Hate Speech Detection in Social Media

Mona Khalifa A. Aljero

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

Doctor of Philosophy
in
Applied Mathematics and Computer Science

Eastern Mediterranean University
January 2022
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 Doctor of Philosophy in Applied Mathematics and Computer Science.

Prof. Dr. Nazim Mahmudov
Chair, Department of Mathematics

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 Doctor of Philosophy in Applied Mathematics and Computer Science.

Assoc. Prof. Dr. Nazife Dimililer
Supervisor

Examining Committee

1. Prof. Dr. Rashad Aliyev
2. Prof. Dr. Hamza Erol
3. Prof. Dr. Hayri Sever
4. Assoc. Prof. Dr. Nazife Dimililer
5. Asst. Prof. Dr. Müge Saadetoğlu

ABSTRACT

Hate speech is a phenomenal issue for social media platforms. Recently a rapid increase in hate speech happened all over social media platforms.

The aim of this thesis is to improve the performance of the current state-of-the-art for binary text classification in terms of hate speech on social media platforms.

The popularity of social media has grown dramatically in recent years. Because of the ease of use and anonymity of the user identity, this increase coincided with the growth of hate speech on social media platforms.

Due to the increasing propagation of hate speech, these platforms must implement an automatic hate speech identification system. Hate speech recognition is a difficult task in text mining, due to the use of colloquial language, intentional or incorrect spelling variations. The limitation of the message size on social media platforms also complicates the task since the context of the message is not readily available. Various approaches have been applied to text classification using supervised machine learning models, unsupervised machine learning models, and ensemble approaches. Nevertheless, these approaches did not acquire sufficient confidence to be implemented on social media platforms to address the classification of hate speech.

Through this thesis, we proposed two models for detecting hate speech on social media platforms. In the first proposed approach, we developed a model using the novel stacking approach, when two levels of classifiers are used for improving hate speech performance. The second approach based on genetic programming (GP), which is an

optimization technique. In the GP approach, a novel mutation technique that combines the standard one-point mutation with a novel feature mutation is employed.

Both proposed methods were tested on four publicly available datasets of varying sizes. The experimental results show an improvement in the performance over the other used approaches in this thesis. The results show that the GP approach improves the performance on all datasets, compared to the state-of-the-art in terms of F1-score. On the other hand, in comparison with the state-of-the-art, the stacking approach improves the performance on three over four of the used datasets.

Keywords: hate speech, text classification, classifier, classifier ensembles, stacking ensemble, text mining, genetic programming, pattern classification.

ÖZ

Bu tezin amacı, sosyal medya platformlarında nefret söylemi tespit etmek için makine öğrenimi yaklaşımlarının kullanımını araştırarak son teknolojiyi geliştirmektir. Kitlelerin günlük yaşamlarında sosyal medyanın yaygın kullanımındaki keskin artışa paralel olarak sosyal medyanın nispeten kontrolsüz doğası ve kullanıcıların kimliğinin saklanabilmesi nedeniyle üretilen küfürlü ve nefret dolu içerik miktarı da artmaktadır. Nefret söyleminin yayılmasının bireyler ve toplum üzerinde ciddi sonuçları olabileceğinden sosyal medya platformları, nefret söylemini tespit etmek ve önlemek için otomatik nefret söylemi tanımlama sistemleri uygulamalıdır. Bununla birlikte, sosyal medyada nefret söyleminin tespit edilmesi, günlük konuşma dilinin kullanılması, kasıtlı veya kasıtsız yanlış yazım varyasyonları nedeniyle zor bir görevdir. Sosyal medya platformlarında mesaj boyutunun sınırlı olması nedeniyle mesajın bağlamının belirlenememesi de görevi karmaşıklaştırmaktadır. Denetimli makine öğrenimi modellerini, denetimsiz makine öğrenimi modellerini ve topluluk yaklaşımlarını kullanan çeşitli sınıflandırma yaklaşımları önerilmiş olsa da hala nefret söylemi tespiti konusunda elde edilen başarı yeterli değildir.

Bu tez ile sosyal medya platformlarında nefret söylemini tespit etmek için iki model önerilmiştir. Önerilen ilk yaklaşımda hem temel sınıflandırıcıların hem de meta sınıflandırıcının aynı özellik setini kullandığı iki seviyeli bir yığınlama mimarisi önerilmiştir. Önerilen ikinci yaklaşım, bir optimizasyon tekniği olan genetik programlamaya (GP) dayanmaktadır. GP yaklaşımında, standart tek noktalı mutasyonu yeni bir özellik mutasyonu ile birleştiren yeni bir mutasyon tekniği kullanılmıştır.

Önerilen her iki yöntem de çeşitli boyutlarda halka açık dört veri kümesi üzerinde test edilmiş ve deneysel sonuçlar, bu tezde kullanılan diğer yaklaşımlara göre performansta bir gelişme olduğunu kanıtlamıştır. Yığınlama yaklaşımı, kullanılan veri kümelerinin dördünden üçünde en son teknolojinin performansını iyileştirmiştir. Ayrıca sonuçlar, GP yaklaşımının performansının tüm veri kümelerinde en son teknolojiyi aştığını göstermektedir.

Anahtar Kelimeler: nefret söylemi, metin sınıflandırması, sınıflandırıcı, sınıflandırıcı toplulukları, yığınlama topluluğu, metin madenciliği, genetik programlama.

DEDICATION

To My Lovely Husband, Daughters, Parents, and Siblings.

ACKNOWLEDGMENT

All praises to Almighty Allah for blessing me to accomplish this thesis successfully.

I would like to express my special thanks of gratitude to my beloved supervisor Assoc. Prof. Dr. Nazife Dimililer for her supervision, support, guidance, and wise feedback in writing this thesis.

I would like to thank my beloved husband Abdulsalam Elfagieh for his all-time encouragement, support, and help throughout my study. To my beloved daughters Maram, Shaima, and Rahaf for their patience, they are the most beautiful thing that happened to me.

Special thanks go to my dear parents, for their constant prayers, unconditional love, encouragement, and care. Thank you for believing in me to complete this journey.

My thanks go to my siblings for their constant support and love over all these years.

To my friends, thank you for your support, understanding, and interesting discussions during the period of my study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional Long Short-Term Memory
BOW	Bag of Word
CNN	Convolutional Neural Network
DT	Decision Tree
E-tree	Extra Trees
GB	Gradient Boosting
GP	Genetic Programming
GPCE	GP classifier expression
GRU	Gated Recurrent Unit
KNN	K-Nearest Neighbors
LR	Logistic Regression
LSTM	Long Short-Term Memory Network
NB	Naïve Bayes
NLP	Natural Language Processing
RF	Random Forest
RNN	Recurrent Neural Network
SVM	Support Vector Machine
TF-IDF	Term Frequency Inverse Document Frequency
USE	Universal Sentence Encoder
XGB	XGBoost

Chapter 1

INTRODUCTION

1.1 Motivation

Hate speech has increased dramatically as a result of the freedom of speech and, to some extent, the anonymity of users on social media platforms. Hate speech is one of the most important concerns on social media sites. Detecting hate speech is critical for victim protection, which has ramifications for the entire society. Nevertheless, due to the nature of material posted on social media, it is difficult to determine whether or not a text involves hate speech.

Despite an increase in research on automatic detection of hate speech, to the best of our knowledge, a completely automated solution has not been implemented yet. Social media platforms try to solve this issue by mainly relying on user's reports of hate speech and blacklists consisting of hate-related or offensive words. Facebook, YouTube, and Instagram all composed such blacklists. Although Facebook built a model called RoBERTa in 2019 to recognize toxic posts, the reliance on user reports for hate speech detection has not yet been eliminated. On the other hand, Twitter does not apply a blacklist but has recently reused the terms regarding hate speech breaches.

None of the currently used approaches can automatically eliminate the hate speech messages before they are shared. For instance, in the case of a user report, an experienced team is necessary to deal with reported posts or tweets due to the

colloquial language and users' manipulation of keywords. Moreover, before being deleted, the reported posts are visible on the social networking platform and thus can be disseminated.

In machine learning, hate speech detection is treated as a classification task. In that context, each tweet or post is annotated as hateful or not. Hate speech on social media platforms is a complicated subject to categorize due to various characteristics such as the person's hateful comments target as the target varies, so would the employed language and the distinguishing characteristics. Moreover, hate-related words are employed in a variety of forms, spellings, and synonyms. Additionally, there is no standard definition of hate speech. Each social media platform has its definition of hate speech. However, all of them agree that hate speech attacks specific targets based on exact characteristics such as religion, race, etc. The social media platform Twitter defines hate speech: "Hateful conduct: You may not promote violence against, threaten, or harass other people on the basis of race, ethnicity, national origin, sexual orientation, gender identity, religious affiliation, age, disability, or serious disease" [1]. While Instagram defines hate speech as: "Attack anyone based on their race, ethnicity, national origin, sex, gender identity, sexual orientation, religious affiliation, disabilities, or diseases" [2]. In Facebook, the definition is as follows: "Direct attack on people based on what we call protected characteristics- race, ethnicity, national origin, disability, religious affiliation, caste, sexual orientation, sex, gender identity, and serious disease" [3].

Hate speech is highly contextual, the same message shared by two different persons may be interpreted or classified differently. Thus, hate speech classification is determined by various circumstances, including the people's histories, the timing of

the comment, media exposure of the tweet, and a host of other concerns [4]. Furthermore, regardless of the context and time, if the person using hate speech is in the same group as the target, such utterances are not considered hate speech.

Even though in general hate speech messages have aggressive and negative content, relying only on such characteristic is not viable. In certain circumstances, the model may incorrectly classify the text due to a lack of hostility or profanity [5]. All these concerns make hate speech detection even more challenging.

Various approaches have been employed to detect hate speech on social media platforms. There are several popular classifiers used as supervised techniques such as Support Vector Machine (SVM), Logistic Regression (LR), Random Forest (RF), K-Nearest Neighbors (KNN), Naïve Bayes (NB), Convolutional Neural Network (CNN), Decision Tree (DT) [6]. Also, as an unsupervised method: Recurrent Neural Network (RNN) [7]**,** Long Short-Term Memory Network (LSTM) [8], Bidirectional Encoder Representations from Transformers (BERT) [9]. Furthermore, the GP and various ensembling algorithms have also been employed in classification tasks [10,11, 12, 13].

The choice of this topic was motivated by the growing interest in hate speech classification and its influence on society. Since one of the most productive social media platforms is Twitter, the experiments mainly use tweets.

1.2 Thesis Contribution

In this thesis, we address the hate speech classification problem, as is a supervised machine learning task. There are two types of classification: binary (two classes) and multi-class (more than two classes). Our work focuses on a binary classification problem (Hate or Not Hate).

The main contributes of this thesis are as follows:

- Improving current state-of-the-art performance in hate speech terms of F1-score.
- Designing a novel stacked ensembling technique for binary text classification of hate speech on social media.
- Designing another model based on the GP with a novel mutation technique and comparing the experimental results with the results produced from other approaches.
- Evaluating the proposed models for the English hate speech detection task.
- Comparing the performances of two proposed models—Stacking and GP— for the English hate speech detection task on four datasets.

1.3 Thesis Outlines

The remaining of this thesis is organized as follows chapters:

Chapter 2 reviews the literature on hate speech on social media platforms. We start by introducing the hate speech topic. Then we give an overview of the main methods namely preprocessing, feature extractions, and classifiers employed to detect hate speech on social media platforms. Chapter 3 gives background knowledge for hate speech detection approaches in the literature. In Chapter 4, we introduce four publicly available datasets of hate speech on social media, which are used in this thesis. Chapter 5 presents the experimental setting and results for the first model of detection, called the "Stacking" technique. In Chapter 6, we describe the second approach applied for detection, which is the GP approach. The last chapter (Chapter 7) summarizes the thesis findings and presents the future work.

1.4 Thesis Related Publications

1. Hate Speech Detection Using Genetic Programming. Third International Conference on Advanced Science and Engineering (ICOASE2020). University of Zakho, Duhok Polytechnic University, Kurdistan Region, Iraq. Date Added to IEEE Xplore: May 31, 2021 [14].
2. Genetic Programming Approach to Detect Hate Speech in Social Media. IEEE Access. Date of publication August 13, 2021. IEEE Access, 9, 115115-115125 [15].
3. A Novel Stacked Ensemble for Hate Speech Recognition. Applied Sciences. Date of publication December 9, 2021. 11(24), 11684; https://doi.org/10.3390/app112411684 [16].

Chapter 2

RELATED WORK

Within the last decade, hate speech detection has become a popular topic in Natural Language Processing (NLP), and there has been a rapid growth in the interest of this task. There are lots of studies that presented automatic approaches for detecting hate speech in social media. However, there are some challenges facing these approaches, such as various definitions of hate speech, dataset accessibility, response-time requirement.

The majority of people have switched to online communication, where they converse and communicate through social media platforms that have created a diversity of virtual worlds. These platforms express all sorts of user's feelings, including hatred. Hate speech can lead to more serious psychological issues, including hate crimes and suicide.

Many researchers have been investigated the hate speech classification task and have contributed to the literature by proposing hate speech detection models and datasets. In this section, we summarize a number of these studies and highlight the hate classification approaches that have been used.

In 2016, Waseem and Hovy [17] addressed hate speech detection on Twitter as a multi-class text classification. Through this research, the authors also created a publicly available dataset from English Twitter consisting of 16k tweets. Their best system

using a combination of character N-gram and linguistic features and LR as a classifier achieved an F1-score of 73.93%.

The largest publicly available dataset for hate speech on English Twitter was formed in 2017 by Davidson et al. [18] and it consists of 24k tweets. Using this dataset, the authors proposed a multi-class classifier for detecting hate speech. They employed a combination of features, namely TF-IDF, N-gram, and sentiment lexicon, and the LR as a classifier. Even though the classifier performance was 90% F1-score, it was biased to the non-hateful class. Thus, misclassifying tweets continuing hate speech as not hate.

For building a hate speech classifier, Badjatiya et al. [19] used three deep learning approaches: CNN, FastText, and LSTM with word embeddings. The authors used a variety of neural networks, GBDT, and word embedding techniques such as GloVe or random embeddings. The authors used machine learning techniques namely, LR, Gradient Boosted Decision Tree (GBDT), SVM, and RF as baselines. For the evaluation, they used Waseem and Hovy [17] dataset. Their method outperformed the baseline methods, with the ensemble strategy of LSTM + GBDT with random embeddings achieving the best result, with a 93% F1-score, surpassing Waseem and Hovy's system by 10.93%.

