Identification of Hate Spreaders on Social Media

Thesis submitted by

Subhalingam D
2018MT10770

under the guidance of

Prof. Niladri Chatterjee

in partial fulfilment of the requirements
for the award of the degree of

Bachelor of Technology

Department Of Mathematics
INDIAN INSTITUTE OF TECHNOLOGY DELHI

April 2022

THESIS CERTIFICATE

This is to certify that the thesis titled **Identification of Hate Spreaders on Social Media**, submitted by **Subhalingam D**, to the Indian Institute of Technology, Delhi, for the award of the degree of **Bachelor of Technology**, is a bonafide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Niladri Chatterjee
Professor
Department of Mathematics
Indian Institute of Technology, Delhi

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my thesis advisor Prof. Niladri Chatterjee (Department of Mathematics, IIT Delhi) and Raksha Agarwal (Research Scholar, Department of Mathematics, IIT Delhi) for their continuous guidance, constructive criticism and valuable suggestions throughout this project. They consistently provided me with all the required resources for my work, encouraged me to follow my ideas and steered me in the right direction whenever they thought I needed it.

I thank the IIT Delhi HPC and the Google Colaboratory facilities for providing computational resources. I am grateful to Francisco Rangel et al., Valerio Basile et al. and Zeerak Waseem et al. for granting access to the datasets.

I would also like to take this opportunity to thank all my friends and teachers, whom I have met till now, for providing me with their unfailing support and continuous encouragement throughout the years. Last but not least, I am grateful to my parents for ensuring I always had everything I needed, especially in this difficult time. They have taught me always to put my best effort in all my endeavours and have consistently set this example for me to follow.

ABSTRACT

KEYWORDS: Hate Speech Spreader; Author Profiling; Twitter

With the internet becoming more accessible to a larger set of population and the anonymity and freedom provided by social media platforms, hate speech propagation has reached new heights in recent times. With evolving scenario, manually inspecting each post is infeasible. Hence developing automatic hate speech detection systems has been a critical area of research in NLP. The present work focuses on detecting users who are keen to spread hate, as identifying spreaders is considered to be more useful than merely classifying individual tweets. To this end, we perform an exploratory data analysis on the given dataset and extract several features, such as user and stylistic features, sentiments, emoticons, stopwords and named-entity mentions, and their several statistical properties. Different baselines are then developed using several handcrafted features, TF-IDF based word representation and flagging those users who post hateful tweets beyond a certain threshold. Finally, we propose a novel model that uses pre-trained word embeddings for encoding the words and incorporates the sentiment scores as weights to mark the importance of the words. It then computes a weighted sum to get the tweet representation and aggregates these to obtain the user representation. The user representation is finally fed to an ML classifier. In particular, we used Logistic Regression, Support Vector Machine, Random Forest, eXtreme Gradient Boosting, Light Gradient Boosting Machine and Neural Network. While it is intuitive that tweets indicating hate tend to carry a high negative sentiment score, our experiments also show that the negative sentiment score is the most important feature among all other features considered. Our model achieves an accuracy of 76% on the test set and outperforms the best model in the competition. We further improve the accuracy by creating an ensemble of models. In future, we plan to investigate these models in more detail.

Contents

List of Tables

List of Figures

ABBREVIATIONS

BERT	Bidirectional Encoder Representations from Transformers	8
CNN	Convolutional Neural Network	8
GloVe	Global Vectors for Word Representation	10
LGBM	Light Gradient Boosting Machine	10
LR	Logistic Regression	8
LSTM	Long short-term memory	11
NB	Naive Bayes	10
NN	Neural Network	15
RF	Random Forest	10
RNN	Recurrent Neural Network	10
RoBERTa	Robustly Optimized BERT Pretraining Approach	13
SVM	Support Vector Machine	10
TF-IDF	Term Frequency–Inverse Document Frequency	9
XGB	eXtreme Gradient Boosting	10

Chapter 1

INTRODUCTION

There has been a massive advancement in information technology in recent times. The number of people using the internet has also increased in the past decade. It is estimated that about 4.95 billion (62.5%) people have used the internet worldwide in January 2022, increasing by 192 million in the past 12 months (equivalent to more than 500,000 new users each day) [1]. The amount of data generated through the internet has also grown exponentially. It is estimated that, on average, about 2.5 quintillion bytes of data were created daily in 2020 and that 463 exabytes of data will be generated on average each day as of 2025 [2].

The growth of the internet has given birth to social media platforms like Twitter, Reddit and Instagram, which allows people worldwide to share their thoughts extensively and have transformed the way information gets propagated today. It is estimated that the number of global users in social media was around 4.62 billion (58.4%) in January 2022, increasing by 10.1% over the past 12 months (equivalent to a rate of over 13 new users per second) [1]. It has been reported that the average global user spends 2 hours 27 minutes on social media each day [1]. In 2020, about half a million new tweets were posted every day, and about four petabytes of data were generated on Facebook every day [2].

Undoubtedly, these platforms have facilitated a convenient exchange of dialogues and information outreach. On the positive side, these platforms have paved the way for some positive societal changes through online social revolutions. On the other hand, the freedom, easy accessibility and anonymity that these platforms offer have put a big question mark on the integrity of different kinds of posts that appear. Some individuals or groups misuse the platform. Hence, identifying and stopping the dissemination of aggressive and harmful content is vital to safeguard the interests of different groups around the world.

One such significant issue to tackle is the spread of Hate Speech. According to Cambridge Dictionary[1], "*Hate Speech*" is defined as "public speech that expresses hate or encourages violence toward a person or group based on something such as race, religion, sex, or sexual orientation". Such messages usually target a particular social group basis religion, caste, colour, race, national origin, sex, sexual orientation, etc., such as women, Muslims, Jews, African-Americans and members of the lesbian, gay, bisexual and transgender (LGBT) community. A study reported that Online Hate Speech has risen by 20% in the UK and US since the start of the pandemic [3].

[1]`https://dictionary.cambridge.org/us/dictionary/english/hate-speech`

Due to the constantly evolving cyberspace and infeasibility to manually verify every content uploaded, Hate Speech propagation has gained new momentum in the recent times, challenging both the policymakers and the research community. Despite several developments and measures to curb the spread of Hate Speech, the problem is far from being solved. Hence, Automatic Hate Speech Detection remains an active and critical area of research in NLP.

In this work, we focus on identifying those user accounts on Twitter that are keen to spread Hate Speech. This thesis is organized as follows. We discuss the dataset we use in Chapter 2 and some previous developments in Hate Speech detection in Chapter 3. Then, we build some baselines and present our proposed model in Chapter 4. Later, we analyze their performance and perform an error analysis in Chapter 5. Lastly, we conclude and discuss some possible future works in Chapter 6.

Chapter 2

DATASET

2.1 Overview

The present work has used the dataset from PAN @ CLEF 2021 Profiling Hate Speech Spreaders on Twitter [4] competition. This corpus has been built by first looking for the users considered potential haters based on two approaches– (i) *keyword-based* (e.g., searching for hateful words mainly towards women or immigrants); (ii) *user-based*, by inspecting users known as haters (e.g., users appearing in reports and/or press) and following their networks. Then, the timelines were collected for these identified users, and the tweets conveying hate were manually annotated. Lastly, those users with more than ten hateful tweets have been labelled as keen to spread hate speech.

The corpus consists of tweets from 300 Twitter users for each of the two languages, English and Spanish. Corresponding to each account, there are 200 tweets, collected and collated together in an XML file. The binary class 0 represents those users who are not keen to spread hate, while the binary class 1 represents those users who are keen to spread hate. The corpus is also completely balanced with respect to class instances with each of the labels having exactly 150 Twitter user accounts, and has already been split into training and test sets in the ratio 2:1. These statistics are tabulated in Table 2.1

	Class 0	**Class 1**
Training set	100	100
Test set	50	50

Table 2.1: Distribution of number of user accounts in the corpus

Each user account is mapped to a unique alphanumeric author ID to ensure anonymity. Further, potential information containing keywords have been masked with special tokens. Particularly, hashtags, user mentions, retweets and URL links have been replaced with `#HASHTAG#`, `#USER#`, `#RT#` and `#URL#` respectively. The format of the data in the dataset is discussed in Section 2.1.1 and some examples from the dataset are listed in Section 2.1.2.

In this work, the focus is on the tweets from the English language only. In all our experiments, we do not use any part of the test set during the entire course of training, including for hyper-parameter fine-tuning. We evaluate the performance of our models with this test set (Chapter 5).

2.1.1 Format

The uncompressed dataset consists in a folder per language. Each folder contains: (i) a XML file per Twitter user account (the name of the XML file correspond to the unique masked author ID); (ii) a `truth.txt` file with the list of authors and the ground truth.

