Hate Speech Detection Using Natural Language Processing

Stanford CS224N Custom Project

Neha Keshari, Durga Malladi, Utkarsh Mittal
Department of Computer Science
Stanford University
nehakesh@stanford.edu, dmalladi@stanford.edu, mittal12@stanford.edu

Abstract

There is a pronounced usage of Hate Speech in Online Social Networks, with potential harm to society and targeted hatred towards communities and individuals. While automated AI based Hate Speech detection mechanisms are in place in these online platforms, detection and classification of Hate Speech remains a hard problem with high false positive rates, wherein offensive but non-hate content is incorrectly classified as hate speech, along with new hate words and variants introduced constantly. In this paper, we tackle the problem of Hate Speech Detection by comparing the ternary (hate, offensive, non-hate) classification performance and model complexity of four different Deep Neural Network models - CNN (baseline), Bi-Directional LSTM with Attention, Pretrained BERT and Finetuned RoBERTa Transformer models - with Accuracy, F1-score and Matthews Correlation Coefficient (MCC) metrics on the validation set. While the baseline CNN model achieves an accuracy of 0.77, the Bi-Directional LSTM, BERT and finetuned RoBERTa models achieve an accuracy of 0.90-0.92. Further, the best MCC performance is achieved by the Transformer models, with pretrained BERT and finetuned RoBERTa models scoring at 0.75 and 0.76 respectively.

1 Key Information to include

- Mentor: Swastika Dutta
- External Collaborators (if you have any): NA
- Sharing project: NA

2 Introduction

Online Social Networks (OSN) and Microblogging websites have grown tremendously over the past few years, attracting more internet users than any other kind of websites. While these forums offer an open space for people to discuss and share thoughts, they also lead to an increase in conflict with a pronounced usage of Hate Speech. Hate Speech - which refers to the use of aggressive, violent or offensive language targeting a specific group of people sharing a common property such as gender, ethnicity, race, beliefs or religion - is now considered a serious world-wide problem with the growth of OSNs, where interactions between people is indirect and people's speech tends to be more aggressive when they feel physically safer.

While Hate Speech is forbidden in these forums, both the nature and sheer magnitude of posts, comments and messages exchanged, make it very hard to control and filter content. The nature of speech is such that very often it is not easy to decide whether a sentence contains hate or sarcasm or just a joke. As an example, consider the following three tweets:

- *Hey dummy, it's been a while since we last read one of your useless comments.*

- *I hate seeing this team lose all the time!*
- *I hate these foreigners!*

The first tweet happens to be a sarcastic joke between two friends, the second tweet is an expression of frustration, while the third tweet is an example of xenophobic hate speech.

Our interest in the topic began with the observation that the problem of automated Hate Speech Detection or generally speaking Text Moderation is not an easy problem to solve, with a lot of moderation systems marking ordinary content as hate language. A significant challenge emerges from the fact that perpetrators intentionally obfuscate certain words by deliberately misspelling them, creating new words never seen by word-based models.

Early work on hate speech detection relied on dictionaries and n-grams of hate words or expressions, but that would have led to the second tweet being classified as hate speech as it contains the word *hate*. Sentiment analysis has the same drawback, as it attempts to detect sentiment polarity of the sentence by detecting positive/negative words or expressions.

3 Related Work

At the core of any computational linguistic system is an understanding of the underlying language and that can be broken down into four main areas (1) Language Modeling, which implicitly captures syntactic and semantic relationships among words or components (2) Morphology, which is concerned with finding segments (roots, stems, prefixes, suffixes) within words (3) Parsing, which examines how different words and phrases relate to each other (4) Semantics, which involves understanding the meaning of words, phrases and sentences.

