Hate Speech Detection in Twitter Data

Navami Jain
Stanford University

Aarushi Patil
Stanford University

Gautam Pradeep
Stanford University

Abstract - With the arrival of new social media platforms comes a responsibility to curb spread of hateful messages. Twitter is at a heightened risk for its unique capability to reach target audiences and key influencers with tweets. This paper aims to experiment different techniques to detect hate speech from Twitter data. We created multiple approaches to compare our baseline, an n-gram sentiment classification algorithm, a linear and non-linear SVM, an LSTM, and a CNN model. Our final results demonstrated the optimal performance from the SVM with a 95.8% accuracy rate, followed by the LSTM with 93.3% accuracy, then our CNN with 90.3% accuracy, and finally our baseline 10-gram with 81.8% accuracy.

***Index Terms*—Support Vector Machine (SVM), Convolutional Neural Network (CNN), Long Short-term Memory (LSTM), Hate Speech, Sentiment Analysis**

I. Introduction

Hate speech in social media is a very dire and prominent issue we face heightened by the globally-connected communication platforms at our fingertips. While companies such as Facebook have piloted algorithms to detect hateful language, they are still far from perfect. As discussed in a recent Science News article, algorithms that are oversensitive to group identifiers such as "black", "gay"', or "transgender" can cause posts with these self-identifying terms from minority groups to be filtered/flagged when they aren't problematic, due to insufficient context. These algorithms may unintentionally prejudice against social media posts from minority groups. This motivates our work to develop algorithms that are context-dependent and minimize their false-positive rate. Researchers find that machine learning models still struggle to detect hate speech, clearly representing how difficult of a task this might be [8]. Some issues are delineated in [8], such as, "[the models] appear[ing] to not sufficiently register linguistic signals that reframe hateful phrases into clearly non-hateful ones (e.g. 'No Muslim deserves to die')." The purpose of our system is to identify hate speech from text, specifically twitter data. With such a system in place, we can automatically detect and flag offensive material on social media and alert followers or friends.

II. Literature Review

A. SVMs and Naive Methods

Schmidt et al [1] summarize a variety of features used in state-of-the-art detection of hate speech. Fundamental features include a bag of words consisting of word and character n-grams [10]. Other features used in detecting abusive content include URL mentions, hashtags, punctuation, word and document lengths, and capitalization [4]. Word generalization utilizes low-dimensional, dense vectorial word representations that are learned through clustering, topic modeling, and word embeddings [9]. These word representations can then be used to build feature vectors. Some sources discuss the use of lexical resources to look up particular negative words in messages such as slurs and insults [6]. For all the methods of building classifiers, they are all predominantly supervised. The most common supervised algorithm in this space is Support Vector Machines [6]. The features implemented in this study include presence of one-grams through five-grams that are derogatory. This approach also considers parsing parts of speech to detect patterns in sentence structure, but warns that these relationships may not delineate well between offensive and non-offensive. While our approach is to parse every word in a tweet and assign appropriate weighting, this model parses for specific phrasing or words. In addition to implementing a SVM, the study also compared results from a Random Forest Decision Tree (RFDT) and Bayesian Logistic Regression

(BLR) which, in contrast to a spatial classification model, operate by learning likelihoods and identifying features that contribute most to classifications. The study compares and combines the results from each of the models.

B. Deep Learning Approaches

When considering deep learning methods, the focus is shifted from manual feature engineering to neural network architecture. The most common network architectures are Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long-Short Term Memory networks (LSTM). [2, 3, 5, 7, 11] utilized some deep learning techniques using either simple word-based or character-based one-hot encoding as input features into the model. The authors discussed the utility of LSTM structures to process inputs with arbitrary sequence lengths, and thus especially pertinent to speech detection in tweets. Vigna et. al. also utilized word polarity as an input feature. For deep learning methods, CNNs can effectively extract word or character combinations such as n-grams, whereas RNNs will learn word or character dependencies, or orderly information, in tweets [11].

III. Dataset

Our input will be twitter data, or specifically text data of tweets that can be classified as hate-speech, offensive language, or neither. These are hand-labeled as hate speech (0), offensive speech (1), or neither (2) by CrowdFlower users. These data are taken from this Kaggle dataset, with 24,783 unique tweets. 1,430 are classified as hate speech, 19,190 are classified as offensive, and 4,163 as neither. Our model will attempt to predict and classify a set of tweets into two of these categories, hate/offensive speech and non-offensive speech. We have pre-processed our data to label anything originally designated as hate speech or offensive as 1 and anything else as 0. We crafted text files to store the label followed by the tweet in each line, which is the form that was suitable for our baseline model.