The format of the XML files is as follows:

```
<author lang="en">
    <documents>
        <document>Tweet 1 textual contents</document>
        <document>Tweet 2 textual contents</document>
        ...
    </documents>
</author>
```

In `truth.txt`, the first column corresponds to the author ID and the second column contains the truth label. The format of file is as follows:

```
b2d5748083d6fdffec6c2d68d4d4442d:::0
2bed15d46872169dc7deaf8d2b43a56:::0
8234ac5cca1aed3f9029277b2cb851b:::1
5ccd228e21485568016b4ee82deb0d28:::0
60d068f9cafb656431e62a6542de2dc0:::1
```

2.1.2 Examples

For illustration, some tweets from users of each class are shown in Table 2.2.

2.2 Exploratory Data Analysis

2.2.1 Preliminary pre-processing

To get started, we have to parse the dataset. Since, every retweet was followed by a user mention, possibly denoting the original author for the tweet, we replace every occurrences of `#RT# #USER#:` at the beginning of every tweet with `##RT##:` [1] to distinguish between a retweet and an actual user mention.

[1]To distinguish between the masked token placeholders we add and those already present, we enclose them between two hashes (i.e., `##...##`) instead of just one (i.e., `#...#`).

Class 0	**Class 1**
RT #USER#: When i go on trips i need airport outfits, day outfits, night outfits, sleeping outfits. Idk why im like this	RT #USER#: People don't believe in Big Foot, but they believe Biden got 80 million votes #URL#
RT #USER#: I pray I dont mishandle what I prayed for...	RT #USER#: Remember that these two liars SPIED ON PRESIDENT TRUMP. They got CAUGHT. And they still lost. #URL#
RT #USER#: I be on websites accepting cookies not even knowing what that is	RT #USER#: HAPPY HALLOWEEN **Dont get tricked, know who youre voting for...** #HASHTAG# #URL#

Table 2.2: Sample tweets by users from each class

2.2.2 Feature Extraction

We first list down different features we extracted for each user.

User stylistic features: These included hashtags, user mentions, URLs and retweets. The counts for these can be obtained by simply counting the occurrences of `#HASHTAG#`, `#USER#`, `#URL#` and `##RT##`.

Statistical stylistic features: For each user, we calculated the number of uppercase, minimum, maximum and total number of characters/words in his/her tweets. Since we do not have the exact hashtags, user mentions, URLs and retweets information, the respective placeholders were removed while computing the character-level statistics. In case of computing word-level statistics, we only removed the retweet placeholder (as they are not actually part of the tweet) and retained the placeholders for hashtags, user mentions and URLs (as each of them would only be a single word if left unmasked). Moreover, we only treat white-spaces as word separators without adhering to any other heuristics (as we expect a space after every punctuation but not before it).

Emoticons: We computed the number of emoticons used by each user using the demoji[2] library of Python.

[2] `https://pypi.org/project/demoji/`

Stopwords: We observed the number of stopwords used by each user using the list of English stopwords from the SpaCy library [5] in Python.

Named-Entity Mentions: This involved identifying different named entities present in the tweets of different users and locating them into four major categories– Persons (PER), Organizations (ORG), Locations (LOC) and Miscellaneous/None of the above (MISC), This was done using the NER model trained on `xx_ent_wiki_sm` corpus from the SpaCy library in Python.

Sentiments: This involved identifying the sentiment– positive, negative or neutral, implied in each tweet of the tweets. We used the TextBlob library [6] in Python for obtaining the sentiments.

We sum the feature values of all users for each class and present an overview[3] of the statistics for some of the features in Table 2.3.

Features	**Class 0**	**Class 1**
`#HASHTAG#`	3757	3392
`#URL#`	8571	6768
`#USER#`	9723	11571
`##RT##`	7633	6090
Upper case characters	71025	75867
Number of characters	1109779	1134313
Upper case words	43584	44012
Number of words	228644	234867
Emoticons	8792	7446
Stop-words	95998	101886
PER	4672	4743
ORG	2611	2387
LOC	2437	2132
MISC	6458	6534
Positive sentiment	6385	6163
Negative sentiment	3679	4482
Neutral sentiment	9936	9355

Table 2.3: Extracted features values overview

2.2.3 Observations

Distinction between the classes based on extracted features: It can be observed that Hate Speech spreaders tend to use a remarkably fewer number of URL links (`#URL#`)

[3]Feature values for each user can be downloaded from `https://subhalingamd.me/btp-hate-speech-profiling/resources/data_pan21_train_en_features.tsv`

and more number of user mentions (`#USER#`). They also retweet (`##RT##`) fewer times and have more tweets that carry negative sentiments. Some notable differences in the number of emoticons and stopwords used can also be observed.

Ratio of distinct tweets: We obtained the number of distinct tweets among the 200 of them for each user and observed that a user who is not keen to spread hate speech had as low as 31 (15.5%) distinct tweets. While looking at the tweets, we observed that most of them were updates about new followers (e.g., "2 people followed me"). Only 64 users from class 0 and 69 users from class 1 have all the 200 tweets different. The frequency distribution of the ratio of distinct tweets to total number of tweets is documented in Appendix A.

Multilingual and code-mixed tweets: While skimming the tweets in the dataset, we observed some tweets containing characters from non-English languages (e.g., Kannada). Hence, we expect models trained on a multi-lingual corpus to perform better than those trained only on an English corpus. This is also the reason behind choosing the `xx_ent_wiki_sm` corpus instead of an English-only corpus like `en_core_web_sm` for finding named-entity mentions.

Chapter 3

RELATED WORKS

There have been several works on Hate Speech detection in recent times. One of the most popular dataset is the HatEval dataset [7], used in SemEval 2019[1] competition. We describe this dataset later in Section 4.2.4. [8] surveys different approaches in the literature and analyzes them. The proposed models range from using hand-crafted features based on linguistics, user stylistics, distributional semantics, etc., to leveraging data-driven approaches using traditional machine learning-based classifiers and deep learning-based architectures.

While several works concentrate on classifying whether a given text contains hate or not, a few of them, apart from submissions in PAN @ CLEF 2021 competition, also study user-activity patterns to flag user accounts involved in spreading hate. ElSherief et al. [9] analyzed the distinctive characteristics of hate instigators and targets based on their profiles, activities and online visibility. They observed that the targets are usually from high-profile backgrounds and that online visibility increases by participating in hate-spreading activities. Chaudhry and Lease [10] studied the use of past utterances as informative prior to improving prediction on new utterances and observed promising results.

In the competition, the overall best performing model [11] leveraged a 100-dimension word embedding representation to feed a Convolutional Neural Network (CNN). This system achieved an accuracy of 73% on the English language test set. However, the best accuracy on the English language test set was 75%, achieved using Bidirectional Encoder Representations from Transformers (BERT) and Logistic Regression (LR). Most of the top-performing models have leveraged a deep transformer architecture, like BERT, for encoding the tweets.

[1]https://alt.qcri.org/semeval2019/

Chapter 4

MODELS AND EXPERIMENTS

4.1 Pre-processing

The data pre-processing pipeline consists of the following steps:

- Replace every occurrences of `#RT# #USER#:` at the beginning of every tweet (if it exists) with `##RT##:` to distinguish between a retweet and an actual user mention (as discussed in Section 2.2.1).
- Replace emoticons with `##EMOJI##` placeholder. The Python library demoji is used to find different emoticons.
- Convert texts in tweets to lower case, except for the placeholders (i.e., `#HASHTAG#`, `#USER#`, `#URL#`, `##RT##`and `##EMOJI##` are left intact in upper case).
- Remove majority of the non-alphanumeric symbols (except '#').
- Normalize line-breaks and multiple white-spaces to single white-space.

We generate a TSV file with the pre-processed tweets and ground truth labels. We directly use the data from this TSV file for all our experiments, possibly with additional pre-processing steps as specified in the respective setups. We use the Pandas [12] library for processing the data.

4.2 Baselines

4.2.1 Term-frequency based representation

In this setup, we merge all the tweets of each user into a single paragraph. To summarize, we now have a paragraph (of tweets) and a ground truth label for each user.

We obtain a sparse vector representation using the paragraph of tweets for each user based on two approaches– (i) Count Vectorizer and (ii) Term Frequency–Inverse Document Frequency (TF-IDF) Vectorizer. We build a vocabulary based on the tokens that appear in all the texts in both these approaches. The former only considers the number of times a token appears in the text, resulting in a bias towards frequently occurring tokens. The latter overcomes this by also considering the number of texts containing the token and penalizing the frequently occurring tokens. The minimum and maximum number of times a token must appear in the collection of texts to be included in the vocabulary is a hyper-parameter that we fine-tune (hence, no stopword removal was performed explicitly). We also fine-tune the best n-gram range for building the vocabulary. The final representation does not capture the ordering between the tweets and between the tokens in each tweet.

We train different statistical machine learning models to classify the given user as "keen to spread Hate Speech" or not based on the extracted representation. Specifically, we use LR, Support Vector Machine (SVM) [13], Naive Bayes (NB), Random Forest (RF) [14], eXtreme Gradient Boosting (XGB) [15] and Light Gradient Boosting Machine (LGBM) [16]. We do not report the performance for XGB and LGBM with Count Vectorizer because of the very long time taken for hyper-parameter fine-tuning.

