TweetBLM: A Hate Speech Dataset and Analysis of Black_Lives_Matter-related Microblogs on Twitter

Sumit Kumar
Birla Institute of Technology
Mesra, India
sumit.atlancey@gmail.com

Raj Ratn Pranesh
Birla Institute of Technology
Mesra, India
raj.ratn18@gmail.com

Subhash Chandra Pandey
Birla Institute of Technology
Mesra, India
s.pandey@bitmesra.ac.in

Abstract

In the past few years, there has been a significant rise in toxic and hateful content on various social media platforms. Recently Black Lives Matter movement came into the picture again causing an avalanche of user-generated response on the internet. In this paper, we have proposed a Black Lives Matter related tweet hate speech dataset- TweetBLM. Our dataset is consists of 9165 manually annotated tweets that target the Black Lives Matter movement. The tweets were annotated into two classes, i.e, "HATE" and "NON-HATE" on the basis of their content related to racism erupted from the movement. In this work, we also generated useful insights on our dataset and performed a systematic analysis of various state-of-the-art models such as LSTM, Bi-LSTM, Fasttext, $BERT_{base}$ and $BERT_{large}$ for the classification task on our dataset. Through our work, we aim at contributing to the substantial efforts of the research community for identification and mitigation of hate speech on the internet.

1 Introduction

The expeditious growth in usage of social media platforms and blogging websites passed 3.8 billion marks of active users that use text as a prominent means for interactive communication. As Twitter has 330 million monthly active users, researchers have been using it as a source of data for hate speech (Garland et al., 2020). Furthermore, it allows us to understand patterns in ethnically diverse and vulnerable audiences. A fraction of users use discriminatory communication intended to insult and intimidate specific groups or individuals due to their gender, race, sexual orientation, or other characteristics that have been an obstructive byproduct of the growth of social media.

Also, The global outbreak of COVID-19 has resulted in a general disturbance in the personal, social, and economic lives of the people. The disruption has resulted in an increased level of anxiety, fear, and an outbreak of sturdy emotions (Ahorsu et al., 2020). This has led to bitter incidents across the world, for instance, acts of verbal and physical abuse, online harassment, aggression (Ziems et al., 2020). One of the incidents among many was the Black Lives Matter protest which started from May 26, 2020, the day after an African-American man, was killed during a police arrest. The Movement peaked on June 6, 2020, and is still undergoing, Reports show that around half a million people turned out in nearly 550 places across the United States.

While efforts to educate about racial justice and counter hate have been made via social media campaigns (e.g. the #BLM campaign), but their success, effectiveness, and reach remain unclear. Moreover, online hate speech has a severe negative impact on the victims, often deteriorating their mental health and causing anxiety (Saha et al., 2019).

Thus, it is critical to study the prevalence and impact of online hate and counter hate speech in the COVID-19 discourse. In this paper, we focused on classifying the tweets relating to the incidents of the Black Lives Matter movement during the Covid-19 outbreak into unique classes. The analysis tells the achievement of the campaign using the proposed tweets data; since the purpose of the movement is to show strong support against discrimination of any form.

And since, Black Lives Matter[1] (BLM) is a decentralized movement advocating for non-violent civil disobedience in protest against incidents of police brutality and all racially motivated violence against African-American people.

Therefore, The **Motivation** of our work is to

[1] https://www.weforum.org/agenda/2016/08/black-lives-matter-movement-explained/

leverage different text classification models for the classification task concerned to hate speech related tweets on the proposed dataset in order to have a close analysis of the people's responses from all around the world in this context.

Contribution The three key contribution in this paper is:

- In this work we have published a novel hate speech detection dataset(3) consisting of over 9165 manually annotated Black Life Matter tweets in two classes- 'Hate' and 'Non-Hate'. The dataset is publicly available[2].
- We also presented a deeper insight(3.3) of our collected dataset.
- We performed a systematic comparative analysis(6) of various deep learning models for hate detection task on our dataset.

