Offensive Language Identification

Kushal Juneja, Udit Narang, Utkrisht Sikka, Vishal Bansal
Indraprastha Institute of Information Technology, Delhi
{kushal19057, udit19120, utkrisht19215, vishal19217}@iiitd.ac.in

1 Problem Definition

Twitter[1] is a social media website where users can write posts(also known as tweets on Twitter). Further, these tweets can be viewed by the followers of the writer. Thus, twitter qualifies to be a social media platform that facilitates sharing of opinions/views/information among the people. But often, the tweets might contain offensive language which should not be circulated.

What is offensive language? It is a language that, according to Oxford University Press, is rude in a way that causes somebody to feel upset or annoyed because it shows a lack of respect. Offensive posts on social media is harmful and potentially dangerous when they target a particular community. Often we find offensive posts being exchanged among people of two enemy countries, two political parties, two religions, or even two personalities. Some of the offensive posts are so harsh, that they could trigger riots, protests and even clashes among different sections of the society. Many times, Twitter has come under great scrutiny and faced backslash from governments and people for offensive tweets by its users. Hence, it becomes all the more important to find ways that could detect offensive tweets, and remove them. Anonymity has led to a surge in instances of offensive language. Social media platforms like Twitter, Facebook, Instagram, and Youtube have a moderation policy for curbing use of offensive language or providing a warning to viewers, when they are about to access offensive content.

In this project, we design and evaluate different machine learning models that given a tweet, can detect whether they are offensive or not. Here offensive is used as an umbrella term to determine any form of profane, abusive, hateful, abusive language being used. This task can be modeled as a binary classification task. We use the OLID dataset[2] to train and test our models. While the original OLID dataset has labels for 3 hierarchical tasks, we only work with labels at the first level, which is a binary classification of whether a tweet is Hateful/Offensive. The first part of our project is to devise ways to convert text into numerical embeddings since neural networks work on numbers and not on strings. The next challenge is to devise a classification algorithm that predicts offensive or not offensive from the numerical embeddings.

2 Related Works

Offensive language detection has gained immense popularity in recent years. A major competition, called SemEval 2019, had its task 6, called OffenseEval, on offensive language detection, categorization of offensive language and offensive language target identification. One of the teams (Guler et al., 2019) used a combination of bi-directional RNN using LSTM cells, CNN, and a Feed Forward Neural Network. They experimented with FastText which they trained on their own and then tried GloVe.twitter embeddings. They found that their FastText embeddings perform better than GloVe as dataset-specific embeddings ideally should capture more details relevant for the task at hand.

Deep Learning approaches were aggressively used by the teams along with ensemble approaches (Zampieri et al., 2019). Some claimed to use external datasets to augment training data. Many top ranked teams reported using BERT along with hyperparameter tuning techniques. Dai et al. (2020) have shown that BERT, with a layer of LSTM and softmax attached at its end, when combined with appropriate preprocessing steps is capable to bag first position in SemEval 2020 Task 12. Some preprocessing techniques they use is emoji to word conversion, Hashtag segmentation, User mention

[1]https://www.twitter.com/

[2]https://scholar.harvard.edu/malmasi/olid

replacement, rare word substitution and truncation of all tweets to a max length of 64.

Bert (Devlin et al., 2019) is a bidirectional Transformer encoder that largely inherits its architecture from the Transformer proposed by Vaswani et al. (2017). Bert uses same architecture in pre-training and fine-tuning, except for the output layers. During pre-training, unsupervised data is used while fine-tuning of BERT is done using labeled data depending on the downstream task. For classification tasks, like offensive language detection, the CLS token is fed into an output layer that can consist of dense, drop out and a final softmax layer.

