A Comparative Study of Deep Learning Methods for Hate Speech and Offensive Language Detection in Textual Data

Yogesh Yadav, Parth Bajaj, Rohan Kumar Gupta, Rohit Sinha
Department of Electronics and Electrical Engineering
Indian Institute of Technology Guwahati, Guwahati, India-781039
Email: {yy170102073, parth170102047, rohan_kumar, rsinha}@iitg.ac.in

2021 IEEE 18th India Council International Conference (INDICON) | 978-1-6654-4175-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/INDICON52576.2021.9691704

***Abstract*—The problem of hate speech on social network sites is very prevalent which is being faced by every major social media platform. Several methods have been explored for the purpose of intent-based text classification. Each method has its own pros and cons concerning the type of intent, size of data set, the maximum length of text, etc. Several approaches have been presented in the literature for the hate and offensive speech detection. The main objective of this work is to present a comparative study among select deep learning methods for hate speech and offensive language detection. These methods include recurrent neural network (RNN), convolutional neural network (CNN), long short-term memory (LSTM) and bidirectional encoder representations from transformer (BERT). We have investigated the effect of class weighting technique on the performance of the deep learning methods. Our study finds that the pre-trained BERT model outperforms the other explored models in case of both unweighted and weighted hate speech classification. For offensive language classification, RNN and CNN model outperforms all other models in case of unweighted and weighted respectively. It came out that, the class weighting technique has considerably boost the classification performance of all four models for hate speech.**

***Index Terms*—Hate speech, Offensive language, NLP, Word2Vec, Deep learning models.**

I. Introduction

Recently, the social media platforms such as Facebook, Instagram, Twitter have gained popularity. With the increased number of users, the propagation of hate speech (HS) and offensive language (OL) on social media platforms is on the rise. This motivates the development of software that can recognize different elements in social media and detect HS and OL. One of the biggest challenges faced while detecting HS is to distinguish it from other offensive contents. Defining HS is very important in this context of distinguishing it from OL on social media platforms. Hate speech is very difficult to define for it being subjective in nature. In the general consensus, hate speech is targeted speech to disrespect a disadvantaged race, community, person or gender. In many countries, there are laws which make it illegal to verbally attack a person on the basis of attribute such as skin colour, ethnicity, community, religion, gender and sexual orientations and also to give threats of violence against anybody. Therefore, there a greater need to protect minorities, groups and individuals who experience hate speech in their daily life [1].

In recent years, several techniques have been proposed for HS and OL detection. In [2], the authors have performed extensive experiments for HS detection in tweets with multiple deep learning architectures. In [3], the authors have proposed a state of art method for HS. The method based on a deep neural network combining CNN and gated recurrent unit (GRU). In [4], the author have used the proposed model (CNN-GRU) in [3] to understand prediction behaviour of deep learning models for HS. In [5], to advance the enactment of deep learning models for HS detection, the authors utilized an ensemble method. Datasets used for HS and OL detection are mainly a collection of posts available on social communication sites, like Twitter [1], YouTube [6], and Reddit [7]. In the real world scenario, the proportion of HS instances are very less compared other categories. Most of the datasets, collected from social media platforms, happen to exhibit such distribution among HS and other categories. It is observed from the literature that HS detection tasks face problems due to the scarcity of HS instances in datasets. In several natural language processing (NLP) chores, transfer learning techniques such as universal language model fine-tuning (ULMFiT) [8], embedding from language models (ELMO) [9], generative pre-trained transformer (GPT) [10], and BERT [11] form an inflection point through major breakthroughs especially in the tasks that have data scarcity. In [11], the authors have shown that the BERT method outperforms all other existing transfer learning-based models. In [12], for HS detection, the authors have used BERT based transfer learning technique. The authors experimented with various fine tuning method to improve the efficacy of hate speech detection. The authors have arrived at the conclusion that the CNN based fine-tuning approach can outperform other investigated approaches.

