevent overfitting; and dense layer – makes the final prediction with help of soft-max function. This model, like the LSTM model, was trained for five epochs while keeping track of relevant metrics.

3.7. Summary

Chapter Three started with an inspection of the dataset used for model training and evaluation – the Twitter hate speech dataset. Then, different feature engineering tasks involved in NLP like text cleaning, tokenization, lemmatization, vectorization, feature transformation, and feature selection were explained. The next focus of this chapter was explaining the underlying structures of the different models used in this research thesis – namely SVM, LSTM, and BERT. The chapter concluded with a detailed explanation of the feature engineering process for each experiment together with the details of the implemented methods.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1. Introduction

This chapter is divided into three parts – metrics, results, and discussion. The metrics section explains the different metrics that were used to evaluate a model's performance. The results section presents the metric scores obtained during model evaluation. Lastly, the discussion section compares the models' evaluation results with one another and with previous research; and offers possible reasons for the obtained results as well as suggestions and ideas for future work.

4.2. Metrics

To evaluate a model's performance during the training and evaluation phase different metrics were used. After a model predicts the classes of data points, the predicted classes and the true classes of the data points are compared using a confusion matrix. In general, the dimensions of the confusion matrix depend on the number of distinct classes, *n*, in the classification problem. Hence, a binary classification problem would result in a 2x2 confusion matrix.

Figure 4.1 shows the confusion matrix for a binary classification problem.

		True class	
		A	B
Predicted class	A	True positive (TP)	False positive (FP)
	B	False negative (FN)	True negative (TN)

Figure 4.1 Confusion matrix for binary classification problem

Corresponding metrics are calculated using the information obtained in the confusion matrix. For this thesis, the following metrics were used: sensitivity, specificity, accuracy, and balanced accuracy.

Sensitivity (recall, true positive rate) quantifies what portion of examples belonging to one class were correctly classified as members of that class. Sensitivity is given by Equation 4.1.

$$Sensitivity = \frac{TP}{TP + FN} \tag{4.1}$$

Specificity (true negative rate) is the proportion of examples belonging to the negative class that were correctly classified as negative. Specificity is displayed in Equation 4.2.

$$Specificity = \frac{TN}{TN + FP} \tag{4.2}$$

Accuracy measures what portion of examples were correctly classified. The formula for calculating accuracy is given by Equation 4.3.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4.3}$$

Balanced accuracy in multiclass problems is the arithmetic mean of the sensitivity metric for all *n* classes of a classification problem. The formula for balanced accuracy is presented in Equation 4.4.

$$Balanced\ accuracy = \frac{1}{n}\sum_{i=1}^{n} Sensitivity_i \tag{4.4}$$

4.3. Results

Three different machine learning models were trained for hate speech detection. The details about preprocessing done and the methods used are in Chapter Three. 60% of the available data was used for the training and validation (where applicable) while the remaining 40% was used for model evaluation. The results of the model evaluation are explained in the following subsections.

4.3.1. Experiment 1: SVM

As mentioned in Chapter Three, no hyperparameter tuning was done during this experiment due to limited time and computation resources. Instead, the model was trained with parameters set to their default values. Following model training, the model's performance was evaluated on the test data.

Table 4.1 Model evaluation results for SVM algorithm

Category	Sensitivity	Specificity	Overall Accuracy	Balanced accuracy
Hate	0.7797	0.9689	0.9238	0.8761
Offensive	0.9373	0.9079		
Neither	0.9111	0.9686		

As Table 4.1 shows, the model correctly classified 77.97%, 93.73%, and 91.11% of all examples belonging to the Hate, Offensive, and Neither class, respectively. The specificity is 96.89% for the Hate class, 90.79% for the Offensive class, and 96.86% for the Neither class. The achieved overall accuracy is 92.38% while balanced accuracy is 87.61%.

