CS230

Investigating the Detection of Abusive Language and Hate Speech on Twitter with Deep Learning

Emiliano L. Rodrigues
Department of Computer Science
Stanford University
erodrig@stanford.edu
Mentor: Pedro Garzon

Abstract

This project aims to explore deep learning methods to predict abusive language and hate speech detected in social media, particularly Twitter. I implemented a series of deep neural networks, including a Recurrent Neural Network and a Convolutional Neural Network in order to analyze the text content of the tweets and then investigated what classes of words and types of sentences were most indicative for the models to classify as either abusive language or hate speech, or neither of these.

1 Introduction

Sentiment Analysis is a field that has evolved a lot and gained a lot of momentum over the past years with more processing power and newer techniques to analyze text and keep track of the contribution of meaning from each word to the sentence as a whole. In this project, I explore the use of these techniques and apply it to a field that has gained a lot of attention lately: hate speech on Twitter.

After preprocessing tweets, which entails cleaning them from unhelpful sentence markers, user handles and other Twitter-specific key words, like RT (Re-tweet), I move on to build models that are able to classify the tweets as containing either hate speech, abusive language or neither of these. Then, I explore what these Neural Networks are doing behind the scenes: Why does the model classify a specific sentence as hate speech?

2 Related work

Classifying tweets as hate speech is a known problem within NLP and there is a number of researchers that have looked into this. There are studies [7] that have used N-grams and TFIDF based approaches to classifying the same three groups of tweets for this project, applying Naive Bayes, SVM and Logistic Regression models and reaching a maximum validation accuracy of about 92%, while other studies [2] have used Deep Learning to tackle a simpler version of this problem where there are only two classes of tweets (hate speech and not hate speech), having reached a validation accuracy of 93% using a RNN with an LSTM Layer, Random Embedding and GBDT.

There are other studies [16] that have also attempted both a binary and a ternary classification approach and have reached 87.4% and 78.4% accuracy respectively, experimenting with different groups of linguistic features and using Random Forest and SVM models. Other authors have looked into more specific issues within the Twitter hate speech umbrella: This study [11] looked into

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

detecting hate speech against black people, and this study [15] performed a linguistic analysis on the type of discourse used in hate speech tweets.

Overall, very little formal research has been done on this issue with the use of Deep Learning techniques, which is why I believe my exploration of the topic is using Neural Networks is very worthwhile.

3 Dataset and Features

The dataset I used consisted of 24,783 tweets. Each tweet was reviewed by three different people, each of whom classified them as containing hate speech, containing abusive language, or neither. The most voted category for each tweet was the classification given to that specific tweet. It is important to note that given the tweets were classified by different people, what these individuals consider as hate speech, abusive language and neither is subject to their own interpretations of these terms.

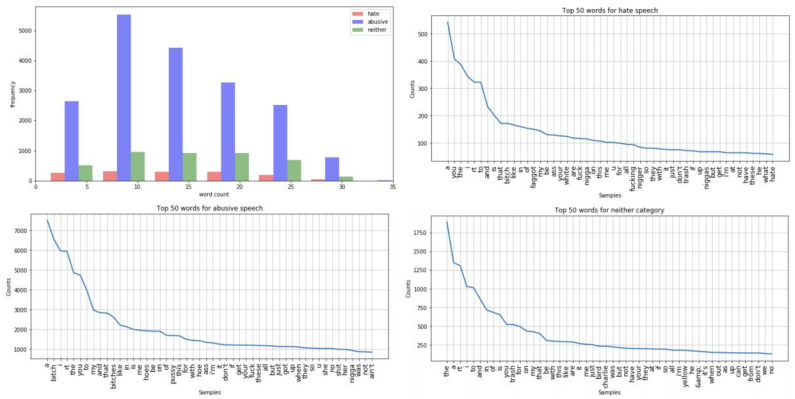
From this raw dataset, we had the following percentage distribution of tweets for each of the aforementioned classes:

Type	% of total
Hate Speech	5.41%
Abusive Lang.	73.42%
Neither	21.16%

I extracted the relevant columns from the CSV file using Pandas, and proceeded to perform basic preprocessing that involved removing links and user handles (@person). Then, I used Keras's Tokenizer class in order to remove all punctuation, numbers and make all the text samples lowercase.

From that, I broke apart each sentence by the words they were composed of, encoding each word using Keras Tokenizer's $texts\_to\_sequences$. Each text extract was then padded at a maximum of 25 words, which is the average word count for a tweet containing 140 characters.