Setup: We build the entire model pipeline in Python using the scikit-learn [17] library and perform a grid search to fine-tune the hyper-parameters with repeated 10-fold cross-validation. The performance of these models is discussed in Section 5.2.1.

4.2.2 Features based representation

In this setup, we use the list of features mentioned in Section 2.2.2 to construct the vector representation for each user. In particular, the entries in the vector will be the value of the features computed by considering all the user tweets. We feed this vector into a machine learning classifier. We experiment with different models, namely LR, SVM, NB, RF, XGB and LGBM.

Setup: We build the entire model pipeline in Python using the scikit-learn library and perform a grid search to fine-tune the hyper-parameters with repeated 10-fold cross-validation. The performance of these models is discussed in Section 5.2.2.

4.2.3 Word-embedding based representation

We do not merge the tweets in this setup, unlike in Section 4.2.1. Instead, we use pre-trained word embeddings and obtain a vector representation for each tweet. We treat all the tweets by the user as a sequence and use Recurrent Neural Network (RNN)-based architecture to classify each user finally. It is analogous to considering each user as a "sentence" and each tweet as a "token" in a traditional setting. We explain the procedure we use for getting the representations.

We use the Global Vectors for Word Representation (GloVe) [18] word embeddings pre-trained on 2B tweets. These embeddings are dense representations in contrast to vectorization methods used in Section 4.2.1. GloVe can capture words that appear in similar contexts as well as words that appear in the context of each other. We replace all the `##EMOJI##` placeholders and remaining punctuations with white spaces and normalize the white spaces again for each tweet. This includes the '`#`' symbols that are present in the placeholders. We split the words at white spaces and get the corresponding GloVe word embedding. To retain the same dimension size for all the tweet representations, we aggregate (or pool) all the vectors we obtain for each word in a particular tweet to obtain a single vector of the same dimension. We use one of the following four techniques to build up this vector dimension-wise– (i) mean-pooling (take the mean of values in the dimension); (ii) max-pooling (take the maximum value in the dimension); (iii) min-pooling (take the minimum value in the dimension); and (iv) abs-max-pooling (take the maximum in terms of the absolute value in the dimension). Upon pooling, we have a sequence of tweet representations for each user. We finally train a Long short-term memory (LSTM) model [19] for classification.

Setup: We build the entire model pipeline in Python using the PyTorch [20] library with PyTorch Lightning [21] wrapper and use the Gensim [22] library for obtaining the GloVe word embeddings. We particularly experiment with 25-dimensional and 200-dimensional GloVe embeddings pre-trained on tweets. The given training set was split in the ratio of 9:1 and used the smaller portion as the validation set. The embedding layer was frozen, and we chose the values for hyper-parameters rather than fine-tuning. The values of hyper-parameters used are tabulated in Table 4.1. AdamW [23] was used as the optimizer with a learning rate of 0.001. The validation set was used for early stopping (with a patience of 3). The random states were seeded with a value of 42 to ensure reproducibility. The performance of these models is discussed in Section 5.2.3.

Entity	**Hyper-parameter**	**Values**
Embedding	Dimensions	$\{25, 200\}$
LSTM	Hidden states	8
	Bidirectional	True
MLP	Dimensions	8×1
AdamW	LR	0.001

Table 4.1: Hyper-parameters values used in Word-embedding based representation models in Section 4.2.3

4.2.4 Counting number of hate tweets

In this setup, we use a two-stage system to obtain the hate score for each tweet made by a user first, then aggregate these scores and use a threshold to flag if the user is a Hate Speech spreader. This is different from previous setups where we directly obtain whether a user is keen to spread Hate Speech or not.

Training a model to classify whether a tweet contains Hate Speech or not requires data with annotations for each tweet. The current dataset (PAN @ CLEF 2021) has annotations only for each user and not for the individual tweets of a user. To deal with this, we make use of some other existing datasets. Particularly, we use the following two datasets:

1. Basile et al. released the HatEval dataset [7] used in SemEval 2019[1] competition. It consists in detecting hateful content in tweets against two targets: *immigrants* and *women*. The label takes a binary value indicating if Hate Speech is occurring against one of the given targets: **1** if occurs, **0** if not. The dataset consists of about 13,000 tweets, and about 42% of them belong to class 1 (i.e., contain Hate Speech). We use the terminology "HatEval" to refer to this dataset in our work.

2. Waseem and Hovy created a dataset [24] with 16,914 tweets. They focused on identifying those tweets with sexist and racist content. Upon annotating, 3,383 were sexist content, 1,972 were racist content, and the remaining 11,559 were neither sexist nor were racist content. Since we are interested in binary labels, denoting whether a tweet contains Hate Speech or not, we grouped those tweets containing sexist content or racist content into one class (class 1, in or case) and assigned class 0 to those that contain neither of them. Hence, only about 32% of the tweets contain Hate Speech. We use the terminology "Zeerak's" (first author's first name) to refer to this dataset in our work.

HatEval dataset already comes with a training, validation and test split, but no such splits are present for the Zeerak's dataset. Hence, we do the split by randomly sampling 25% of the data to be the test set in such a way that the proportions of class labels in each of the splits are the same as the input dataset (stratified split). Moreover, the dataset has a class imbalance (i.e., the number of data points in each class is not the same). We randomly undersample the majority class in the training set to tackle this.

Upon preparing the dataset, we have to obtain a vector representation for each tweet for the first stage of the system. Since the tweets (in the PAN @ CLEF 2021 dataset and the datasets mentioned above) come from different distributions (e.g., different time frames, targets, etc.), To get rid of problems that might occur due to out of vocabulary (OOV) tokens, we restrain ourselves from using Count/TF-IDF Vectorizer-based representation (like in Section 4.2.1). Instead, we stick to using the feature vectors (like in Section 4.2.2). We use the same set of features from Section 2.2.2, but in this case, we get the feature values for each tweet. We feed this vector into a machine learning classifier to obtain the tweet-level labels. We experiment with different models, namely LR, SVM, NB, RF, XGB and LGBM.

Hate scores for each tweet made by a user in the PAN @ CLEF 2021 dataset can be obtained with the above model. Now the idea is to "just count how many tweets made by

[1]https://alt.qcri.org/semeval2019/

a user contain Hate Speech". If this count is more than a particular *threshold*, predict the user as a Hate Speech Spreader. The threshold is a hyper-parameter to be figured out using the training set by experimenting with different values.

To make a fair comparison of this methodology with other baselines, we use a pre-trained language model to get the hate scores rather than training a model from scratch (in the first stage). In our case, we use the base Robustly Optimized BERT Pretraining Approach (RoBERTa) [25] model, fine-tuned on tweet data and further fine-tuned for the Hate Speech prediction task [26]. Note that since this model was trained on a vast corpus, it has an extensive set of vocabulary, in contrast to Count/TF-IDF Vectorizer-based models that have a limited vocabulary based on the words it sees in the current training data. Hence, problems due to the presence of OOV words mentioned earlier are not relevant in this case. Pre-trained models are also known to capture the intricacies of Natural Languages like word synonyms, context-based word disambiguation, etc., in contrast to Count/TF-IDF Vectorizer-based models, due to the difference in the methods used while training. As done earlier, we then count the number of hate-containing tweets to determine if the user is a Hate Speech Spreader.

Setup: We build the entire model pipeline in Python. For undersampling the training data, we use the imbalanced-learn [27] library. For training the ML classifiers, we use the scikit-learn library and perform a grid search to fine-tune the hyper-parameters with repeated 10-fold cross-validation. We directly use the `cardiffnlp/twitter-roberta-base-hate`[2] model checkpoint from the Hugging Face Transformers library [28] and do not perform further fine-tuning. To get the best threshold value, we iterate over different values and choose the one that gives the best accuracy on the training set. The performance of these systems is discussed in Section 5.2.4.

4.3 Our Model

4.3.1 Motivation

It is intuitive that Hate Speech either consists of abusive or harmful words or indirectly implies them, thus carrying a high negative sentiment score. If a model can give more importance to these particular words than the rest, it might help the model better capture the presence of Hate Speech in some text, especially if it is wordy. This can be conceptualized as the "*attention*" weights used in the transformer-based architectures [29]. The word-embedding of each word is multiplied by a learnt scalar weight, denoting the importance of that word in the context. However, unlike these, we do not make our model learn the weights

[2]https://huggingface.co/cardiffnlp/twitter-roberta-base-hate

or the word embeddings as we have a small dataset. Rather, we use the "sentiment scores" from the SentiWordNet lexical resource [30] as the weights and the GloVe embeddings pre-trained on Twitter for representing each word. We then take a weighted sum to get the tweet representation, aggregate these tweet representations and use an ML classifier to predict whether the user is a Hate Speech Spreader or not. Our model is described in detail in the following section.

4.3.2 Methodology

1. **Further pre-processing:** We remove `##RT##`, `##EMOJI##`, `#URL#` and `#USER#`placeholders and replace the `#HASHTAG#`placeholder with the word "hashtag" from the data. Then, any remaining punctuations are replaced with white spaces and normalize the white spaces again. We use this "cleaned" data in the following steps.