A. Train/Validation/Testing Splits

We first split our data into training, validation, and test sets. We allocated 90% of the data as training, 5% as validation, and 5% as testing. This splitting designates 22304 tweets as training examples, 1240 as validation, and 1240 as testing.

B. Pre-processing

For each split, we parsed each line in our file to separate the label (1 or 0) from the actual tweet. For each tweet, we removed any punctuation, cut any non-alphabetic tokens, and filtered out any NLTK "stop" words. We developed a dictionary to keep track of the count of each unique word in the entire dataset and used this to filter out any words from a tweet that had an overall dataset count of less than 2. Finally, we had three separate lists of cleaned training, validation, and testing tweets and three lists of training, validation, and testing labels. We utilized a Keras Tokenizer and fit it on the cleaned, training tweets and encoded each of the three tweet lists into integer sequences using the Tokenizer, and then padded the sequences so each tweet sequence is of the same length. This pre-processing was necessary for the LSTM and CNN implementations.

IV. Baseline

As described above, we performed a simple sentiment classification based on counts of n-grams in a tweet. We used a function extractCharacterFeatures to parse out and count frequency of all 10-grams in a tweet. The value of the features were equal to their frequency in the tweet; a more frequent value is given a greater weighting. The baseline model learned over 1.7 million weights and obtained a training accuracy of 99.8% accuracy and a validation accuracy of 81.8% accuracy.

V. Evaluation Metric

Evaluation of hate speech detection was assessed by the accuracy of our models while training and fine-tuning during validation through built-in functions in the tensorflow library.

VI. Main Approach

A. Support Vector Machine

We implemented linear support vector machine (SVM) and non-linear SVM. We use a Tfidf vectorizer to extract the most relevant words in the tweets based on their term frequency (or TF) and inverse document frequency (IDF). Term frequency

measures the number of times a word appears in a document (or in this case, a tweet), while IDF is a measure of how frequently the term appears in the set of tweets. We see that a high weight is associated with high term frequency in a given tweet and low frequency of the term in the collection of tweets. Other parameters we had to tune included min_df, specifying when terms with a small document frequency should be ignored and likewise max_df when terms with too large of a frequency should be ignored. The inputs to the SVM are the extracted features from Tfidf vectorizers and the outputs are +1 and -1, corresponding to offensive and non-offensive language, respectively. With the SVM, we've used a Linear kernel and the Gaussian radial basis function to draw the hyperplane on the data. We then trained and fitted our data, looking at error, using the SVM and scikitlearn library. The Gaussian Radial Basis Kernel is listed as Equation (1).

$$K(x_1, x_2) = exponent(-\gamma||X_1 - X_2||^2 \quad (1)$$

B. Deep Learning

The two deep learning models that we used were an LSTM (Long Term Short Memory Recurrent Neural Network) and a CNN (Convolutional Neural Network). The data was cleaned and preprocessed in the same way for both of these models. Specifically, our implementation used vectors that indexed words into the vocabulary created during the preprocessing. This vocabulary was essentially a dictionary of word counts. The tensorflow and keras libraries were used to help us implement these. The loss function used for both the LSTM and CNN was the binary cross-entropy function.

Listed here is the general form of the cross entropy loss function:

$$CE = -\sum_{i}^{C} t_i \log(s_i) \quad (2)$$

We used the binary cross-entropy loss for our CNN:

$$CE = -\sum_{i=1}^{C'=2} t_i \log(s_i) = -t_1 \log(s_1) - (1-t_1) \log(1-s_1 \quad (3)$$

where $s_2 = 1 - s1$ and $t_2 = 1 - t_1$. t and s are the groundtruth for their respective classes C_1 and C_2.

C. LSTM

The activation function used in the LSTM was the sigmoid function. For the optimization function, we used adam. We also experimented with other activation functions like RMSProp, and adam resulted in the best results.

D. CNN

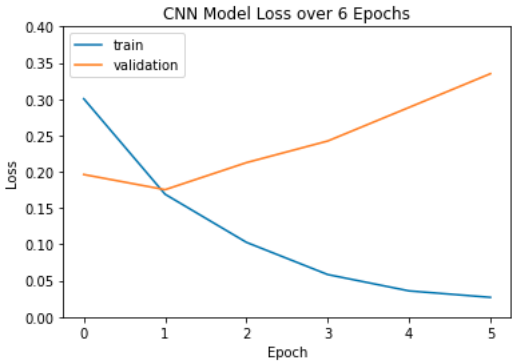
The CNN we used was a simple 1D CNN with one convolutional layer and one pooling layer. The activation functions used in the CNN were ReLu and sigmoid. We also considered using tanh activation function, but this did not produce as good results. For the optimization function, we used adam. We also experimented with other activation functions like RMSProp, and adam resulted in the best results.