3.1 Data Analysis



4 Methods

In this project, I implemented three different Neural Network structures in order to see how they compare and which one would be most appropriate for this specific classification task. All models were trained with 300d GloVe embeddings. The first model was a Recurrent Neural Network, where I used a Bidirectional LSTM Layer. The second one consisted of a Convolutional Neural Network, where I used a Convolutional 1-dimensional layer, as well as a Max Pooling layer. Finally, I implemented a model mixing the Convolutional Layer with the Bidirectional LSTM Layer.

When applied to NLP problems, the Recurrent Neural Networks, particularly those with the Long Short-Term Memory (LSTM) layer, keep track of important information from earlier parts of the sentence to create a better understanding of the semantics of the sentence and better predict an output.

The additional Bidirectional layer that wraps the LSTM layer makes this awareness of longer distance relationships between words a two way street, where words at the beginning are weighted according to those at the end, and vice-versa, according to relevance.

In NLP, the Convolutional layer for the Convolutional Network, along with the Max Pooling layer, focuses the analysis on particular parts (words) of the sentence that contribute more significantly to the classification process at hand.

The loss function of choice for all the models was the Softmax function, using Keras' Sparse Categorical Cross-Entropy built-in loss, that essentially applies log loss with respect to each of the categories to predict which one the sample is most likely to belong to. Its formula for this study case is as follows:

$$-\frac{1}{3}\sum_{i=1}^{3}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

5 Experiments/Results/Discussion

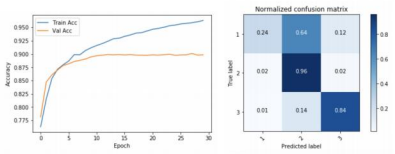
For reference, in the confusion matrices below, 1 stands for hate speech, 2 for abusive language, and 3 for neither.

5.1 Convolutional Neural Network

The CNN had the following layer set up: Trainable Embedding Layer $\rightarrow$ 1D Spatial Dropout (0.2 rate) $\rightarrow$ 1D Convolutional Layer (16 filters, kernel size 3, stride 1) $\rightarrow$ 1D Max Pool Layer (2x2 filters) $\rightarrow$ Dropout Layer (0.2) $\rightarrow$ Dense Softmax Layer (3 nodes). My chosen learning rate was 0.001 and the mini-batch size was 32. These hyper-parameters were the ones I got the best results with after trying different learning rates such as 0.01, 0.005, 0.0001, and different mini-batch sizes, including 20, 15, and 100.

I trained this network for a different number of epochs and with different layer configurations in order to experiment with different dropout rates for the Spatial Dropout and Dropout layers and number of nodes for the Convolutional Layer, as well experimenting with the addition of a Dense ReLu layer with different node counts.

I created a testing set composed of a random sample of 20% of the dataset, and a validation set also composed of 20% of the remaining 80% of the initial dataset. The Validation and Training Accuracy for each of the epochs are as follows, as well as the confusion matrix:



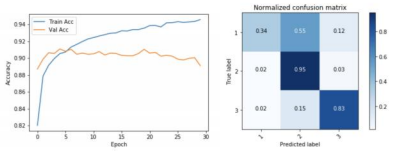
We can see that the model started overfitting approximately after the 5^{th} epoch, with training accuracy going up to more than 95% at 30 epochs, while validation accuracy became constant and peaked at 90.09% around the 28^{th} epoch.

When we look at the confusion matrix, we see that the model is very successful at classifying tweets marked as abusive language, with a 96% accuracy overall for this category, while it does poorly for tweets classified as hate speech, classifying only 24% of these correctly, and mistakenly labeling them as abusive language instead. This is likely due to the similarities pointed out earlier between abusive language and hate speech, which seem to have very similar vocabulary frequencies. When it comes to tweets classified as neither hate speech nor abusive language, the model also performs well and is able to classify 84% of these correctly.

