Toxic Comment Classification

Marcel Colvin 912033961

https://github.com/MarcelColvin/Toxic_comments

Executive Summary:

For the final project of Sta141c, I decided to try and solve a recent kaggle competition: Toxic Comment Classification. The goal of this competition is to not only classify which comments are toxic, but also other categories of toxicity such as “obscene”, “threat”, “insult”, and “identity hate”. This means that most toxic comments will need to be classified into one or more categories. Using the training data provided, I was able to create a model that classified correctly with 96% accuracy. Then I decided to apply my data set to classify New York Times comments and validate the success based on sentiment analysis.

Data:

The training data given for the Toxic Comments competition consists of 159,571 rows of unique wikipedia comments and 6 binary classes for each level of toxicity. The supplied data also included a test data set, with 153,164 rows of just comments to test the models on a new dataset. The New York Times comments data set has 2,176,364 rows of comments from 2017 and 2018 which I also obtained through Kaggle in their public data sets.

Methods:

Word2Vec

For my first method in toxic comment classification, I decided to turn each comment in the training and test data sets into vectors using google’s word2vec algorithm using the gensim package. First I had to tokenize my data set and remove stopwords. In order to turn each word into a vector and take the average vector per sentence, I used google’s pretrained model to change each word into a vector. I used google’s pretrained model because this will give my vectors a more accurate distance since it is based on a larger corpus of documents. Then taking the average vector of each comment will theoretically group comments that use more toxic words, since toxic words will be closer to each other.

Next I decided to use Sklearn's Multi-Level Perceptron Classifier (MLP) to fit the vectors. I chose this model because this neural network because it is not sensitive to linearity and the word vectors are scaled. The neural network will be able to distinguish which values in each toxic vector correspond to toxicity and be able to properly weight them. Then when new data with the same attribute is entered it will be properly classified. I decided to create 6 models, one for each category of toxicity. This allows for there to be 6 binary classifiers, this makes it easier for multilabel classification. Although there is definitely correlation between classifying something as toxic and identity hate, I assume here that any comment that is an insult will also be classified as toxic. Therefore having six different binary models should classify just as well.

After running a partial fit on each model and running a prediction on the test data set, I submitted my final result to Kaggle to check how well my model did. This model, using word2vec and an MLP Classifier, had 93.35% accuracy on the test set. This value was not high on the Kaggle leaderboard.

Using the same word2vec data, I attempted to apply sklearn's logistic regression model. I thought that this model might do better because using a simpler method such as measuring success by the log of the odds ratio might increase accuracy. Although we are trying to classify by certain traits that have weights while using the MLP Classifier, doing a logistic regression will simplify the classification process. This model actually did score higher, with 95.09% accuracy. Therefore, using a logistic regression model, simply based on the odds of a comment being toxic, outperformed the MLP Classifier on the word2vec data.

TFIDF

In order to try to get better results, I decided to turn to creating a Term Frequency Inverse Document Frequency (TFIDF) Vector for each comment with a maximum of 5000 features per comment using Sklearn's TFIDFVectorizer. I decided to take the TFIDF with text that had been cleaned. This is a similar method to google's word2vec, in the sense we are turning comment data into numbers, but this method just tracks the frequency of words compared to its frequency in the corpus of documents. This does not depend on words' usage in common language for distance, but simply how frequent each word is used in each comment vs how frequent it is used in all comments.

So, using the TFIDF vector data with the MLP Classifier, it was able to bump the MLP up to 95.21% accuracy, almost as good as the logistic regression on the word2vec

model. Then applying the same TFIDF with logistic regression, it had an accuracy of 96.30% on the test set.

From this, we can draw the conclusion that the TFIDF is a better way of vectorizing comments. There may be many reasons as to why the TFIDF vectorizer improve the results. Firstly, google's word2vec algorithm could have not had all the words in the corpus used by the Kaggle competition. In this case, I decided to generate a random vector to act as a replacement. These vectors had no meaning, but could have skewed the error if comments had multiple words spelled wrong or unknown strings. Another possibility is that the TFIDF simply made better vectors. These vectors are more relevant to the corpus, which was cleaned, while the pre-trained word2vec algorithm had multitudes of the same word e.g. cabin, Cabin, cabins. These words had a much closer distance between each other, but may have made a larger distance between words and actual words they were near. The TFIDF also may have had a better way of compiling the entirety of the comment. While the word2vec took an average vector, the TFIDF vector consisted of frequencies of 5000 of the most important term frequencies. The data could have also made the TFIDF succeed. Simply put, the reason that the TFIDF functions better than the pre-trained word2vec model is that it uses frequencies in its own corpus, rather than distances in language. Therefore, for comments in the same data set, the wikipedia comments, is much better at classifying.

Is this Algorithm Applicable Outside the Wikipedia Comments?

Next, I decided to apply the best classification model, the TFIDF logistic regression, to a couple months of New York Times Comments. Since there are around 2 million comments and no real solution to what is toxic and what isn't, I decided to take the comments that the model classified as > 80% probability that it is toxic. This made a set of comments with a size around 17,000. This is fairly low, since there are around 2 million comments in the original data set. After some investigation into the data, I found that the New York Times does not attract toxicity or does an extremely good job of moderating its comment boards. I also looked into what it classified as an insult, and found that the word "kill" appeared in almost all of these types of comments. Sadly, kill was not being used as an insult to another commenter, but simply was commenting on the article, that involved death. Therefore, it seems as though, in an environment that

seems to have a higher level of education, and less toxicity, the model will latch onto specific words that are associated with toxicity, but miss the point of the comment.