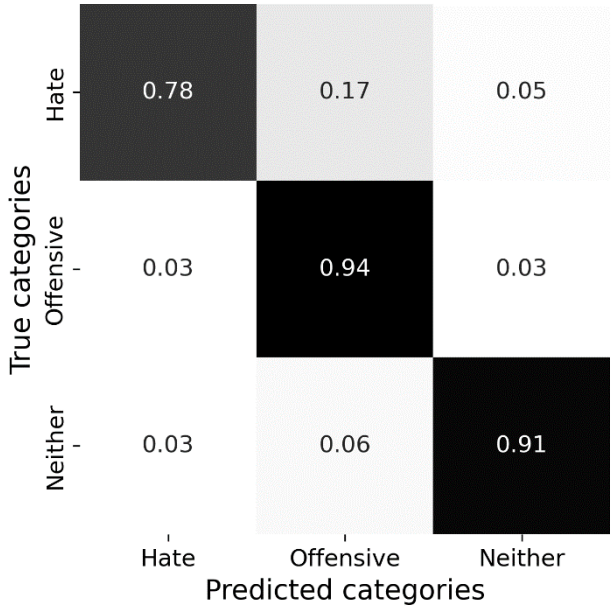

Figure 4.2 Confusion matrix for SVM algorithm

From Figure 4.2**,** it can be seen that 17% of all examples belonging to the Hate class were misclassified as Offensive while 5% were misclassified as Neither. The Neither and Offensive examples were misclassified as Hate in 3% of cases. Lastly, the Neither class was misclassified as Offensive in 6% of cases.

4.3.2. Experiment 2: LSTM

The experiment was conducted, as stated in Chapter Three, with the default hyperparameters available in the Keras library. One of the important hyperparameters used in neural networks is the epoch hyperparameter – the number of times that the model will be retrained on the complete train data. The LSTM network was trained with epoch=5, the default value. However, retaining the model for a high number of epochs can lead to overfitting. To prevent this, it is crucial to keep track of the accuracy and loss values.

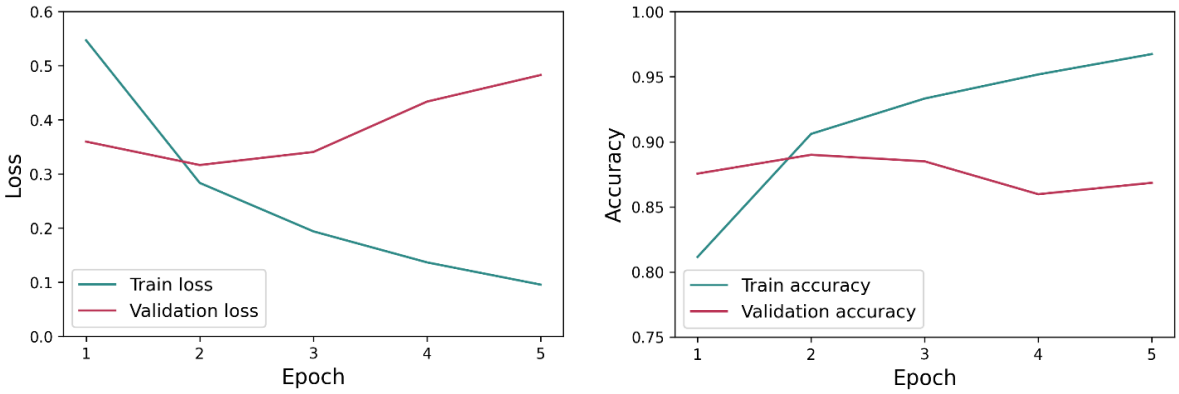

Figure 4.3 Loss and accuracy per epoch for LSTM

From Figure 4.3, it can be seen that as the model was retrained the accuracy kept increasing while the loss kept decreasing; however, only on the train data. On the other hand, for the validation data, the accuracy kept decreasing and the loss kept increasing. To ensure that the maximum possible accuracy is achieved on previously unseen data, the model's weights had to, therefore, be set to the weights obtained after epoch 2. After adjusting the weights of the model, the model was tested on new data. The obtained performance metrics are shown in Table 4.2.

