MultiClass Hate Speech Detection

Drishya Uniyal
IIIT Delhi
drishya21119

Harshit Gupta
IIIT Delhi
harshit21028

Jahnvi Kadia
IIIT Delhi
jahnvi21123

Ramyanee Kashyap
IIIT Delhi
ramyanee21139

Abstract

Hate speech is generally an offensive discourse given to a party or an individual. Platforms like Twitter, Reddit, and more, are some sources from which data related to hate speech can be scraped and used for analysis. We can classify Hate Speeches as Hate, Offensive, and more. For this experiment, the dataset contains these two categories. To classify the given tweets, we have performed experiments using sklearn models and convolutional neural networks (CNN) that are pre-trained on word vectors for sentence-level classification tasks. Models have been compared, calculating the accuracy, precision, recall, and f1-score.

1 Problem Definition

In today's free world, much textual data is generated daily in and out. Most of this data is being generated by the usage of Social media platforms which are free for usage and are available to the general public all over the world, provided they have access to the internet. Some of these companies are - Facebook, Twitter, etc. These social media companies provide the best anonymity to their users. But some people use this anonymity to write offensive texts on these platforms for their benefit and ill-intentions, which sometimes also leads to violent hate crimes. Since the amount of data generated on these platforms is huge, it is difficult to stop the spread of hate speeches on such social media platforms.

Social media companies use multiple ways to curb such hate speeches or texts. They have teams of human moderators who keep an eye out for hateful or offensive content and remove them from their platforms before it spreads. Although these social media companies have put in place multiple measures to curb hate speeches, some cases that go unnoticed result in hate crimes. So now, these companies try to use the computational power they have access to and create Classic Machine Learning and Deep Learning models for analysis and classification of Hate Speech from the content on their platforms.

Classical Machine Learning models which are popular and can be used for our purpose are Logistic Regression, Decision Tree Classifiers, Support Vector Machines, etc. We have also used multiple Deep Learning models like CNNs, RNNs and LSTM. The main usefulness provided by RNN is that it holds sequence information over time, which helps hold context about the data. But it also suffers from problems like Vanishing and Exploding gradients. Due to the dependence on its previous content, parallelization is also impossible in RNNs. Hence more advanced Deep Learning models like LSTM and transformers were introduced and used in state-of-the-art models. We will use the mentioned methods for classifying the tweets into Hate, Offensive, and Neither.

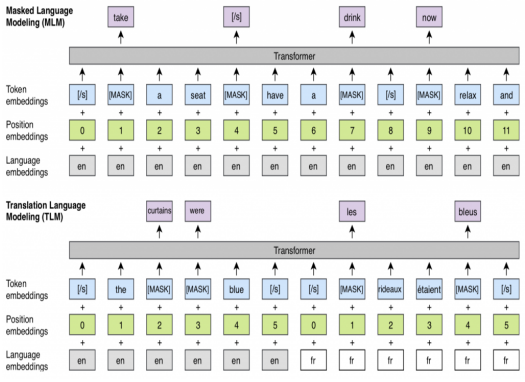


Figure 1: BERT for Text classification

2 Related Works

Davidson et al., 2017(2) talks about training a multi-class classifier to distinguish between hate, offensive, and neither. The bag of words approach increases false positives with high recall, leading to the tweets' misclassification. The linguistic

features are used to distinguish between hate speech and offensive language. The features were lowercase, stemmed and POS-tagged unigram, bigram and trigram, and TF-IDF weighted each. The logistic regression, linear SVM, naive Byes, decision trees, and random forest models were tested using 5-fold cross-validation. Logistic regression and linear SVM comparatively gave better results. Lexical methods give inaccurate results in identifying hate speech. Automated classification methods give comparatively higher accuracy in differentiating these three classes.