BERT can gain significant performance improvements when trained longer, with larger batch sizes, on longer sequences, and with a dynamic masking approach (Liu et al., 2019). Roberta was reported to outperform previous BERT on classification tasks of GLUE, namely MNLI, QNLI, RTE and STS-B. Authors of RoBERTa found training of BERT to be very sensitive to Adam epsilon term. HateBERT, a BERT model retrained on RAL-E, a large scale dataset of Reddit comments which are offensive, abusive and hateful, has shown to outperform original BERT on OffensEval, AbusEval and HatEval tasks of SemEval 2019 (Caselli et al., 2021).

fBERT is a BERT model (Sarkar et al., 2021), retrained on SOLID dataset, one of the largest corpus of English language for offensive language identification. In their model, they tokenized the input sentences using WordPiece Embeddings. The tokenized embeddings are then converted to their contextualized embeddings by bert model. The model has 12 transformer encoders, 768 hidden layers and 12 self-attention heads. Retraining is carried out using masked language modeling, where 15% of input tokens are selected and some of them are masked, others replaced with randomly chosen other tokens. fBERT was reported to give better Macro F1 than HateBERT and BERT on HatEval, OLID and HS & O dataset.

Hamdy (2021) has given a fine-tuned model of a very efficient implementation of FastText called BlazingText that gives comparable results as BERT with a faster training time.Further, the research work, also presents a comparison of a combination of models and embeddings , like BERT, RoBERTa, BlazingText, ALBERT, XLM-RoBERTa, TFIDF, Doc2vec, on OLID2019 and OLID 2020 datasets. The study concludes that RoBERTa performs best on both OLID2019 and OLID2020 dataset, however, all models were beaten by BlazingText in terms of time taken in Training.

Further, since, social media comes in so many languages now-a-days, so it is necessary that offensive langauge detection algorithms and models are also designed for multilingual text. In this regard, XLM-R (Connea et al., 2020), is a great performing model. XLM-R is a transformer-based multilingual masked language model that has been pre-trained on text from 100 languages. The model beats multilingual BERT on classification task on XNLI by showing an improvement of 14.6% in accuracy.It also shows an improvement of 2.4% on F1-score on NER.

3 Methodology

3.1 Evaluation Metrics

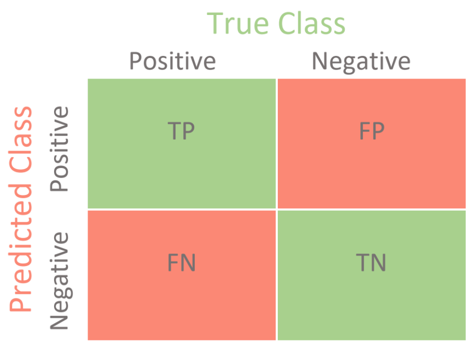


Figure 1: Confusion Matrix

3.1.1 Accuracy

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

Figure 2: Accuracy

3.1.2 F1 Score

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2\frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{2\mathrm{tp}}{2\mathrm{tp} + \mathrm{fp} + \mathrm{fn}}$$

Figure 3: F1 Score

3.1.3 Macro F1 Score

Macro F1 score is computed by taking average of the F1 scores computed for each positive and negative class.

3.2 Dataset

Our dataset consists of a trainset and a testset. Trainset consist of 13240 rows and test consist of 860 rows. Trainset consist of 3 columns, namely, tweet, label, and id. Testset consist of only two columns tweet and id. For an offensive tweet, Label is OFF, while for a not offensive tweet, it is NOT. Since test set does not contain any labels , we split the trainset into training and validation data. For all the models (except for BERT and its variants), we have validation size = 0.33*total training size. We report macro F1 and accuracy on the validation set for each model. The maximum length of any tweet in training set is 103.