Al-Hassan and Al-Dossari [20] proposed a deep learning approach to detect hate speech on the Arabic Twitter dataset. They applied an SVM classifier as a baseline and four deep learning techniques namely, LSTM, CNN-LSTM, CNN-GRU, and GRU. The authors compared the results of baseline and the deep learning approaches in terms of F1-score and showed that all deep learning approaches outperformed the

SVM. Among the deep learning methods proposed by the researches, the CNN-LSTM classifier achieved the highest performance with a 73% F1-score.

In [21], various machine learning techniques namely: SVM, RF, and NB, have been used for hate speech detection on Indonesian Twitter. They addressed detection as a multi-classification hate speech on a set of three public datasets [22, 23, 24] as well as their newly created dataset. They achieved the best result with 77.36% accuracy by utilizing RF as a classifier and unigram as a feature.

One of the most frequently used ensemble algorithms in text classification is stacking. Wolpert [25] proposed the stacking method in 1992. One of the benefits of this method is that the misclassification of each Base-Level classifier is decreased [25]. It has been tested on a variety of real-world datasets and has outperformed stand-alone models [26, 27]. This strategy is also popular in several competitions, such as Kaggle[1]. It has a high rate of adoption across Kaggle users; because of its outstanding results on real-life datasets and adoption by top participants.

Kokatnoor and Krishnan [28] introduced a stacked weighted optimization technique for binary hate speech classification. They used a five classifiers combination using NB, RF, LR, soft voting, and hard voting. The researchers compared the proposed system to stand-alone classifiers, and they found that the proposed system produced better results, with a 95.54% accuracy.

[1] http://kaggle.com

In [29], Gao and Huang employed an ensemble model using LR and LSTM to detect hate speech. The authors evaluated each model separately as a stand-alone classifier then combined them. The ensemble model improved both of the stand-alone models in terms of F1-score by 7% on the Fox News corpus. MacAvaney et al. [30] presented a stacked ensemble. In their model, multiple view SVMs, where each SVM used different features, were combined. They evaluated their model in terms of accuracy and F1-score on four datasets. The authors also explored the difficulties of automatic hate speech identification in social media.

On the HatEval dataset from SemEval-2019 Task 5, Nourbakhsh et al. [11] used an ensemble technique with three classifiers, SVM, RF, and BiLSTM, for hate speech detection. They employed N-gram in the SVM and RF as features, and word embedding in the BiLSTM. The stacked ensemble of the three classifiers had an F1-score of 75.2% on the development set, but obtained F1-score on the test set was 39.2%. The authors point out that the dataset was not shuffled before the split, also the training and testing sets are vastly different. Indurthi et al. [31], on the other hand, used only SVM with one feature, namely universal sentence encoder (USE). They tested their model on a publicly accessible HatEval dataset from SemEval-2019 Task 5, and their model scored 65.1% F1-score.

In [32], Fauzi and Yuniarti proposed an ensemble approach using NB, KNN, RF, and SVM for hate speech detection on the Indonesian Twitter dataset. The authors firstly trained and evaluated each classifier separately as stand-alone classifiers. To implement the proposed ensemble, all classifiers were combined using majority voting with hard voting and soft voting approaches. The authors compared the single classifiers with ensembles and showed that the ensemble approach using majority

voting with the soft voting approach achieved the best performance with an 84.1% F1-score.

Zimmerman et al. [12] introduced an ensemble method in which ten CNNs with varied weight parameters and initializations are combined. For the evaluation of the suggested method, two datasets were used, with an 85:15 ratio for training and testing sets. As a feature, the authors employed word embedding. The proposed method outperformed the individual classifiers by an average of 1.97 percent on the F1-score proving that the ensemble approach can achieve substantial performance over an individual classifier.

In [13], Zhang et al. combined CNN and Gated recurrent unit network (GRU) to present a model to detect hate speech. They used word2vec as a feature. The authors tested their method on a variety of datasets. In comparison to the state-of-the-art and baselines, the authors got better results.

In 1994, Koza [33] first introduced the GP, which is an evolutionary algorithm that develops a solution based on Darwin's principle of survival of the fittest. Individuals represent programs in the GP architecture, and GP improves programs to offer the best solution [34].

Kuo et al. [35] used the GP approach to build a binary classifier for categorizing credit-card applications into approving and disapproving categories. In the GP framework classifiers were expressed as rules, and they were evolved based on correctness and complexity, using novel genetic operators used to exclude or combine the rules based on their correlations, reducing the complexity of the tree.

Muni et al. [36] proposed a novel model for developing a multi-class classifier. Five datasets were utilized to evaluate their model, two of the datasets are binary and three are multi-class. These real-life datasets were about Iris flowers, breast cancer, liver disorder, vehicle, and landsat-TM3 satellite image. They used a GP framework with only a single run to get the optimal solution. The selection method employed was the Roulette wheel to select the trees based on their unfitness.

Kishore et al. [37] proposed a GP model for multiclassification tasks. They evaluated their model on a real-life dataset for flower species classification. The authors designed a GP classifier expression (GPCE) for a specific class, which is recognized as one of the binary classes; all the other classes are recognized as the other class. The strength of association measure was employed with every GPCE to handle the problem of conflict since the GPCE identifies its class based on the strength of association value. Similar to [36] the authors used the Roulette wheel scheme to choose parents for representation and crossover. They were able to achieve 96% accuracy by employing this strategy.

GP proved to be a successful approach in many other classification tasks, including image classification such as in [38], and in diagnosing diseases such as diabetes in [39, 40] and breast cancer in [41] achieving an accuracy of 99.6%, 87.0%, 78.5%, and 98.94%; respectively.

Chapter 3

BACKGROUND

In this chapter, the background, methods, and materials on hate speech recognition as a subtask of text classification are presented. Moreover, brief information on existing strategies for hate speech classification in social media platforms are given. Text classification can be defined as the task of assigning a text to a predefined group of categories or classes based on its context [42]. One of the applications of text classification is the detection of hate speech.

Recently the number of users of social media platforms has significantly increased. This growth in usage and anonymity leads to the proliferation of hate speech extensively on these platforms. Uncontrolled spread of hate speech may result in societal problems or psychological and eventually loss of users. Therefore, the need to an automatic detection method is necessary.

In 2021, Antonakaki et al. [43] ranked Twitter as the world's third most popular social media network. A high number of individuals use Twitter to express their opinions including their hatred and outrage toward another person or group. Specifically, this high traffic results in the collection of a high amount of information. Therefore, most hate speech studies utilize data from Twitter. It should be noted that all of the datasets used in this thesis are texts obtained from Twitter in English.

3.1 Hate Speech Classifications

In general, all hate speech recognition follows the same general workflow shown in Figure 3.1.

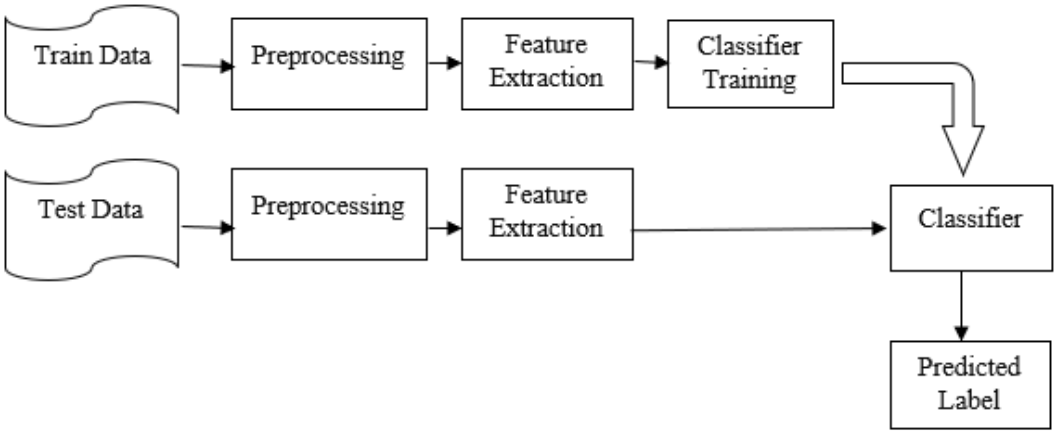


Figure 3.1: Flowchart of the general hate speech recognition

As shown in the figure, the first step is preprocessing where the data is prepared for the subsequent phases. This is followed by features extraction to extract from the data and other sources the properties that will be used to train the classifiers. If a supervised learning approach is used, the data must be labeled with the correct class. After the classifiers are trained “unknown” test data is classified by the classifiers. These steps are discussed in detail in the following subsections.

3.1.1 Preprocessing

Preprocessing is the process of removing extraneous information that does not contribute to the classification and the noise such as misspelling, symbols, and repeated characters or words from the text with the aim of making it more meaningful. This process results in decreasing the size of the dataset, and the training time and is expected to increase the performance.

It is possible to use a combination of various preprocessing steps to clean and prepare the text for the subsequent phases. The most frequently used preprocessing steps are listed below:

- Tokenization: splitting the text into individual tokens or words. Tokenization process is important as generally the list of tokens is used as input of other preprocessing steps as well as the feature extraction phase.
- Removal of the stop-words: eliminating words such as *is, at, you, will, then, the, or, and, what, that* occur very frequently in text and thus carry little or no information. There are several libraries or lists such as NLTK[2], spacy[3], and scikit-LEARN[4] that can be used for stop-word removal. The stop-word list may change from one language to the next. It should also be noted that stop-word lists may be modified to fit the scope of the text mining application.
- Removal of the punctuation: such as (% ^ & () - % ^ & [ ] { } ; : * _" \ , < > . / # $? @ ! ~ ') from the text.
- Stemming: reducing the words in the text to their base form by eliminating affixes from words.
- Lemmatizing: reducing the words in the text to their meaningful base form considering the context including grammatical information about part of speech.
- Letter-case modification: primarily consists of converting all words in the text to the same case such as lower case letters.

[2] https://www.nltk.org/

[3] https://spacy.io/

[4] https://scikit-learn.org/stable/

- Removal of URL and user mentions: eliminating the URL and user mentions as they do not add anything to the text. In some applications instead of removing these special tokens completely, they can be replaced by placeholders.
- Removal of numbers: eliminating the numbers that do not add any useful information to the task.
- Removal of one-character words: generally used to eliminate single characters that may be entered by mistake or may result from other preprocessing steps. These single character words are not expected to carry useful information.
- Removal of white space: elimination of two or more consecutive white spaces.
- Emoji or special symbol processing: deals with the special visual representation used for conveying emotions or sentiments. of the user. There are two options for dealing with such representations. The first option is to replace them with an actual text, while the second option is to eliminate them.

3.1.2 Feature extraction

Feature extraction process transforms the raw data into features that are representative properties of the data samples. The aim of feature extraction is to eliminate redundant and irrelevant information and develop a set of features that could be used instead of the raw data in order to efficiently perform subsequent tasks such as categorization or prediction. The widely used features in the hate speech and text mining domain are briefly explained in the following subsections.

3.1.2.1 Bag of Words (BOW)

BOW feature represents the words in the text without consideration of their location. When the BOW feature is used, the text is represented as an array where each item of the array represents a word. The array values may take the values 1 or 0 representing the existence or absence of a word or a frequency value or integer or floating point values representing the number of occurrences of the words or any other measure. Even though BOW is a robust, simple, and efficient feature, it has some disadvantages such as disregarding the ordering, semantics, and syntactic role of the word, which can lead to misclassification of the text. Therefore, employing BOW as a stand-alone feature can cause an increase in false positive rate which results in a low accuracy in text classification.

3.1.2.2 N-gram

N-gram refers to collecting a sequence of N words or characters in a document. When N is one, it is called unigram when N is 2 it is called bigram and when N is equal to three, it is called trigram. As an example, for 'Hate speech detection …' the character bigrams are *Ha, at, te,* etc and the word bigrams are *Hate speech, speech detection,* etc. The character N-gram proved to be better than the word N-gram approach for abusive language detection as it is more predictive as a feature [44].

3.1.2.3 Word Embedding

Word Embedding feature represents text such as words, phrases, or paragraphs as vectors of real numbers. A large corpus is often used for training word embedding feature [45] where the resulting feature represents words with similar meanings using similar vectors. In contrast to BOW, the words that have similar contexts will be given a similar representation. The two approaches of word embedding that are frequently used are:

- Word2Vec**:** used with unsupervised approaches. It is an effective feature based on a neural network [46]. A unique vector in the numerical form will be assigned to every word. In vector space, the vectors that represent similar words (mathematically) will cluster together. The semantic meaning is taken into account in Word2Vec. In 2014, Tomas Mikolov has developed this technique and applied it to text data to learn word embedding [47]. There are two ways to establish Word2Vec: Continuous Bag of Words (CBOW) and Skip Gram [48].
- Paragraph2vec: assigns a unique vector to each paragraph or document. Every word in the paragraph is assigned a unique vector as well. It differs from the word2vec only in terms of the input since Paragraph2vec employs paragraph and word instead of just a word.

3.1.2.4 Sentence Embedding

Sentence Embedding feature represents text as a vector of real numbers. An example of this feature is the Universal Sentence Encoder (USE) which encodes the text into a high dimensional vector [49]. The inputs of the USE are the text, which can be words or sentences, and the outputs are vectors with 512 dimensions. USE can capture the semantic information of the sentence and represent it as a vector, which provides a good illustration of the context in the entire sentence. This representation of the entire text is difficult to achieve by other features.

3.1.2.5 Sentiment Opinion Words

Sentiment Opinion Words feature takes the text, in the form of a sentence or paragraph, as an input and gives a tag as output. The output reflects the polarity of the text based on the frequency of opinion words. The sentiment polarity of text containing hate speech is mainly negative; therefore, the sentiment feature can be used with

unorganized text such as social media comments and posts to classify unstructured text.