In this paper, our aim is to perform a comparative study of some pre-existing deep learning methods to detect HS and

978-1-6654-4175-9/21/$31.00 ©2021 IEEE

OL in textual data. We have investigated the effect of class weighting technique on the enactment of the deep learning architectures. In Section II, we present a brief review of salient existing deep learning methods considered in this study. Section III describes the textual database and the experimental setups employed for the study. The experimental results are presented in Section IV. Salient findings are discussed in Section V. Finally, the paper is concluded in Section VI.

II. Review of Deep Learning Methods

Recently, various deep learning methods have been used for HS and OL detection. In this section we have discussed them in detail.

A. CNN

A CNN takes the input in the form of matrix and applies a convolutional kernel onto it. The main idea with this is to form a more efficient system to analyze the intricacies of an input. CNN model is initially used and evolved in computer vision domain [13]. A 2-D matrix, representing the pixel values of an image, is used as an input to the CNN architecture. For NLP, the input to the CNN architecture is in the form of 1-D [14]. Generally, A CNN model utilises the following layers to process input data.

- *Input layer*: A layer having input either in 1-D or 2-D matrix form.
- *Convolution layer*: This is the layer where magic happens. In this, a small sized integer kernel is periodically applied to the input matrix. This process is depicted mathematically in Equation 1.

$$\mathbf{S}(i,j) = (\mathbf{I} * \mathbf{K})(i,j) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\mathbf{I}(m,n)\mathbf{K}(i-m, j-n) \tag{1}$$

  where $\mathbf{K}$ is the kernel which convolute with the input matrix $\mathbf{I}$ to produce convolution matrix $\mathbf{S}$ [15].
- *Activation layer*: This layer applies a nonlinear activation function such as sigmoid or tanh or RELU for limiting the dynamic range of the entries of the convolution sum matrix.
- *Pooling layer*: The resultant convolution matrix after applying activation layer can become extremely big and redundant. Its size cab be compressed by doing average or max pooling to adjacent cells.

For employing the CNN for HS or OL detection, each sentence is represented as a matrix composed of individual word vectors. This allows us to achieve lower error rate.

B. RNN

A RNN is one designed for sequential input. At a time step, it takes the current input and the inputs at previous time steps. Basically, RNN can correlate its current input to the previous inputs, as it conveys information about the previous inputs in a hidden layer that is used to compute the output of a given input. RNNs have a property called long short-term memory which makes them very accurate at predicting sequences. The functional relationship is shown in Equation 2.

$$\boldsymbol{h}^{(t)} = \boldsymbol{f}(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}, \boldsymbol{\theta}) \tag{2}$$

where, the current concealed state $\mathbf{h}^{(t)}$ relies on the present input $\mathbf{x}(t)$, the past concealed state $\mathbf{h}^{(t-1)}$ and the network parameter set $\boldsymbol{\theta}$. Thus, the RNNs allows to capture the frequency of occurrence of a particular word or the occurrence of it with a closely correlated word, etc [16]. This in turn enable in predicting the common patterns in textual samples belonging to hate speech or offensive language.

C. LSTM

A LSTM model [16] works very similarly to the RNN, except it provides a lot more flexibility and increases performance compared to the traditional RNN. This is due to the fact that the LSTM model employs *memory blocks*, composed of self-connected memory cells and "Gates". These memory blocks are used to store the state of the network at any given time, allowing for faster computation. And the "Gates" like input, output or forget allow the system to better control the flow of information. Overall, this increases efficiency of the RNN model .

D. BERT

A BERT [11] is a transfer learning based model. It uses Transformer [17] to learn the context. Unlike LSTM and RNN, which trains language model (LM) unidirectionally, BERT trains LM bidirectionally. A unidirectional approach inherently limits the context learning. To mittigate this limitation, BERT, in pre-training, uses two strategies:

- **Masked LM:** It is a technique that allows the the training model itself to test the BERT model by randomly "Masking" a subset of input tokens and asking the model to predict them.
- **Next Sentence Prediction:** This strategy used to help BERT treat whole sentences as single meaningful entities. This help to identify the relationships between sentences, in which the model is asked to predict whether or not a sentence 'B' is the next sentence of a sentence 'A', where sentence 'A' and 'B' can be generated from any monolingual corpus.

fine-tuning is used, after the pre-training step, to adjust the parameters of pre-trained model according to a downstream task. In this study, detection of HS and OL is the downstream task.