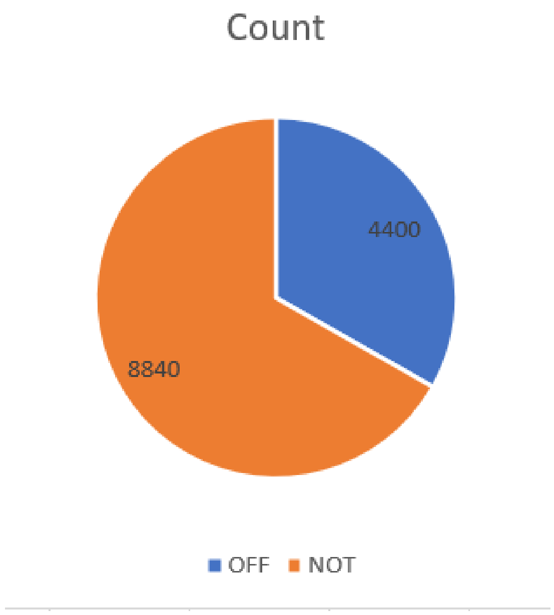


Figure 4: Data composition

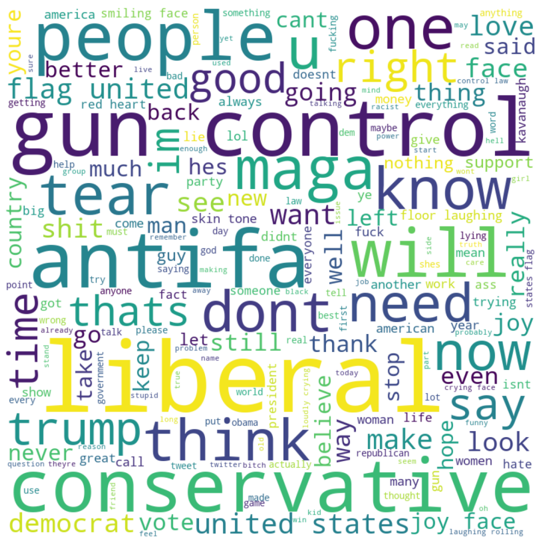
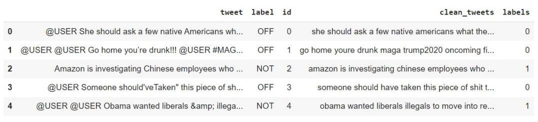


Figure 5: Word Cloud

3.3 Preprocessing

For ML based models and lstm, we used a common pre-processing procedure. First, we removed "@USER", &lt, &gt, numbers, and URLs from the tweets. Then, we replaced ampersand (&) with 'and', and converted webpage apostrophe to its ASCII 39. Then, we lowercased the tweets. Then, we removed decontractions i.e. converted d'ont to do not, n't to not, etc. Then, we removed punctuations and emojis. Then, non ASCII characters were removed. Finally, leading and ending trailling spaces were removed.



	tweet	label	id	clean_tweets	labels
0	@USER She should ask a few native Americans wh...	OFF	0	she should ask a few native americans what the...	0
1	@USER @USER Go home you're drunk!!! @USER #MAG...	OFF	1	go home youre drunk maga trump2020 oncoming fi...	0
2	Amazon is investigating Chinese employees who ...	NOT	2	amazon is investigating chinese employees who ...	1
3	@USER Someone should'veTaken" this piece of sh...	OFF	3	someone should have taken this piece of shit t...	0
4	@USER @USER Obama wanted liberals &amp; illega...	NOT	4	obama wanted liberals illegals to move into re...	1

Figure 6: Pre-Processed Data

3.4 Experiments

3.4.1 ML-Based Experiments

We have extracted the following features from the processed dataset.

- TFIDF
- VADER Sentiment Score
- Doc2Vec embedding
- Glove embedding

Using TFIDF as our train features, we trained machine learning models like Logistic regression, Multinomial Naive Bayes, Passive agressive classifier, decision tree classifier and random forest classifier.

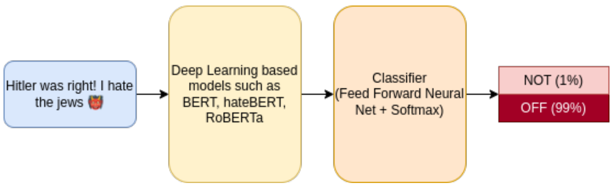
We then extracted the VADER sentiment score for each tweet in our dataset and appended the sentiment score with TFIDF features obtained earlier. Thus using TFIDF + Sentiment score as our features we trained Logistic Regression, Support Vector classifier, Passive Aggressive classifier

Doc2Vec embeddings were extracted and combined with the sentiment and TFIDF. The combined features were used to train Logistic Regression, Passive Aggressive Classifier and Support Vector Classifier.

We then used Glove embedding. To get glove embedding for the entire tweet, we did two experiments. In the first experiment, we extracted the glove embedding for each word in the sentence and then finally took the mean of each word embedding to get the sentence embedding.