Table 4.2 Model evaluation results for LSTM algorithm

Category	Sensitivity	Specificity	Overall Accuracy	Balanced Accuracy
Hate	0.2465	0.9957	0.9183	0.7117
Offensive	0.9678	0.7734		
Neither	0.9207	0.9681		

Examples belonging to the Offensive and Neither class were correctly classified as such in 96.78% and 92.07% of cases, respectively. On the other hand, the model successfully predicted the Hate class only in 24.65% of cases. Specificity for the Hate class is 99.57%, for Offensive class 77.34%, and for Neither class 96.81%. The overall accuracy achieved by the model is 91.83%. Balanced accuracy, on the other hand, is 71.17%.

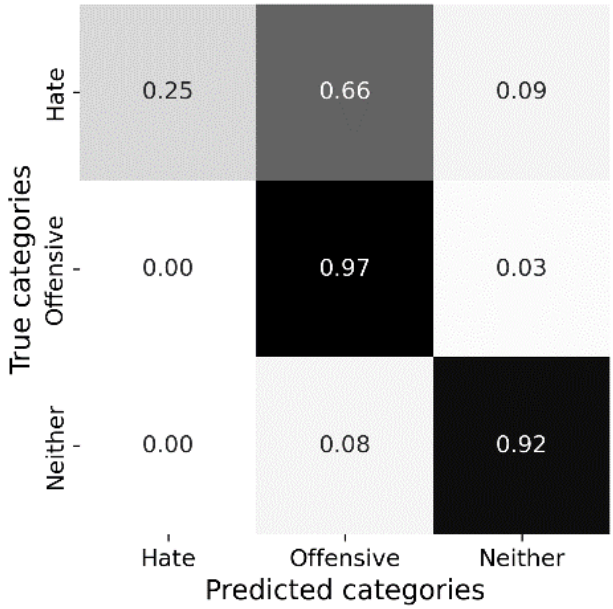

Figure 4.4 Confusion matrix for LSTM algorithm

As can be seen from Figure 4.4, the Hate class was mostly misclassified as Offensive – 66% of the examples being misclassified as Offensive. Further, the Hate class was also misclassified as Neither in 9% of examples.

4.3.3. Experiment 3: BERT

The last experiment – training the BERT model – was done according to the steps explained in Chapter Three. Like the LSTM mode, the BERT model has the epoch hyperparameter. The epoch hyperparameter was left with the default value of 5 during model training. Again, to determine if the model will perform successfully on previously unseen data, it was crucial to keep track of the loss and accuracy during model training.