3.1.2.6 Lexical Features

Lexical features are utilized to provide more understanding of a word depending on its context and make it easier to forecast their meanings [50]. Many authors utilized the use of such terms as a feature, attempting to rely on the popular perception that hateful texts involve abuse and trauma words. To collect such kinds of information, lexical features need to comprise such prediction words. There are several freely searchable lists of common hate words from the web. Some lists concentrate on a specific serotype of hate speech, such as the ones used by [51], which are based on gender expression[5], disability[6], and ethnicity[7]. Other authors expressly developed their lists, especially for their study. [52] utilized a lexicon that includes good verbs and adjectives. [53] created a dictionary of abusing and insulting language. The public lexicon dictionary SentiWordNet is described by [54] for opinion mining. [55] compiled a list of verbs that support or promote violent behavior. Moreover, the Stanford NLP group utilized NLP parser[8] to extract the grammar dependency in a sentence.

3.1.2.7 Term Frequency – Inverse Document Frequency (TF-IDF)

TF is a measure of a word corresponding to the frequency of that word in a document. IDF, on the other hand, is calculated using the frequency of a word in a set of documents. IDF feature tends to correlate a low weight to each word if it appears

[5] https://en.wikipedia.org/wiki/List_of_LGBT_slang_terms

[6] https://en.wikipedia.org/wiki/List_of_disability-related_terms_with_negative_connotations

[7] https://en.wikipedia.org/wiki/List_of_ethnic_slurs

[8] https://nlp.stanford.edu/

frequently throughout the documents and high weight to words that appear seldom throughout the documents. Mazzonello et al. [56] used the following formula for TF-IDF:

$$\mathrm{TF}(w, \mathrm{c}) = \frac{|\text{occurrence of w in c}|}{|\text{words in c}|} \tag{1}$$

$$\mathrm{CT} = \sum_{\mathrm{C_w} \in \mathrm{C}} \frac{|\text{documents in } \mathrm{C_w} \text{ in which w appears}|}{|\text{documents in } \mathrm{C_w}|} \tag{2}$$

$$\mathrm{IDF}(\mathrm{w}) = \log \frac{|\mathrm{C}|}{\mathrm{CT}} \tag{3}$$

$$\mathrm{TF} - \mathrm{IDF}\ (\mathrm{w}, \mathrm{c}) = \ \mathrm{TF}(\mathrm{w}, \mathrm{c}) * \ \mathrm{IDF}(\mathrm{w}) \tag{4}$$

where (C): The set of all the classes c.

TF (w, c): The frequency of word w in class c.

$\mathrm{C_w}$: The class c in which word w appears.

IDF (w): The percentage of documents in class c in which w appears.

3.1.2.8 Brown Clustering (BC)

The brown clustering algorithm groups the equivalent distributional information together in one class. In 1992, Brown et al. [57] defined it as clustering words into classes by applying a statistical algorithm for classifying words based on how often they appear. A corpus of vocabulary is the input of this feature and based on the contexts; the output is a hierarchical vocabulary clustering that provides leaves in a binary tree. These leaves have distinctive bit-string (0-1). The vocabularies cluster in classes according to the similarity of the bits. By changing the number of bits to be compared, you can modify the number of clusters. This feature is statistical and does not require a lexicon or training data.

3.1.3 Classifiers

3.1.3.1 Support Vector Machine (SVM)

SVM is one of the most popular supervised machine learning approaches. Joachims [58] developed SVM as a model for high-dimensional feature text categorization in 1998. SVM is non-probabilistic, and in its basic form it is used on data, that can be linearly divided to provide an identification of two classes. SVM maximizes the margin between the two classes. Therefore, it has good generalization and less chance to overfit [59]. The SVM has been seen used effectively for sentiment prediction [60, 61].

3.1.3.2 Naïve Bayes (NB)

NB is a supervised machine learning approach. NB is a probabilistic classifier placed on the Bayes theorem with speculation of the strong independence of all words. This approach computes the word's probability then assigns a tag to the high probability words as an output. NB is simple to design, and it is capable of working on large size of datasets.

3.1.3.3 Logistic Regression (LR)

LR is a supervised machine learning method. This classifier is statistical and probabilistic; it has two types: binary classification and multi-classification. LR considers as an efficient classifier especially on large size of datasets. It is not a complex classifier, also it has less risk of overfitting [62]. LR is suitable for binary classification tasks as it has the sigmoid function, as the LR hypothesis states that the sigmoid function is limited to a value between 0 and 1, where the threshold is 0.5. Therefore, the class is negative in case the sigmoid function is greater than 0.5, and positive when the sigmoid function is less than 0.5. The name of LR came from this function that it has, which is called sigmoid or logistic.

3.1.3.4 Recurrent Neural Network (RNN)

RNN is an unsupervised machine learning method. This classifier is suitable for sequential data. The Recurrent Neural Network contains three layers namely: input, hidden, and output [63]. It can predict a word and understand the context depending on the previous word. Therefore, the inputs of this classifier are the previous output plus the current input which, provides gain with NLP as the previous word has the rest of the meaning of the current word. The drawback of this classifier is the long-term dependencies in a sentence, as it cannot remember this type of sentence. Therefore, it might produce a low performance.

3.1.3.5 Long Short-Term Memory Network (LSTM)

An unsupervised machine learning method is used for classification problems. It is a strong kind of deep neural network; RNN [64]. There are three gates in this classifier namely: input, output, and forget gates. As an advantage of this classifier, it overcomes the problem of remembrance of the long-term dependencies in a sentence through the use of forget gates. Therefore, it can prevent the overfitting issue [65]. Because of the memory cell, this classifier is good at catching long-term dependencies, which sets it apart from RNN.

3.1.3.6 K-Nearest Neighbors (KNN)

KNN is a supervised machine learning method. It is widely known as a lazy learning algorithm due to its simplicity as there is no training stage. It is built on the assumption that similar data are located in the same neighborhood. KNN performance relies crucially on the method used to compute the similarity between various examples [66]. The drawback of this classifier is the long time required to search through the whole training set to find the nearest neighbors to provide a prediction, and this issue gets worse with the increase of the dataset size.

3.1.3.7 Convolutional Neural Network (CNN)

CNN is a supervised machine learning method. It is a special kind of neural network. The architecture of the CNN is hierarchical, typically contains three layers, the first layer is an input layer, the second one is a hidden layer, and lastly an output layer. The hidden layer typically consists of three layers, convolution, pooling, and fully connected layers. There are various variants of the CNN models [67]. CNN was firstly designed for image classification, therefore, in text classification, the words in the text are converted to vectors.

3.1.3.8 Bidirectional Encoder Representations from Transformers (BERT)

An unsupervised machine learning method. It is a neural network for natural language processing, presented in 2018 by Jacob and his team. BERT can apply bidirectional training the prediction of the context of the word will be provided according to the adjacent words [68].

3.1.3.9 Decision Tree (DT)

The DT classification structure is a tree, with root, branch, and leaf nodes. The root represents the entire dataset, while the branches represent the subsets and the leaves indicate the prediction. Starting with the root node, select the suitable feature by attribute selection measurement, and then make a decision by splitting the node using an algorithm. To get to the leaf nodes, repeat these steps. Overfitting is a critical flaw in this model.

3.1.3.10 Random Forest (RF)

RF is a supervised algorithm. It is a large number of DT classifiers evolved by Breiman in 2001 [69]. Every single tree predicts a class; the output will be the class that achieved the majority votes. Stochastic subsets from the training dataset are used to train each tree, with various features used to split the dataset. An advantage of this

phase is to ensure that the single decision trees are unconnected; therefore, the predictions will be more accurate. Unlike the single DT, RF can avoid overfitting. By increasing the number of decision trees, the time of training increases as well.

3.1.3.11 Extra-tree (E-tree)

E-tree is a type of ensemble classification technique that combines the results of numerous less correlated DT ensembled in a forest to provide a classification result. It separates nodes randomly based on given cut-points. This classifier argues that precise randomization of the cut-point together with ensemble averaging should be capable of minimizing variance more effectively than other approaches that have weak random methods [70].

The E-tree model is a version of the RF model that uses the whole training set to train the individual classifiers, whereas the RF trains them on subsets of the training data.

3.1.3.12 Gradient Boosting (GB)

GB is a supervised machine method. Boosting is a type of ensemble technique where a set of weak low performance classifiers are used to build a strong high performance classifiers; the weak classifiers are mostly a group of DT classifiers. The set produces a loss function for each classifier in the set, which determines the model effectiveness. Every classifier in the set will be added to an iteration. This classifier aims to reduce the prediction error as it provides a boosting model. It also offers the benefit of being able to parallelize and distribute training across clusters.

Extreme gradient boosting (XGB) is a special implementation of GB that is faster than GB. XGB uses a penalty function, as well as some computational methods that take advantage of a computer's equipment to speed up algorithms.

3.1.4 Genetic Programming (GP)

This subsection provides an overview of GP concepts and the steps required to adapt GP to the binary classification task.

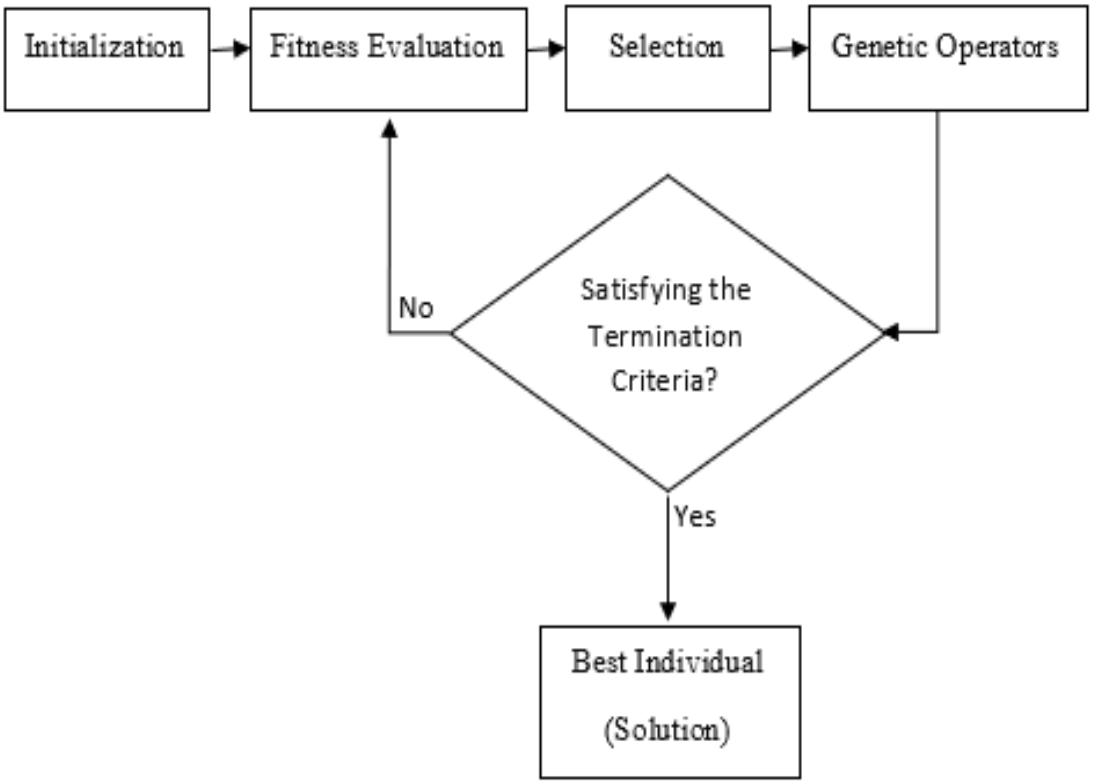


Figure 3.2: Flowchart of the GP approach

GP is considered an extension of the genetic algorithm; where the individuals represent solutions of a problem and they are evolved until an optimal solution is found. The GP process applies a search strategy to find an optimal solution using a computer program without prior knowledge from the user regarding the structure of the solution.

As shown in Figure 3.2. the GP process consists of several steps. The commonly employed steps involved in GP are as follows:

- GP starts with generating an initial random population of individuals or solutions.
- A fitness function is used to evaluate each individual.
- Two parents are chosen.
- Genetic operators are applied to the parents with the probability to create new individuals.
- The previous three steps are repeated until a solution is achieved or the termination criteria are met.
- The best individual of the population according to the fitness function value is returned.

The process starts with the initialization step where an initial population is formed from randomly generated trees. To simulate evolution, the parents are chosen at random, giving priority to the fittest individuals. As a result, each new generation contains the offspring of the randomly chosen fitter parents. Each individual is evaluated using the fitness function. Individuals are evolved via genetic operators such as crossover, reproduction (replication), and mutation to produce new populations from offspring. When the termination criteria are satisfied, the process is terminated, and the best individual is returned.

In the GP framework, each individual or chromosome represents a program or solution. As shown in Figure 3.3, leaf nodes are terminals which consist of variables, constants, and ephemeral and internal nodes are primitives which are in general functions such as mathematical functions or logical or conditional operators as shown in Table 3.1.

Table 3.1: Sample of functions and operators employed by GP approach

Function	Contents
Mathematical Functions	+, -, *, sin, cos, log, sqrt, tan, tanh, and protected division
Logical Operators	AND, OR, NOT
Conditional Operator	if-then-else

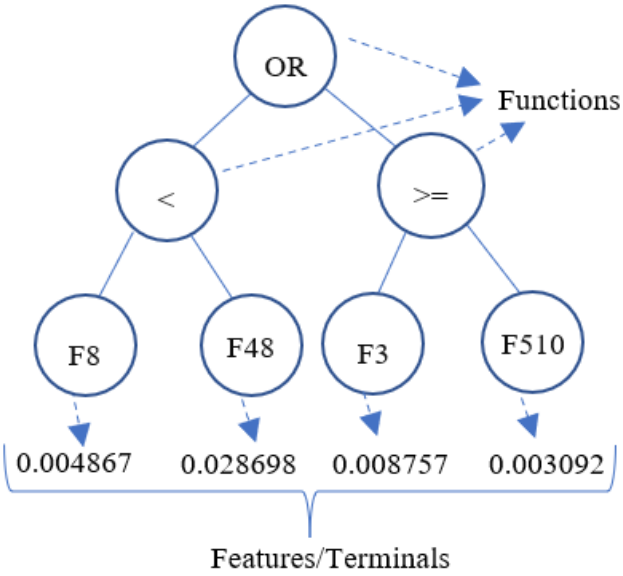


Figure 3.3: A sample classification tree employed by the GP model where the leaf nodes contain features denoted by F##

3.1.4.1 Initialization

The first step of the GP approach is the initialization of the population known as generation zero. To randomly initialize to the population, there are various methods to use. The popular methods Full method, Grow method, and Ramped-half-and-half method are briefly described below:

- Full method: The depth of each tree is equal to the maximum depth used; therefore, all generated trees will have equal depth.