For the second experiment, we defined a maximum length for the sentence. For the training set the maximum length of the sentence was 103. We defined the maximum length as 113, i.e. 10 words more than the maximum length for the training set. Thus the dimension of sentence embedding becomes 113(maximum words)*50(embedding length for each word)

3.4.2 DL-Based Experiments


Hitler was right! I hate the jews
Deep Learning based models such as BERT, hateBERT, RoBERTa
Classifier (Feed Forward Neural Net + Softmax)
NOT (1%)
OFF (99%)

We have extracted the features from the following DL-pretrained Models and fine tune them for our processed Dataset:

- BERT Base
- BERT Large
- HateBert
- DistilBert Based
- DistilRoberta Base
- Roberta-Base
- BERT+CNN

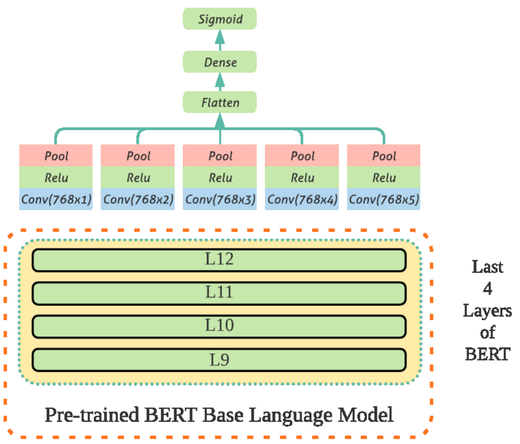


Figure 7: CNN+BERT Architecture

Using Last Layer Embeddings For all models, we have used the last layer of CLS token Embeddings and then fed to a classifier consisting of sets of dense,RELU Activation layers and a Softmax layer. The models used here were pretrained models from the Hugging Face library and fine-tuned using the OLID dataset provided to us. For this section we have used 10000 tweets for training and validation because of the limit provided by hugging face auto-training interface.

For reducing dataset size from 13k to 10k, we followed two approaches. In the first approach, we reduces the number of NOT offensive tweets only since the number of NOT offensive tweets were greater than then number of offensive tweets. In the second approach, we used the trainTestSplit function provided by the sklearn. TrainTestSplit method maintains the ratio of tweets in both the classes before and after the split.

We got better validation results by using the trainTestSplit method, hence we have reported the results only for the dataset obtained using the trainTestSplit method.

Using the last 4 layers of Embeddings For Roberta-base, Bert-Base, DistilRoberta-Base, BERT-CNN above models we used the last 4 layers CLS Token embeddings and concatenated them to form the new feature vector. After extracting the features we feed them to sets of dense, RELU activation layers and Softmax layer.

For BERT+CNN we passed the embeddings to 4 filters each of sizes{2,3,4,5} and each of 32 filters. After that, each passed to RELU activation and max pooled we obtain new set of features which we then passed to the classifier consisting of sets of dense, GELU activation layers and a Softmax layer.

For LSTM, we first, tokenize the tweets using keras' preprocessing text tokenizer. Our LSTM architecture consists of an embedding layer, then bidirectional LSTM layer, then a dense layer with activation 'relu', followed by a dropout layer, and finally a sigmoid layer. We also augment dataset with SOLID testset and compute the metrics again.

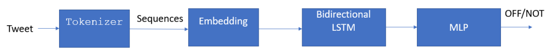


Figure 8: LSTM Pipeline

4 Experimental Results

4.1 ML-Based

4.1.1 TFIDF

Model	Accuracy	F1 Score
Logistic Regression	0.75	0.66
Multinomial NB	0.72	0.62
Passive Agressive	0.70	0.66
Decision Tree	0.71	0.66
Random Forest	0.75	0.66

4.1.2 TFIDF + Sentiment Score

Model	Accuracy	F1 Score
Logistic Regression	0.756	0.70
SVC	0.76	0.71
Passive Agressive	0.76	0.71

4.1.3 TFIDF + Sentiment Score + Doc2Vec Embeddings

Model	Accuracy	F1 Score
Logistic Regression	0.76	0.71
SVC	0.76	0.712
Passive Agressive	0.76	0.715

4.2 DL-Based

4.2.1 Using Last Layer Embeddings

Model	Accuracy	F1 Score
BERT-Large	0.80	0.856
HateBert	0.79	0.85
Roberta-Base	0.80	0.86
Bert-Base	0.80	0.86
Distil-Base	0.80	0.85

As mentioned in the previous section, only 10,000 data points were used for training these models. Hence the validation set used here would be different from the validation set used in other models.