The rest of the paper is organized as follows: Section 2 talks about the related work done in the field of hate speech detection using ML and DL models. Section 3 contains steps for dataset collection and presents dataset analysis. In section 4, we discuss the methodology we adopted to develop the Hate speech detection model. Section 5 details the survey of baseline models and the experiment setup. Followed by section 6 where we put together the experiment results and conduct a detailed analysis. Finally, with section 7 we conclude the paper.

2 Related Work

In the past few years, Research has been done in the field of natural language processing which involves analyzing and exploring Hate speech hidden in textual representations. For instance, Previous works relied on binary classification such as (Kwok and Wang, 2013), (Djuric et al., 2015) and, by (Nobata et al., 2016). Till now, researchers have proposed several methods for hate speech detection over the past years, varying from classical learning methods, to modern deep learning approaches.

(Taylor et al., 2017) made use of Twitter data to identify online hate speech communities by creating Neural Embedding Models that capture word similarity. Using graph expansion and PageRank scores to bootstrap initial hate speech seed words which enriches bootstrapped words to learn out-of-dictionary terms that bare some hate speech relation and behave like code words.

[2]https://doi.org/10.5281/zenodo.4000539

(Malmasi and Zampieri, 2017) explored content and hate speech analysis of Twitter related posts and applied standard lexical features and a linear Support Vector Machine (SVM) classifier. They used three groups of features extracted for their experiments and the character 4-gram model achieves the best accuracy in the experiment.

Some earlier works (Weber et al., 2013) use sentiment words as features to augment other contextual features. Particularly, They let the model learn its representation based on training data which is generally a deep learning model. The latest DL architectures of text processing generally contain a word embedding layer, focused to capture the semantics meaning of words, mapping each word in the input sentence into a vector of low-dimension as in (Mikolov et al., 2013). The following layers learn relevant latent feature representations, where the processed information is fed into a classification layer that predicts the label of the input sentence.

(Badjatiya et al., 2017) performed hate speech detection, specifically to detect racism and sexism by implementing various deep learning architectures. The architectures included Convolutional Neural Networks (CNNs), Long Short-Term Memory Networks (LSTMs), and FastText (Joulin et al., 2016), along with various features like TF-IDF and Bag of Words (BoW) vectors. Their LSTM classifier with arbitrary embeddings implied to get significantly improved performance compared to baseline methods.

(Gambäck and Sikdar, 2017) also focuses on hate speech detection on Twitter, and used some feature embeddings, such as one-hot encoded character n-gram vectors and word embeddings. According to them, they outperform the baseline in terms of precision and F1-score but not on recall. Similarly, (Park and Fung, 2017) also made use of CNNs with character and word level inputs for the same task. They analyzed using two different scenarios, performing the classification using three labels; none, sexist, or racist at once. Also beginning with the case of detecting 'abusive language' and then further investigate using 'sexist' and 'racist'. Their experiments show that, in general, two cases can have the same performance. But, Their deep learning model does not outperform the traditional methods when it comes to the two-step approach.

Most recently, (Elmadany et al., 2020) explored models based on the Encoder from Transformers(BERT) Model for the purpose of detecting of-