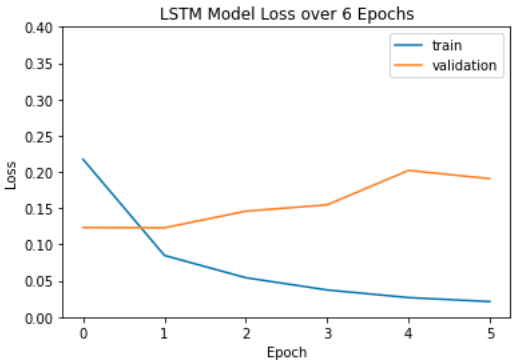
VII. Results

A. Summary Statistics

Model	Baseline	SVM (Linear)	SVM (Non-linear)
Training Accuracy	0.998	0.968	0.989
Validation Accuracy	0.818	0.958	0.952
Test Accuracy	0.791	**0.952**	0.948

Model	LSTM	CNN
Training Accuracy	0.993	0.989
Validation Accuracy	0.945	0.920
Test Accuracy	**0.936**	0.902





VIII. Error Analysis

A. Baseline

For our baseline model, we chose to count the frequency of all 10-grams as features for each tweet. While 7-gram or 8-gram frequencies may have produced improved accuracy, we observed a tradeoff between accuracy and training time. Choosing 10-gram features proved to be fast while also demonstrating proof of concept. We report a train error of 0.001345 and a validation error of 0.1816.

B. SVM

For our linear kernel support vector machine, the model had an accuracy of 94.0% on the validation set, thus demonstrating a smaller error compared to our baseline. We initialized our parameters to be $min_df = 200$ and $max_df = 0.8$. We first observed that decreasing the min_df seemed to decrease accuracy.

Decreasing to 50 led to a longer training time but greater accuracy –
Training time: 10.449;
Prediction time: 0.415s
Accuracy: 0.9572

Decreasing to 10:
Training time: 13.025s;
Prediction time: 0.541s
Accuracy: 0.9588

Increasing the max threshold from 0.8 to 0.95 or 1.0 didn't change the accuracy. Max parameter generally didn't have much of an effect.

After training the linear kernel, we observed several misclassified results, some featured here:

1) "Weekend is here. What an amazing week this has been. Let's use this extended weekend to celebrate our successes my fellow queer folk." [-1] 1
2) "Welding with your shirt off is the redneck version of tanning lol" [1] -1
3) "What's worse than Ebola ? The F-ing retards trying to make it a race issue. Al Sharpton has caused more American death than ebola." [1] -1 "When this test is over today I'm making a pan of brownies. I'm going to lay in bed all day amp; eat them until my heart is content. 128524;" [-1] 1

Where the first and fourth were misclassified as hateful speech while the second and third were misclassified as non-hateful. The misclassifications arise from several reasons. First, acronyms like "F-ing" were likely not weighted by the Tf-idf vectorizer. The abundance of such shortened versions of words may explain the growth in accuracy when min_df parameter was lowered. In addition, because an overwhelming majority of the tweets in the dataset were offensive or hateful, most tweets contained offensive words that ultimately didn't carry high weight following the Tf-idf algorithm. Instead, words such as 'this', 'start', 'woman', or specific hashtags used sparsely by users. This may explain why non-offensive tweets were repeatedly flagged as being hateful. When finally applied to the test dataset (1240 tweets), 60 of them were misclassified; 36 false positives and 24 false negatives.

The SVM using a Gaussian Radial Basis Function did not improve accuracy. In fact, many of the misclassified tweets from a linear kernel remained misclassified. On the test dataset, 125 were classified; 70 as false positives and 55 as false negatives.

C. LSTM and CNN

In general, the misclassified tweets in both of these were misclassified for similar reasons. In fact, 50 percent of the misclassified LSTM tweets were also misclassified by the CNN. The CNN in particular had trouble classifying tweets that used swear words/derogatory terms (depending on the identity of the person using them) that can be used positively or negatively based on the context. For example: The CNN misclassified the following tweets:

1) ”Why would date ugly girl also huge bitch” 1 0
2) ”c**n hair” 1 0
3) ”boy yung ni**a told dat” 0 1
   Note that the first number is the prediction and the second number is the actual

The first two are examples of the swear words/slurs being used in a hate speech context, while the third example shows a way that the slur is used in a non-hate speech context. The sociological phenomenon of groups reclaiming slurs and the fact that curse words have massive positive and negative utility is something that the model has difficulty with.