Mutanga et al., 2020(4) compared the DistilBERT transformer method with attention-based RNN and other transformer baselines for hate speech detection in tweets. The recurrent neural network(RNN) failed to capture the long-term dependencies, which led to the loss of semantic detailed information, but transformers overcame this drawback. RNN-based long short-term memory(LSTM) and gated recurrent units(GRU) overcame long-term dependencies problems. RNN faces exploding gradient and vanishing gradient problems during backpropagation training that LSTM overcame by minimizing the vanishing gradient. The performance of LSTM decreases as the sequence length increases. Hence, to overcome this problem, the attention mechanism came into existence. Attention captures long-term dependencies irrespective of the distance between input and output sequences and assumes that "every word in each sentence is relevant." When attention is used with RNN, the parallel sequencing is not processed and affects the processing time. By shifting to self-attention and transformers, parallelization and long-term sequence feature capturing are allowed. This is useful while dealing with imbalanced datasets. The evolution of BERT has led to many algorithms, such as RoBERT, XLNET, and DistilBERT. The results show that DistilBERT outperforms all transformer-based and attention-based LSTM methods.

Kim et al., 2014 (1) shows how the small changes in the hyperparameters tunning and static vectors can accomplish better results on multiple benchmarks. The word vectors obtained from the unsupervised neural language model is trained to a simple CNN with one convolution layer. Despite

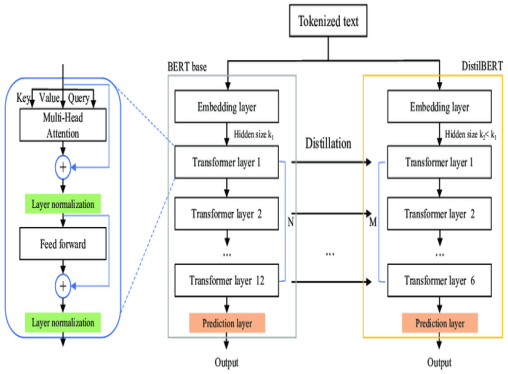


Figure 2: The DistilBERT model architecture and components(https://www.researchgate.net/figure/The-DistilBERT-model-architecture-and-components)

using very little hyperparameter tuning, it gives very good results on multiple benchmarks. Furthermore, the results are improved when task-specific vectors are learned by fine-tuning. The author says that the supervised pre-training of word vectors is important in deep learning for NLP.

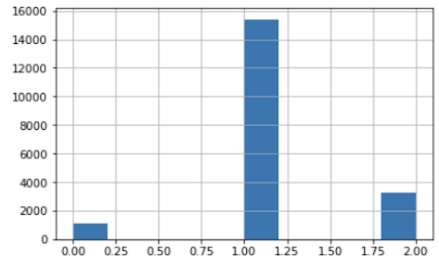
3 Methodology

3.1 Experimental Dataset

The image below shows the structure of the dataset provided. It consists of three columns, Label, Tweet and its Id.

label	tweet	id
1	"@KeyshawnSwag: Lmfao this cat started beating the shit out of me" my nigga	0
1	RT @digaveliavelife: Lol I be eatin da shit outta sum pussy. Ill never be a serial p	1
1	RT @QueenReenie_: How bitch how? “@_Vontethekidd: How is this ok	2
2	Floppy bird chalmers over here taking after his hero LeBron	3
1	"Boolin' in the bando wit a few bitches"	4

Figure 3: Dataset



label	tweet	id
0	1137	1137
1	15398	15398
2	3291	3291

Figure 4: Class Distribution in Training Dataset

The main aim of the experiment is to perform Multiclass Hate Speech Detection on the dataset provided with classes - Hate, Offensive, and Neither. The training dataset contains 19826 tweets and labels 0, 1, and 2. Class 'Hate' has 1137 tweets, Class 'Offensive' has 15938 tweets, and Class 'Neither' has 3291 tweets. The test dataset contains 4957 tweets on which the final predictions will be used for testing purposes.

Table I. Class Distribution		
Label	**Count**	**Percentage**
0	1137	5.73
1	15398	77.66
2	3291	16.59

3.2 Experimental Preparation

For most of the models tested throughout, the training dataset was split into 67% of the training dataset and 33% of the validation dataset. The validation dataset helps in reviewing the model performances. Python was used as the main language for the development of models. Some important libraries include Sklearn, HuggingFace, PyTorch and many more.