2. **Stopwords removal:** Stop words are those words that frequently occur in a corpus, do not add much meaning and can be safely ignored without sacrificing the meaning of the sentence. By removing stop words, we remove the low-level information from our text to give more focus to the important information. We use the list of "english" stopwords from the NLTK library and remove all those words in the data that are part of this list.

3. **Getting POS tags:** SentiWordNet requires the Part of Speech (POS) tag along with the word to obtain the correct set of sentiment scores. We focus only on nouns, verbs, adjectives and adverbs as these are usually the sentiment carrying words. We obtain the POS tags from the NLTK library.

4. **Getting sentiment scores and word embeddings:** SentiWordNet also has a different set of sentiment scores depending on the sense in which the word is used (denoted by "*sense numbers*"). For example, "*field*" might mean "knowledge domain" or "piece of land". Since we do not have a method to obtain the word sense, we use the most commonly used sense. The lexicon gives each word a positive, neutral and negative sentiment score. We get the word embeddings from GloVe pre-trained on 2B tweets. If the word occurring in a tweet is not present in at least one of SentiWordNet or GloVe, we simply ignore the word. We obtain the sentiment scores from SentiWordNet using the NLTK library and the GloVe word embeddings from the Gensim library.

5. **Constructing the feature space:** We have the sentiment scores (positive, neutral and negative) and word embeddings (say, d-dimensional) for each word in a tweet. We first compute the weighted sum of all the embeddings, the weights being the positive scores. Similarly, we take the weighted sum using the neutral scores as weights, and then with the negative scores as weights. We concatenate these three d-dimensional vectors obtained by summing to obtain the tweet representation. Hence, for a d-dimensional word embedding, we construct a $3d$-dimensional feature space. More formally, if we represent a tweet as $< w_1, w_2, \ldots, w_n >$, where w_i is the i^{th} word in the tweet, and if each w_i has positive, neutral and negative sentiment scores s_i^{pos}, s_i^{neut} and s_i^{neg} and embedding e_i, then we obtain the tweet-level representation t as:

$$t^{pos} = s_1^{pos} e_1 + s_2^{pos} e_2 + \cdots + s_n^{pos} e_n$$

$$t^{neut} = s_1^{neut}e_1 + s_2^{neut}e_2 + \cdots + s_n^{neut}e_n$$
$$t^{neg} = s_1^{neg}e_1 + s_2^{neg}e_2 + \cdots + s_n^{neg}e_n$$
$$t = concat(t^{pos},\ t^{neut},\ t^{neg})$$

Now that we have the representation of tweets made by a user, we have to aggregate them to obtain the user representation. In our case, we use mean pooling, i.e., we take the mean of embedding values of all the tweets of a user along each dimension. Hence, if we have tweets $t_1, t_2, \ldots, t_n$ for a user, we obtain the user representation as:

$$u = \frac{1}{n}\sum_{i=1}^{n} t_i$$

6. **Classifier:** Finally, we obtained the input for the classifier model, i.e., the $3d$-dimensional user representation. We experiment with different ML classifiers, namely LR, SVM, RF, XGB, LGBM and Neural Network (NN). The output will be a binary label– 1 if the user is a Hate Speech Spreader and 0 if not. Like in previous experiments, we build the entire model pipeline in Python using the scikit-learn library and perform a grid search to fine-tune the hyper-parameters with repeated 10-fold cross-validation. The performance of these models is discussed in Section 5.3.

4.4 Ensemble

We pick the best classifiers in terms of accuracy from term-frequency based representation baselines (Sections 4.2.1), counting the number of hate tweets baselines (Section 4.2.4) and our models (Section 4.3), and create a max-voting classifier to combine their predictions. Each participating model independently predicts which class a given input belongs to (“*vote*”), and the class that receives the maximum *vote* is regarded as the final output. In our case, we have only binary labels and three participating models. Hence, the class that receives at least two *votes* will be the final output class. Note that no additional hyperparameters are introduced in this step.

Chapter 5

RESULTS

5.1 Evaluation Scheme

The metric used in the competition for evaluating the models is accuracy. Additionally, we assess the model using precision, recall and F1-score in our work. The performance results presented in this chapter, along with some additional information like best hyper-parameter values obtained upon fine-tuning, can also be found at `https://subhalingamd.me/btp-hate-speech-profiling/results.html`. We now define these metrics formally in the following subsection.

5.1.1 Definitions

We use the following notations in the definition of the metrics.

- **TP** (True Positive): Model predicts hate spreader and is actually hate spreader.
- **FP** (False Positive): Model predicts hate spreader but is actually not hate spreader.
- **FN** (False Negative): Model predicts not hate spreader but is actually hate spreader.
- **TN** (True Negative): Model predicts not hate spreader and is actually not hate spreader.

The metrics mentioned earlier can be defined as follows:

$$\begin{aligned}
\text{Accuracy} &= \tfrac{TP+TN}{TP+FP+FN+TN} \\
\text{Precision} &= \tfrac{TP}{TP+FP} \\
\text{Recall} &= \tfrac{TP}{TP+FN} \\
\text{F1-score} &= \tfrac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}
\end{aligned}$$

5.2 Baselines

5.2.1 Term-frequency based representation

From Table 5.1, it can be observed that both TF-IDF+LR and TF-IDF+SVM have the best performance in terms of accuracy (69%). However, TF-IDF+NB has the highest F1-score (73.43%) among all the models. It also has a very high recall score of 94%, i.e., it produces the least low false negatives. TF-IDF+SVM also has a competitive F1-score score of 71.02%. Moreover, TF-IDF vectorization performs better than simple Count vectorization.

Representation	**Classifier**	**Accuracy**	**Precision**	**Recall**	**F1-score**
Count	LR	0.6500	0.6531	0.6400	0.6465
	SVM	0.6600	0.6600	0.6600	0.6600
	NB	0.6200	0.6111	0.6600	0.6346
	RF	0.6500	0.6596	0.6200	0.6392
TF-IDF	LR	**0.6900**	0.6792	0.7200	0.6990
	SVM	**0.6900**	0.6667	0.7600	0.7103
	NB	0.6600	0.6026	0.9400	0.7344
	RF	0.6100	0.6000	0.6600	0.6286
	XGB	0.6000	0.6136	0.5400	0.5745
	LGBM	0.6200	0.6034	0.7000	0.6481

Table 5.1: Performance of models trained with Term-frequency based representation (Section 4.2.1)

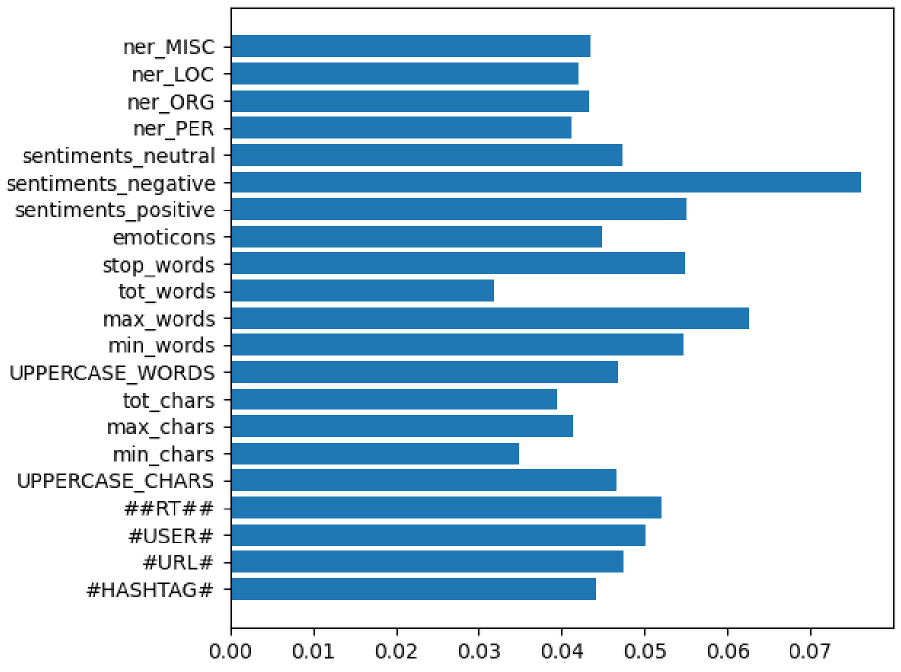
5.2.2 Features based representation

From Table 5.2, we can observe a significant performance gap with this approach compared to the models from Section 5.2.1. XGB performs the best among all the models in this setup.