TABLE I
DISTRIBUTION OF TWEETS OF DAVIDSON DATASET [1] AMONG THREE CLASSES.

Class	Distribution (%)
HS	5.77
OL	77.43
Neither	16.80

III. EXPERIMENTAL SETUP

A. Dataset

This study has been performed on the Davidson dataset [1] consisting of 24,783 tweets manually labelled by volunteers into 3 classes viz. HS, OL and neither using particular guidelines about the definition of HS as discussed earlier in this paper. The distribution of tweets of the datasets among three classes is shown in Table I. It can be observe that inside the dataset, the number of tweets corresponding to each category differs a lot, due to which the training of models get biased.

B. Pre-processing

It is a process of performing series of operations on input data. It is used to transform the input data in order to make it suitable for machine learning algorithms. The following pre-processing operations are used most widely in NLP tasks:

1) Tokenization: In order to model the textual data, tokenization is the foremost step. It is used in order to chop a input text sentence into smaller units called tokens. In this step, words in an input sentence, that are irrelevant and may disrupt the training process, are converted into simple lists of semantic words. Word tokenization is most widely used in which words are treated as tokens. The tokens of a corpus are used in order to create the vocabulary of the corpus. The vocabulary contains all unique tokens of the corpus.

2) Word Embedding: Feeding a sequence of words/token into a neural network is not possible. Neural networks only takes numerical data as input. Word embedding is a process to represent a word by a unique numerical value. Word representations are broadly categorised into two categories: fixed and distributed.

One-hot encoding is a popular technique of fixed word representation. In this technique, each word in a corpus is represented by a vector of size equals to the vocabulary size plus 1. Where extra 1 is added for out of vocabulary token. The vector is filled with zeros except 1 at a unique place. Since the vector is sparse, it requires large memory for computation. Also, the representation doesn't encodes Semantics which makes learning process of neural networks very difficult.

Distributed word representation is utilized in order to mitigate the limitations of fixed word representation. Word2Vec is one of the widely used distributed word representation. It is developed by Gensim [18]. It uses neural networks to form vectors for each word by learning the word associations from a large word corpus. It detects the context around which the word is used and correlates it with other words in similar contexts. For instance, the words "good" and "great" would have largely correlated vectors, as they are often used around the same context. In this paper, we have used word2vec that implemented using skip-gram model.

C. Implementation

We have implemented CNN, RNN and LSTM using Keras library. Whereas, BERT model is implemented using Ktrain pre-trained model. In all models, to get the output of trinary classification, a softmax activation function of 3 output cells is applied.

For the implementation of CNN model, We have calculated the maximum length of tweets to define the input size. Here, three convolutional layers are used having three kernels of sizes 3, 4 and 5, in which max pooling is applied. Then we feed the convolutional matrix to the flattened layer consisting of 6272 neurons having ReLU activation function. To avoid overfitting, a dropout layer of 0.7 is used between different layers of our CNN.

For RNN and LSTM models, the dimesion of input layer is 300, fed to the dense hidden layer consisting of 128 neurons having ReLU activation function. We have used a dropout layer of 0.7 to avoid overfitting.

A pre-trained $\mathbf{BERT}_{BASE}$ model [19], having 12 layers and 768 neurons in the hidden layer is implemented. The pre-trained mode is fine-tuned using Davidson dataset [1] for HS and OL detection.