- Grow method: In contrast to the Full method, there is no limit on the depth of the tree. The nodes of the tree select randomly from the leaf or internal nodes. The output is trees with different depths and structures.
- Ramped-half-and-half method: This method presented by Koza [33], is a combination of two methods, Grow and Full methods. Half of the initial population was created by the Grow method and the other half by the full method.

3.1.4.2 Selection

One of the most important processes of the genetic or evolutionary framework is the selection of appropriate parents that will mate and reproduce to form a new generation. It is crucial to ensure that the fittest individuals are more probable to be chosen as parents. But it is also important to ensure diversity. There are various methods for selection; the most common methods are Tournament and Roulette wheel methods [71].

- Tournament selection: In this approach, k individuals in the population are chosen randomly. Among these k individuals, the fittest is selected as a parent. k is known as tournament size and determines the selection pressure where higher selection pressure results in the selection of fitter individuals.
- Roulette wheel selection: The probability of an individual being chosen is directly proportional to its fitness. Thus, the individuals with higher fitness values have a higher probability of being chosen as parents.

3.1.4.3 Genetic Operators

The GP approach used genetic operators to generate a new individual or offspring from the selected parents. There are three popular genetic operators, namely crossover, reproduction, and mutation [33].

- Crossover: The crossover operator is applied on two parents selected using one of the selection methods to generate two new individuals or offsprings. The offsprings are created by exchanging sub-trees between the two parents. In this process, a subtree rooted at a randomly selected point in a parent is exchanging with another subtree thus, chosen in the other parent.
- Reproduction: This operator copies the selected parent to the next generation as a new individual without any alternation. The reason behind that is to reserve the good individuals.
- Mutation: The mutation operator produces a new individual or offspring by replacing a random sub-tree of a single parent with a randomly generated sub-tree. The mutation can lead to rapid growth in the size of the new individual's tree; therefore, in general, a control mechanism to limit the maximum tree depth is employed. The aim of using mutation is to increase diversity among individuals.

3.1.4.4 Fitness Function

Choosing an appropriate fitness function that can be used to identify the fitness or success of individuals is crucial for the GP process. In each generation, the fitness function is used to evaluate the fitness of all individuals. It is used to determine how good the optimal solution obtained from the GP approach is.

3.1.4.5 Termination Criteria

Since the aim of the GP framework is to find a solution to a problem. The evolution may end when the target is achieved. Nevertheless, it is also possible to terminate the GP after the completion of the given number of generations, even when the optimal solution is not found. When the termination criteria is met the solution will the highest fitness is the optimal solution.

3.1.4.6 Weakness and Strengths

There are several strengths and weaknesses of the GP approach. The main strengths may be listed as follows:

- The randomness provided by the GP in different steps such as the initialization, selection of the parents, and choosing the point in parents to apply the genetic operations leads to variety in solutions that allow exploration of a wider search space.
- A comprehensive knowledge of the issues and their solutions is not required in the GP approach.
- The interpretability of the GP solution, as the obtained solution can be understandable in human terms.
- It can eliminate unnecessary features for optimal solution.

While the main weaknesses are:

- Evolution in GP framework may result in a solution that grows unpredictably complicated without any improvement in the performance or fitness, this issue is also known as Bloat [71]. It caused problems in the performance. Bloat can happen because of introns, which are parts of the solution that do not add anything to the solution in terms of fitness; while increasing the size.
- Time-consuming, because of long training time.
- No guarantee for the optimal solution.

3.1.5 Evaluation Metrics

For evaluating the performance of a classifier, numerous performance measures such as Confusion Matrix, Accuracy, Precision, Recall, Specificity, F1-score.

3.1.5.1 Confusion matrix

It is a metric to measure the classifier performance in supervised machine learning. The diagonal values True Positive (TP), and False Negative (FN) give the number of correct predictions. The best confusion matrix has high diagonal values. The other values in the metric represent the False Negative (FN), and False Positive (FP) which are the false predictions.

Table 3.2: Confusion matrix for binary classification

		Predicted Class	
		Positive Class	**Negative Class**
Actual Class	**Positive Class**	TP	FN
	Negative Class	FP	TN

The common measure for the effectiveness of the classifier's performance is accuracy. That is defined as the ratio between the number of correct predictions by the classifier and the total number of cases as show in equation 5.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{5}$$

Inspection of the accuracy formula shows that in the case of an imbalanced dataset, it is possible to get a high overall accuracy even when the minority class is misclassified. Precision or positive predictive value is the proportion of correct positive prediction to all of the positive predicted values. The low Precision may reflect the huge number of FP.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6}$$

Recall or true positive rate is the proportion of correct positive prediction to all of the positive's values. In another way, it is the positive predictions divided by the actual

positive classes. It is also known as sensitivity. The low Recall may reflect the huge number of FN.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7}$$

Specificity or True negative rate measures the correctly negative predicted values.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{8}$$

F1-score is the harmonic mean between Precision and Recall. It is a commonly used metric for measuring the effectiveness of a classifier especially in the case of the imbalanced dataset. F1-score reflects the balance between the Recall and Precision.

$$\text{F1} - \text{score} = \frac{2(\text{Precision} * \text{Recall})}{\text{Precision} + \text{Recall}} \tag{9}$$

Chapter 4

DATASETS

This chapter presents an overview of the datasets used throughout this dissertation. The proposed models were developed and tested using four publicly available English Twitter datasets retrieved from different repositories as shown in Table 4.1. The datasets were selected with different sizes.

Table 4.1: Datasets employed in this thesis

Dataset	**Repositories**
HatEval	http://hatespeech.di.unito.it/hateval.html
Davidson	https://data.world/thomasrdavidson/hate-speech-and-offensive-language
COVID-HATE	http://claws.cc.gatech.edu/covid/#dataset
ZeerakW	https://github.com/ZeerakW/hatespeech/blob/master/NAACL_SRW_2016.csv

4.1 HatEval Dataset

The HatEval dataset was released as a part of a competition in 2019, from SemEval Task 5. It addressed the detection of hate speech against immigrants and women on Twitter. It consists of three sets namely: train, development, and test sets containing 9000, 1000, and 3000 tweets respectively. HatEval is a binary classification dataset annotated with two classes: *Hate* and *Not Hate*. The targets of hateful tweets in this dataset are immigrants and women. In the training set, 39.76% of the tweets are targeted at immigrants and 44.44% are targeted women, while in the testing set, 42.0%

and 42.0% towards immigrants and women respectively. On the other hand, 60.24% and 55.24% of *Not Hate* tweets towards immigrants and women respectively in the training dataset, while in the testing dataset, 58.0% and 58.0% of *Hate* tweets towards immigration and women respectively.

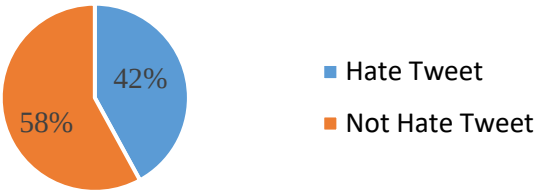


Figure 4.1: Class distribution of HatEval train set

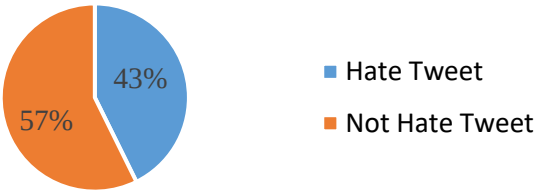


Figure 4.2: Class distribution of HatEval development set

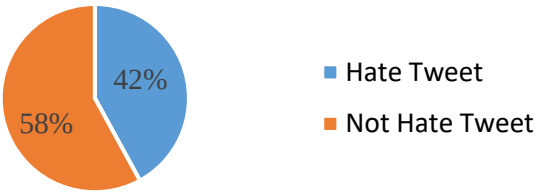


Figure 4.3: Class distribution of HatEval test set

4.2 Davidson Dataset

This dataset is considered the largest publicly available dataset for hate speech. It consists of 24783 tweets, produced in 2017 by Davidson et al. [18]. The annotations

or labels of the Davidson dataset are *Hate*, *Offensive*, and *Neither*. In this dissertation, hate speech detection is treated as a binary classification problem. Therefore, the tweets with labels *Hate* and *Offensive* combined and labeled as *Hate* tweet, while the tweets with the label *Neither* relabeled as *Not Hate* tweet [72].

The Davidson dataset where 17% of the tweets are categorized as *Not Hate*, and 83% are categorized as *Hate* is an imbalanced dataset. Therefore, we applied a synthetically oversampling technique (SMOTE) on the training set to deal with this issue [73].

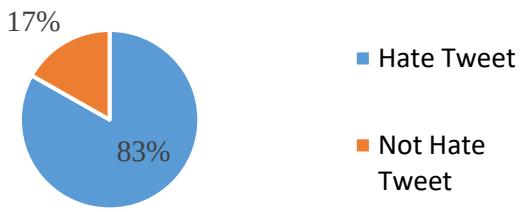


Figure 4.4: Class distribution of Davidson dataset

4.3 COVID-HATE Dataset

COVID-HATE dataset is a small dataset with 2319 tweets released in 2020 related to COVID-19 hateful tweets against Asian people, created by Ziems et al. [74]. This dataset has four labels namely *Hate*, *Counter-Hate*, *Neutral* and *Non-Asian Aggression*, with 678, 359, 961, and 321 tweets respectively. In this dissertation, the tweets with labels *Hate* and *Non-Asian Aggression* combined to *Hate* label, while *Counter-Hate* and *Neutral* set to the *Not Hate* label.

Figure 4.5: Class distribution of COVID-HATE dataset

4.4 ZeerakW Dataset

The ZeerakW dataset compressed 16135 tweets, annotated as *Racism*, *Sexism*, and *Neither* [75]. The tweets with *Racism* and *Sexism* labels combined to perform the *Hate* label, while the tweets with *Neither* label set to *Not Hate* label instead.

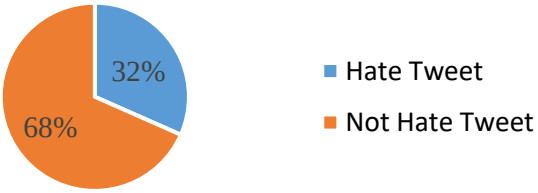


Figure 4.6: Class distribution of ZeerakW dataset

All the datasets have been split into two sets namely, train (80%) and test sets (20%), except for HatEval. HatEval was available as training, development, and test sets.

Table 4.2: Distribution of *Not Hate* and *Hate* tweets in all datasets

Dataset	Size	Not Hate Tweet	Hate Tweet
HatEval	13000	7530	5470
Davidson	24783	4163	20620
COVID-HATE	2319	1320	999
ZeerakW	16135	11033	5102

Chapter 5

STACKING APPROACH

This chapter presents the experimental setup and results for binary hate speech classification using a novel stacking classifier ensemble. The four datasets described in Chapter 4 are employed to show the effectiveness of the proposed stacked ensemble compared to single classifiers, standard stacking ensemble, and simple majority voting ensembles as well as the state-of-the-art. Parts of this chapter were previously published in [16].

5.1 Experimental Settings

We employed the Gensim package in Python using Python 3.7 for the implementation of the proposed approach. The general principle behind the stacked ensemble is to use predictions from the classifiers at the first level as input to the classifiers at the subsequent levels. The novelty of the proposed stacking architecture is that in addition to the predictions of the first level classifiers we use the features input to the first level. The figure and tables of this chapter are from our work published in [16].

HatEval dataset is available as pre-divided into three parts train, development, and test sets. To develop and test the proposed stacking ensemble approach, we divide the remaining three datasets using 80:20 ratio as training and testing datasets respectively. We further divide the training set into training and development sets using a 90:10 ratio respectively.

Table 5.1: Distribution of *Not Hate* and *Hate* tweets in train, test, and development sets of HatEval dataset

	Not Hate Tweet (0)	**Hate Tweet (1)**	**Total**
Train	5217 (58%)	3783 (42%)	9000
Development	573 (57%)	427 (43%)	1000
Test	1740 (58%)	1260 (42%)	3000
Total	7530 (58%)	5470 (42%)	13000

We use the training set to train the first level classifiers referred to as Base-Level classifiers and the development set to train the second level classifier referred to as Meta-Level classifier. The final performance of the proposed system is given using the predictions on the previously unseen test set.

Table 5.2: Distribution of *Not Hate* and *Hate* tweets in the train, test, and development sets of Davidson dataset

	Not Hate Tweet (0)	**Hate Tweet (1)**	**Total**
Train	3010 (17%)	14833 (83%)	17843
Development	327 (17%)	1656 (83%)	1983
Test	826 (17%)	4131 (83%)	4957
Total	4163 (17%)	20620 (83%)	24783

Table 5.3: Distribution of *Not Hate* and *Hate* tweets in train, test, and development sets of COVID-HATE dataset

	Not Hate Tweet (0)	**Hate Tweet (1)**	**Total**
Train	965 (58%)	704 (42%)	1669
Development	106 (57%)	80 (43%)	186
Test	249 (54%)	215 (46%)	464
Total	1320 (57%)	999 (43%)	2319

Table 5.4: Distribution of *Not Hate* and *Hate* tweets in train, test, and development sets of ZeerakW dataset

	Not Hate Tweet (0)	**Hate Tweet (1)**	**Total**
Train	7935 (68%)	3682 (32%)	11617
Development	877 (68%)	414 (32%)	1291
Test	2221 (69%)	1006 (31%)	3227
Total	11033 (68%)	5102 (32%)	16135

Figure 5.1. shows the architecture of the proposed stacking approach which consists of four stages namely preprocessing, feature extraction, and two levels of classifiers. The classifiers of the first level are called Base-Level classifiers. In order to avoid overfitting, we chose a diverse set of machine learning architectures as Base-Level classifiers. We employed seven Base-Level classifiers namely SVM, LR, KNN, NB, RF, E-tree, and XGB. As the second level classifier, also known as Meta-Level classifier, we employed LR due to the shorter training time and the fewer number of parameters [76, 77]. As shown in the figure, the same set of features are used as input to both the Base-Level classifiers and Meta-Level classifier.