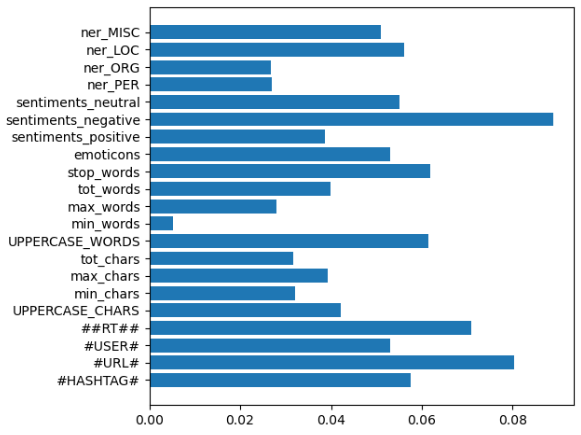
Classifier	**Accuracy**	**Precision**	**Recall**	**F1-score**
LR	0.5800	0.5870	0.5400	0.5625
SVM	0.5200	0.5278	0.3800	0.4419
NB	0.5100	0.5116	0.4400	0.4731
RF	0.5800	0.5741	0.6200	0.5962
XGB	**0.6200**	0.6154	0.6400	0.6275
LGBM	0.5500	0.5455	0.6000	0.5714

Table 5.2: Performance of models trained with Features based representation (Section 4.2.2)

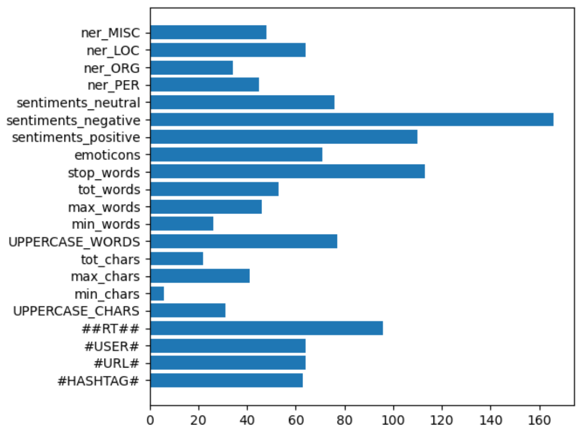
We plot the feature importance from the trained XGB, RF and LGBM models in Figure 5.1. The “negative sentiment” feature has the highest importance in all of these. This strong observation further strengthened our claim in Section 4.3 and motivated us to work towards our proposed model. We do not perform further (recursive) hyper-parameter tuning for the models in this setup as we had a solid plan to move forward with.



(a) XGB



(b) RF



(c) LGBM

Figure 5.1: Feature importance from trained XGB, RF and LGBM models in Section 4.2.2

5.2.3 Word-embedding based representation

From Table 5.3, we can observe that the models' performances are worse than the previous two setups. The fact that the training set is small (only 200 examples in training set) could be one of the reasons for not being able to train robust deep learning models from scratch.

Embedding dimension	Pooling	Accuracy	Precision	Recall	F1-score
25	Mean	0.5200	0.5161	0.6400	0.5714
	Max.	0.5200	0.5179	0.5800	0.5472
	Min.	0.5500	0.5455	0.6000	0.5714
	Abs. Max.	0.5500	0.5556	0.5000	0.5263
200	Mean	0.4900	0.4923	0.6400	0.5565
	Max.	0.4500	0.4444	0.4000	0.4211
	Min.	0.5000	0.5000	0.5200	0.5098
	Abs. Max.	0.5100	0.5091	0.5600	0.5333

Table 5.3: Performance of models trained with Word-embedding based representation (Section 4.2.3)

5.2.4 Counting number of hate tweets

As the first step, we train the ML models using the training set of HatEval and Zeerak's datasets and test the performance on their corresponding test sets. Then, we combine the training set of Hatval and Zeerak's datasets, train the ML models on the combined training set, and test the performance of each test set separately. We tabulate the results in Table 5.4.

The models trained on the Zeerak's dataset seem to have significantly better performance than those trained on the HatEval dataset. For instance, the best performing model for the Zeerak's dataset, RF, got an accuracy of 70.14% on the test set. XGB is also close with an accuracy of 70.12%. On the other hand, the accuracy of the models trained on HatEval is less than 50%. We investigate the poor performance of the models on the HatEval dataset. We evaluate them on the validation set of the HatEval dataset. Note that we performed the 10-fold cross-validation on the training set only and that the validation set was not used at any point. We tabulate the results in Table 5.5.

The accuracy looks far better than that obtained with the test set, with XGB having an accuracy of up to 67% when trained only with the HatEval dataset and 65% when trained with the combined dataset. Additionally, we also observed a similar performance gap between the validation set and the test set in the paper submitted by the FERMI team [31], the top-performing team in the competition. Their best performing model had an accuracy of 65% on the test set but only 57.13% on the validation set, while the best accuracy achieved on the validation set was 71.18%. Such a difference is most probably due to the difference between the distributions in the validation and the test sets.

Another interesting observation from Tables 5.4 and 5.5 is that the performance has also dropped when more data is added. For instance, the performance of the models trained only on the Zeerak's dataset is higher than those trained on the combined dataset. A similar trend can be seen in the case of the performance of models on the validation set of the

Training data	**Test data**	**Model**	**Accuracy**	**Precision**	**Recall**	**F1-score**
HatEval	HatEval	LR	0.4507	0.4280	0.9151	0.5832
		SVM	0.4470	0.4259	0.9103	0.5803
		NB	**0.4763**	0.4044	0.5222	0.4558
		RF	0.4640	0.4343	0.9135	0.5887
		XGB	0.4740	0.4388	0.9040	0.5908
		LGBM	0.4637	0.4337	0.9056	0.5865
Zeerak's	Zeerak's	LR	0.6586	0.4744	0.6243	0.5391
		SVM	0.6889	0.5110	0.6383	0.5676
		NB	0.5630	0.3865	0.6235	0.4772
		RF	**0.7014**	0.5237	0.7345	0.6115
		XGB	0.7012	0.5239	0.7204	0.6067
		LGBM	0.6993	0.5213	0.7322	0.6090
HatEval + Zeerak's	HatEval	LR	0.4813	0.4296	0.7167	0.5372
		SVM	0.4820	0.4323	0.7444	0.5469
		NB	0.4740	0.4058	0.5437	0.4647
		RF	0.4857	0.4332	0.7286	0.5434
		XGB	**0.5007**	0.4430	0.7341	0.5526
		LGBM	0.4850	0.4315	0.7119	0.5373
HatEval + Zeerak's	Zeerak's	LR	0.5940	0.4170	0.6768	0.5161
		SVM	0.5848	0.4165	0.7433	0.5339
		NB	0.5571	0.3647	0.5185	0.4282
		RF	**0.6749**	0.4946	0.7448	0.5945
		XGB	0.6712	0.4910	0.7700	0.5997
		LGBM	0.6695	0.4893	0.7604	0.5954

Table 5.4: Performance of models trained for classifying the tweets in Section 4.2.4

Training data	**Test data**	**Model**	**Accuracy**	**Precision**	**Recall**	**F1-score**
HatEval	Validation set of HatEval	LR	0.6640	0.5697	0.8712	0.6889
		SVM	0.6660	0.5784	0.8033	0.6725
		NB	0.5820	0.5128	0.4215	0.4627
		RF	0.6550	0.5693	0.7892	0.6614
		XGB	0.6700	0.5826	0.8009	0.6746
		LGBM	0.6540	0.5667	0.8056	0.6654
HatEval + Zeerak's	Validation set of HatEval	LR	0.6270	0.5474	0.7307	0.6259
		SVM	0.6220	0.5467	0.6721	0.6029
		NB	0.6170	0.5570	0.5035	0.5289
		RF	0.6220	0.5476	0.6604	0.5987
		XGB	0.6500	0.5750	0.6909	0.6277
		LGBM	0.6350	0.5620	0.6581	0.6063

Table 5.5: Performance of models on the validation set of the HatEval dataset, trained for classifying the tweets in Section 4.2.4

HatEval dataset, i.e., the performance of the models trained only on the HatEval dataset is higher than those trained on the combined dataset. Hence, more data does not always mean better performance– the quality of the data is also important. In this case, this probably occurs because the data in both these datasets come from different distributions.

We now move to the next stage of getting the prediction for a user based on the number of tweets containing Hate Speech. We choose one model from each of the three training data setups in Table 5.4. In particular, we choose RF for the Zeerak's dataset case (the best performing model) and XGB for the HatEval case (the best model after NB) and the combined dataset case (the best model on the HatEval's test set and second, yet close to the first, on the Zeerak's test set). We also include the fine-tuned transformer-based RoBERTa model described in Section 4.2.4.

We move back to the PAN @ CLEF 2021 dataset and get the hate scores for each tweet. We now need to determine the "threshold", which will determine the minimum number of Hate Speech containing tweets to be present to flag the user as a Hate Speech Spreader. This value is determined by iterating over different values (ideally from 1 to 200, in our case) and choosing the one that gives the best results on the training set. We perform this experiment with the models mentioned above and report the results, along with the best threshold value determined, in Table 5.6.