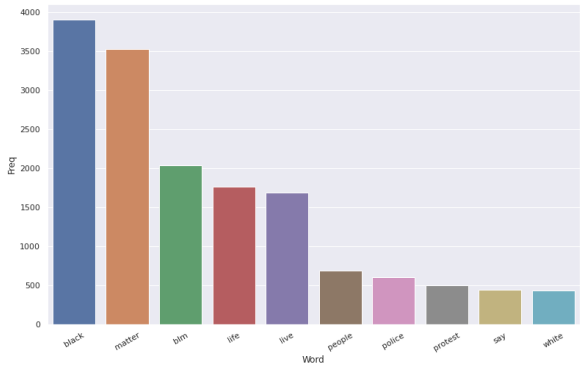

Figure 1: 10 most frequent unigram

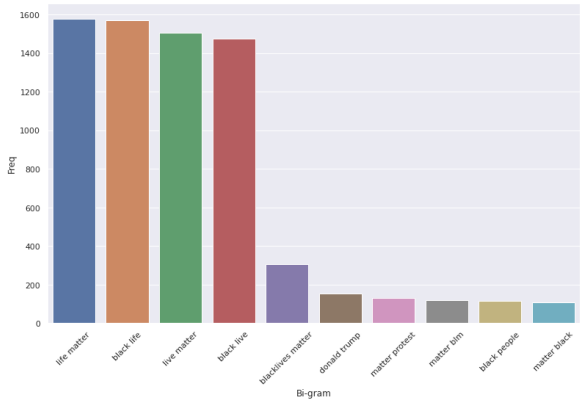

Figure 2: 10 most frequent bi-gram

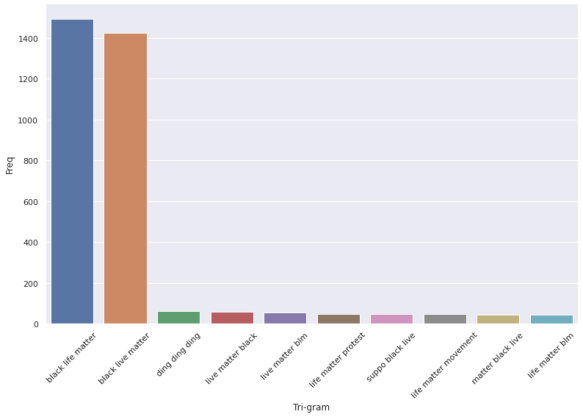

Figure 3: 10 most frequent tri-gram

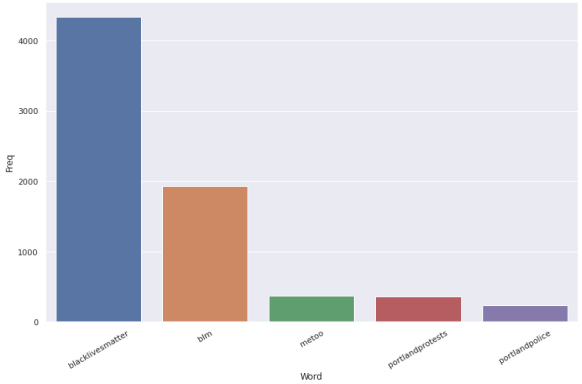

Figure 4: 5 most frequent hashtags

fensive and hateful contents. They fine-tuned one sentiment analysis model and one emotion detection model on their training data.

3 Dataset

This section will explain the generation process and description of the dataset that we introduced in this paper. We condense the approach for collecting and Pre-processing the user-generated dataset through the tweets to come up with a final dataset. We have summarised the features of the dataset through some examples 1, along with the data annotation schemes and guidelines.

3.1 Data Collection

We crawled Twitter data using the Tweepy[3] which is a Python library for accessing Twitter Application Programming Interface (API[4]), and collected a sample of tweets between 27th May 2020 and 26th July 2020. The sample consisted of an average of 1,025,286 million tweets per day. Out of the 7,346,842 tweets, English written tweets were considered for the dataset, and users having more than 150 followers were selected for further steps in order to remove the spam tweets written by bots.

To extract Black Lives Matter movement-related tweets, we build a set of keywords related to the usage of hashtags in the semantic sentence (e.g., #BLM, #BlackLivesMatter) in both lowercase and uppercase. The following keywords such as `Atlanta protest`, `BLM`, `ChangeTheSystem`, `JustieForGeorgeFloyd` were used to collect hate speech specific tweets. Hence, The final collected dataset contains 9165 tweets.

3.2 Data Annotation

The gathered data were annotated based on the categories mentioned in Table 2. The categories were chosen based on the frequency of the occurrence of hate and non-hate text associated with tweets.

A general description of each class is given below.