Therefore, in my analysis, I decided to look at the sentiment of the comments that were classified as toxic. Using NLTK's Valence Aware Dictionary and Sentiment Reasoner (VADER) and I was able to find the sentiment of every comment in the New York Times comments. This data is given to us in a magnitude of how positive, negative and neutral a comment is. It also returns a compound score of the comment, which is based on a polynomial curve of its sentiment and does not depend on the score of the positive, negative and neutrality of the comment. This allowed me to compare the sentiment on average of the toxic comments to the comments in general to see if there was any change in the distribution. From appendix A, it can be seen that the compound distribution of sentiment seems to be more negative, which means that the comments classified as toxic had more negative sentiment. Therefore it can be concluded that the model does do well in predicting toxic comments, but relies on the corpus of comments having more interaction and disagreement, rather than a dataset that has far more neutral comments than anything else.

Future Directions and Discussion:

This project could be expanded to analyze more forums where people talk between each other and further honed to be better at analyzing toxicity. I assume that the point of this competition is to create a online moderation bot that might be able to detect comments that are hateful and remove them from their website. This, in terms of a kaggle competition is a good start, but the corpus of comments is far too small and only uses one data source to generalize it to all websites and comments. Also, there are problems with the model limiting users freedom of speech and censoring if this model is ever actually deployed.

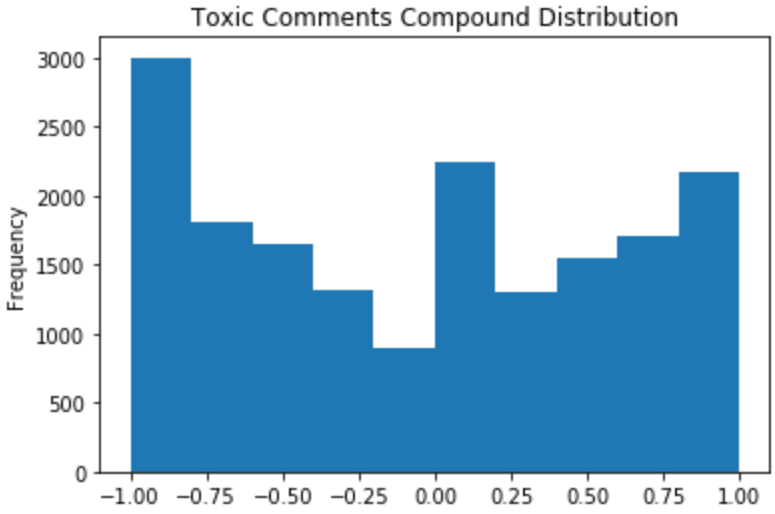
Conclusion:

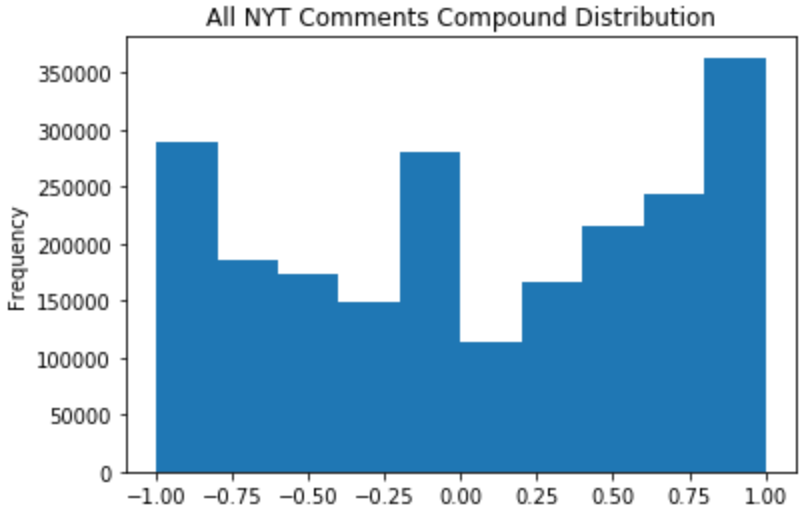
Through the methods used, the Word2Vec algorithm and TFIDF, for this dataset, the TFIDF worked slightly better than the Word2Vec algorithm. In terms of modeling, the MLP classifier also worked slightly worse than the logistic regression model. I was able to create a model using the TFIDF and logistic regression that classified correctly 96.30% of the comments. In terms of the kaggle leaderboard for the project the best score was around 98.90% accuracy, so this project is not too far off. Also, when I

applied it to the New York Times comments, it was able to find comments with more negative sentiment, but because of a lack of toxicity, the results were not exciting.

Appendix:

A.





Toxic Comments
Mean of Negative Sentiment: 0.12228214710378754
Mean of Positive Sentiment: 0.11163892191835083
Mean of Neutral Sentiment: 0.7660216862012379
Mean of Compound Sentiment: -0.052416544929505864
All Comments
Mean of Negative Sentiment: 0.09928626093795935
Mean of Positive Sentiment: 0.11829671920689967
Mean of Neutral Sentiment: 0.782388060085638
Mean of Compound Sentiment: 0.06153051704583903