5.2 Recurrent Neural Network

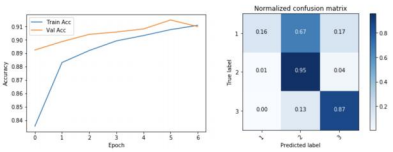
The RNN had the following layer set up: Non-trainable Embedding Layer $\rightarrow$ Bidirectional LSTM Layer (8 nodes total) $\rightarrow$ Dropout (0.2 rate) $\rightarrow$ Softmax Dense Layer (3 nodes). My chosen learning

rate was 0.001 and mini-batch size was 32. Unlike the CNN, I did not train the embedding weights, and for this network the test split and the validation split were both 10% of the dataset. The Validation and Training Accuracy, as well as the confusion matrix for this model were as follows:



We see that the model starter overfitting just like the CNN around the 5^{th} epoch, with training accuracy going up to 94.56% at 30 epochs, and with validation accuracy peaking at 91.08% and then decreasing and finishing at 89.11% in the last epoch. Looking at the confusion matrix, this model performed worse than the CNN for both the abusive language and the neither classes, but was better by 10% compared to the CNN in classifying tweets that were labeled as hate speech.

Because this model started overfitting very early, I also implemented a version of it using Early Stopping with a patience of 1, so training was automatically stopped after one epoch of no improvement in validation accuracy. The Validation and Training accuracies, as well as the confusion matrix for this same model with early stopping activated were the following:

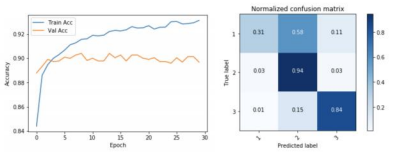


The RNN with Early Stopping performed better than the previous two models when classifying tweets as neither, and performed just as well as the non-early stopping RNN for abusive tweets, but performed much worse than the two previous models for tweets classified as hate speech, only classifying 16% of these correctly. The validation accuracy for this model was 90.99% at the end of the last training epoch. This is most likely due to the fact that the model was trained for only 7 epochs, compared to 30 for the previous two, so the backpropagation effect wasn't enough yet to improve the accuracy across the three categories. However, the advantage of this model is that it didn't overfit the training data.

5.3 Convolutional + Recurrent Neural Network

My last model implementation mixed a Convolutional Layer with a Bi-LSTM layer in order to see if the results would be any better than these two configurations separately. The layer set up for the network was: Non-trainable Embedding Layer $\rightarrow$ 1D Spatial Dropout Layer (0.2 rate) $\rightarrow$ Bidirectional LSTM Layer (8 nodes total) $\rightarrow$ 1D Convolutional Layer (16 filters, kernel size 3, stride 1) $\rightarrow$ 1D Global Max Pooling Layer $\rightarrow$ Dropout Layer (0.2 rate) $\rightarrow$ Dense Softmax Layer (3 nodes). The chosen learning rate was 0.001 and the mini-batch size was 32.

The Accuracy across epochs and confusion matrix follow:



This model performed better than the Convolutional Model, with a higher accuracy for tweets classified as hate speech (31% vs. 24%), but the same accuracy percentage in testing for the neither category (84%). Despite fairly good accuracy distributions for the three classes compared to the previous models, it didn't outperform the first RNN trained for 30 epochs, that had better accuracies for hate speech and abusive language labeled tweets. When it comes to accuracy for validation and training, the model overfit the training data, just as the previous ones, with training accuracy peaking

at 93.13% at the last epoch, while validation accuracy peaked in epoch 9 at 90.41%, decreasing to 89.69% at the final epoch.

5.4 Summary of Models

Model	Max Val Accuracy	Test Acc. Hate	Test Acc. Abusive	Test Acc. Neither
Conv1D (CNN)	90.09%	24%	96%	84%
Bi-LSTM (RNN)	91.08%	34%	95%	83%
Bi-LSTM (RNN) + ES	90.99%	16%	95%	87%
Bi-LSTM + Conv1D	90.41%	31%	94%	84%

5.5 Analysis of Predictions

It is interesting to look at the sentences that were miss-classified by the CNN model in the test set, so I went ahead and did that. It would be somewhat redundant to perform the same analysis for the other models, so I chose to focus on the CNN. For example, these are two samples that were classified as abusive language, but were labeled originally as hate speech: *"gucci mane in jail and dropping mixtapes every month and you hoes cant even text back"*, *"im feeling pretty fuckin ghetto smh"*. In these examples and personally, I believe these sentences would be better classified as abusive/offensive language as the model predicted, but that is not what they were labeled as. Even more interestingly, the following is a tweet labeled as hate speech that the model classified as neither: *"i hope my future wife who ever she might be is sleeping or studying instead if being a hoe right now nigga shutcho asss up"*. This tweet in particular (the last 4 words) was a reply to the previous sentence. Although its author was negating the previous user's hateful comment, his reply also contained hate speech, but somehow the model assumed this was not classified as hate speech nor abusive language. Perhaps the model was aware of this negating assumption, but it was still unable to catch the hate speech/abusive language in the reply.