Some of the early work showed that writing patterns are effective in text classification tasks such as sarcasm detection (8)(2), multi-class sentiment analysis and sentiment quantification(7). In (4) the authors extract patterns of hate speech and expressions, and use these together with unigrams, sentiment-based and semantic features to detect hate speech on Twitter. Specifically, the extracted features include sentiment-based features (score of positive and negative words, slang words, emoticons, hashtags), semantic features (number of exclamation/question/full-stop marks, all-capitalized words, quotes, interjections, laughing expressions, words in tweet), unigram features (part-of-speech PoS tag of a noun, verb, adjective, adverb) and pattern features (mapping of sentiment and non-sentiment words into a corresponding PoS tag).

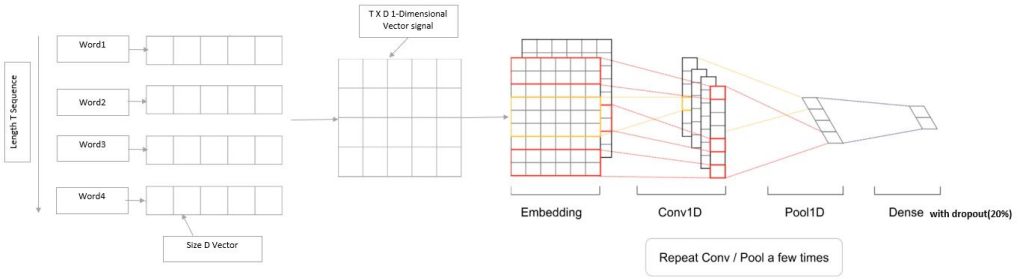
In recent years, several studies have explored the use of Transformer-based models, such as BERT and RoBERTa, for hate speech detection. For instance, in the paper "Hate Speech Detection with BERT using PyTorch Lightning" by Budhraja and Agarwal (2020) (1), the authors proposed a BERT-based model for detecting hate speech in social media. They fine-tuned the pre-trained BERT model on a hate speech dataset and achieved state-of-the-art performance. Similarly, in the paper "RoBERTa: A Robustly Optimized BERT Pretraining Approach" by Liu et al. (2019) (6), the authors introduced RoBERTa, a new pre-training approach that achieved state-of-the-art performance on a range of NLP tasks, including hate speech detection.

4 Approach

We explore four different models to tackle the problem of "Hate" vs. "Offensive" vs. "Neither" speech detection.

4.1 Baseline CNN model

The model architecture consists of an embedding layer, a convolution layer, max pool layer, and a dense layer with dropout. The convolution layer uses 128 filters (kernels) and a kernel size of 3. Max pooling layer was used in all layers in the network. GloVe embeddings were used with a 1D CNN model. Max sequence length of 100 was applied to limit the length of the input sequence after word embedding. Next, the data was passed through three convolutions and three max pool layers. In the last pooling layer, Global max pooling was used. Lastly, we have a couple of dense layers with a dropout of 0.2 to avoid model overfitting, with a sigmoidal activation to give the probability of all three binary classifications.



4.2 Bidirectional LSTM model with attention layer

The Bi-LSTM model uses a deep neural network architecture for text classification, where the input data are sequences of tokens of variable length. The model consists of an embedding layer followed by two bidirectional LSTM layers, each with 128 units and a dropout rate of 0.2. The bidirectional LSTM layers process the input sequence in both directions, which helps capture contextual information from the surrounding words. Using two bidirectional LSTM layers helped to improve the model's ability to capture complex and abstract relationships between words in an input sequence, and ultimately improve its performance on text classification tasks. Next, the model includes an attention layer that computes attention scores for each word in the input sequence based on its importance to the classification task. After the attention layer, the model includes a dropout layer with a rate of 0.2 to prevent overfitting. Then, a flatten layer is added to convert the 3D output from the previous layers to a 2D output. Finally, a dense layer with a softmax activation function is added to output a probability distribution over the three possible classes. Pre-trained Glove word embedding is used with the model architecture designed by us. (10) and (5) were referenced for the implementation of the attention layer and Glove embedding layer, respectively.