3.3 Preprocessing

Before classifying a text into one of the above mentioned categories, we need to process it and make it fit the model for better training. According to the model, the preprocessing steps have been followed differently. Removing the unwanted part of the text makes it better for training.
The tweets in train and test data have been preprocessed for the same purpose.Common steps used for preprocessing are lowering the case of the text removal of URLs, hashtags, punctuations, numbers, stopwords, and special characters. In advance of this, the Davidson model replaces the following with another word, like searches the URL in the string and replaces it with the word URL HERE; this provides better training. After these basic cleanups, Contractions like - "he's" were changed to "he is." Typos and slang were also preprocessed and either removed from the tweets or replaced by some sensible word. For training BERT, the model has directly passed and handled the data.

3.4 Proposed Methods

The dataset was trained and tested using a handful of linear Machine Learning and Deep Learning models. For the linear machine learning models from sklearn, TfidfVectorizer has been used for extracting features from the text data and converting the text data into numerical features. A handful of linear models like Decision Tree Classifier, Logistic Regression, and Support Vector Classifiers were used. The data was also trained by Deep Learning models like CNN, RNN, and LSTM.

During the project's initial phase, we explored the training dataset consisting of 19826 samples labeled 0, 1, and 2. The dataset was highly imbalanced, with 80.3% being in class 1(offensive) alone, while class 2 and class 3 shared 25% and 5.7% data samples, respectively. We started by base-lining the task with various machine-learning-based models trained on TF-IDF vectors of the tweets. The results were underwhelming due to the enormous imbalance among the classes. The best baseline model was Support Vector Classifier and Gradient Boosting Classifier and had an f1-score(macro) of 68%.

In the second phase, we experimented with TF-IDF vectors combined with handcrafted features as introduced by [site Davidson paper]. These handcrafted features were used to perform two types of training: Training a standalone model on the entire feature set. Pipelining models first select features and then perform classification on the selected features. Feature selection was performed for the same model used for classification.

While running the experiments on these classifiers, 10Logistic regression classifier (penalty="l2", C=0.01,multi_class='multinomial', solver='lbfgs,' class_weight='balanced', verbose = 2,n_jobs=-1)and Random Forest classifier (max_depth=15, n_jobs = -1, class_weight = 'balanced', n_estimators = 500, random_state = 0) gives remarkable results on both these settings with weighted f1 score on validation dataset as 85% and 85% respectively.

In the third phase, we moved ahead with various deep-learning models. As proposed by the initial paper on TextCNN by Kim et al., 2014, the architecture uses sentences represented as vectors of words. These words are then converted into 300D vectors giving us a 2D representation for each sentence. Approaches for creating these words vectors are: CNN-rand: (the basic model) where embeddings are randomly assigned to each word. CNN-static: word2vec is used to provide word embeddings. Unknown words are randomly initialized. These em-

beddings are kept fixed. CNN-non-static: As above, but the vectors are fine-tuned (changed during training) CNN-multichannel: Two sets of word vectors are used. Fine-tuning is Done. Convolutions are performed on these 2D representations with different size windows (3,4,5) and then max pooled. The final predictions are made.

The data training has been done for the above-mentioned models, and the f1-score has been calculated along with other evaluation metrics. The parameters used are loss='categorical_crossentropy' and optimizer='adam'; the other remains standard. Apart from TextCNN, LSTM and Bi-LSTM have also been trained on the model, which performs better than CNN.

Attention-based models such as the DistilBERT and the BERT performed encouragingly well in the raw tweets themselves without the need for any preprocessing. Pre-trained models from the Huggingface Transformers library were fine-tuned on our classification dataset with a weighted cross-entropy loss, where class weights are calculated as:

$$ClassWeight_i = 1 - \frac{n\_samples_i}{total\_samples}$$

Where $ClassWeight_i$ is the weight of the $i'th$ class and $n\_samples_i$ is the number of samples having label i and $total\_samples$ is the total number of training samples.