D. Class Weighting Technique

In this study, the used dataset [1] has skewed distribution of the classes which can lead to bias results. To mitigate this bias, we analysed the affect of giving weights to each class. To compute weight of each class, the formula mentioned in Equation 3 is used.

$$w_j = \frac{n}{m \times n_j} \tag{3}$$

Where w_j is the weight of j^{th} class, n is the total number of samples is the dataset, m is the total number of classes, n_j is the total number of samples in the j^{th} class.

E. Comparison Metrics

The classification results of each method are presented in the form of confusion matrix. The confusion matrix enabled us to visually analyse the performance of our models to classify texts into different categories. It allowed us to see how accurate our models are performing to identify each category. We have also presented the results in terms of other distinct metrics like

TABLE II
COMPARISON OF CLASSIFICATION PERFORMANCE OF CNN, RNN, LSTM AND BERT MODELS IN TERMS OF DISTINCT METRICS.

CNN Model				
	Unweighted		**Weighted**	
Metrics	**Micro avg**	**Weighted avg**	**Micro avg**	**Weighted avg**
Precision	.72	.88	.69	.88
Recall	.66	.89	.71	.87
F1 score	.67	.88	.70	.88
Accuracy	.89		.87	
RNN Model				
	Unweighted		**Weighted**	
Metrics	**Micro avg**	**Weighted avg**	**Micro avg**	**Weighted avg**
Precision	.74	.85	.60	.86
Recall	.58	.87	.73	.78
F1 score	.60	.85	.63	.80
Accuracy	.87		.78	
LSTM Model				
	Unweighted		**Weighted**	
Metrics	**Micro avg**	**Weighted avg**	**Micro avg**	**Weighted avg**
Precision	.68	.83	.59	.83
Recall	.56	.85	.66	.77
F1 score	.58	.83	.60	.80
Accuracy	.85		.77	
BERT Model				
	Unweighted		**Weighted**	
Metrics	**Micro avg**	**Weighted avg**	**Micro avg**	**Weighted avg**
Precision	.75	.90	.67	.92
Recall	.74	.91	.80	.76
F1 score	.75	.91	.67	.81
Accuracy	.91		.76	

accuracy, precision and recall, F1 score. These metrics presents overall performance. Below we have parented the overview of the metrics:

- **Accuracy**: This is one of the most popular metrics. It denotes the proportion of the instance that are correctly classified over total instances.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Where, *TP*/*TN* denotes the proportion of instances that truly belongs to positive/negative class and also classified into positive/negative class. *FP*/*FN* denotes the proportion of instances that truly belongs to negative/positive class but classified into positive/negative class.

- **Precision**: It is the ratio of the instances that are correctly predicted positive to the total positive instances.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

- **Recall**: It is the ratio of number of positive instances that are predicted correctly to total number of instances in the positive class.

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

- **F1 score**: In few cases, accuracy may lead to inaccurate results. F1 score provides comparatively more accurate result by taking recall and precision into account.

$$F1score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{7}$$

TABLE III
CONFUSION MATRIX FOR MULTI-CLASS CLASSIFICATION AMONG THE CLASSES HS, OL, AND NEITHER (N) CLASSES FOR CNN, RNN, LSTM AND BERT MODELS.

CNN Model						
	Unweighted (%)			**Weighted (%)**		
Class	**HS**	**OL**	**N**	**HS**	**OL**	**N**
HS	16.49	70.25	13.26	38.35	49.82	11.83
OL	1.27	95.20	3.53	4.96	91.48	3.56
N	0.97	13.56	85.47	2.66	12.71	84.62
RNN Model						
	Unweighted (%)			**Weighted (%)**		
Class	**HS**	**OL**	**N**	**HS**	**OL**	**N**
HS	7.89	85.3	6.81	56.63	28.32	15.05
OL	0.52	96.73	2.75	13.4	77.73	8.88
N	0.12	30.39	69.49	6.66	8.96	84.38
LSTM Model						
	Unweighted (%)			**Weighted (%)**		
Class	**HS**	**OL**	**N**	**HS**	**OL**	**N**
HS	10.04	81.36	8.60	41.94	41.94	16.13
OL	1.01	95.56	3.43	12.59	80.35	7.06
N	0.48	38.26	61.26	5.33	19.37	75.30
BERT Model						
	Unweighted (%)			**Weighted (%)**		
Class	**HS**	**OL**	**N**	**HS**	**OL**	**N**
HS	36.56	54.48	8.96	79.57	10.39	10.04
OL	3.14	94.47	2.39	23.94	73.39	2.67
N	0.73	7.51	91.77	10.17	2.06	87.77