Training data	**Model**	**Accuracy**	**Precision**	**Recall**	**F1-score**	Threshold
HatEval	XGB	0.5400	0.5357	0.6000	0.5660	80
Zeerak's	RF	0.4800	0.4808	0.5000	0.4902	60
HatEval + Zeerak's	XGB	0.5800	0.5714	0.6400	0.6038	80
—	`cardiffnlp/twitter-roberta-base-hate`	**0.6900**	0.8065	0.5000	0.6173	10

Table 5.6: Performance obtained by counting the number of Hate Speech containing tweets (Section 4.2.4)

It can be observed that the system using the RoBERTa model for obtaining the tweet scores has the best accuracy of 69% on the test set. We also obtained the same accuracy with Term-frequency based representation baselines in Section 5.2.1. It flags any user as Hate Speech Spreader if they have posted at least ten tweets containing Hate Speech. It is also interesting to note that this number is similar to the threshold used for annotating the users during dataset preparation (Section 2.1).

5.3 Our Model

GloVe Twitter consists of word embeddings in four different dimensions– 25d, 50d, 100d and 200d. Usually, the smaller dimensional ones are useful for syntactic tasks, and the larger ones are useful for semantic tasks. However, which one is most suitable can be determined only through experiments and hence, we present the performance of our proposed models for all the four dimensions in Table 5.7 and plot their accuracies in Figure 5.2.

Embedding dimension (d)	**Model**	**Accuracy**	**Precision**	**Recall**	**F1-score**
25	LR	**0.7600**	0.7407	0.8000	0.7692
	SVM	0.7300	0.7255	0.7400	0.7327
	RF	0.6800	0.6667	0.7200	0.6923
	XGB	0.6500	0.6415	0.6800	0.6602
	LGBM	0.6900	0.6792	0.7200	0.6990
	NN	0.6000	0.5962	0.6200	0.6078
50	LR	0.7200	0.7037	0.7600	0.7308
	SVM	0.7200	0.7115	0.7400	0.7255
	RF	**0.7300**	0.7170	0.7600	0.7379
	XGB	0.7200	0.6897	0.8000	0.7407
	LGBM	0.7200	0.6964	0.7800	0.7358
	NN	0.6400	0.6250	0.7000	0.6604
100	LR	0.7100	0.6981	0.7400	0.7184
	SVM	0.6900	0.6792	0.7200	0.6990
	RF	0.7100	0.7234	0.6800	0.7010
	XGB	**0.7200**	0.7391	0.6800	0.7083
	LGBM	0.7000	0.7273	0.6400	0.6809
	NN	0.7100	0.6721	0.8200	0.7387
200	LR	0.7100	0.6842	0.7800	0.7290
	SVM	0.7200	0.6897	0.8000	0.7407
	RF	**0.7500**	0.7193	0.8200	0.7664
	XGB	0.7200	0.7037	0.7600	0.7308
	LGBM	0.7100	0.6981	0.7400	0.7184
	NN	0.6900	0.6610	0.7800	0.7156

Table 5.7: Performance of our proposed models (Section 4.3) for different GloVe word embedding dimensions

It can be observed that many of these models have outperformed our previous baselines. LR with 25d GloVe word embeddings has obtained the best performance with an accuracy of 76% and an F1-score of 76.92% on the test set. Among the 50d, 100d and 200d word embeddings setup, the best performing models in terms of accuracy are RF, XGB and RF respectively. The best performing model has 7% more accuracy than our best baseline models. It has also outperformed the best model in the competition by 1%.

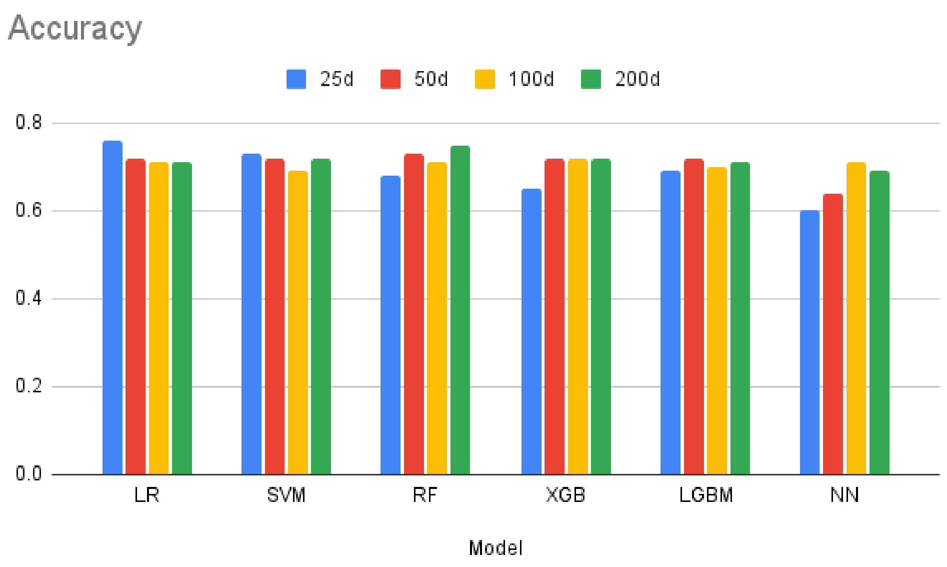


Figure 5.2: Accuracies of our proposed models (Section 4.3) for different GloVe word embedding dimensions

Comparing the models without using a weighted sum: We are interested in studying if introducing the sentiment scores as weights have actually helped improve the performance. To test this out, we build very similar models, except that we take the weight to be "*one*" for all the words, i.e., we take an unweighted sum in Step 5 of the process (Section 4.3.2). We present the results in the case of 25d GloVe word embeddings only (the setup in which the best performance was obtained) in Table 5.8. We also plot the accuracies for easier comparision in Figure 5.3. It can be observed that the performance drops significantly when the weighted sum is replaced with an unweighted sum. This shows that the introduction of sentiment scores as weights is useful.

Setup	**Model**	**Accuracy**	**Precision**	**Recall**	**F1-score**
Our Model (weighted sum)	LR	0.7600	0.7407	0.8000	0.7692
	SVM	0.7300	0.7255	0.7400	0.7327
	RF	0.6800	0.6667	0.7200	0.6923
	XGB	0.6500	0.6415	0.6800	0.6602
	LGBM	0.6900	0.6792	0.7200	0.6990
	NN	0.6000	0.5962	0.6200	0.6078
Unweighted sum	LR	0.6400	0.6296	0.6800	0.6538
	SVM	0.6500	0.6316	0.7200	0.6729
	RF	0.6500	0.6531	0.6400	0.6465
	XGB	0.6000	0.6000	0.6000	0.6000
	LGBM	0.6600	0.6538	0.6800	0.6667
	NN	0.6300	0.6667	0.5200	0.5843

Table 5.8: Comparing the performance of our proposed models (Section 4.3) when the weighted sum using sentiment scores is replaced with an unweighted sum

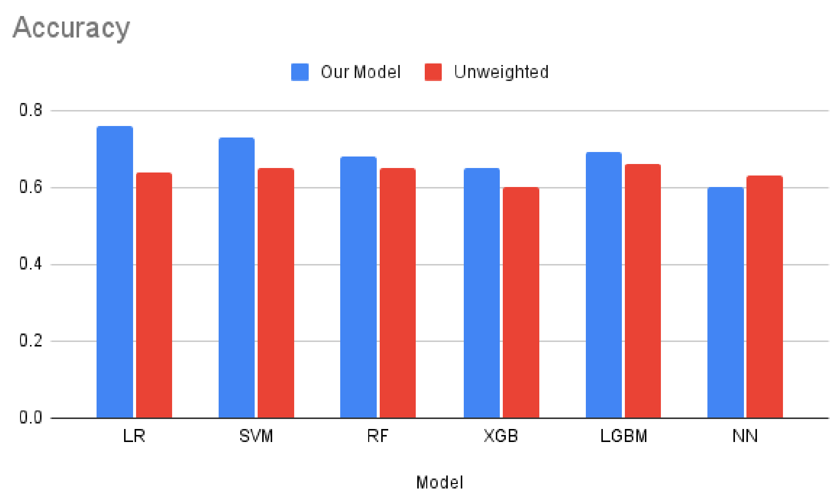


Figure 5.3: Comparing the accuracies of our proposed models (Section 4.3) when the weighted sum using sentiment scores is replaced with an unweighted sum

Ablation	Model	Accuracy	Precision	Recall	F1-score
N/A	LR	0.7600	0.7407	0.8000	0.7692
	SVM	0.7300	0.7255	0.7400	0.7327
	RF	0.6800	0.6667	0.7200	0.6923
	XGB	0.6500	0.6415	0.6800	0.6602
	LGBM	0.6900	0.6792	0.7200	0.6990
	NN	0.6000	0.5962	0.6200	0.6078
-stopwords removal	LR	0.7300	0.7170	0.7600	0.7379
	SVM	0.6800	0.6552	0.7600	0.7037
	RF	0.6500	0.6596	0.6200	0.6392
	XGB	0.6700	0.6809	0.6400	0.6598
	LGBM	0.6500	0.6415	0.6800	0.6602
	NN	0.6400	0.6296	0.6800	0.6538
+lemmatization +stemming	LR	0.7200	0.6964	0.7800	0.7358
	SVM	0.7200	0.7115	0.7400	0.7255
	RF	0.7100	0.6981	0.7400	0.7184
	XGB	0.7000	0.6786	0.7600	0.7170
	LGBM	0.6700	0.6349	0.8000	0.7080
	NN	0.6800	0.6607	0.7400	0.6981
max-pooling	LR	0.5700	0.5745	0.5400	0.5567
	SVM	0.5800	0.5833	0.5600	0.5714
	RF	0.6400	0.6296	0.6800	0.6538
	XGB	0.6100	0.6170	0.5800	0.5979
	LGBM	0.5800	0.5690	0.6600	0.6111
	NN	0.5600	0.5500	0.6600	0.6000