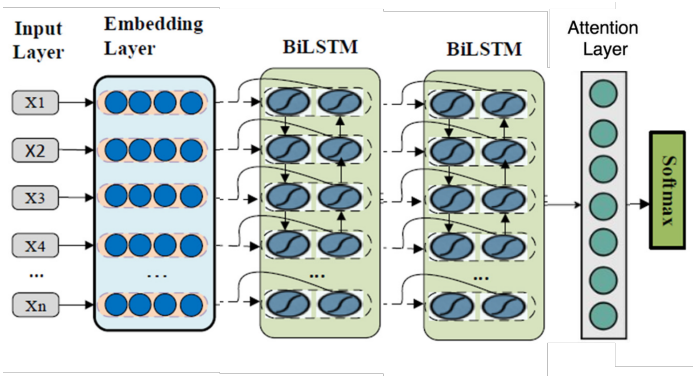


Figure 1: Architecture of Bidirectional LSTM Model

4.3 Pretrained Bert with Adamw optimizer Transformer model

Bert (3) is Transformer model architecture. It uses multilayer bidirectional transformer encoders for language representations. There are two types of BERT models are introduced namely BERTBase and BERTLarge based on the depth of model architecture. For this project, we used Base Bert which has 12 layers of transformers block with a hidden size of 768 (Embedding size) and a number of self-attention heads as 12 and has around 110M trainable parameters.

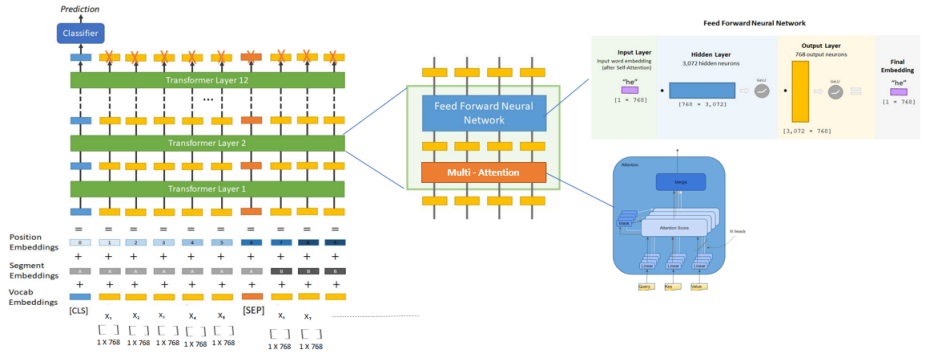
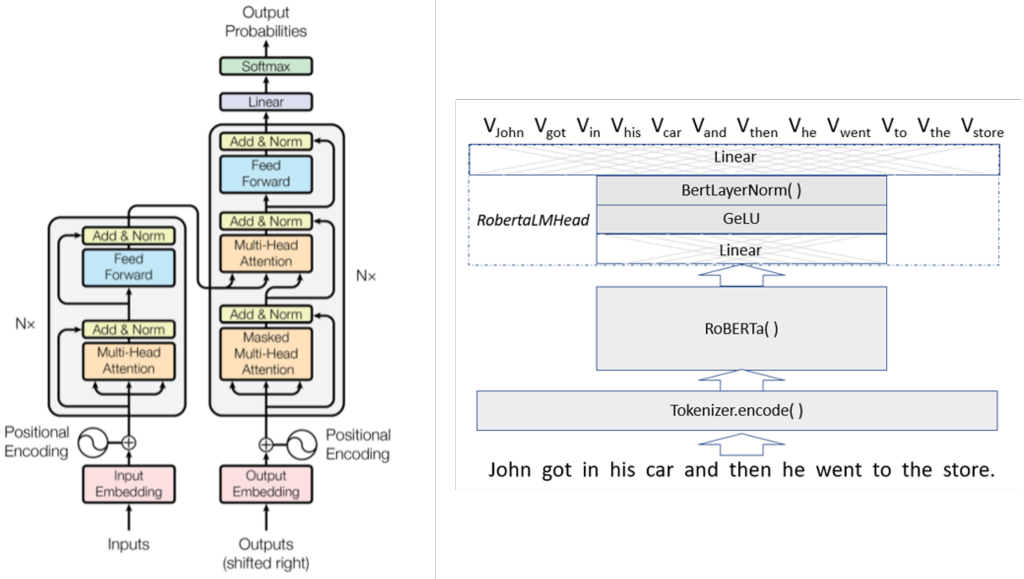