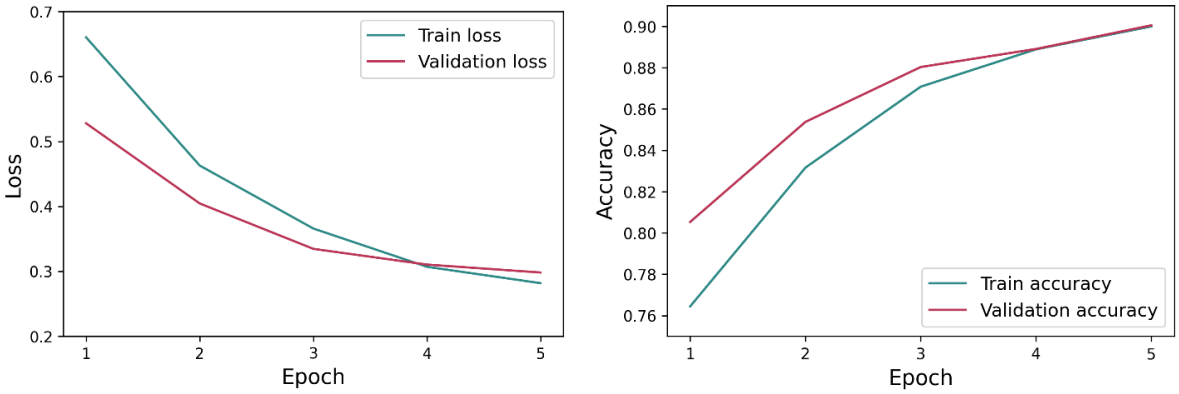

Figure 4.5 Loss and accuracy per epoch for BERT

As can be deducted from Figure 4.5, unlike the LSTM model, the loss kept decreasing with the number of epochs while the accuracy kept increasing – for train and validation data – reaching maximum accuracy and minimal loss after 5 epochs. Hence, the model weights used for evaluating performance on the test data were the weights obtained at the end of training.

Table 4.3 Model evaluation results for BERT algorithm

Category	Sensitivity	Specificity	Overall Accuracy	Balanced Accuracy
Hate	0.3147	0.9801	0.9043	0.7322
Offensive	0.9401	0.8288		
Neither	0.9417	0.9539		

Table 4.3 shows the optimal model successfully labeled the examples belonging to the Offensive and Neither classes in 94.01% and 94.17% of cases, respectively. On the contrary, the model failed to classify examples belonging to the Hate class in 69% of cases – misclassifying 50% of examples as Offensive and 18% as Neither, referring to Figure 4.6. The achieved specificities are 98.01%, 82.88%, and 95.39% for the Hate, Offensive, and Neither class, respectively. The optimal model achieved 90.43% overall accuracy and 73.22% balanced accuracy.

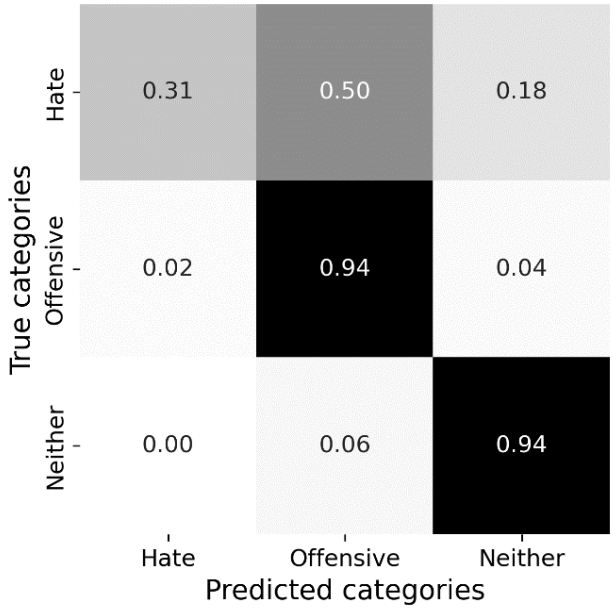

Figure 4.6 Confusion matrix for BERT algorithm

4.4. Discussion

Combining the evaluation results of the models in experiments 1 through 3 Table 4.4 was obtained.

Table 4.4 Accuracy and balanced accuracy for all three models

Model	Overall Accuracy	Balanced Accuracy
SVM	**0.9238**	**0.8761**
LSTM	0.9183	0.7117
BERT	0.9043	0.7322

Before model training and performance evaluation, it was expected that the BERT model would offer the best performance as it is considered a state-of-the-art model for NLP tasks; however, model evaluation revealed that the SVM model performed the best with respect to overall accuracy, with 92.38% overall accuracy, while BERT reached an overall accuracy of 90.43%. Moreover, the LSTM model was also expected to perform better than the SVM

model – as this model enables backward and forward dependency between features; however, the model achieved 91.83% overall accuracy. However, because the dataset in question is imbalanced with respect to the target variable perhaps a more accurate measure of performance is the balanced accuracy – as it treats all classes equally important. However, SVM has also the greatest balanced accuracy – 87.61%. The second-best model, with respect to balanced accuracy, was the BERT model which achieved 73.22%. Lastly, the worst-performing model was LSTM with a balanced accuracy of 71.17%.