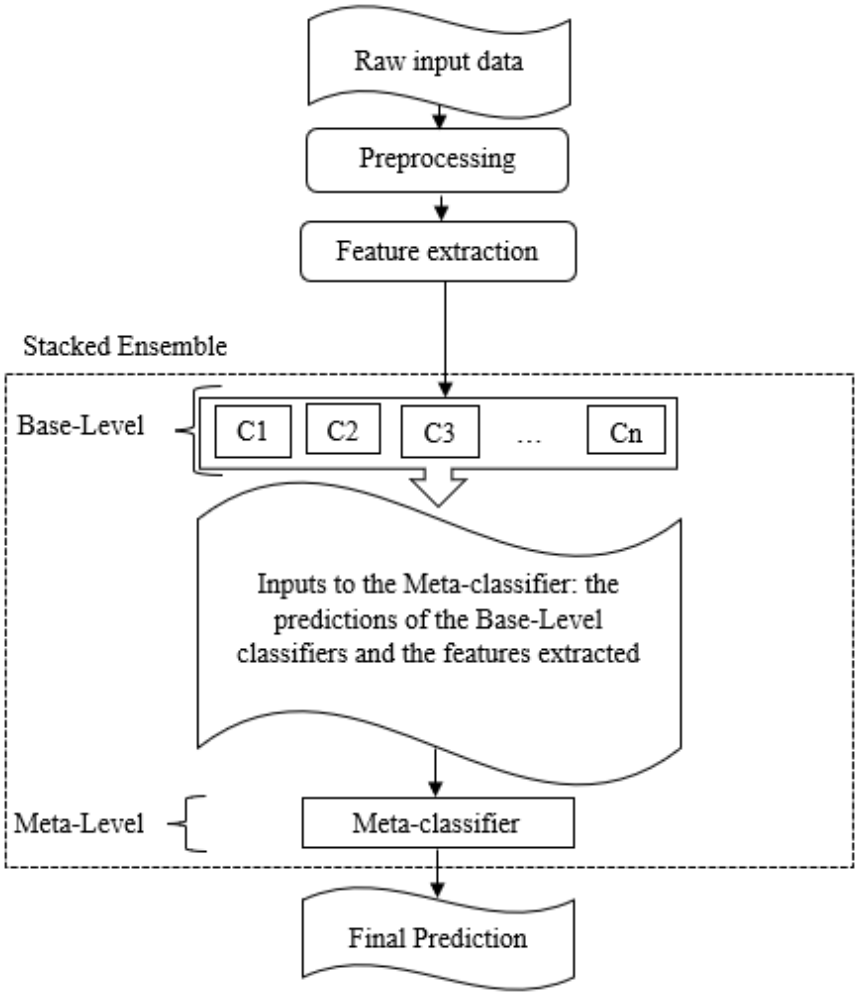


Figure 5.1: Proposed stacking ensemble architecture

At the preprocessing stage first, we applied the following steps on the train, development, and test sets:

- Each tweet is tokenized.
- Hashtag symbols '#' are removed.
- Mentions '@' are removed.
- URLs are removed.
- All characters are changed to lowercase.
- All words are stemmed.
- Words with 1 character are removed.

We used a combination of two features word2vec and USE to train all Base-Level classifiers. The extracted features from the training set are fed as an input to the Base-Level classifiers. To train the Meta-Level classifier, pre-trained Base-Level classifiers make the predictions on preprocessed development set, which forms the input to the Meta-Level classifier together with the features used as input to the Base-Level classifiers. When the Meta-Level classifier training phase is completed the Meta-Level classifier computes the final decision of the proposed stacking ensemble for previously unseen data using the combination of the Base-Level classifier predictions and the features.

5.2 Results and Discussion

5.2.1 Results from proposed stacking experiment

For each dataset, all Base-Level classifiers were trained using a training set and their performance on the development set were presented in Table 5.5. Five different combinations of these Base-Level classifiers were used in the stacking system. It can be seen that in this experiment the best performing classifier for each dataset varies, XGB is always in the top 3.

Table 5.5: F1-score of the Base-Level classifiers on development sets

Base-Level classifier	HatEval	Davidson	COVID-HATE	ZeerakW
KNN	0. 6061	0.9296	0. 6982	0.7480
LR	0. 6434	0.9343	0. 7761	**0.8406**
SVM	0.6180	**0.9517**	0. 7632	0.8398
NB	0. **6845**	0.8669	0. 7201	0.7402
RF	0. 6443	0.8671	**0. 7963**	0.6081
E-tree	0. 5808	0.9358	0. 7745	0.7180
XGB	0. 6745	0.9500	0. 7960	0.8253

Bold entries show the highest performance for each dataset

The development set is used to train the Meta-Level classifier in each proposed stacking ensemble. We explored every possible combination of the seven individual classifiers in our experiment. The five highest performing stacking combinations in terms of F1-score are shown in Table 5.6.

Table 5.6: F1-score of proposed stacking ensemble using different Base-Level classifier combinations on test sets

Base-Classifier combination	**HatEval**	**Davidson**	**COVID-HATE**	**ZeerakW**
SVM, LR, XGB	**0.6551**	**0.9713**	**0. 7301**	**0.7392**
KNN, SVM, NB	0.6405	0.9613	0. 6920	0.7049
LR, NB, RF	0.6407	0.9601	0. 7136	0.7226
KNN, LR, NB	0.6410	0.9710	0. 7261	0.7150
SVM, LR, E-tree	0.6428	0.9710	0. 7255	0.7253

Bold entries show the highest performance for each dataset

As shown in the table for all four test sets used, the Base-Level classifier combination (SVM, LR, XGB) has the highest performance compared to the other combinations. On the test set, Table 5.6 demonstrates the performance of the proposed stacking system.

5.2.2 Results from single classifier experiment

Table 5.7 shows the performance of single classifiers on test sets in terms of F1-score on test sets.

It can be observed that the LR classifier performed the best on two of the datasets, HatEval and ZeerakW. The single classifiers SVM and XGB, respectively, achieved the best F1-score for the other two datasets Davidson and COVID-HATE.

Table 5.7: F1-score of the Base-Level classifiers on test sets

Classifier	HatEval	Davidson	COVID-HATE	ZeerakW
KNN	0.5885	0.9281	0. 6580	0.6037
LR	**0.6407**	0.9365	0. 7146	**0.7179**
SVM	0.6394	**0.9530**	0. 6843	0.7030
NB	0.6024	0.8745	0. 6539	0.5508
RF	0.6016	0.8797	0. 6710	0.6274
E-tree	0.6031	0.9397	0. 6542	0.6747
XGB	0.6353	0.9498	**0. 7246**	0.7058

Bold entries show the highest performance for each dataset

The Base-Level classifiers were tested on the development in the proposed stacking experiment and test sets in the single classifier experiment of each dataset, and the results are given in Tables 5.5 and 5.7, respectively. Close inspection of HatEval test and development sets show that the frequently used words in these two sets are different. This difference is expected to affect the Meta-Level classifier's performance. The discrepancy between the train and test sets has also been indicated by Paula et al. [78] for their system's poor performance. Tables 5.5 and 5.7 showed that, as previously stated, the data in the test, development, and training segments of the dataset differ, which is reflected in the Base-Level classifiers' performance. Furthermore, it may be argued that using the development set to train the Meta-Level classifier is harmful.

5.2.3 Results from majority voting experiment

We conducted an experiment with the majority voting approach using the same combinations of the proposed stacking approach, in order to compare both approaches. Table 5.8 shows that for all datasets, the majority voting ensemble of (LR, NB, RF) consistently has the worst performance. RF is an ensemble strategy that comprises tree estimations, which are provided by each tree's majority voting [79]. Throughout most

scenarios, within (LR, NB, RF) combination, RF concurs with LR, NB, or both in incorrect predictions, resulting in misclassification in the majority voting strategy. Furthermore, RF has better performance with multiclassification tasks [79].

Table 5.8: F1-score of majority voting ensembles on results on the test sets

Classifiers	HatEval	Davidson	COVID-HATE	ZeerakW
SVM, LR, XGB	0. 6296	0.9521	0. 7050	**0.7221**
KNN, SVM, NB	0.6170	0.9509	0. 7188	0.7005
LR, NB, RF	0.6135	0.9309	0. 6950	0.6280
KNN, LR, NB	0.6167	0.9547	**0. 7208**	0.6289
SVM, LR, E-tree	**0.6307**	**0.9552**	0. 7081	0.7210

Bold entries show the highest performance for each dataset

As indicated in Table 5.8, the results obtained using the majority approach are not as good as those obtained using the proposed stacking approach.

On both the HatEval and Davidson datasets, the majority voting ensemble with (SVM, LR, E-tree) provided the maximum F1-score. On the COVID-HATE dataset, the (KNN, LR, NB) combination achieved the best F1-score. Finally, the majority voting ensemble utilizing (SVM, LR, XGB) combination achieved the maximum F1-score for the ZeerakW dataset. Furthermore, LR was still in the combination that achieved the maximum performed F1-score for binary classification on every test set in the majority voting (SVM, LR, XGB), (KNN, LR, NB), and (SVM, LR, E-tree).

5.2.4 Results from standard stacking experiment

The difference between the standard stacking experiment and the proposed stacking is that we only utilize the Base-Level classifier predictions as input to the Meta-Level classifier (in the proposed stacking approach, the Meta-Level classifier employs both

the features and the predictions of the Base-Level classifiers). Table 5.9 shows the binary classification performance of the standard stacking in terms of F1-score for each dataset. On the HatEval dataset, the results obtained by this method are quite low, with the best F1-score for the (SVM, LR, E-tree) combination being 59.43%. The best results were found using the (LR, NB, RF), (SVM, LR, XGB) and (KNN, LR, NB) combinations on the COVID-HATE, Davidson, and ZeerakW datasets, respectively.

Table 5.9: F1-score of standard stacking on the test sets

Classifiers	**HatEval**	**Davidson**	**COVID-HATE**	**ZeerakW**
SVM, LR, XGB	0.5910	**0.9517**	0.7036	0.6657
KNN, SVM, NB	0.5746	0.9434	0.6623	0.6806
LR, NB, RF	0.5804	0.9412	**0.7047**	0.6754
KNN, LR, NB	0.5873	0.9437	0.7046	**0.6815**
SVM, LR, E-tree	**0.5943**	0.9491	0.6859	0.6472

Bold entries show the highest performance for each dataset

In the proposed stacking approach, we found that LR and SVM were the most successful models out of all the models applied on the test sets in terms of the F1-score. The best two combinations in the proposed stacking approach namely, (SVM, LR, E-tree), and (SVM, LR, XGB), already have LR and SVM in combination. These combinations may be the best because they utilized LR as a Meta-Level classifier, but LR was also the highest in the other two experiments that did not employ a Meta-Level classifier. Among Base-Level classifiers, on two datasets namely, HatEval and ZeerakW, LR had the highest F1-score. Additionally, LR was among the combinations that produced the best F1-score on all datasets in the majority voting (SVM, LR, XGB), (KNN, LR, NB), and (SVM, LR, E-tree).

Table 5.10 shows the best binary classification performance on all test sets from the four experiments in this chapter. According to the experimental results on the test sets, our proposed stacking approach outperforms all other three approaches in terms of F1-score. Moreover, it can be seen that the standard stacking approach is consistently ranked as the lowest performing classifier.

Table 5.10: F1-score of the best performing system in the four experiments on test sets

Approaches	HatEval	Davidson	COVID-HATE	ZeerakW
Proposed Stacking	**0.6551**	**0.9713**	**0.7301**	**0.7392**
Standard Stacking	0.5943	0.9517	0.7047	0.6815
Single Classifiers	0.6307	0.9530	0.7208	0.7179
Majority Voting	0.6407	0.9552	0.7246	0.7221

Bold entries show the highest performance for each dataset

Table 5.11 summarize the state-of-the-art on three datasets, HatEval, Davidson, and ZeerakW for binary classification There was no published binary classification work on the COVID-HATE dataset until the time of writing this thesis.

Table 5.11: F1-score and classification model of the state-of-the-art on all datasets

State-of-the-Art	Dataset	F1-Score	Model	Features
Indurthi et al. [31]	HatEval	65.1%	SVM	USE
Zhang et al. [13]	Davidson	94.0%	CNN, GRU (Ensemble)	word2vec
Zhang et al. [13]	ZeerakW	74.0%	LR	N-gram and user features

The result of the state-of-the-art on HatEval dataset was 65.1% achieved by using SVM as a classifier and sentence embedding feature [31]. The highest performance of 74.0% F1-score for binary classification on the ZeerakW dataset was achieved by

applying LR with a combination of N-gram and user's features [13]. Due to the size and the uniformity of the data Davidson dataset contains, many researchers used it for classification problems. The highest published result on this dataset for binary classification is a 94.0% F1-score, which is achieved using CNN and GRU with word embedding as a feature [13].

Table 5.12: F1-score of the state-of-the-art and the proposed stacking ensemble on all datasets

Dataset	State-of-the-art	Proposed Stacking Ensemble
HatEval	65.1 [31]	**65.5**
Davidson	94.0 [13]	**97.1**
COVID-HATE	-	**73.0**
ZeerakW	**74.0** [13]	73.9

Bold entries show the highest performance for each dataset

For each of the four datasets, Table 5.12 compares the proposed approach to the state-of-the-art. On the Davidson dataset, the highest published performance is by Zhang et al. [13]. Our proposed stacking approach outperformed this result by 3.1%, indicating that the stacked ensembling approach we used and the combination of word2vec and USE played a role in increasing the classification performance.

In both HatEval and Davidson datasets, the proposed stacking approach outperformed the state-of-the-art. Only on the ZeerakW dataset, the state-of-the-art outperformed the proposed stacking approach by merely 0.1%. This is due to the fact the state of the art was achieved using feature engineering specifically for the said dataset whereas we opted to use the same features for all datasets. Indeed, for this dataset, Waseem and Hovy [17] discovered that character N-gram outperformed other features and

moreover that other features have a negative impact on performance. Furthermore, they also showed that integrating these features with the LR enhances performance.