6 Conclusion/Future Work

In sum, it was interesting to analyze the issue of hate speech on Twitter with this dataset, which didn't only contain two different labels where tweets were classified as either hate speech or not, but with a more nuanced distinction where abusive tweets were also a class. The initial data analysis was indicative of the difficulty we noticed in the models to distinguish between hate speech and abusive language, with all of the models miss-classifying a large proportion of hate speech labeled tweets as abusive language.

Training the networks for 30 epochs, although leading to overfit, seemed to be good because all the models were more able to more correctly classify hate speech tweets, whereas when I used early stopping for the second model, the accuracy for classifications of the hate speech class was almost half of that for the models that were trained for more iterations.

The model that best predicted across the 3 categories was the RNN trained for 30 epochs, and that is given to its simplicity in terms of the construction of the network, united with the underlying robustness of the Bidirectional LSTM Layer.

For future work, it would be interesting to increase the dropout rate in the dropout layers for all models in order to reduce overfitting, as well as change the stride for the convolutional layers, and train for even more iterations to see how well the model is able to learn. It would also be interesting to search for more tweets that are classified as hate speech in order to even out the number of tweets belonging to each of the three classes.

7 Contributions

This was a one-person project, and as such I worked on all stages of the investigation. Here's the link to the GitHub repository with the code used to train the models: `https://github.com/emilianolr/cs230-project/`

References

[1] Abadi, Agarwal et al. "Tensorflow". 2015. https://tensorflow.org

[2] Badjatiya, Pinkesh, et al. “Deep Learning for Hate Speech Detection in Tweets.” *ArXiv.org*, American Physical Society, 1 June 2017, arxiv.org/abs/1706.00188.

[3] Brownlee, Jason. “How to Use Word Embedding Layers for Deep Learning with Keras.” *Machine Learning Mastery*, 30 May 2018, machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/.

[4] Brucher,Perrot, et al. "Scikit-learn: Machine Learning in Python".*Journal of Machine Learning Research*. 12, 2825-2830. 2011.

[5] Chollet, François, et al. "Keras". 2015. https://keras.io

[6] Davidson, Thomas. “t-Davidson/Hate-Speech-and-Offensive-Language.” *GitHub*, 14 Mar. 2018, github.com/t-davidson/hate-speech-and-offensive-language.

[7] Gaydhani, Aditya, et al. “Detecting Hate Speech and Offensive Language on Twitter Using Machine Learning: An N-Gram and TFIDF Based Approach.” *ArXiv.org*, American Physical Society, 23 Sept. 2018, arxiv.org/abs/1809.08651.

[8] Hunter, John D. "Matplotlib: A 2D Graphics Environment". *Computing in Science & Engineering*. 9, 90-95. 2007. DOI:10.1109/MCSE.2007.55

[9] McKinney, Wes. "Data Structures for Statistical Computing in Python". *Proceedings of the 9th Python in Science Conference*, 51-56. 2010.

[10] Joshi, Prateek. “Comprehensive Hands on Guide to Twitter Sentiment Analysis with Dataset & Code.” *Analytics Vidhya*, 12 Dec. 2018, www.analyticsvidhya.com/blog/2018/07/hands-on-sentiment-analysis-dataset-python/.

[11] Kwok, Irene, and Yuzhou Wang. "Locate the Hate: Detecting Tweets against Blacks." *AAAI*. 2013.

[12] Oliphant, Travis E. "A guide to NumPy". *Trelgol Publishing*. 2006. USA.

[13] Pérez, Granger. "IPython: A System for Interactive Scientific Computing", *Computing in Science & Engineering*, 9, 21-29. 2007. DOI:10.1109/MCSE.2007.53

[14] Wahome, Ronald. “This Is How Twitter Sees The World : Sentiment Analysis Part One.” *Towards Data Science*, Towards Data Science, 7 Sept. 2018, towardsdatascience.com/the-real-world-as-seen-on-twitter-sentiment-analysis-part-one-5ac2d06b63fb.

[15] Waseem, Zeerak, and Dirk Hovy. "Hateful symbols or hateful people? predictive features for hate speech detection on twitter." *Proceedings of the NAACL student research workshop*. 2016.

[16] Watanabe, Hajime, et al. “Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection.” *IEEE Access*, vol. 6, 15 Feb. 2018, pp. 13825–13835., doi:10.1109/access.2018.2806394.