The distilbert-base-cased and distilbert-base-uncased models were trained on our downstream task for 3 epochs with a batch size of 64, a learning rate of 1e-5, and weight decay of 0.01. The bert-base-cased and bert-base-uncased are fine-tuned for 2 epochs with a batch size of 8, a learning rate of 1e-5, and a weight decay of 0.01.

3.5 Model Parameters

For Davidson, we have used two paradigms for training the model. The first one is in which the paper feature selection was performed using a skeletal version of the same classifier as used in the final classification task. The second is where all the features were used to train the model.

4 Experimental Results

Analysis: After comparing all the models and submitting the predictions, BERT-uncased predicted

Table II. ML Models

Algorithm	Parameters	F1-Macro	F1-Weighted
Multinomial Naive Bayes (tuned)	alpha=0.1	0.60	0.84
Logistic Regression	default	0.66	0.87
Decision Tree	criterion="entropy"	0.67	0.87
Linear SVC	kernel='rbf', C=1,gamma='auto'	0.68	0.87
Gradient Boosting	n_estimators=100, learning_rate=1.0, max_depth=1	0.68	0.87
Davidson (Random Forest)	max_depth=15, class_weight= 'balanced', n_estimators=500	0.68	0.85
Davidson (Logistic Regression)	class_weight= 'balanced', penalty= 'l2', solver= 'lbfgs'	**0.70**	**0.85**

Table III. DL Models

Algorithm	Parameters	F1-Macro	F1-Weighted
CNN-static	loss = 'categorical_crossentropy', optimizer='adam'	0.41	0.74
CNN-random	loss = 'categorical_crossentropy', optimizer='adam'	0.51	0.80
CNN-trainable	loss = 'categorical_crossentropy', optimizer='adam'	0.57	0.85
Bi-LSTM	loss = 'categorical_crossentropy', optimizer='adam'	0.65	0.86
LSTM	loss = 'categorical_crossentropy', optimizer='adam'	0.68	0.88
DistillBert-uncased		0.76	0.87
DistillBert-cased(Fine-Tuned)	batch size=64, lr=1e-5, weight decay=0.01, epochs=3	0.76	0.91
Bert-cased	batch size=8, lr=1e-5, weight decay=0.01, epochs=2	0.77	0.91
Bert-uncased	batch size=8, lr=1e-5, weight decay=0.01, epochs=2	0.79	0.91

the test data with the highest accuracy of 79.78 when trained with unprocessed data.

5 Contributions

Drishya Uniyal: Survey of Hate Speech and its existing models, training Sklearn Models like Logistic Regression, Decision Tree, and Boosting. Deep Learning models like TextCNN (variations of CNN with different embeddings); distilled BERT. Davidson Model with Logistic Regression. (3)

Ramyanee Kashyap: Exploratory data analysis, experimentation and tackling of data imbalance using class weights, K-fold cross validation, SMOTE; using handcrafted features from (2) to train on Logistic Regression and Random Forest classifiers, along with performing feature selection and grid search for optimal parameters; fine tuning of BERT and DistilBERT models (cased/uncased).

Harshit Gupta: Training Sklearn Models Multinomial Naive Bayes, Linear SVC and Deep learning models LSTM, Bi-LSTM.

Jahnvi Kadia: Fine Tuning Sklearn Models.
Equal contribution in the report.

References

[1] CHEN, Y. Convolutional neural network for sentence classification. Master's thesis, University of Waterloo, 2015.

[2] DAVIDSON, T., WARMSLEY, D., MACY, M., AND WEBER, I. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media* (2017), ICWSM '17, pp. 512–515.

[3] DAVIDSON, T., AND ZARECKI, J. Github repository hate-speech-and-offensive-language.

[4] MUTANGA, R. T., NAICKER, N., AND OLUGBARA, O. O. Hate speech detection in twitter using transformer methods. *International Journal of Advanced Computer Science and Applications 11*, 9 (2020).