Even though the proposed stacking approach outperformed the best performance on the HatEval dataset which is achieved by Indurthi et al. [31] using SVM and sentence embeddings features, the performance gain is not as significant as on the Davidson dataset. Basile et al. [80] present the findings of the SemEval-2019 competition for Task 5 HatEval. Zhang et al. [13] used LR as a binary classification model combined with N-gram and user characteristics to produce the best result for binary classification on the ZeerakW dataset.

It is important to highlight that no individual classifier or combination of classifiers can provide the best performance for all different kinds of datasets. Categories and targets of hate speech, the context of the tweets, the ratio of hate to non-hate tweets show variation among the datasets [81]. These particular characteristics of the datasets make it difficult to adapt the systems with custom built architectures and features to the constantly changing hate speech domain.

Chapter 6

GENETIC PROGRAMMING APPROACH

This chapter presents a genetic programming framework to generate an efficient binary hate speech classifier. A novel hybrid mutation operator is proposed, and the four datasets described in Chapter 4 are employed to show the effectiveness of this new mutation operator. A comparison with the state-of-the-art is also provided. Parts of this chapter were previously published in [15].

6.1 Experimental Settings

We employed a GP framework named Distributed Evolutionary Algorithms in Python (DEAP). The proposed approach was implemented in Python 3.7. A GP model used in this work is based on tree representation where the internal nodes are called primitive nodes and represent mathematical functions, relational operations, logical operators, and a conditional operator, and the leaf nodes called Terminals represent features.

With the proposed GP approach, we used the splitting of 80% training and 20% for testing sets, as shown in the next tables.

The figures and tables of this chapter are from our work published in [15].

Table 6.1: Distribution of *Not Hate* and *Hate* tweets in train, test, and development sets of HatEval dataset

	Not Hate Tweet (0)	**Hate Tweet (1)**	**Total**
Train	5217 (58%)	3783 (42%)	9000
Development	573 (57%)	427 (43%)	1000
Test	1740 (58%)	1260 (42%)	3000
Total	7530 (58%)	5470 (42%)	13000

Table 6.1 is the same as Table 5.1. It is included here to increase the readability of the chapter.

Table 6.2: Distribution of *Not Hate* and *Hate* tweets in the train, test, and development sets of Davidson dataset

	Not Hate Tweet (0)	**Hate Tweet (1)**	**Total**
Train	3350 (16%)	16476 (84%)	19826
Test	846 (17%)	4111 (83%)	4957
Total	4196 (17%)	20587 (83%)	24783

Table 6.3: Distribution of *Not Hate* and *Hate* tweets in train, test, and development sets of COVID-HATE dataset

	Not Hate Tweet (0)	**Hate Tweet (1)**	**Total**
Train	1075 (58%)	780 (42%)	1855
Test	254 (55%)	210 (45%)	464
Total	1329 (57%)	990 (43%)	2319

Table 6.4: Distribution of *Not Hate* and *Hate* tweets in train, test, and development sets of ZeerakW dataset

	Not Hate Tweet (0)	**Hate Tweet (1)**	**Total**
Train	8848 (69%)	4060 (31%)	12908
Test	2226 (69%)	1001 (31%)	3227
Total	11074 (69%)	5061 (31%)	16135

The data is prepared for DEAP by preprocessing and extracting the USE feature. The preprocessing steps with the GP model are tokenization, removing the stop-words, removing the punctuation, and stemming. The proposed GP approach is presented in Figure 6.1.

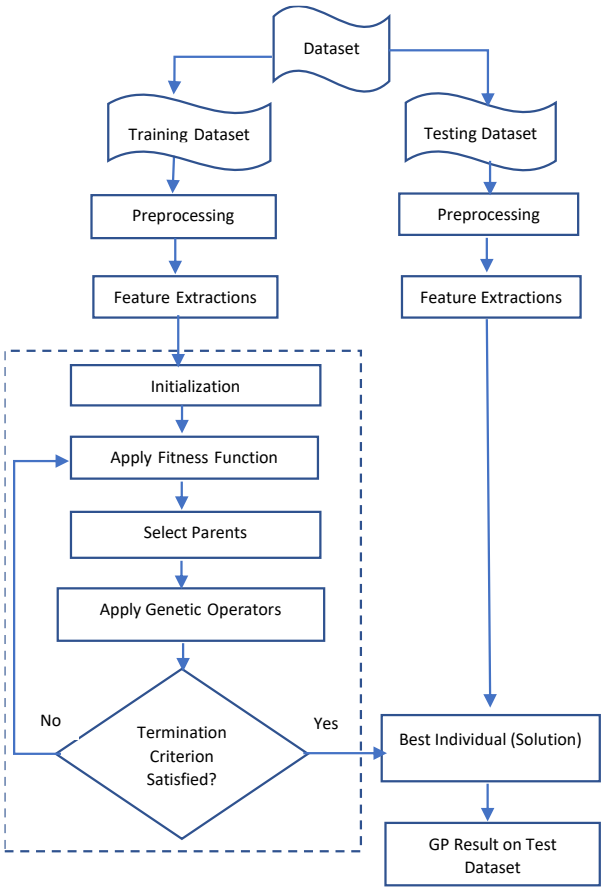


Figure 6.1: Flowchart of the GP approach

6.1.1 GP process

In the proposed GP model, 200 individuals are evolved over 500 generations. Every individual is a hate speech model represented as a tree in the proposed GP model. The Ramped-half-and-half method is used to generate the initial population with an initial maximum depth tree of 2, while the maximum tree depth during the evolution process is 4. The F1-score is used as a fitness function to measure the performance of the

model. Parents for crossover and mutation operators are selected using tournament selection with a tournament size of 20. The genetic operators, mutation, and crossover are used with probabilities of 20% and 80% respectively. The termination criteria used is the completion of the generation; therefore, after 500 generations the best individual will be provided as a result. The GP parameters which are used in the experiment are listed in Table 6.5. A total of 10 runs have been performed due to the randomization of the GP model. The GP model outperformed all other models on all datasets.

Table 6.5: GP parameters

Parameter	**Value**
Population Size	200
Max Number of Generations	500
Max Initialization Depth	2
Max Process Tree Depth	15
Selection Type	Tournament
Tournament Size	20
Crossover Probability	80%
One-point Mutation Probability	20%
Feature Mutation Probability	20%
Creation Method	Ramped-half-and-half
GP Representation	Tree
Max Evolution Depth	4
Stop Condition	Max number of generations
Training Set	80%
Test Set	20%

The GP framework is used with a hybrid mutation operator that makes use of the USE feature. In the hybrid mutation approach, one of the standard one-point mutation and the novel feature mutation techniques are used randomly. The standard one-point mutation produces a new individual by changing a subtree at a randomly chosen node with another randomly generated subtree, as shown in Figure 6.3 (a).

Algorithm 1 Hybrid Mutation

```
1: Procedure Mutation
2:  Initialization
3:  Mp ← Probability of mutation
4:  Fp ← Probability of feature mutation
5:  begin
6:  for each offspring Si do
7:     Generate a random number m ϵ [0,1]
8:     if m < Mp then
9:        Generate a random number n ϵ [0,1]
10:       if n < Fp then
11:             for each feature node Ni do
12:                Generate random feature node Nf
13:                Replace Ni by Nf
14:             end for
15:       else
16:             Select random point (Pt) in Si tree
17:             Generate random subtree (Rs)
18:             Replace the subtree rooted at Pt with Rs
19:       end if
20:    end if
21:  end for
22: end procedure
```

The mutation technique randomly chooses between two mutations standard one-point mutation, novel feature mutation as presented in the algorithm above. The novel feature mutation replaces each feature node randomly with another feature from the 512-dimensional elements of the USE. The novel feature mutation technique only affects the features; the rest of the tree's structure and nodes are unaffected, as shown in Figure 6.3 (b).

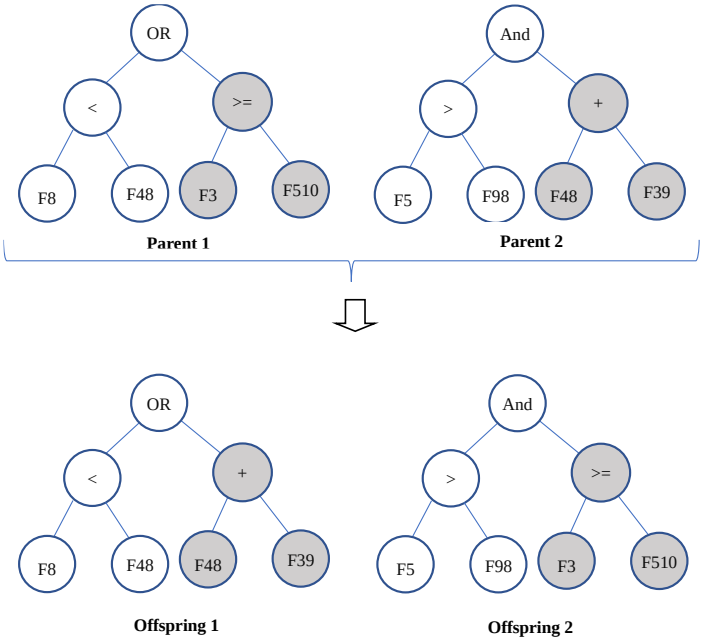
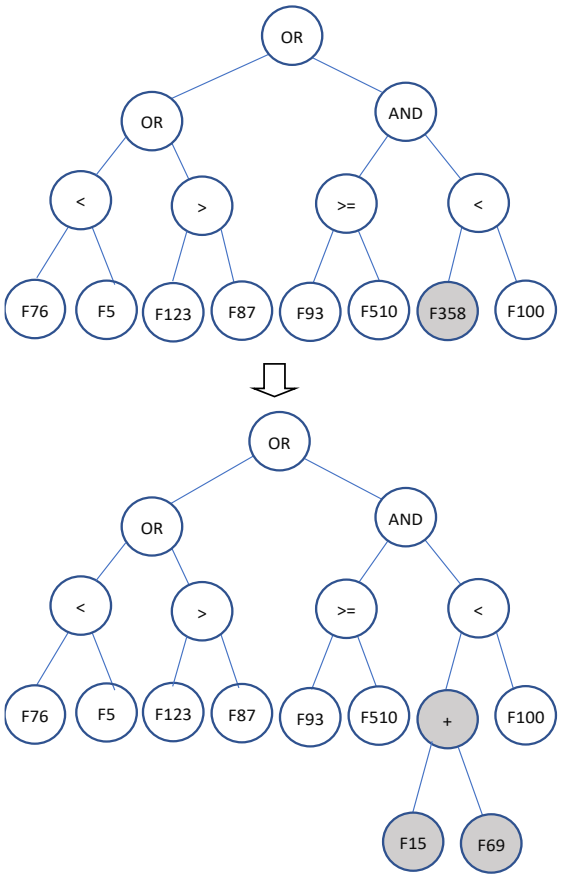
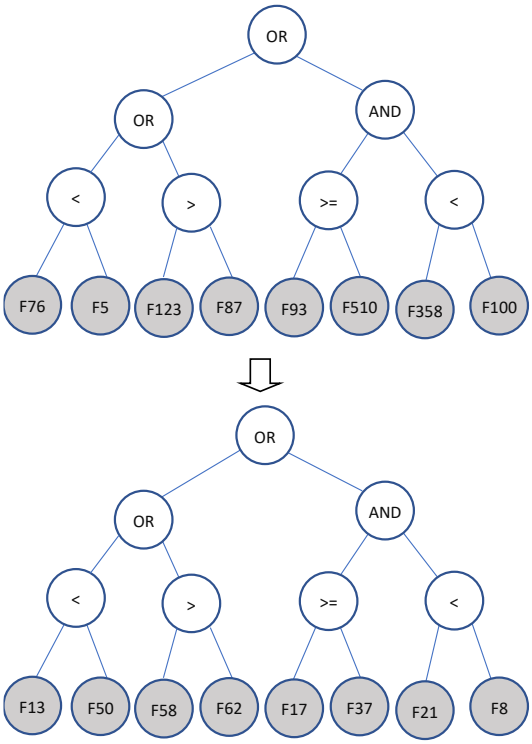


Figure 6.2: Example of a standard one-point crossover



(a) Standard one-point subtree mutation



(b) Novel feature mutation

Figure 6.3: Example of hybrid mutation techniques employed in the proposed GP model

6.2 Results and Discussion

We discuss the results of the proposed GP technique in this section and compare them to other results on the same four datasets.

We applied three different settings of the mutations. We utilized the standard one-point mutation in the first experiment, where a randomly generated subtree replaces a random subtree in the offspring. In the second experiment, we used a novel method named feature mutation, in which the offspring structure is left intact, but all of the feature nodes are changed with a given probability. The third experiment is the proposed approach, where the standard one-point and novel feature techniques of mutation are combined.

Table 6.6 illustrates the comparison between the three experiments and the state-of-the-art on the four datasets, also the features and the classifiers of the state-of-the-art are shown. The proposed GP approach outperforms all other results on the four datasets employed. As Table 6.6 shows the feature mutation technique is only slightly better than standard mutation but the hybrid mutation improves the standard mutation by more than 1%.

Table 6.6: F1-score of the state-of-the art and different GP approaches on all datasets

Dataset Name	**State-of-the-art**	**Standard mutation**	**Feature mutation**	**Hybrid mutation**
HatEval	65.10 [31]	72.27	72.61	**75.30**
COVID-HATE	-	74.47	75.03	**77.59**
ZeerakW	74.00 [13]	74.97	75.15	**77.33**
Davidson	94.00 [13]	93.34	93.45	**94.40**

The proposed GP approach outperformed the state-of-the-art in terms of F1-score on the HatEval test set with 10.2%, moreover, this result outperformed the other results from the other two experiments. The proposed GP approach on COVID-HATE achieved an F1-score of 77.35%, which is higher than the other two experiments of the standard and feature mutations with 74.47% and 75.03% F1-score, respectively, proving the effectiveness of this approach. On the ZeerakW test set, the GP model with a standard one-point mutation achieved a 74.97% F1-score, which is the lowest performance among the GP models; however, there was an improvement in the result achieved with the feature mutation by 0.18%. The best performance was a 77.33% F1-score, which was attained by hybrid mutation. GP proposed approach resulted in an F1-score of 94.40% on the Davidson test set which surpasses the best score by 0.4%.