Aforementioned, the results of the experiments were unexpected as it was thought that BERT, a state-of-the-art model, would perform the best. One of the possible reasons why the SVM model performed the best is that, unlike the BERT and LSTM models, the SVM model was fed additional features that the two models were not – POS tag and sentiment features. As mentioned in Chapter Three, the BERT and LSTM models were trained on features that were created within the models themselves – in the BERT encoder layer for the BERT model, and the Text vectorization layer for the LSTM model. Moreover, while the LSTM model performs good in theory due to its' ability to depict long dependencies; the depiction of these dependencies is heavily reliant on the computational powers available for model training, consistent with findings in [16]. Another possible reason for the performance of the models is due to the errors present in the dataset itself. As the authors of the dataset mentioned in their paper [11], they have found that the coders who annotated the labels of the individual tweets were perhaps biased and made errors while annotating e.g., they would classify a tweet as hateful because it contained a slur even though the slur was used in a positive context.

Table 4.5 Summary of results obtained by research done on the same dataset

Author	Method	Reported results
T. Davidson et al. [11]	Logistic Regression	Overall Accuracy: 91% Sensitivity for Hate class: 0.61
H. Watanabe et al. [12]	C4.5 Decision Tree	Balanced accuracy: 78% Sensitivity for Hate class: 0.70
R. Mutanga et al. [16]	DistilBERT	Overall Accuracy: 92%
Best performing model in thesis	**SVM**	**Overall Accuracy: 92%** **Balanced accuracy: 88%** **Sensitivity for Hate class: 0.78**

Comparison to previous solutions for detection of hate speech was done only on research that used the same dataset. Furthermore, while the mentioned research conducted multiple experiments with different algorithms only the best-performing methods are reported in Table 4.5. Work [11] done by the original authors of the dataset yielded an overall accuracy of 91% using the Logistic Regression algorithm. The best-performing model obtained during the writing of this thesis, the SVM model, managed to improve upon their results by 1%. Moreover, the SVM model performed better at predicting the Hate class – achieving 0.78 sensitivity for the Hate class compared to 0.61 sensitivity by the authors. [12] reported a balanced accuracy of 78% using the C4.5 algorithm – 10% less than SVM. The sensitivity rate for the Hate class was also lower – 0.70 compared to 0.78 in this thesis. Finally, the DistilBERT classifier in [16] achieved 92% accuracy – same as SVM – however the DistilBERT model is more complex than the SVM model. One of the possible reasons why the model trained in this work performed better is the feature selection done, as part of the feature engineering process, which reduced the dimensionality of the data.

Certainly, there can be improvements to the work done in this research thesis. Firstly, the dataset that was used was obtained from Twitter where users use colloquial language and

are not using proper grammar or paying attention to the correct spelling of words, which was concluded during the exploratory analysis of the data. Hence, the first improvement suggested for such cases is to include spelling correction as a preprocessing step. Secondly, there was class imbalance present in the dataset. Random under-sampling was considered at one point of the research; however, due to fear of shrinking the dataset and losing, possibly, valuable data the idea was not implemented. To resolve this issue perhaps more sophisticated techniques for class imbalance resolution can be used, like Synthetic Minority Oversampling Technique (SMOTE) or data augmentation. Lastly, as aforementioned, no hyperparameter tuning was done during model training due to limited time and computational resources; hence, finding the optimal hyperparameters for the individual models would improve the performance measures.

4.5. Summary

Chapter Four displayed the results obtained during model evaluation of the trained models. The SVM, BERT, and LSTM models achieved 87.61%, 73.22%, and 71.17% balanced accuracy, respectively. All three models showed good results in identifying speech that is considered clean, or offensive. The SVM model was the most successful at identifying hate speech – with a sensitivity of 0.7797 for the Hate class. Comparison to previous research done on the same dataset revealed an improvement in accuracy by 1-10%. Despite the satisfactory results, further improvement might be possible by including spelling correction in the feature engineering phase, solving the class imbalance issue in the dataset, and hyperparameter tuning.