4.2.2 Using last 4 layers of Embeddings

Model	Accuracy	F1 Score
BERT-Base	0.77	0.80
DistilRoberta-Base	0.76	0.79
Roberta-Base	0.75	0.79
BERT-CNN	0.77	0.80

4.2.3 LSTM

Model	Accuracy	F1 Score
Baseline	0.73	0.68
Augmented data from SOLID	0.80	0.78

5 Analysis

5.1 ML-Based Model Analysis

The best model for TFIDF features gave 66 per cent macro F1 score. Adding VADER sentiment score helped to increase the macro F1 score to 71 per cent. Then we tried combining doc2vec embeddings to sentiment score and TFIDF features, we observe that adding doc2vec embeddings did not improve the macro F1 score.

For each combination(TFIDF, TFIDF+sentiment, TFIDF+sentiment+doc2vec), we also predicted the majority voting based predictions, but the results for the majority voting classifier were also same as the results given by the best model in each category.

Using Glove based embedding did not help in improving the F1 score. Macro F1 score while using Glove embedding(Experiment 1) was coming out to be 62 per cent and for the Experiment 2, it was coming out to be 55 per cent

5.2 DL-Based Model Analysis

We used huggingface library to get pretrained transformer models such as BERT, RoBERTa etc. We fine tuned these models by adding a classification layer in front of it and training it. By conducting different experiments in which we change the number of layers of transformer based models trained, we observe that the best results are given by RoBERTa-base. The transformer models give better performance than the Machine Learning based models. The data augmentation with SOLID dataset could not be done on BERT-based models as it required GPU for training, that too for 3 hours per epoch.

6 Contribution of Each Member

Kushal: Literature Review, ML models TFIDF, DL transformer models such as BERT, RoBERTa.

Udit: ML models such as TFIDF, TFIDF + sentiment, TFIDF + sentiment + doc2vec, Glove embedding and DL based models such as BERT, RoBERTa and experimenting various models using huggingface auto training library.

Utkrisht: LSTM, data augmentation on SOLID testset, ensemble approach by giving equal weights to BERT and LSTM probabilities, common preprocessing for ML models, bert-large-based model, Literature Review mentioned in related section,

Vishal: DL-based models such as CNN+BERT, DistilRoberta-Bert,Roberta-Base, and BERT-Base testing on multiple last layers.Literature Review on CNN+BERT.

Equal contribution of all members in report making.

References

Tommaso Caselli, Valerio Basile, Jelena Mitrovic, and Michael Granitzer. 2021. Hatebert: Retraining bert for abusive language detection in english.

Alexis Connea, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzman, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin and Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale.

Wenliang Dai, Tiezheng Yu, Zihan Liu, and Pascale Fung. 2020. Kungfupanda at semeval-2020 task 12: Bert-based multi-task learning for offensive language detection. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, pages 2060–2066.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186.

Batuhan Guler, Alexis Laignelet, and Nicolo Frisiani. 2019. Frisio41 at semeval-2019 task 6: Combination of multiple deep learning architectures for offensive language detection in tweets. *Computation and Language Repository*, arXiv:1903.08734v2.

Ehab Hamdy. 2021. Neural models for offensive language detection.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Diptanu Sarkar, Marcos Zampieri, Tharindu Ranasinghe, and Alexander Ororbia. 2021. fbert: A neural transformer for identifying offensive content. Findings of the Association for Computational Linguistics: EMNLP 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Kaiser Lukasz, and Illia Polosukin. 2017. Attention is all you need. In *31st Conference on Neural Information Processing Systems*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*, pages 75–86.