It is seen that the best results on each dataset were achieved using different architectures and features. In [31], SVM was used as a binary classification model with USE feature on the HatEval dataset. The LR, which is suitable for large datasets, was applied as a classifier on the ZeerakW dataset with a combination of N-gram and user features as features. In the Davidson dataset, a deep learning approach CNN, which is a black-box model, was employed with the word2vec feature by [18].

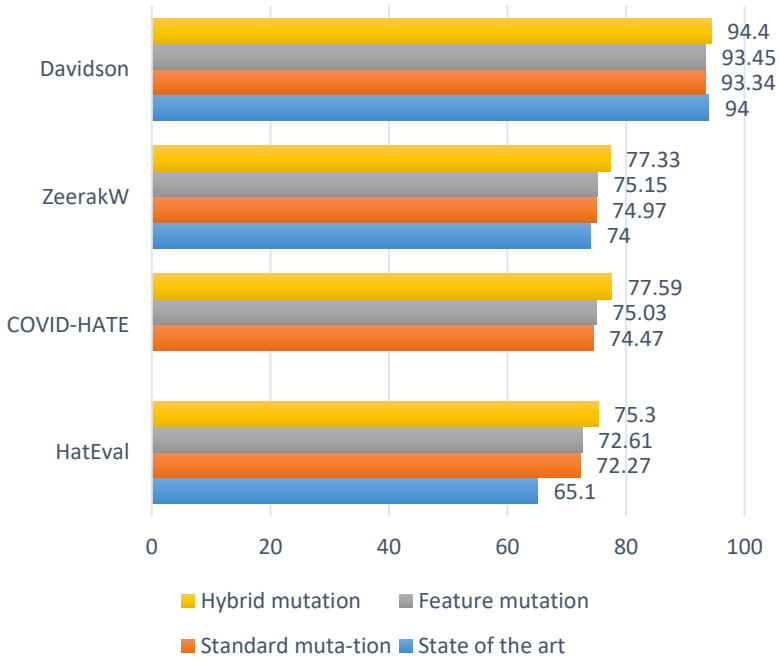


Figure 6.4: The F1-score with different approaches on all datasets

There are many factors that contribute to the scores of a GP system. Increasingly the number of generations allows the evolution of new individuals and it was seen empirically that increasing the number of generations and population size during training improved performance on the test dataset. Furthermore, the dataset's size influences performance because a large dataset provides a broad search space for evolution. Another factor that affects the performance is the ratio of positive and

negative classes in the dataset. Consequently, to balance the unbalanced dataset, an oversampling or undersampling technique is recommended. Furthermore, the GP parameters can be customized to improve performance for each dataset.

The performance of 10 runs of the GP proposed approach on the HatEval test set has been illustrated in Figure 6.5. It can be observed that runs 6, 7, 9, and 10 have the lowest performance of the initial generation among the ten runs with an F1-score of 57.96%. Even though one of the lowest performances in the initial generation was run 10, it achieved the best individual performance by 76.33% F1-score in the final generation on the HatEval test set.

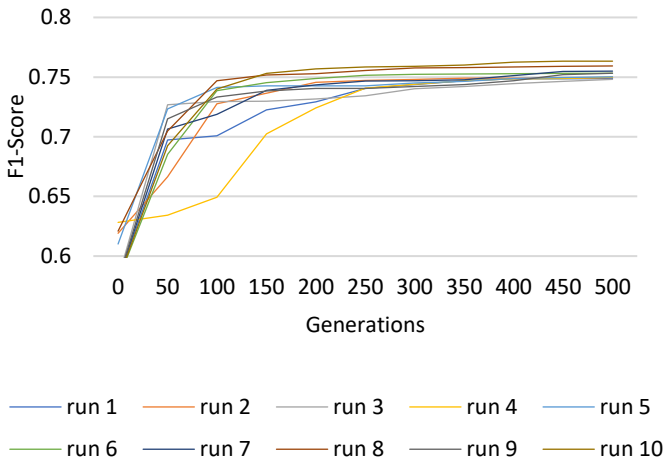


Figure 6.5: F1-score of the best individual in ten runs on HatEval test set

Moreover, Figure 6.6 shows the proposed GP approach on the COVID-HATE dataset with ten runs. The lowest performance in the initial generation was in run 5 with a 56.46% F1-score. On the other hand, run 10 achieved the highest initial generation with a 72.67% F1-score.

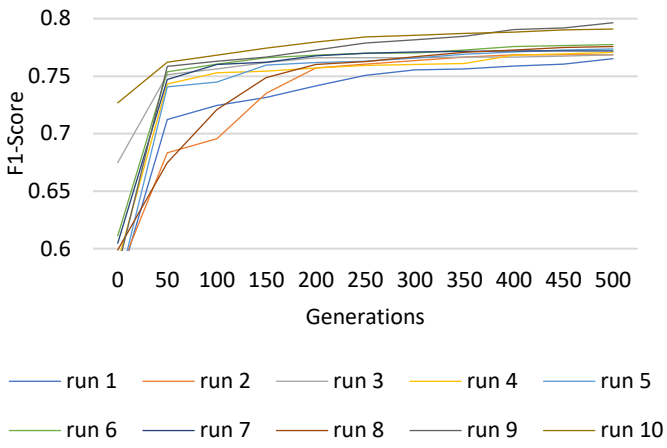


Figure 6.6: F1-score of the best individual in ten runs on COVID-HATE test set

For the ZeerakW test set, the 10 runs are demonstrated in Figure 6.7. The lowest score in the initial generation was in run 9 with a final generation of 77.7% F1-score, while the highest final generation was stated by run 10 with a 78.3 % F1-score.

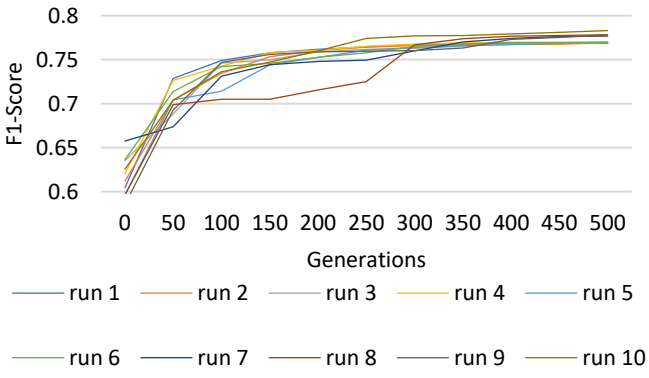


Figure 6.7: F1-score of the best individual in ten runs on ZeerakW test set

Figure 6.8 shows the 10 runs using the proposed approach on the Davidson test set. The highest initial generation was reached by run 4 with 75.76% F1-score and final generation with 94.2%, while the lowest was by run 6 with 57.67% F1-score. Run 9

reached the best final generation by scoring 95.53% F1-score, while the initial generation was 62.7% F1-score.

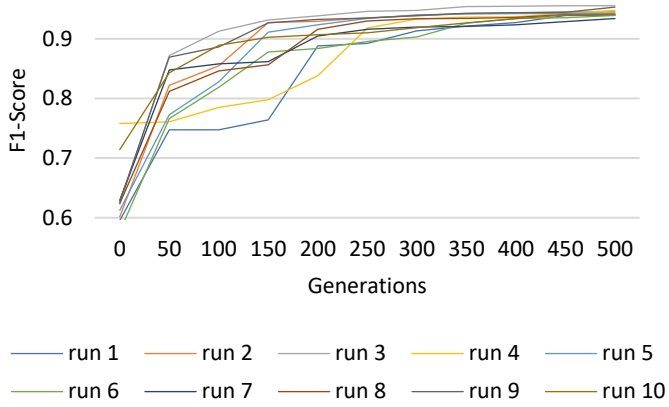


Figure 6.8: F1-score of the best individual in ten runs on Davidson test set

As an observation from all the performance Figures 6.5 to 6.8, the highest initial generation does not necessarily lead to the best final result. From all ten runs on all used test sets, there was just one case of getting the best initial and final generation within the same run which was obtained by run 9 on the Davidson test set. On the other hand, the best final result may occur in the same run that started with the lowest initial score as seen in run 10 on the HatEval test set. In terms of the best individual as shown in Figures 6.5 to 6.8 the convergence of all runs was towards a similar range of performance. This proves that the hybrid mutation produced sufficient variation successfully across the individuals which lead the proposed GP model to the optimal results despite the starting populations. Overall, the experimental results from the proposed GP approach illustrate the superiority and effectiveness of this approach, with outperforming results on all datasets. Figures 6.9 - 6.12 show the standard deviation of the performance in terms of F1-score for the 10 runs on each dataset.

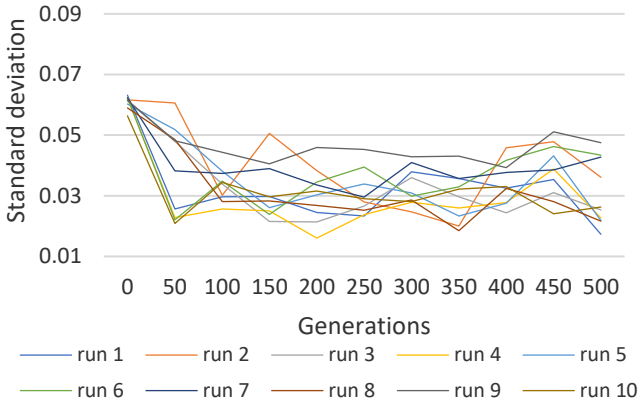


Figure 6.9: Standard deviation comparison of ten runs on HatEval test set

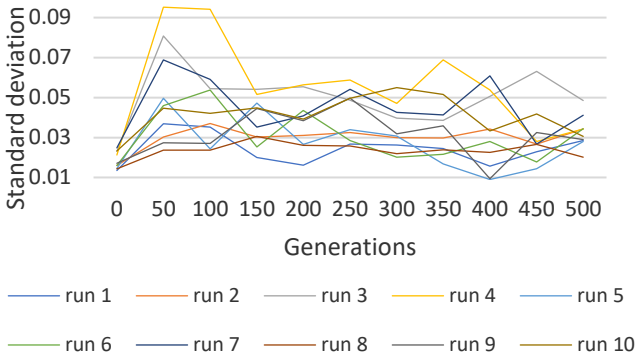


Figure 6.10: Standard deviation comparison of ten runs on COVID-HATE test set

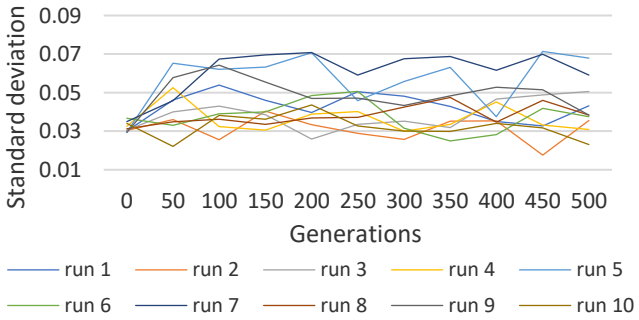


Figure 6.11: Standard deviation comparison of ten runs on ZeerakW test set

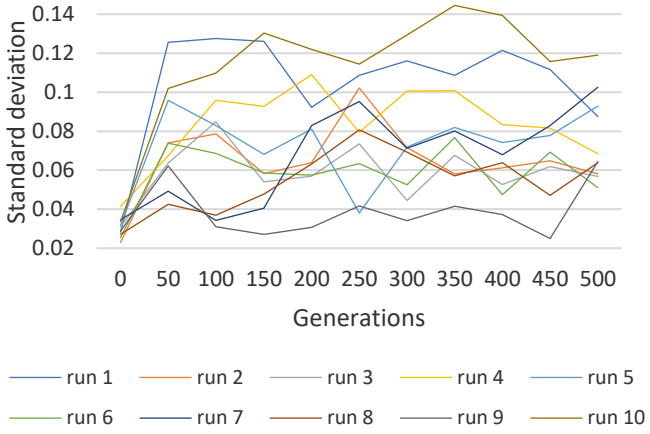


Figure 6.12: Standard deviation comparison of ten runs on Davidson test set

Table 6.7 presents the best standard deviation achieved by the hybrid mutation with 0.2%. The highest standard deviation in all experiments was 0.0613 on the Davidson test set by the hybrid mutation which is still acceptable. This can be related to the huge size of this dataset.

Table 6.7: The standard deviation of the F1-scores on each dataset

	Standard Deviation		
Dataset Name	**Standard Mutation**	**Feature Mutation**	**Hybrid Mutation**
HatEval	0.**0292**	0.0283	0.0221
COVID-HATE	0.**0410**	0.0405	0.0233
ZeerakW	0.0323	0.**0326**	0.0301
Davidson	0.0534	0.0505	**0.0613**

Chapter 7

CONCLUSION AND FUTURE WORK

Various approaches have been employed to classify text into binary or multi classes thus far. Due to the popularity of social media platforms and the danger of hate speech dissemination on them, there is an urgent need to automatically detect hate speech. In this thesis, we introduced two approaches for binary text classification. The performance of the two approaches has been evaluated and tested on four different datasets from English Twitter.

The novel stacking approach employed two levels of classifiers, Base-Level and Meta-Level. The training set is used to train the Base-Level classifiers. The Meta-Level classifier is then trained using the development set. The inputs to the Meta-Level classifier are the predictions of the Base-Level classifiers together with the features used for training the Base-Level classifiers. Our model was used to classify the preprocessed unseen test set. We compared the performance of the proposed approach to the performance of single classifiers, standard stacking, majority voting ensembles, and the state-of-the-art on all of the analyzed datasets. The results showed that discriminating between hateful and non-hateful tweets is a difficult process. On all datasets, the proposed stacking approach achieved the best F1-score when compared to, standard stacking, single classifiers, and majority voting. Moreover, utilizing the same combinations of features (word2vec and USE) as well as the same Base-Level classifiers combination of (SVM, LR, XGB) with Meta-Level classifier LR, the

proposed stacking approach outperformed the state-of-the-art on three out of four datasets. Even though it can be argued that the classifiers are trained in two steps, this is justified because it is not repetitious. The features which may be valuable for correct classification are connected to the structure and composition of the data to be classified, therefore fine-tuning the feature engineering step may increase performance even more. More extensive datasets with a longer timeline would better represent the time-varying character of tweets and help classifiers to capture the dynamic nature of hate speech.

