Abstract

In an effort to combat hateful, offensive speech prevalent throughout several social media platforms, we implemented a Long Short Term Memory model (LSTM) to identify and classify racist tweets. The main advantage of an LSTM is that it prevents the vanishing gradient problem present in other recurrent neural networks, as it discards unimportant inputs. Built using a Keras library with Tensorflow backend, the model consisted of four layers: embedded, LSTM, dropout and activation layer using a sigmoid function. It was trained on two datasets with an F1 scoring used to analyze the accuracy.

1. Introduction

In recent years, America has seen an alarming rise in the number of hate crimes perpetrated against minorities per day, since the election of a president that encourages hate speech. (Levin and Reitzel 2018). Although hate speech is simply speech, there is a direct link between speech inciting violence against minorities and actual violence against minorities. For example, preceding the tragic shooting of a Pittsburgh synagogue a few weeks ago, the murderer actively posted anti-Semitic comments on social media, and even announced his intention hours before the crime (New York Times).

Hate speech is undeniably harmful, yet social media platforms are slow to act against it due to the unscalable and subjective nature of internet moderation - rst, a user must post content for review; then, a human moderator must view and judge it; and finally, someone must act upon it. By producing a model for automatic detection and classification of abusive language, we hope to expedite the process of addressing hate speech and eventually reduce the amount of hate speech that leads to hate crimes.

2. Related Work

Most work in the classification of hate speech in Tweets stems from Waseem’s dataset of tweets tagged as racist, sexist, or neither. Waseem himself used this dataset to train a logistic regression classifier and found that using character n-grams of length 4 provides better results compared to word n-grams, and that gender as a feature provides better results than length, location, or a combination of the three, with an F1 score of 73.93%, comparing four versions of the Convolutional Neural Network (CNN) classifier to a logistic regression one. The first used random word vectors as feature embeddings, the second used word2vec word vectors, the third used character n-grams, and the fourth one combined character n-grams and word2vec word vectors.

Gambck and Sikdar found that each CNN model outperformed Waseem and Hovy’s logistic regression model; within the CNN models, the second one built off of only word2vec word vectors provided the overall best result, with an F1 score of 78.29% (Gambck and Sikdar, 2017).

Davidson et al. propose a one versus-rest multiclass approach to classification, differentiating between hate speech, offensive language, and neither (2017). They define hate speech as language that is used to expresses hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group, whereas offensive language includes potentially inflammatory terms used in a non-hateful manner. For example, hate speech would include a white person using the n-word, whereas offensive language would include the world f**k.

Davidson et al. do not make use of Waseem’s dataset, instead curating their own database of Tweets containing possible hate speech and annotated by a group of amateur annotators. After trying out various models, such as naive Bayes, decision trees, and random forests, they obtained an F1 score of 90% using a logistic regression model with L2 regression. However, they admit that 40% of their hate speech is misclassified as offensive language (Davidson et
Park and Fung suggest a two-step classification of abusive language and, like most of their predecessors, make use of Waseems dataset (2017). The first step detects the abusive language, then passes on that information to the second step, which classifies it into a specific type of hate speech. Influenced by previous CNN models such as the CharCNN and WordCNN, general purpose CNNs that take in characters and words as input, respectively, Park and Fung designed a CNN classifier called the HybridCNN, which takes in both characters and words as input. They found that their HybridCNN model outperformed the CharCNN and WordCNN models; moreover, they posit that using HybridCNN for the first step of abusive language detection and logistic regression for the second step of racism vs. sexism classification performs the best (Park and Fung, 2017).

3. Dataset and Model

Two datasets were obtained for training and testing of our model. The first dataset, taken from Davidson et. al, contained over 20,000 tweets and used crowd-sourcing to assign each tweet one of three labels: offensive language, hate speech, neither. We will refer to this as the amateur labeled dataset. The second dataset, which will be referred to as the expert labeled dataset, was an open-source dataset of approximately 17,000 tweets each represented by a unique ID, collected by Zeerak Waseem and Dirk Hovy over 2 months and annotated by their team. They labeled tweets as either racist, sexist or neither. We focused on two labels: racist or not racist. While the database was large, only 1,972 tweets were labeled as racist, so our dataset was actually quite small. To remedy this unbalanced dataset, tweets labeled as racist were duplicated and spread throughout in a new dataset entitled Modified Expert Labeled Tweets.

We utilized the Keras library on top of Tensorow to implement a Long-Short Term Memory Model to accomplish this task. Combining Keras with Tensorow allowed for more control over the model. The primary reason for choosing to use an LSTM network was that it solved the problem of the vanishing gradient that other recurrent neural networks experienced. Rather than layers, an LSTM contains building blocks, and each block contains three gates: input, forget, output. Thus, based on the weights and computation of the inputs, the network understands which inputs are important and which can be discarded. Our model specifically contained four layers: embedding layer, LSTM layer, dropout layer, activation layer using sigmoid function. The embedding layer mapped the words of each tweet to a vector and the dropout layer was included to reduce training time and prevent overfitting.

4. Method

A Twitter API was used to retrieve the tweets from the datasets, which included a unique ID for each tweet and its corresponding label. 64% of the dataset was used for training and 20% was used for testing. Following that, one hot encoding and padding was applied to each tweet. Because our dataset was imbalanced, we decided to duplicate rare data points, in order to improve the models learning. We then varied how our dataset was preprocessed and embedded. Finally, F1 scoring was used to analyze the accuracy of the model. The link to the Github repository can be found at the end of this paper.

5. Results

The ratio of racism to non-racism was originally 1:40, so we duplicated racist tweets in the training dataset to produce a 1:1 ratio of racism to non-racism and solve the imbalanced dataset problem. Before balancing our datasets, we had misleadingly high accuracies of 0.9833 and 0.7763 for the Waseem expert labeled and Davidson amateur labeled datasets, respectively. This was because the data was massively biased towards the non-racist tweets, causing our model to label the entire test dataset as non-racist. Duplication of tweets classified as racist fixed the data imbalance, but decreased our accuracies to around 0.53 for each dataset until we fixed our problem with the max length of tweets.

We originally set our max_length to 140 (which padded each tweet so that they would be the same length of 140), because the max length of a tweet is 140 characters; thus, in the worst case scenario, every character in the tweet could be a word. However, we found that this caused the model to label every tweet with the same value. Thus, we reduced our max_length to 30, in order to match the true length of the longest tweet in our dataset. This improved our F1 score dramatically, as seen in Fig. 1 and Fig. 2.
Changing how the tweets were represented from one-hot encoding to GloVe improved our F1 score by 1 point on the Davidson dataset. For the Waseem dataset, the overall F1 score remained the same; however, classification of racist tweets improved by 9 points.

After optimizing the LSTM model, we compared our results to that of the original results on the same datasets and found that LSTM improved the F1 score by 10-20 points. We produced an F1 score of 99, compared to an F1 score of 73.89 using a logistic regression on the same Waseem dataset.1 We also produced an F1 score of 93, compared to an F1 score of 90 using logistic regression with L2 regularization.2

6. Conclusion

Using an LSTM model to classify tweets for racism significantly improved the F1 score, compared to results by the original authors on the same dataset. Thus, it appears that using an LSTM model for classification of tweets may perform better than classification using linear or logistic regression models. It is also clear that the way the data is preprocessed and embedded has an impact on results. For future steps, an LSTM layer used in tandem with other RNN layers, or a CNN layer, may improve performance.

7. References


Github Repository
https://github.com/yrachel/racismclassifier.git