Table 5.9: Performance of our proposed models (Section 4.3) with ablations: (i) no stop-words removal; (ii) using lemmatized/stemmed form of an OOV word; and (iii) max-pooling

Ablation study: We make minor modifications in the model setup and observe how the performance changes with these modifications. In particular, we do the following: (i) do not perform stopwords removal in Step 2 (denoted as "*-stopwords removal*"); (ii) use the lemmatized or stemmed form of the word if it is not found in one of the SentiWordNet or GloVe vocabulary, if the lemmatized/stemmed forms are present, in Step 4 (denoted as "*+lemmatization +stemming*"); and (iii) use max-pooling, i.e., taking maximum value in each dimension, to aggregate the tweet representations and obtain a user representation in Step 5 (denoted as "*max-pooling*"). Note that we only perform one of these modifcations at a time. We tabulate the performance in the case of 25d GloVe word embeddings only (the setup in which the best performance was obtained) in Table 5.9. We plot the accuracies in Figure 5.4. At least one of the positive, neutral or negative sentiment scores is non-zero for a word (in fact, they sum up to one). Hence, not removing the stopwords adds noise to some dimensions of the user representation, which might be the probable reason for the drop in performance.Moreover, some sentiments carrying words like "very" are included in the list of stopwords. Using the lemmatized or stemmed forms of the words must help incorporate some of the OOV words. However, we can also observe performance drops in some of the classifiers. Further investigation might be required before concluding in this regard. We can also observe a significant drop in performance when mean-pooling is replaced with max-pooling for aggregating the tweet representations.

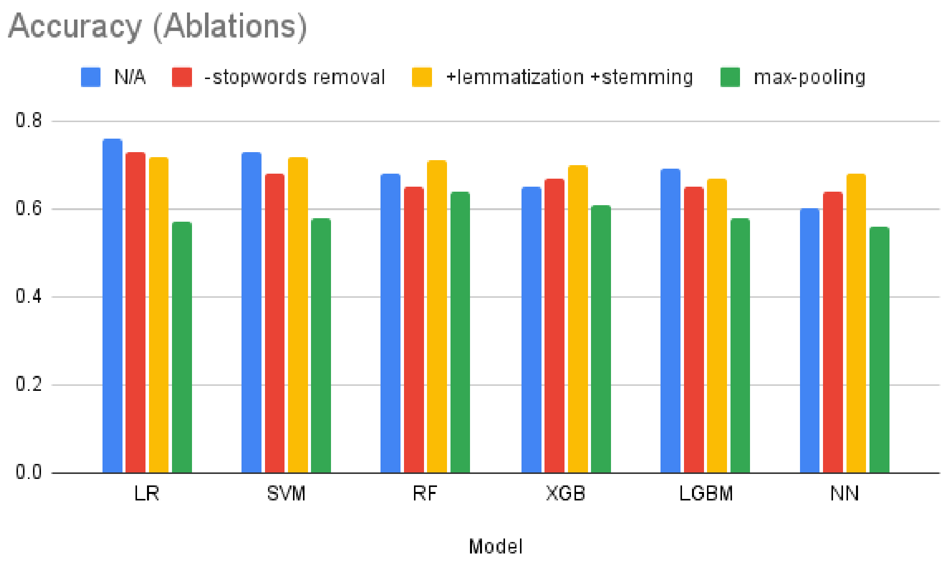


Figure 5.4: Accuracies of our proposed models (Section 4.3) with ablations: (i) no stopwords removal; (ii) using lemmatized/stemmed form of an OOV word; and (iii) max-pooling

5.4 Ensemble

Clearly, the fine-tuned RoBERTa model and the LR classifier with 25d GloVe word embeddings have the best performance in Section 5.2.4 and Section 5.3, respectively. However, since both TF-IDF+LR and TF-IDF+SVM have the same test accuracy in Section 5.2.1, we create two different ensembles, including one in each. Hence, we have two max-voting classifiers: (i) TF-IDF/LR + RoBERTa + 25d/LR; and (ii) TF-IDF/SVM + RoBERTa + 25d/LR. Their performance on the test set is tabulated in Table 5.10.

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
TF-IDF/LR + RoBERTa + 25d/LR	**0.7700**	0.8000	0.7200	0.7579
TF-IDF/SVM + RoBERTa + 25d/LR	0.7600	0.7708	0.7400	0.7551

Table 5.10: Performance of ensemble models (Section 4.4)

We can observe that TF-IDF/LR + RoBERTa + 25d/LR has achieved an accuracy of 77% on the test set, which is a further 1% increase from the standalone 25d/LR model.

5.5 Error Analysis

We perform a detailed error analysis only for the models we proposed (Section 4.3) in this section.

Containing internet slang/typos	**Possibly hate carrying words**
im, dont, thats, yall, lol, aint, youre, cant, gt, gotta, wanna, didnt, doesnt, hes, lets, ive, lm*o, isnt, gonna, rt, theyre, yo, whats, bro, ya, shes, ima, theres, lil, ppl, wasnt, tryna, wont, ur, arent, wouldnt, dems, oh, uk, dr, havent, heres, cuz, couldnt, idk, nah, finna, omg, whos, gon, youll, blm, smh, w, vs, bruh, idc, imma, id, youve, shouldnt, outta, lm**o, thru, yep, til, hasnt, lm*oo, lt, gov, fbi, dem, shouldve, gf, goin, gf, goin	biden, women, n, as, f**king, n***a, joe, covid, democrats, f**k, media, b***hes, sh*t, friends, americans, america, donald, children, guys, states, t*, wt*, obama, f**ked, pelosi, girls, bbc, facebook, cuomo, trumps, god, tommy, republicans, trump, killed, voted, voters, coronavirus, sorry, a*, mad, arrested, h**s, disgusting, illegals, following, m*s, jews, antifa, muslims, as*, whites, cnn, f**kin, liberals, boys, patriots, girl, migrants, elections, cops, immigrants, hillary, guns, chris, meghan, blacks, protesters, womens, females, politicians, a*s, babies, conservatives, d**k, boy, racist

Table 5.11: Sample OOV words for our model

Sample OOV words: We drop those words that are not part of the SentiWordNet or GloVe word embeddings vocabulary (in Step 4). We collected a list of such most commonly occurring OOV words and manually classified them into two categories– (i) containing internet slang or typos; and (ii) possibly hate carrying words. We tabulate them in Table 5.11. Some words are OOV because of invalid/incorrect POS tags obtained while parsing the tweets. Parsing tweets remains an active research area in NLP in itself, and hence, we do not move towards it further. We can also observe that most of the words in the second category are named-entity mentions (e.g., person names, locations, organizations, etc.). Even if we manually assign sentiment scores using heuristics, obtaining word embeddings will remain a challenge because it will require scratch training.

Confusion matrix: We plot the confusion matrix for the LR model with 25d GloVe word embeddings, which had the best performance in Section 5.3, in Figure 5.5. We observe that 14 users have been classified as Hate Speech Spreaders even though they are not (False Positives). 10 users have been classified as non-Hate Speech Spreaders even though they are (False Negatives).

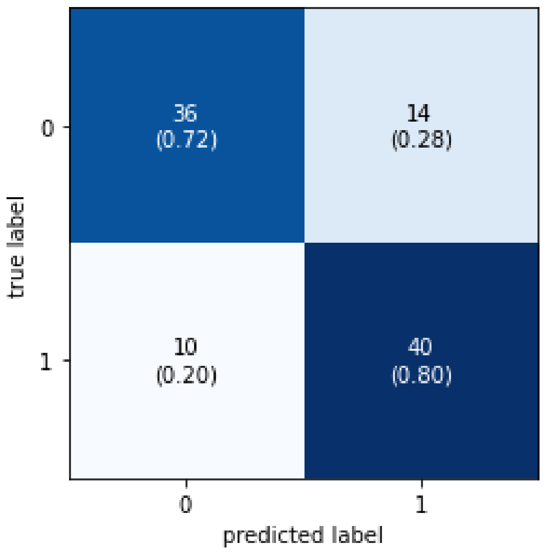


Figure 5.5: Confusion matrix of our best performing model

Number of tweets with abusive word mentions: Among the misclassified users, we find the number of tweets that contain the usage of abusive words. We use the list of abusive words from a GitHub repository[1] and tabulate the number of tweets that contain at least one, two and three abusive words for all the 24 users in Table 5.12. Since the proportion of tweets containing abusive words is rather high for the 14 False Positive users, the proposed system wrongly labels them as positive. In a similar vein, the proportion of tweets having abusive words is relatively small for the 10 False Negative users and are therefore wrongly labelled as negative by our model.