IV. RESULTS

In this section, we present the results of select deep learning methods on Davidson dataset [1] for HS and OL detection. These methods include CNN, RNN, LSTM and BERT models.

In the Table II, the performance metrics of all four methods used for the classification is shown. For further insights about the categorical classification performance we have shown confusion matrices in the Table III. Both the tables show the performances of unweighted and weighted classification for all four models.

CNN model fared the worst of all models for HS classification and gave 38.35% accuracy on HS for weighted case. However, it is significantly better than the unweighted case. On the other hand, for OL class, the CNN model gave best performance with 91.48% accuracy in the weighted case. From

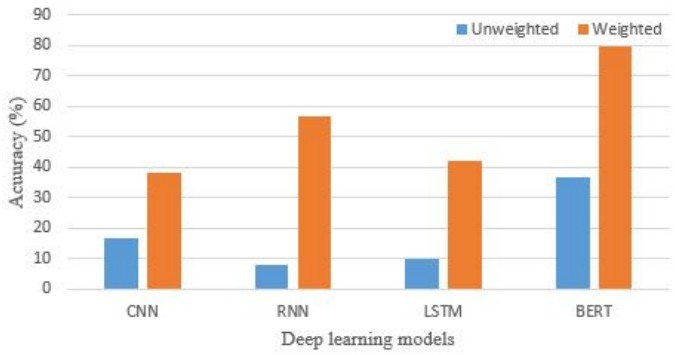

Fig. 1. Comparison of performance of distinct deep learning models in weighted and unweighted classification settings for HS detection.

Table I it can be observed that the overall accuracy (87%) is highest in the weighted case.

From Table III it can be observed that the improvement in accuracy for hate speech, caused by class weights, is highest for RNN model.

For HS, LSTM model worked similar to the CNN model. It gave a marginal (3.59%) improvement over 38.35% classification performance achieved by the CNN model. Whereas, for OL case, the LSTM model yielded a degradation of 11.13% with respect to the classification performance achieved by the CNN model.

BERT model performed the best for HS and yielded the classification performances of 79.57% and 36.56% for weighted and unweighted cases, respectively. However, for OL case, the BERT model gave the worst performances for both weighed and unweighted classification settings. From Table II, it can be observed that it gave best overall accuracy 91% in the unweighted case. However, the performance in the weighted case is severely degraded.

Figure 1 shows the performance comparison among the models utilized in this work. It can be observe that, for HS, the accuracy of all four deep learning models has improved significantly in the weighted setting in contrast to the unweighted one.

V. Discussion

In the real world scenario, the proportion of HS instances are very less compare other categories. The Davidson dataset [1], used in this study, represents the real world scenario. In this study, class weighting technique is used to balance the dataset. We have investigated its effect on the individual class accuracy as well as overall accuracy of distinct deep learning models. From Table II, it can be observe that the overall performance of each deep learning model has deprive upon utilization of class weighting technique. However, from Table III and Figure 1, it can be observe that the accuracy of HS has significantly improved upon utilization of class weighting technique. To verify this, we can refer the existing works in the literature. In [4], the accuracy of a state-of-the-art CNN-GRU model for HS on Davidson dataset is reported as 31% which is significantly lower than the accuracies of the all models developed in this study. In [12], the accuracy of HS for best performing fine tuning based BERT model on Davidson dataset is reported as 29.58% which is also significantly lower than the accuracies of the all models developed in this study. Further, on comparing with the BERT model accuracy reported in [12], the weighted BERT model developed in this work achieves 49.99% absolute improved accuracy for HS classification.