Mention of Hateful text: This category contains tweets containing information that show hate, at-

[3] https://www.tweepy.org/

[4] https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/api-reference/get-search-tweets

Label	Examples
Hate	"A convicted felon with a history of domestic violence allegedly stabbed his girlfriend to death with a kitchen knife"
	"Marques was shot by this old Portuguese man of three bullets for the simple reason that he was black #BlackLivesMatter"
	"BlackLivesMatter white people who say, i grew up poor so i had no white privilege don't understand what white privileged truly means"
	"#BlackLivesMatter are a bunch of #terrorists, Trump must win #BlueLivesMatte #All-LivesMatter @realDonaldTrump"
	"#BlackLivesMatter Wake Up you are Pawns Being used by the #Democrats Your Lives mean nothing to them! They are insane"
Non Hate	"#BlackLivesMatter This should not be a controversial statement. #BLM"
	"#BlackLivesMatter Trump supporters are seeding violence. This is also happening in Seattle, Portland, and the areas"
	"#BlackLivesMatter White lives matter too"
	"#BlackLivesMatter #CorneliusFredericks We Demand Justice For Cornelius Fredericks"
	"#Blm We cannot change the past, but we can change the consequences for the better"

Table 1: Example tweets from our collected dataset

tacks or demeans a group based on race, ethnic origin, religion, gender, age, or sexual orientation and gender identity.

Mention of Non-Hateful text: This category of tweets contains texts that have information that is neutral and doesn't follow the above-mentioned category or doesn't harm or demeans the emotion and sentiment of a person, group, community, and culture in any way.

Annotation of the dataset was done by five human annotators of linguistic background and proficiency in English to detect the presence of hate speech related to Black Lives Matter. In this work, we use 5 annotators to annotate the gathered dataset. The annotators were males having age between 20-25, out of which 3 are undergraduate students and 2 are Masters student. To begin with annotations, We collected 11029 tweets from the crawling process. Three annotators annotated each tweet separately. We considered those tweets for our dataset which have 100% agreement between at least two annotators among the three annotators. And the final label was decided by the 100% agreement of the remaining other two annotators. The tweets were deleted if there is no agreement between the remaining other 2 annotators. This gathered data is reliable for performing experiments. Finally, we get 9165 number of tweets consisting of 3084 hate speech and 6081 non-hate speech. The class distribution of the TweetBLM dataset is given in Table 3.

Name	Annotation Class
Mention of Hateful text	1
Mention of Non-Hate text	0

Table 2: Label and ID associated with each class

Class	Number of sentence text
Hate	3084
Non-Hate	6081
Total	9165

Table 3: Class distribution of TweetBLM datset

3.3 Dataset Analysis

In this section, we analyzed our collected dataset in order to generate some useful insights. We discovered the top 10 most frequent unigrams, bi-grams, and tri-grams present in the dataset. We also extracted the most frequent hashtags present in the tweets.

Data Preprocessing : Before conducting the analysis and experiments, we preprocessed tweets by firstly converting them to lowercase representation. We also made the tweets free from any unnecessary elements such as username, mentions, links, retweets. We used NLTK[5], a Python module for text processing that removed the English stopwords and performed lemmatization of tweets.

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

Method : Scikit-learn's CountVectorizer[6] module could be used to convert a collection of text documents to a vector of term/token counts. Using CountVectorizer we tokenized the text of the tweet, build a vocabulary of known words, and extracted features from the text of the tweet. As shown in the figure 1, 2 and 3, we then extracted and visualised the top 10 most frequent unigrams, bi-grams and tri-grams present in our tweet dataset. We also extracted the top 5 most frequently used hashtags(see figure 4) present in the tweets.

4 Methodology

In this section, we have sequentially discussed the architecture of various classification models used in our experiment. We used the Random Forest-based classification model. CNN and RNN(LSTM and BiLSTM) based deep learning models. We also used FastText and BERT($BERT_{base}$ and $BERT_{large}$) pretrained encoder for building classification model.

4.1 Random Forest (RF)

RF is an ensemble learning classifier that merges different decision tree classifiers for class prediction (Injadat et al., 2016). The model comprises several decision trees each of which is trained using random subsets of features. The prediction of RF is done through majority voting of the predictions of all the trees in the forest. Following is the description of the RF algorithm as in (Malik et al., 2011):
(i) Select T number of trees
(ii) Select m number of variables for splitting each node, m<<M, where M is the total number of input variables.
(iii) Populate trees while utilizing the below methods:

- Given N training samples, we construct a sample of size N while replacing and growing a tree from the obtained sample.
- Choose m variables randomly from m to get the finest split while populating the tree at each node.
- Let the tree grow to its maximum without any hindrance.