Figure 2: Pretrained Bert Transformer with Adam optimizer

4.4 Ternary Finetuned RoBERTa Transformer Model

The RoBERTa-Large model (12) is based on a Transformer architecture with 24 layers with hidden size of 1024, 16 Attention Heads with a Head size of 64. It is trained a large corpus comprising of BOOKCORPUS (13) and English WIKIPEDIA, with batch size of 8k, Dropout of 0.1, Adam optimization with no Gradient clipping. We fine-tuned this model further, using the 25K ternary labeled dataset described in section 5.1.



5 Experiments

5.1 Data

We used Hate Speech and Offensive Language dataset shared by Twitter as a part of the Kaggle Competition (11). This dataset consists of over 25K tweets with labeled ternary classification of Hate vs. Offensive vs. Neither. The collected data was in a raw format, which required several steps of curating and processing (1) Removal of usernames starting with @ (2) Removal of http links, hashtags, RT, special characters except apostrophe (’), emojis and short words with less than 3 characters (3) Stemming, which removes the suffix from a word and reduces it to its root word (4) Lemmatization, which considers the context and switches any kind of word to its base root mode.

5.2 Evaluation method

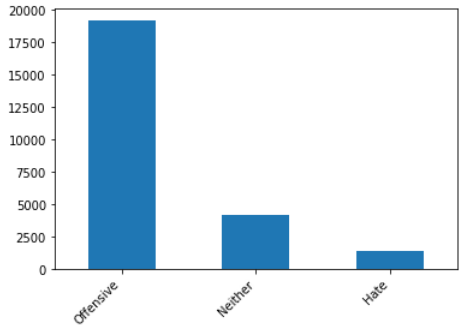


Figure 3: Distribution of data labels

We split the data into Train and Validation Sets in all scenarios. Each scenario was evaluated with a combination of metrics - Accuracy, Precision, Recall and F1 Score - to take a comprehensive view of the efficacy of algorithms. Given that the dataset was unbalanced with 77% of text corresponding to a single label, we also used Matthews Correlation Coefficient (MCC) to more accurately capture the performance of algorithms.

5.3 Experimental details

Baseline CNN Model: We used a Train/Val/Test split of 0.8, 0.1, 0.1 respectively. The batch size was set to 128, with a learning rate of 0.01 for RMSProp optimizer, 20% for Dropout when applicable and increased the number of epochs from 10 to 100. For data embedding, we used Glove embedding (9), max sequence length of 100, max vocab size of 20,000 and tested embedding dimensions of 100 and 200.

Bidirectional LSTM model with attention layer: We used a Train/Test split of 0.8, 0.2 respectively. The model was trained using categorical cross-entropy as the loss function and Adam as the optimizer. The F1 score was used as an evaluation metric, with a weighted average calculated for the three classes. The batch size was set to 128 and dropout rate to 0.2. For data embedding, we used Glove embedding (9), max sequence length of 50, and max vocab size of 50,000.

Pretrained Bert with Adam Optimizer: We used Pre-trained Bert and utilized the hugging face Bert sequence classification library. This comes with the predefined architecture of the Bert model that has 12 Bert layers and a single classification layer on top of that. There are 109m parameters in the Bert model. The weights of those 12 layers are trained but the weights for the classification layer are randomly initialized. In the first layer, the embedding size is 768 and the max length has 512 tokens. We used Adam optimizer with a weight decay fix, which made updates to the weights in the training loop. We ran the model for ten epochs, with a batch size of 32, and applied the early stopping so that model does not overfit. We also implemented a scheduler to take advantage of learning rate decay (making sure that model takes larger steps initially and smaller steps later).