CHAPTER FIVE

CONCLUSION

The problem of hate speech online has been a discussed issue since the emergence of social media. Hate speech can be defined as any kind of communication in speech, writing, or behavior that calls for hate and/or violence towards a group of people based on their race, nationality, religious beliefs, gender, sexuality, or other identifiers. Proving the correlation between hate speech online and real-life hate crimes have been and will continue to be the focus of research studies. Regardless of whether hate speech online actually does inspire real-life crimes, it certainly does evoke the feeling of alienation among the members of communities and minority groups that fall victim to this type of speech. Therefore, identifying and removing this type of online content is crucial in countering this issue. Social medial platforms pay efforts to censor such content with the help of users who report instances of hate speech. The reported content is then manually reviewed to determine whether it is actual hate speech. However, the plethora of content that is produced and reported daily makes manual reviewing a time-consuming and inefficient process. Moreover, the lack of personal bias by the reviewers cannot be completely ensured. This thesis proposed a solution to the problem of hate speech detection based on natural language processing (NLP) methods.

The research in this thesis focused on the detection of hate speech in textual posts obtained from the social media platform Twitter. The used dataset contained 25k examples – 77% of examples were offensive language, 17% were hate speech, and 6% were neither. Three different machine learning algorithms were used to train models – Support Vector Machines (SVM), Long Short-Term Memory (LSTM), and Bidirectional Encoder Representations from Transformers (BERT). NLP-specific feature engineering techniques were used during the data

preprocessing phase – such as lemmatization, tokenization, and TF-IDF. Despite the initial expectation that BERT, a state-of-the-art model, would show the best performance the SVM model outperformed BERT. SVM achieved a balanced accuracy of 87.61%. In particular, it managed to achieve 0.7797 sensitivity for hate speech tweets – which is crucial as hate speech detection was the main focus of this work. BERT and LSTM got 73.22% and 71.17% balanced accuracy, as well as 31.47% and 24.65% sensitivity for hate speech class, respectively. Further, comparison to previously done research on the same dataset revealed an improvement in accuracy of 1-10%. Although the achieved results are satisfactory given the limited time and computational resources, further improvement is certainly possible by expanding the feature engineering phase and using hyperparameter tuning for optimization of the models.

The initial research idea for this thesis was to create a machine learning model that would classify hate speech instances in the Bosnian/Croatian/Serbian (B/H/S) languages; however, no datasets were available for this type of task. Efforts were made to obtain such a dataset by contacting a Bosnian online news forum, Klix.ba, that reported high levels of hate speech on their platform; however, no reply was received from their end and this initial idea was not explored further. Therefore, the idea for future research would be the creation of a dataset that would enable the training of a hate speech classifier for the B/H/S languages. Moreover, the compilation of such a dataset could be a step toward a better understanding of colloquial language in the B/H/S languages and the basis for further research in this area.

REFERENCES

[1] "Prosecutors say Dylann Roof 'self-radicalized' online, wrote another manifesto in jail - The Washington Post." https://www.washingtonpost.com/news/post-nation/wp/2016/08/22/prosecutors-say-accused-charleston-church-gunman-self-radicalized-online/ (accessed Apr. 11, 2022).

[2] "Pittsburgh Synagogue Massacre Suspect Was 'Pretty Much a Ghost' - The New York Times." https://www.nytimes.com/2018/10/28/us/pittsburgh-shooting-robert-bowers.html (accessed Apr. 11, 2022).