(iv) In order to classify node X, utilize the majority voting to classify the label class.

4.2 CNN

In this subsection, we described the Convolution Neural Networks (Fukushima, 1988) for classification and also outlines the methods for text classification specifically. Convolutional neural networks are multistage trainable neural network architectures developed for classification tasks (LeCun et al., 1998). Each stage contains different layers as summarized below:

- **Embedding Layer:** The function of an embedding layer is to transform the text inputs into a form that can be used by the CNN model. Here, each word of a text document is transformed into a dense vector of fixed size.
- **Convolutional Layer:** A Convolutional layer comprises of several kernel matrices that perform the convolution mathematical operation on their input and process an output matrix of features upon the addition of a bias value.
- **Pooling Layer:** A pooling layer performs dimensionality reduction of the input feature vectors. It uses sub-sampling to the output of the convolutional layer matrices combing neighboring elements. we have used the max-pooling function for the pooling.
- **Fully Connected Layer:** : A classic fully connected neural network layer is connected to the Pooling layers via a Dropout layer in order to prevent overfitting. The softmax activation function is used for defining the final output of this layer. The following objective function is commonly used in the task:

$$E_w = \frac{1}{n} \sum_{p=1}^{P} \sum_{j=1}^{N_l} (o_{j,p}^{L} - o_{j,p})^2 \qquad (1)$$

where P is the number of patterns, $o_{j,p}^{L}$ is the output of j^{th} neuron that belongs to L^{th} layer, N_l is the number of neurons in output of L^{th} layer, $y_{j,p}$ is the desirable target of j^{th} neuron of pattern p and y_i is the output associated with an input vector x_i to the CNN.

In order to minimize the cost function E_w , we use Adam Optimizer (Kingma and Ba, 2014).

[6] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

4.3 RNN

Recurrent neural networks (RNN) have been used to produce promising results on different tasks, along with language model and speech recognition (Kombrink et al., 2011; Graves and Schmidhuber, 2005). An RNN predicts the current output conditioned on long-distance features by keeping a memory based on previous information.

An input layer represents features at time t. One-hot vectors for words, dense vector features such as word embeddings, or sparse features usually represent an input layer. An input layer has the same dimensionality as feature size. An output layer represents a probability distribution over labels at time t and also has the same dimensionality as the size of the labels. Compared to the feedforward network, an RNN holds a relation between the previous hidden state and current hidden state. This relation is made through the recurrent layer, which is designed to store history information. The following equation is used to calculate the values in the hidden, and output layers:

$$\mathbf{h}(t) = f(\mathbf{U}\mathbf{x}(t) + \mathbf{W}\mathbf{h}(t-1)) \quad (2)$$

$$\mathbf{y}(t) = g(\mathbf{V}\mathbf{h}(t)) \quad (3)$$

where U, W, and V are the connection weights to be computed during training, and f(z) and g(z) are sigmoid and activation functions as given below:

$$f(z) = \frac{1}{1+e^{-z}} \quad (4)$$

$$g(z_m) = \frac{e^{z_m}}{\sum_k e_k^z} \quad (5)$$

For the purpose of sequence tagging, we used Long Short Term Memory (LSTM) and Bidirectional Long Term Short Memory (Bi-LSTM) as in (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005; Graves et al., 2013).

LSTM networks use purpose-built memory cells to update the hidden layer values. Therefore, they may perform better at finding and utilizing long-range dependencies in the data, unlike a standard RNN. The following equation implements the LSTM model:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (6)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (7)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (8)$$

$$h_t = o_t \tanh(c_t) \quad (9)$$

For a given time, both past and future input features can be accessed in the sequence tagging task. Therefore, we can also utilize a bidirectional LSTM network (Bi-LSTM) as proposed by the author (Graves et al., 2013).