[1]`https://github.com/manoelhortaribeiro/HatefulUsersTwitter/blob/master/data/extra/bad_words.txt`

at least 1	at least 2	at least 3
50	10	1
41	5	0
26	6	0
51	9	3
35	4	0
16	0	0
4	0	0
12	1	0
10	1	0
11	1	0
4	0	0
29	5	0
4	0	0
17	3	0

(a) False Positives

at least 1	at least 2	at least 3
2	0	0
35	2	0
1	0	0
0	0	0
4	0	0
5	0	0
23	4	0
2	0	0
1	0	0
4	0	0

(b) False Negatives

Table 5.12: Number of tweets with at least one, two and three abusive words for the misclassified users

False Positive User	**False Negative User**
#USER# #USER# #USER# B***h you know wt* I mean, actin like you cant tell smh	JJ McCartyney Show 03-10-2021 3-5PM ET Wednesday #URL# via #USER#
#USER# #USER# #USER# You didnt tag me first but Im in b***h	JJ McCartney previews today's special 7th Anniversary special!! #URL# via #USER#
My mood is s**t when my mind is in the wrong place	#USER# Cory Booker is a lying sack of c**p. Terrori/hate crimes committed by whites/bigots? Prove it, you s***bag.
F**king hate doing s**t last min	IQ Al Rassooli is JJ's guest LIVE from 3 to 5pm ET on The JJ McCartney Show #HASHTAG# #HASHTAG# #USER# #URL#

Table 5.13: Some samples of tweets made by two misclassified users

Samples of tweets from misclassified users: Among the misclassified users, we show a few tweets shared by two users; one was falsely classified as positive, and the other was falsely classified as negative in Table 5.13. In the case of the False Positive user, even though the user uses abusive words, it is not targeted toward a particular individual or group. The user uses it casually, and such instances do not count under Hate Speech. This is one of the weaknesses of our model, i.e., it is not entirely aware of the intent. In the case of the False Negative user, a larger proportion of the tweets give updates for a show. However, specific tweets contain Hate Speech (e.g., the third one), but these are quite small in number. Hence, the model has misclassified the user. This is a limitation of any ML model.

Chapter 6

CONCLUSION AND FUTURE WORKS

We focused on identifying those user accounts on Twitter that are keen to spread Hate Speech in this work. We described the dataset used and performed an exploratory data analysis in Chapter 2. We also extracted several features, such as user and stylistic features, sentiments, emoticons, stopwords and named-entity mentions, and their several statistical properties. We used them for constructing baselines in the later sections.

In Chapter 4, we built some baselines and proposed a novel model. The baselines included: (i) using the term frequencies (like TF-IDF Vectorization) to obtain user representation from the tweet texts; (ii) using the handcrafted features to construct the user representation; (iii) using the GloVe word embeddings to represent tweets and an LSTM for classification; and (iv) using a two-stage system to flag the user as Hate Speech Spreader if the number of hate containing tweets is beyond a certain threshold. Our proposed model leveraged the pre-trained word embeddings for encoding the words and incorporated the sentiment scores as weights to mark the importance of the words. It computed a weighted sum to get the tweet representation and aggregated these to obtain the user representation, which was finally fed to an ML classifier like LR, SVM, RF, XGB, LGBM or NN. Finally, we constructed a max-voting classifier (ensemble) using the different models.

We discussed the performance of different models in Chapter 5. The best accuracy we obtained with the baseline models was 69%. We also observed that the "negative sentiment" score had the highest feature importance in baselines developed using the handcrafted features (Section 5.2.2), which further motivated us to work towards our proposed model. Our model achieved an accuracy of 76% on the test set, which was 1% more than the best performance obtained in the competition. The ensemble model consisting of TF-IDF+LR, RoBERTa and LR classifier with 25d GloVe word embeddings from Sections 4.2.1, 4.2.4 and 4.3 achieved an accuracy of 77%, a further 1% increase. Finally, we performed an error analysis where we discussed some limitations of our model.

In the future, we plan to conduct a deeper analysis of the results and investigate these models in more detail. We would also like to address some of the limitations discussed in Section 5.5.

Appendix A

FREQUENCY DISTRIBUTION OF RATIO OF DISTINCT TWEETS

Ratio	Frequency
1.000	64
0.995	8
0.990	5
0.985	3
0.980	2
0.975	1
0.970	1
0.955	4
0.945	1
0.940	1
0.930	1
0.925	1
0.910	1
0.890	1
0.885	1
0.875	1
0.840	1
0.770	1
0.480	1
0.155	1

(a) Class 0 (Not keen to spread hate speech):

Ratio	Frequency
1.000	69
0.995	14
0.990	5
0.985	5
0.975	1
0.970	1
0.955	2
0.935	1
0.850	1
0.775	1

(b) Class 1 (Keen to spread hate speech)

Table A.1: Frequency distribution of the ratio of distinct tweets to total number of tweets

Bibliography

[1] “Digital around the world,” 2022. [Online]. Available: https://datareportal.com/global-digital-overview

[2] J. Bulao, “How much data is created every day in 2022?” 2022. [Online]. Available: https://techjury.net/blog/how-much-data-is-created-every-day/

[3] M. Baggs, “Online hate speech rose 20% during pandemic: ’we’ve normalised it’,” Nov 2021. [Online]. Available: https://www.bbc.com/news/newsbeat-59292509

[4] F. RANGEL, B. CHULVI, G. L. D. L. PEÑA, E. FERSINI, and P. ROSSO, “Profiling hate speech spreaders on twitter,” Mar. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5637013

[5] M. Honnibal and I. Montani, “spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing,” 2017, to appear.

[6] S. Loria, “textblob documentation,” *Release 0.15*, vol. 2, 2018.

[7] V. Basile, C. Bosco, E. Fersini, D. Nozza, V. Patti, F. M. Rangel Pardo, P. Rosso, and M. Sanguinetti, “SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter,” in *Proceedings of the 13th International Workshop on Semantic Evaluation.* Minneapolis, Minnesota, USA: Association for Computational Linguistics, Jun. 2019, pp. 54–63. [Online]. Available: https://aclanthology.org/S19-2007

[8] M. S. Jahan and M. Oussalah, “A systematic review of hate speech automatic detection using natural language processing,” 2021.

[9] M. ElSherief, S. Nilizadeh, D. Nguyen, G. Vigna, and E. Belding, “Peer to peer hate: Hate speech instigators and their targets,” *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 12, no. 1, Jun. 2018. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/15038

[10] P. Chaudhry and M. Lease, “You are what you tweet: Profiling users by past tweets to improve hate speech detection,” *CoRR*, vol. abs/2012.09090, 2020. [Online]. Available: https://arxiv.org/abs/2012.09090

[11] M. Siino, E. Di Nuovo, I. Tinnirello, and M. La Cascia, “Detection of hate speech spreaders using convolutional neural networks,” in *CLEF*, 2021.

[12] W. McKinney *et al.*, “Data structures for statistical computing in python,” in *Proceedings of the 9th Python in Science Conference*, vol. 445. Austin, TX, 2010, pp. 51–56.

[13] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, p. 273297, Sep. 1995. [Online]. Available: https://doi.org/10.1023/A:1022627411411

[14] Tin Kam Ho, “Random decision forests,” in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, 1995, pp. 278–282 vol.1.

[15] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’16. New York, NY, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785

[16] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, “Lightgbm: A highly efficient gradient boosting decision tree,” *Advances in neural information processing systems*, vol. 30, pp. 3146–3154, 2017.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] J. Pennington, R. Socher, and C. Manning, “GloVe: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: https://aclanthology.org/D14-1162

[19] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[21] W. Falcon and The PyTorch Lightning team, “PyTorch Lightning,” 3 2019. [Online]. Available: https://github.com/PyTorchLightning/pytorch-lightning

[22] R. Rehurek and P. Sojka, “Gensim–python framework for vector space modelling,” *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, vol. 3, no. 2, 2011.

[23] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” 2019.

[24] Z. Waseem and D. Hovy, “Hateful symbols or hateful people? predictive features for hate speech detection on twitter,” in *Proceedings of the NAACL Student Research Workshop*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 88–93. [Online]. Available: http://www.aclweb.org/anthology/N16-2013

[25] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” 2019.

[26] F. Barbieri, J. Camacho-Collados, L. Neves, and L. Espinosa-Anke, "Tweeteval: Unified benchmark and comparative evaluation for tweet classification," 2020. [Online]. Available: https://arxiv.org/abs/2010.12421

[27] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365.html

[28] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations.* Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[30] S. Baccianella, A. Esuli, and F. Sebastiani, "SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining," in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10).* Valletta, Malta: European Language Resources Association (ELRA), May 2010. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2010/pdf/769_Paper.pdf

[31] V. Indurthi, B. Syed, M. Shrivastava, N. Chakravartula, M. Gupta, and V. Varma, "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.* Minneapolis, Minnesota, USA: Association for Computational Linguistics, Jun. 2019, pp. 70–74. [Online]. Available: https://aclanthology.org/S19-2009