Ternary Finetuned RoBERTa Transformer Model: We used a Train/Test split of 0.9, 0.1 respectively as we fine-tuned a pretrained RoBERTa model over 22,272 examples. The batch size was set to 32, with a learning rate of 2e-5 on 355M trainable parameters and we trained the model over 3 epochs with early stopping to ensure that the model does not overfit.

5.4 Results

The results are depicted in the Tables below.

Table 1: Training & Validation Set Results - Baseline CNN

Parameters	Train/Val	Accuracy	F1 Score	Precision	Recall
EmbedSize=100, Dropout, Adam	Train	0.8400	0.8390	0.8505	0.8281
	Val	0.7550	0.7490	0.7597	0.7394
EmbedSize=100, Dropout, RMSProp	Train	0.7803	0.7798	0.7814	0.7782
	Val	0.7667	0.7649	0.7664	0.7634

Table 2: Classification Report - BiDirectional LSTM With Attention Layer

Metric	Train/Val	Precision	Recall	F1-Score	Accuracy	MCC
Offensive	Train	0.96	0.97	0.97		
	Val	0.93	0.95	0.94		
Hate	Train	0.67	0.53	0.59		
	Val	0.47	0.34	0.39		
Neither	Train	0.94	0.96	0.95		
	Val	0.84	0.86	0.85		
Macro Average	Train	0.86	0.82	0.84		
	Val	0.75	0.72	0.73		
Weighted Average	Train	0.94	0.94	0.94		
	Val	0.89	0.90	0.89		
Consolidated	Train				0.94	0.84
	Val				0.90	0.72

Table 3: Classification Report - Pretrained Bert with Adam Optimizer Transformer

Metric	Train/Val	Precision	Recall	F1-Score	Accuracy	MCC
Offsensive	Train	0.93	0.96	0.95		
	Val	0.95	0.96	0.96		
Hate	Train	0.56	0.25	0.35		
	Val	0.48	0.36	0.41		
Neither	Train	0.86	0.91	0.89		
	Val	0.86	0.88	0.87		
Macro Average	Train	0.79	0.71	0.73		
	Val	0.76	0.73	0.75		
Weighted Average	Train	0.90	0.91	0.90		
	Val	0.91	0.92	0.92		
Consolidated	Train				0.91	0.75
	Val				0.92	0.75

Table 4: Classification Report - Ternary Finetuned RoBERTa Transformer

Metric	Train/Val	Precision	Recall	F1-Score	Accuracy	MCC
Offsensive	Train	0.96	0.93	0.95		
	Val	0.96	0.93	0.94		
Hate	Train	0.47	0.52	0.50		
	Val	0.43	0.41	0.42		
Neither	Train	0.87	0.96	0.91		
	Val	0.83	0.96	0.89		
Macro Average	Train	0.77	0.80	0.78		
	Val	0.74	0.77	0.75		
Weighted Average	Train	0.92	0.91	0.92		
	Val	0.91	0.90	0.90		
Consolidated	Train				0.91	0.78
	Val				0.90	0.76

6 Analysis

Baseline CNN Model: We achieved the highest validation accuracy when RMSProp and Dropout were used. Given that the train and validation accuracy is close, we believe the model is not overfitting. However, the baseline train accuracy was low, which we improved with better models.

BiLSTM with Attention layer: The BiLSTM model with attention layer showed good performance on "Offensive" and "Neither" classes. However, the performance needs to be improved for the "Hate" class. Overall, the macro average F1-score, which is the average F1-score across all classes, is lower on the validation set compared to the training set. The model's Matthew's correlation coefficient, which accounts for class imbalance, is 0.84 for the training and 0.72 for the validation sets.

Overall, this suggests that the model may be overfitting on the training set and may need further finetuning and regularization techniques to improve its generalization performance.

Pretrained Bert Model with Adam Optimizer: The model performed well on the "Offensive" and "Neither" classes with high precision, recall, and F1-score for both training and validation sets. However, the model struggled with the "Hate" class, as shown by the low precision, recall, and F1-scores. The F1-score for the "Hate" class is low, with 0.35 on the training set and 0.41 on the validation set, indicating that the model is not performing well for the "Hate" class. The model's Matthew's correlation coefficient, which accounts for class imbalance, is 0.75 for both the training and validation sets. This indicates that the model is not overfitting.