The GP approach is based on the Darwinian principle implemented as a GP approach with a hybrid mutation technique. This technique contributes to the improvement of the evolutionary process, which evolves the best individual. The proposed GP approach used only one feature called USE. The hybrid mutation strategy selects one of two mutations with probability: standard one-point or novel feature mutation. Four publicly available datasets of English Twitter messages of various sizes were used to evaluate the proposed approach. On the four datasets, a comparison of performances with various approaches was performed. The proposed GP model's experimental findings demonstrated its efficacy and superiority, with performance improvement on all datasets that outperformed all existing classification approaches with no post-processing. Additionally, the findings collected using the proposed hybrid approach outperformed those obtained using other methods, with standard mutation and feature mutation. It has been demonstrated that utilizing the proposed GP model improved the F1-score. It was found that expanding the number of generations could lead to even better results.

As future work, there is a number of points we are planning to address. The proposed approaches treats text categorization as a binary classification issue. Adapting the presented approaches to multiclass classification is considered a future study, therefore, our models will be tweaked for the multiclassification challenge, although a binary decision for identifying hate speech on highly active social media platforms is more helpful. Moreover, as future work, we will apply both proposed models on other datasets retrieved from different social media platforms such as Facebook, Instagram, YouTube. Additionally, future research will focus on the imbalanced datasets and the development of a method for resampling, as the real world datasets are mostly imbalanced.

REFERENCES

[1] Twitter, The Twitter Rules. (2021, July 15). Retrieved from https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy.

[2] Instagram, Help Center. (2021, July 15) Retrieved from https://help.instagram.com/477434105621119.

[3] Facebook, Hate Speech. (2021, July 15). Retrieved from https://www.facebook.com/communitystandards/hate_speech.

[4] Schmidt, A., & Wiegand, M. (2017, April). A survey on hate speech detection using natural language processing. In *Proceedings of the fifth international workshop on natural language processing for social media* (pp. 1-10).

[5] Dinakar, K., Reichart, R., & Lieberman, H. (2011, July). Modeling the detection of textual cyberbullying. In *fifth international AAAI conference on weblogs and social media.*

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

[7] Das, A. K., Al Asif, A., Paul, A., & Hossain, M. N. (2021). Bangla hate speech detection on social media using attention-based recurrent neural network. *Journal of Intelligent Systems*, *30*(1), 578-591.

[8] Bisht, A., Singh, A., Bhadauria, H. S., & Virmani, J. (2020). Detection of hate speech and offensive language in Twitter data using LSTM model. In *Recent trends in image and signal processing in computer vision* (pp. 243-264). Springer, Singapore.

[9] Mou, G., Ye, P., & Lee, K. (2020, October). Swe2: Subword enriched and significant word emphasized framework for hate speech detection. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (pp. 1145-1154).

[10] Graff, M., Miranda-Jiménez, S., Tellez, E., & Ochoa, D. A. (2019, June). INGEOTEC at SemEval-2019 Task 5 and Task 6: A genetic programming approach for text classification. In *Proceedings of the 13th International Workshop on Semantic Evaluation* (pp. 639-644).

[11] Nourbakhsh, A., Vermeer, F., Wiltvank, G., & van der Goot, R. (2019, June). sthruggle at SemEval-2019 task 5: An ensemble approach to hate speech detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation* (pp. 484-488).

[12] Zimmerman, S., Kruschwitz, U., & Fox, C. (2018, May). Improving hate speech detection with deep learning ensembles. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

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

[14] Aljero, M. K. A., & Dimililer, N. (2020, December). Hate speech detection using genetic programming. In *2020 International Conference on Advanced Science and Engineering (ICOASE)* (pp. 1-5). IEEE.

[15] Aljero, M. K. A., & Dimililer, N. (2021). Genetic programming approach to detect hate speech in social media. *Ieee Access*, *9*, 115115-115125.

[16] Aljero, M. K. A., & Dimililer, N. (2021). A novel stacked ensemble for hate speech recognition. *Applied Sciences*, *11*(24), 11684.

[17] Waseem, Z., & Hovy, D. (2016, June). Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop* (pp. 88-93).

[18] Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017, May). Automated hate speech detection and the problem of offensive language. In *Proceedings of the International AAAI Conference on Web and Social Media* (Vol. 11, No. 1).

[19] Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017, April). Deep learning for hate speech detection in tweets. In *Proceedings of the 26th international conference on World Wide Web companion* (pp. 759-760).

[20] Al-Hassan, A., & Al-Dossari, H. (2021). Detection of hate speech in Arabic tweets using deep learning. *Multimedia Systems*, 1-12.

[21] Ibrohim, M. O., & Budi, I. (2019, August). Multi-label hate speech and abusive language detection in Indonesian twitter. In *Proceedings of the Third Workshop on Abusive Language Online* (pp. 46-57).

[22] Alfina, I., Mulia, R., Fanany, M. I., & Ekanata, Y. (2017, October). Hate speech detection in the Indonesian language: A dataset and preliminary study. In *2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS)* (pp. 233-238). IEEE.

[23] Putri, T. T. A. (2018). Analisis dan deteksi hate speech pada sosial twitter berbahasa indonesia. *Master's thesis, Faculty of Computer Science, Universitas Indonesia.*

[24] Ibrohim, M. O., & Budi, I. (2018). A dataset and preliminaries study for abusive language detection in Indonesian social media. *Procedia Computer Science*, *135*, 222-229.

[25] Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, *5*(2), 241-259.

[26] Oza, N. C., & Tumer, K. (2008). Classifier ensembles: Select real-world applications. *Information fusion*, *9*(1), 4-20.

[27] Tumer, K., & Oza, N. C. (2003). Input decimated ensembles. *Pattern Analysis & Applications*, *6*(1), 65-77.

[28] Kokatnoor, S. A., & Krishnan, B. (2020, November). Twitter hate speech detection using stacked weighted ensemble (swe) model. In *2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)* (pp. 87-92). IEEE.

[29] Gao, L., & Huang, R. (2017). Detecting online hate speech using context aware models. *arXiv preprint arXiv:1710.07395.*

[30] MacAvaney, S., Yao, H. R., Yang, E., Russell, K., Goharian, N., & Frieder, O. (2019). Hate speech detection: Challenges and solutions. *PloS one*, 14(8), e0221152.

[31] Indurthi, V., Syed, B., Shrivastava, M., Chakravartula, N., Gupta, M., & Varma, V. (2019, June). Fermi at semeval-2019 task 5: Using sentence embeddings to identify hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (pp. 70-74).*

[32] Fauzi, M. A., & Yuniarti, A. (2018). Ensemble method for indonesian twitter hate speech detection. *Indonesian Journal of Electrical Engineering and Computer Science*, *11*(1), 294-299.

[33] Koza, J. R. (1994). Genetic programming: on the programming of computers by means of natural selection. *Statistics and computing*, *4*(2), 87-112.

[34] Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekart, A., Esparcia-Alcázar, A. I., ... & Yannakakis, G. N. (Eds.). (2010). *Applications of Evolutionary Computation: EvoApplications 2010: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC, Istanbul, Turkey, April 7-9, 2010, Proceedings, Part I* (Vol. 6024). Springer.

[35] Kuo, C. S., Hong, T. P., & Chen, C. L. (2007). Applying genetic programming technique in classification trees. *Soft Computing*, *11*(12), 1165-1172.

[36] Muni, D. P., Pal, N. R., & Das, J. (2004). A novel approach to design classifiers using genetic programming. *IEEE transactions on evolutionary computation*, *8*(2), 183-196.

[37] Kishore, J. K., Patnaik, L. M., Mani, V., & Agrawal, V. K. (2000). Application of genetic programming for multicategory pattern classification. *IEEE transactions on evolutionary computation*, *4*(3), 242-258.

[38] Chen, Z., & Lu, S. (2007, October). A genetic programming approach for classification of textures based on wavelet analysis. In *2007 IEEE International Symposium on Intelligent Signal Processing* (pp. 1-6). IEEE.

[39] Aslam, M. W., Zhu, Z., & Nandi, A. K. (2013). Feature generation using genetic programming with comparative partner selection for diabetes classification. *Expert Systems with Applications*, *40*(13), 5402-5412.

[40] Aslam, M. W. (2013). *Pattern recognition using genetic programming for classification of diabetes and modulation data* (Doctoral dissertation, University of Liverpool).

[41] Guo, H., & Nandi, A. K. (2006). Breast cancer diagnosis using genetic programming generated feature. *Pattern Recognition*, *39*(5), 980-987.

[42] Dalal, M. K., & Zaveri, M. A. (2011). Automatic text classification: a technical review. *International Journal of Computer Applications*, *28*(2), 37-40.

[43] Antonakaki, D., Fragopoulou, P., & Ioannidis, S. (2021). A survey of Twitter research: Data model, graph structure, sentiment analysis and attacks. *Expert Systems with Applications*, *164*, 114006.

[44] Mehdad, Y., & Tetreault, J. (2016, September). Do characters abuse more than words?. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue* (pp. 299-303).

[45] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[46] Lee, H. S., Lee, H. R., Park, J. U., & Han, Y. S. (2018). An abusive text detection system based on enhanced abusive and non-abusive word lists. *Decision Support Systems*, *113*, 22-31.

[47] Le, Q., & Mikolov, T. (2014, June). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188-1196). PMLR.

[48] Zhao, R., Zhou, A., & Mao, K. (2016, January). Automatic detection of cyberbullying on social networks based on bullying features. In *Proceedings of the 17th international conference on distributed computing and networking* (pp. 1-6).

[49] Cer, D., Yang, Y., Kong, S. Y., Hua, N., Limtiaco, N., John, R. S., ... & Kurzweil, R. (2018). Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

[50] Abozinadah, E. (2017). *Detecting abusive arabic language twitter accounts using a multidimensional analysis model* (Doctoral dissertation, George Mason University).

[51] Burnap, P., & Williams, M. L. (2016). Us and them: identifying cyber hate on Twitter across multiple protected characteristics. *EPJ Data science*, *5*, 1-15.

[52] Spertus, E. (1997, July). Smokey: Automatic recognition of hostile messages. In *Aaai/iaai* (pp. 1058-1065).

[53] Razavi, A. H., Inkpen, D., Uritsky, S., & Matwin, S. (2010, May). Offensive language detection using multi-level classification. In *Canadian Conference on Artificial Intelligence* (pp. 16-27). Springer, Berlin, Heidelberg.

[54] Uysal, E., Yumusak, S., Oztoprak, K., & Dogdu, E. (2017, April). Sentiment analysis for the social media: A case study for turkish general elections. In *Proceedings of the SouthEast Conference* (pp. 215-218).

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

[56] Mazzonello, V., Gaglio, S., Augello, A., & Pilato, G. (2013, September). A study on classification methods applied to sentiment analysis. In *2013 IEEE Seventh International Conference on Semantic Computing* (pp. 426-431). IEEE.

[57] Brown, P. F., Della Pietra, V. J., Desouza, P. V., Lai, J. C., & Mercer, R. L. (1992). Class-based n-gram models of natural language. *Computational linguistics*, *18*(4), 467-480.

[58] Joachims, T. (1998, April). Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning* (pp. 137-142). Springer, Berlin, Heidelberg.

[59] Bennett, K. P., & Campbell, C. (2000). Support vector machines: hype or hallelujah?. *ACM SIGKDD explorations newsletter*, *2*(2), 1-13.

[60] Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. *arXiv preprint cs/0205070*.

[61] Imtiaz Khan, N., Mahmud, T., & Nazrul Islam, M. (2021). COVID-19 and black fungus: Analysis of the public perceptions through machine learning. *Engineering Reports*, e12475.

[62] Dreiseitl, S., & Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, 35(5-6), 352-359.

[63] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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

[65] Liu, Y., Ji, L., Huang, R., Ming, T., Gao, C., & Zhang, J. (2019). An attention-gated convolutional neural network for sentence classification. *Intelligent Data Analysis*, *23*(5), 1091-1107.

[66] Weinberger, K. Q., Blitzer, J., & Saul, L. K. (2006). Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems* (pp. 1473-1480).

[67] Zhang, Y., & Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

[68] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[69] Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123-140.

[70] Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, *63*(1), 3-42.

[71] Poli, R., Langdon, W. B., McPhee, N. F., & Koza, J. R. (2008). A Field Guide to Genetic Programming. lulu. com. *With contributions by JR Koza*.

[72] Salminen, J., Hopf, M., Chowdhury, S. A., Jung, S. G., Almerekhi, H., & Jansen, B. J. (2020). Developing an online hate classifier for multiple social media platforms. *Human-centric Computing and Information Sciences*, *10*(1), 1-34.

[73] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321-357.

[74] Ziems, C., He, B., Soni, S., & Kumar, S. (2020). Racism is a virus: Anti-asian hate and counterhate in social media during the covid-19 crisis. *arXiv preprint arXiv:2005.12423*.

[75] Malmasi, S., & Zampieri, M. (2018). Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, *30*(2), 187-202.

[76] Van Thin, D., Le, L. S., & Nguyen, N. L. T. (1991). Nlp@ uit: Exploring feature engineer and ensemble model for hate speech detection at vlsp 2019. *TRAINING*, *5*, 3-51.

[77] Verma, G., Chhaya, N., & Vinay, V. (2020). " To Target or Not to Target": Identification and Analysis of Abusive Text Using Ensemble of Classifiers. *arXiv preprint arXiv:2006.03256*.

[78] Fortuna, P., & Nunes, S. (2019, June). Stop PropagHate at SemEval-2019 Tasks 5 and 6: Are abusive language classification results reproducible?. In *Proceedings of the 13th International Workshop on Semantic Evaluation* (pp. 745-752).

[79] Lorena, A. C., Jacintho, L. F., Siqueira, M. F., De Giovanni, R., Lohmann, L. G., De Carvalho, A. C., & Yamamoto, M. (2011). Comparing machine learning classifiers in potential distribution modelling. *Expert Systems with Applications*, 38(5), 5268-5275.

[80] Basile, V., Bosco, C., Fersini, E., Debora, N., Patti, V., Pardo, F. M. R., ... & Sanguinetti, M. (2019). Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *13th International*

Workshop on Semantic Evaluation (pp. 54-63). Association for Computational Linguistics.

[81] Alshalan, R., & Al-Khalifa, H. (2020). A deep learning approach for automatic hate speech detection in the saudi twittersphere. *Applied Sciences*, 10(23), 8614.