4.4 FastText

(Joulin et al., 2016) showed that a simple linear classifier can compete with complex deep learning algorithms in text classification. It can be trained to the accuracy achieved with complex deep learning algorithms efficiently, even without using high-performance GPU. FastText uses a bag of words (BOW) and a bag of n-grams as features for text classification. It averages the n-gram features to represent a tweet, trained on stochastic gradient descent with a linearly decaying learning rate, followed by the softmax in the final layer.

4.5 BERT

(Devlin et al., 2018) proposed BERT which is a powerful transformer based model that has been successful in achieving state-of-the-art results on various NLP tasks. It stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. BERT is a contextualized word presentation model, pre-trained using bidirectional transformers (Vaswani et al., 2017). Basically, It utilizes the work for predicting the next sentence and thus, learns the embeddings with a larger context.

The transformer architectures such as $BERT_{base}$ and $BERT_{large}$ will be used for performance analysis. Also, we use a pool of labeled training examples for fine-tuning BERT for hate speech detection task using the balanced set of proposed annotated data. We performed the fine-tuning of the $BERT_{base}$ and $BERT_{large}$ model by building a custom classification head on top of both models. The classification head was consist of a dropout layer(p=0.05) followed by a linear layer(size = 768) with Mish(Misra,

Model	**Accuracy**	**F1**	**Precision**	**Recall**
Random Forest	77.35%	0.775	0.785	0.767
LSTM	76.21%	0.767	0.775	0.759
BiLSTM	77.58%	0.789	0.855	0.733
CNN	79.46%	0.796	0.802	0.791
FastText	82.77%	0.829	0.833	0.825
$BERT_{base}$	87.45%	0.869	0.921	0.823
$BERT_{large}$	**89.13%**	**0.889**	**0.934**	**0.850**

Table 4: Performance score of various models.

2019) activation function followed by an another dropout layer and a final linear layer(size = 768). The averaged pool of sequential output from 12 encoding layers of used as the custom classifier head's input. $BERT_{base}$ uses a 12-layered transformer, 12 attention heads, and 110 million parameters. On the other hand, $BERT_{large}$ uses a 24-layered transformer, 16 attention heads, and 340 million parameters.

5 Experiments

In this section, we described the experiment setup for various classification model used in the experiment. We divided the tweets into train and test dataset. Out of total 9165 tweets, we used 80% of the data for training the models and rest 20% of the data was used for the validation. The hyperparameters were fine-tuned on the validation dataset. For the **Random Forest**, we set the *n_estimators* = 100, *max_depth* = 5, *max_features* = 'auto' and rest of the parameters were set to default configuration. For **CNN** and **RNN** models, we set the max length(*max_length*) of input sequence as 120 and used GloVe embedding (Pennington et al., 2014) of size 300 at embedding layer. In **CNN**, the convolution layer had *filters* = 300, *kernel_size* = 3, *stride* = 1 with relu activation(Nair and Hinton, 2010). For **RNN** models, we used a *SpatialDropout*(p = 0.2), single LSTM(with dropout = 0.2) and BiLSTM(with dropout = 0.2) layer were for the LSTM and BiLSTM model respectively. In both CNN and RNN model, a fully connected layer classify the tweet into hate and non-hate class. We used *categorical_crossentropy* loss with Adam optimizer(Kingma and Ba, 2014). For the **BERT** models- $BERT_{base}$ and $BERT_{large}$, the model had the learning rate of 1e-4 with Adam optimizer(Kingma and Ba, 2014) and textitcategorical_crossentropy as loss function. For the experiment, the model was fine-tuned for 5 epochs. The Random Forest, CNN and RNN models were trained for 25 epochs.

6 Results and Discussion

In this section, we have summarised the result obtained in our experiment and also discussed the performance of various classification models on our dataset. As seen in the table 4, the $BERT_{base}$ and $BERT_{large}$ was the best performing models. They were able to surpass the other models with the highest accuracy score of 87.45% and 89.13% respectively. The possible explanation is that due to the large data used for pretraining of the $BERT_{base}$ and $BERT_{large}$ models and transfer learning increased their contextual understanding and models were able to generalize better. Following the BERT model, the FastText model was able to achieve an accuracy of 82.77. In the case of RNN models, BiLSTM performed better than the LSTM model by achieving higher accuracy of 77.58% which was 1.37% more than LSTM. The CNN model outperformed both RNN models with an accuracy score of 79.46%. Random Forest reported an accuracy score of 77.35%, which was a slight improvement of 1.14% over the LSTM model.