In conclusion, the model performed well on the "Offensive" and "Neither" classes but needs further improvement to better classify instances of the "Hate" class. It may require additional fine-tuning, a larger dataset, or a more sophisticated architecture to improve its performance.

Finetuned RoBERTa Transformer Model: On the validation set, the model performed well on the "Offensive" and "Neither" classes with high F1-scores of 0.94 and 0.89 respectively, but the F1-score was much lower at 0.42 for "Hate" class. Meanwhile, the MCC score on this imbalanced dataset is 0.76, which we consider as acceptable performance. All the corresponding values on the training set were very similar, so we believe the model is not overfitting.

To further improve performance, we need to explore a few options (1) Get more data for "Hate" class (2) Increase the number of parameters to increase model complexity (3) Balance the dataset prior to finetuning.

7 Conclusion

The project involved building and evaluating three different models (beyond a baseline CNN model) for multi-class text classification. The Ternary Finetuned RoBERTa Transformer outperformed the other models, achieving higher scores in all metrics except for hate speech. Throughout the project, we used various NLP techniques and models, including experimenting with BiLSTM architecture,

and Pretrained Bert, and finetuned RoBERTa models, to enhance performance and gain in-depth knowledge of Transformer models and their applicability to text classification tasks.

Our accomplishments include constructing three different NLP models that can accurately categorize tweets as offensive, hate speech, or neither. We also assessed the models using several metrics (Precision, Recall, F1-score, MCC) to gain a comprehensive understanding of their performance. All three models yielded comparable results on the validation set with no sign of overfitting, while the Finetuned RoBERTa Transformer achieved the best performance.

That being said, we believe the performance needs to be better as we ran into a few limitations. The dataset used was small and unbalanced (few "Hate" labels), and may not be representative of all text classification tasks. We also believe BERT might be pretrained over a large dataset that is biased towards certain demographics or viewpoints, thereby limited by the language and expressions it has been trained on, and may struggle with detecting new or evolving forms of hate speech.

Future work could entail (1) Training over much larger, diverse and balanced datasets (2) Experimenting with model complexity and architecture further (3) Tuning the hyperparameters.

References

[1] Raghav Budhraja and Bhanu Agarwal. 2020. Hate speech detection with bert using pytorch lightning. *arXiv preprint arXiv:2010.12688*.

[2] A. Rappaport D. Davidov, O. Tsur. pp. 107-116, July 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proc. 14th Conference on Computational Natural Language Learning*.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

[4] T. Ohtsuki H. Watanabe, M. Bouazizi. Volume 6, February 2018. Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection. In *IEEE Access*.

[5] Hamish. January 2019. Bidirectional lstm in keras with glove embeddings.

[6] 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. *arXiv preprint arXiv:1907.11692*.

[7] T. Ohtsuki M. Bouazizi. pp. 1-6, December 2016. Sentiment analysis in twitter: From classification to quantification of sentiments within tweets. In *Proceedings of IEEE GLOBECOM*.

[8] T. Ohtsuki M. Bouazizi. Volume 4, pp. 5477-5488, 2016. A pattern-based approach for sarcasm detection on twitter. In *IEEE Access*.

[9] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

[10] Mehreen Saeed. September 2022. Adding a custom attention layer to a recurrent neural network in keras.

[11] Andrii Samoshyn. June 17 2020. Hate speech and offensive language dataset.

[12] Naman Goyal Jingfei Du Mandar Joshi Danqi Chen Omer Levy Mike Lewis Luke Zettlemeyer Veselin Stoyanov Yinhan Liu, Myle Ott. A robustly optimized bert pretraining approach. In *arXiv:1907.11692*.

[13] Richard Zemel Ruslan Salakhutdinov Raquel Urtasun Antonio Torralba Sanya Fidler Yukun Zhu, Ryan Kiros. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books.