VI. Conclusion

This paper presents a comparative study of some pre-existing deep learning methods to detect HS and OL in textual data. These methods include CNN, RNN, LSTM and BERT models. We have investigated the effect of class weighting technique on the enactment of the deep learning methods. Our study finds that the pre-trained BERT model outperformed the other explored models for HS in case of both unweighted and weighted classification. We believe that it is because of the property of BERT to measure the relation between sentences by treating them as whole units. BERT allows for the space for sentence interpretation as a whole, instead of simply word interpretation, like all the other models. For OL classification, RNN and CNN has surpassed all other investigated models in case of unweighted and weighted respectively. It came out that the class weighting technique has considerably boost the classification performance of all four models for HS. After the verification from the existing works in the literature, we arrived at the conclusion that the class weighting technique can improve the classification accuracy of HS belonging to an unbalanced dataset.

References

[1] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *Proc. of the 11th International AAAI Conference on Web and Social Media*, 2017, pp. 512–515.

[2] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep learning for hate speech detection in tweets," in *Proc. of the 26th International Conference on World Wide Web Companion*, no. 2. ACM Press, 2017, p. 759–760.

[3] Z. Zhang, D. Robinson, and J. Tepper, "Detecting hate speech on twitter using a convolution-GRU based deep neural network," in *Proc. of the European Semantic Web Conference: The Semantic Web*, vol. 10843, 2018, pp. 745–760.

[4] C. Wang, "Interpreting neural network hate speech classifiers," in *Proc. of the 2nd Workshop on Abusive Language Online (ALW2)*. Association for Computational Linguistics, 2018, pp. 86–92.

[5] S. Zimmerman, U. Kruschwitz, and C. Fox, "Improving hate speech detection with deep learning ensembles," in *Proc. of the Eleventh International Conference on Language Resources and Evaluation*, no. 05, 2018, pp. 2546–2553.

[6] R. Ottoni, E. Cunha, G. Magno, P. Bernardina, W. Meira Jr., and Almeida, "Analyzing right-wing youtube channels: Hate, violence and discrimination," in *Proc. of the 10th ACM Conference on Web Science*, 2018, p. 323–332.

[7] A. Mittos, S. Zannettou, J. Blackburn, and E. D. Cristofaro, "And We Will Fight For Our Race!" A Measurement Study of Genetic Testing Conversations on Reddit and 4chan," in *Proc. of the 14th International AAAI Conference on Web and Social Media*, 2020, pp. 452–463.

[8] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proc. of the 56th Annual Meeting of the Association for Computational Linguistics"*, no. 07, 2018, pp. 328–339.

[9] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, no. 06, 2018, pp. 2227–2237.

[10] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," *Technical Report, OpenAI*, 2018.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proc. of the North American Chapter of the Association for Computational Linguistics*, 2019, pp. 4171–4186.

[12] M. Mozafari, R. Farahbakhsh, and N. Crespi, "A BERT-based transfer learning approach for hate speech detection in online social media," in *Proc. of the Complex Networks and Their Applications VIII*, 2019, pp. 928–940.

[13] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, no. 01, 2012.

[14] Y. Kim, "Convolutional neural networks for sentence classification," *Empirical Methods in Natural Language Processing*, no. 08, p. 1746–1751, 2014.

[15] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *ArXiv*, vol. abs/1511.08458, 2015.

[16] M. M. Lopez and J. Kalita, "Deep learning applied to NLP," *ArXiv*, vol. abs/1703.03091, 2017.

[17] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. of the 31st Conference on Neural Information Processing System*, 2017, pp. 1–11.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. of the 1st International Conference on Learning Representations*, 2013.

[19] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: On the importance of pre-training compact models," *ArXiv*, vol. abs/1908.08962, 2019.