[3] K. Müller *et al.*, "Fanning the Flames of Hate: Social Media and Hate Crime *," 2020, Accessed: Apr. 11, 2022. [Online]. Available: https://ssrn.com/abstract=3082972

[4] G. R. Stone, "Hate Speech and the U.S. Constitution", Accessed: Apr. 11, 2022. [Online]. Available: https://chicagounbound.uchicago.edu/journal_articles

[5] "The EU Code of conduct on countering illegal hate speech online | European Commission." https://ec.europa.eu/info/policies/justice-and-fundamental-rights/combatting-discrimination/racism-and-xenophobia/eu-code-conduct-countering-illegal-hate-speech-online_en (accessed Apr. 11, 2022).

[6] "Hate speech and violence." https://www.coe.int/en/web/european-commission-against-racism-and-intolerance/hate-speech-and-violence (accessed Apr. 11, 2022).

[7] "United Nations Strategy and Plan of Action on Hate Speech." https://www.un.org/en/genocideprevention/documents/advising-and-mobilizing/Action_plan_on_hate_speech_EN.pdf (accessed Apr. 11, 2022).

[8] "HATE SPEECH | meaning in the Cambridge English Dictionary." https://dictionary.cambridge.org/dictionary/english/hate-speech (accessed Apr. 11, 2022).

[9] "What is Natural Language Processing? | IBM." https://www.ibm.com/cloud/learn/natural-language-processing (accessed Apr. 11, 2022).

[10] I. Kwok and Y. Wang, "Locate the Hate: Detecting Tweets against Blacks," 2013. [Online]. Available: http://tempest.wellesley.edu/~ywang5/aaai/paper.html.

[11] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated Hate Speech Detection and the Problem of Offensive Language," 2017. [Online]. Available: www.aaai.org

[12] 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.

[13] F. Del *et al.*, "Hate me, hate me not: Hate speech detection on Facebook," 2017, Accessed: Mar. 24, 2022. [Online]. Available: http://www.alexa.com/topsites

[14] B. Evkoski, A. Pelicon, I. Mozetičid, N. Ljubeš Ić, P. K. Novak, and J. Stefan, "Retweet communities reveal the main sources of hate speech," *PLOS ONE*, vol. 17, no. 3, p. e0265602, May 2022, doi: 10.1371/JOURNAL.PONE.0265602.

[15] N. Dennis Gitari, Z. Zuping, H. Damien, and J. Long, "A Lexicon-based Approach for Hate Speech Detection," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 10, no. 4, pp. 215–230, 2015, doi: 10.14257/ijmue.2015.10.4.21.

[16] R. T. Mutanga, N. Naicker, and O. O. Olugbara, "Hate Speech Detection in Twitter using Transformer Methods," *IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 11, no. 9, 2020, Accessed: Apr. 21, 2022. [Online]. Available: www.ijacsa.thesai.org

[17] K. Miok, B. Škrlj, D. Zaharie, and M. Robnik-Šikonja, "To BAN or Not to BAN: Bayesian Attention Networks for Reliable Hate Speech Detection," *Cognitive Computation*, vol. 1, p. 3, Jan. 2022, doi: 10.1007/s12559-021-09826-9.

[18] R. Gomez, J. Gibert, L. Gomez, and D. Karatzas, “Exploring Hate Speech Detection in Multimodal Publications.” Accessed: Mar. 24, 2022. [Online]. Available: https://www.hatebase.org/

[19] D. Sarkar, R. Bali, and T. Sharma, “Feature Engineering and Selection,” in *Practical Machine Learning with Python: A Problem-Solver’s Guide to Building Real-World Intelligent Systems*, D. Sarkar, R. Bali, and T. Sharma, Eds. Berkeley, CA: Apress, 2018, pp. 177–253. doi: 10.1007/978-1-4842-3207-1_4.

[20] J. J. Webster and C. Kit, “Tokenization as the initial phase in NLP,” 1992.

[21] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O’Reilly Media, Inc., 2009.

[22] N. Habash, O. Rambow, and R. Roth, “MADA+TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization,” *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools (MEDsAR)*, Jan. 2009.