7 Conclusions

In this paper, we presented the Black Lives Matter dataset. We collected a large number of Black Lives Matter movement-related user-generated data from an online platform. Using our dataset, we compared various state-of-the-art models as an attempt to develop a hate speech detection system. In our experiment, we discovered that the large language models such as ($BERT_{base}$ and $BERT_{large}$) outperformed other baseline models as pretraining and transfer learning enhanced the textual feature representation and therefore improved the contextual understanding of deep learning model. We also presented deeper insights into

our dataset by extracting frequent n-grams and hashtags. The experiment results show that one application of our approach can potentially be the identification and filtering of hateful textual contents on social media platform but there is still room for improvement. We believe that this dataset would enable computer scientists to design and develop a more sophisticated, intelligent, and feasibly available advance hate speech detector system. Future work and possible experiments that can be done such as (i) Expanding our data for annotating more tweets that would be beneficial for the research community, (ii) Providing additional features for fine-grained classification.

References

Daniel Kwasi Ahorsu, Chung-Ying Lin, Vida Imani, Mohsen Saffari, Mark D Griffiths, and Amir H Pakpour. 2020. The fear of covid-19 scale: development and initial validation. *International journal of mental health and addiction*.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th international conference on world wide web*, pages 29–30.

AbdelRahim Elmadany, Chiyu Zhang, Muhammad Abdul-Mageed, and Azadeh Hashemi. 2020. Leveraging affective bidirectional transformers for offensive language detection. *arXiv preprint arXiv:2006.01266*.

Kunihiko Fukushima. 1988. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hate-speech. In *Proceedings of the first workshop on abusive language online*, pages 85–90.

Joshua Garland, Keyan Ghazi-Zahedi, Jean-Gabriel Young, Laurent Hébert-Dufresne, and Mirta Galesic. 2020. Countering hate on social media: Large scale classification of hate and counter speech. *arXiv preprint arXiv:2006.01974*.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

MohammadNoor Injadat, Fadi Salo, and Ali Bou Nassif. 2016. Data mining techniques in social media: A survey. *Neurocomputing*, 214:654–670.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Stefan Kombrink, Tomáš Mikolov, Martin Karafiát, and Lukáš Burget. 2011. Recurrent neural network based language modeling in meeting recognition. In *Twelfth annual conference of the international speech communication association*.

Irene Kwok and Yuzhou Wang. 2013. Locate the hate: Detecting tweets against blacks. In *Twenty-seventh AAAI conference on artificial intelligence*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Arif Jamal Malik, Waseem Shahzad, and Farrukh Aslam Khan. 2011. Binary pso and random forests algorithm for probe attacks detection in a network. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 662–668. IEEE.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting hate speech in social media. *arXiv preprint arXiv:1712.06427*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Diganta Misra. 2019. Mish: A self regularized non-monotonic neural activation function. *arXiv preprint arXiv:1908.08681*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153.

Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on twitter. *arXiv preprint arXiv:1706.01206*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Koustuv Saha, Eshwar Chandrasekharan, and Munmun De Choudhury. 2019. Prevalence and psychological effects of hateful speech in online college communities. In *Proceedings of the 10th ACM Conference on Web Science*, pages 255–264.

Jherez Taylor, Melvyn Peignon, and Yi-Shin Chen. 2017. Surfacing contextual hate speech words within social media. *arXiv preprint arXiv:1711.10093*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Ingmar Weber, Venkata R Kiran Garimella, and Alaa Batayneh. 2013. Secular vs. islamist polarization in egypt on twitter. In *Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining*, pages 290–297.

Caleb Ziems, Bing He, Sandeep Soni, and Srijan Kumar. 2020. Racism is a virus: Anti-asian hate and counterhate in social media during the covid-19 crisis. *arXiv preprint arXiv:2005.12423*.