Another general issue for both the CNN and LSTM is that tweets that have identity words that are positive are misclassified as negative:

1) ”What black folk take kids riding ghetto point youth amp say” 0 1
   Note that the first number is the prediction and the second number is the actual

While the accuracy for these models is relatively high, the 5% misclassification rates disproportionately affect those of marginalized identities, either by allowing hate speech to stay online or by suppressing tweets that are actually positive but may use identity words.

IX. Future Work

As mentioned in the paper, the dataset we used contained an overwhelming proportion of tweets classified as offensive or hateful. Given the range of hateful tweets, it was often difficult to separate non-hateful tweets. If we were to repeat this analysis, we could find datasets with roughly equal numbers of hateful and non-hateful language. In addition, we noticed the sparse use of shortened versions of words (use of “F-ing”). As another preprocessing step, we could have created a dictionary of all words in the tweets as keys and their corresponding values as all possible shortened versions. For example, the values for key “phone” would include “phn”, “phon”, etc. We can thus infer the true meaning of words and clean up the tweet before running the model. While this would add to computational complexity, we observed several tweets contained such shortenings, so parsing their meaning may become significantly easier. We would like to work on methods that will reduce the incorrect classification of identity based tweets as hate speech - perhaps using a higher level intelligence model for these ambiguous tweets specifically and understanding what techniques work best for these in particular. Adding features about the user who posted that tweet might also be useful: for example, a Black person using a slur targeting Black people is different than a white person using that slur. This may allow for a better classification. Furthermore, greater performance in our LSTM and CNN would be possible with more optimization on the number of layers and hidden units utilized as well as the batch size. Overall, this task was performed at a high accuracy by all the models we experimented with in comparison to our 10-gram baseline. Hate speech detection is definitely a very complex task as hate speech itself varies in so many ways. This paper aimed to understand the fundamentals of how various techniques compare with each other in detecting general hate/offensive speech but much work is still to be done to implement these methods to reduce the spread of hate speech across social media.

X. Code

All of the code for this project along with the labeled data can be found **here.**

References

[1] Anna Schmidt and Michael Wiegand. A survey on hate speech detection using natural language processing. In International Workshop on Natural Language Processing for Social Media, pages 1–10. Association for Computational Linguistics, 2017. doi:10.18653/v1/W17-1101.

[2] Björn Gambäck and Utpal Kumar Sikdar. Using convolutional neural networks to classify hate speech. In Proceedings of the First Workshop on Abusive Language Online, pages 85–90. Association for Computational Linguistics, 2017. doi:10.18653/v1/W17-3013.

[3] Fabio Del Vigna, Andrea Cimino, Felice Dell’Orletta, Marinella Petrocchi, and Maurizio Tesconi. Hate me, hate me not: Hate speech detection on Facebook. In Proceedings of the First Italian Conference on Cybersecurity, pages 86–95. CEUR Workshop Proceedings, 2017.

[4] Irene Kwok and Yuzhou Wang. Locate the hate: Detecting tweets against blacks. In Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, AAAI'13, pages 1621–1622, Menlo Park, California, United States, 2013. Association for the Advancement of Artificial Intelligence.

[5] Jo Ho Park and Pascale Fung. One-step and two-step classification for abusive language detection on Twitter. In ALW1: 1st Workshop on Abusive Language Online, pages 41–45, Vancouver, Canada, 2017. Association for Computational Linguistics. doi:10.18653/v1/W17-3006.

[6] Pete Burnap and Matthew L. Williams. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. Policy and Internet, 7(2):223–242, 2015. doi:10.1002/poi3.85.

[7] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. Deep learning for hate speech detection in tweets. In Proceedings of the 26th International Conference on World Wide Web Companion, WWW '17 Companion, pages 759–760, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee. doi:10.1145/3041021.3054223.

[8] Wiggers, Kyle. "Researchers Find Machine Learning Models Still Struggle to Detect Hate Speech." VentureBeat, VentureBeat, 6 Jan. 2021, venturebeat.com/2021/01/06/researchers-find-machine-learning-models-still-struggle-to-detect-hate-speech/.

[9] William Warner and Julia Hirschberg. Detecting hate speech on the World Wide Web. In Proceedings of the Second Workshop on Language in Social Media, LSM '12, pages 19–26. Association for Computational Linguistics, 2012.

[10] Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. Detecting offensive language in social media to protect adolescent online safety. In Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust, SOCIALCOM-PASSAT '12, pages 71–80, Washington, DC, USA, 2012. IEEE Computer Society. doi:10.1109/SocialComPASSAT.2012.55.

[11] Ziqi Zhang, David Robinson, and John Tepper. Detecting hate speech on Twitter using a convolution-GRU based deep neural network. In Proceedings of the 15th Extended Semantic Web Conference, ESWC'18, pages 745–760, Berline, Germany, 2018. Springer. doi:10.1007/978-3-319-93417-4$_4$8.