[23] C. D. Manning and P. Raghavan, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[24] M. Chary, S. Parikh, A. F. Manini, E. W. Boyer, and M. Radeos, “A Review of Natural Language Processing in Medical Education,” *West J Emerg Med*, vol. 20, no. 1, pp. 78–86, Jan. 2019, doi: 10.5811/westjem.2018.11.39725.

[25] M. Boudchiche and A. Mazroui, “A hybrid approach for Arabic lemmatization,” *International Journal of Speech Technology*, vol. 22, no. 3, pp. 563–573, Sep. 2019, doi: 10.1007/S10772-018-9528-3.

[26] D. Jurafsky and J. H. Martin, *Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition Third Edition draft*, Third Edition Draft. 2022.

[27] H. Yousuf and S. Salloum, “Survey analysis: Enhancing the security of vectorization by using word2vec and CryptDB,” *Adv. Sci., Technol. Eng. Syst. J.*, vol. 5, no. 4, pp. 374–380, 2020.

[28] A. K. Singh and M. Shashi, “Vectorization of Text Documents for Identifying Unifiable News Articles,” *IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 10, no. 7, 2019, Accessed: Jun. 04, 2022. [Online]. Available: www.ijacsa.thesai.org

[29] X. Wan, “Influence of feature scaling on convergence of gradient iterative algorithm,” *Journal of Physics: Conference Series*, vol. 1213, no. 3, p. 032021, 2019, doi: 10.1088/1742-6596/1213/3/032021.

[30] S. Gopal, K. Patro, and K. Kumar Sahu, “Normalization: A Preprocessing Stage,” *International Advanced Research Journal in Science, Engineering and Technology*, vol. 2, 2015, doi: 10.17148/IARJSET.2015.2305.

[31] J. Li *et al.*, “Feature Selection: A Data Perspective,” *ACM Comput. Surv.*, vol. 50, no. 6, Dec. 2017, doi: 10.1145/3136625.

[32] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Pearson, 2005.

[33] R. Gomez, J. Gibert, L. Gomez, and D. Karatzas, “Exploring Hate Speech Detection in Multimodal Publications,” *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1459–1467, Mar. 2020, doi: 10.1109/WACV45572.2020.9093414.

[34] N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, and N. Bhamidipati, “Hate speech detection with comment embeddings,” in *WWW 2015 Companion - Proceedings of the 24th International Conference on World Wide Web*, May 2015, pp. 29–30. doi: 10.1145/2740908.2742760.

[35] B. Lantz, *Machine Learning with R*. Packt Publishing, 2013.

[36] K. Smagulova and A. P. James, “Overview of long short-term memory neural networks,” *Modeling and Optimization in Science and Technologies*, vol. 14, pp. 139–153, 2020, doi: 10.1007/978-3-030-14524-8_11.

[37] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/NECO.1997.9.8.1735.

[38] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, pp. 4171–4186, Oct. 2018, doi: 10.48550/arxiv.1810.04805.

[39] W. de Vries, A. van Cranenburgh, A. Bisazza, T. Caselli, G. van Noord, and M. Nissim, “BERTje: A Dutch BERT Model,” Dec. 2019, doi: 10.48550/arxiv.1912.09582.

[40] M. Masala, S. Ruseti, and M. Dascalu, “RoBERT – A Romanian BERT Model,” pp. 6626–6637, Jan. 2020, doi: 10.18653/V1/2020.COLING-MAIN.581.

[41] N. Ljubešić and D. Lauc, “BERTi\’c -- The Transformer Language Model for Bosnian, Croatian, Montenegrin and Serbian,” *Proceedings of the 8th BSNLP Workshop on Balto-Slavic Natural Language Processing, BSNLP 2021 - Co-located with the 16th European Chapter of the Association for Computational Linguistics, EACL 2021*, pp. 37–42, Apr. 2021, doi: 10.48550/arxiv.2104.09243.