JSC270 A4

Tiger Jin, Ethan Chan

Work Breakdown

Tiger was responsible for part 1. For part 2, we chose our topic, and at a high level, planned out what our investigation would look like jointly. Then, Tiger was responsible for the processing and analysis, while Ethan completed the presentation, visualizations, and report.

Part I: Sentiment Analysis with a Twitter Dataset

A)

Positive: 18042 Negative: 15397 Neutral: 7712

Positive: 43.84% Negative: 37.42% Neutral: 18.74%

B) Completed in the colab notebook.

C) Completed in the colab notebook.

D) Completed in the colab notebook.

It could be a scenario in which punctuation influences the tone of the sentence. For example, “Everything is fine.” can be considered neutral/positive, but “Everything is fine!” could be negative depending, as under certain conditions it might sound sarcastic.

E) We used PorterStemmer.

F) Completed in the colab notebook.

G)

The length of my vocabulary is 37901 for the training set, and 8598 for the test set. The combined vocabulary for them has length 39744. The number of columns (features) from the matrix of counts represents the number of distinct words in the vocabulary, thus the size of the vocabulary.

H)

Training accuracy: 0.7850 Test accuracy: 0.6796

Top 5 words for neutral and their counts: food: 712, price: 1370, covid: 2177, 19: 2198, coronaviru: 3817

Top 5 words for positive and their counts: supermarket: 3319, store: 3931, covid: 5361, 19: 5558, coronaviru: 7521

Top 5 words for negative and their counts: thi: 3230, store: 2697, covid: 4758, 19: 4925, coronaviru: 6745

I)

It would not be appropriate to fit an ROC curve in this scenario. An ROC curve is used to evaluate the performance of a binary classification model, which plots the true positive rate against the false positive rate at different classification thresholds. In this scenario, the model variable is a multi-class classification model. ROC curves are not applicable to multi-class classification problems because they only consider binary decisions.

J)

Training accuracy with TF-IDF: 0.7159 Test accuracy with TF-IDF: 0.6419

Training accuracy with count vectors: 0.7850 Test accuracy with count vectors: 0.6796

The accuracy gets worse when we use TF-IDF vectors instead of count vectors, with the training accuracy decrease by 0.07 and test accuracy decrease by 0.03 approximately.

K)

Training accuracy with lemmatization: 0.7252 Test accuracy with lemmatization: 0.6443

Training accuracy with stemming: 0.7159 Test accuracy with stemming: 0.6419

The accuracy of using lemmatization improved a little compared to the accuracy of using stemming. The training accuracy improved by 0.01 and test accuracy improved by 0.003 approximately.

Bonus:

The Naive Bayes model is generative. A generative model can generate new data that is similar to the training data, while a discriminative model can predict the class of a new data point, but it cannot generate new data. The Naive Bayes model assumes that the features are conditionally independent given the class. This assumption allows the model to learn the joint probability distribution of the features and the class.

Part II Report: Detecting and Classifying Hate Speech

Tiger Jin, Ethan Chan

Google Colab Link: JSC270_2024_Assignment4.ipynb

1 Problem Description and Motivation

As technology and the internet grows increasingly intertwined with our lives, the presence of social media has only risen, and accompanying it, hate speech. Racism, xenophobia, sexism, and bigotry of all kinds can be found across the web, posted by people who don't fear consequences as a result of the anonymity that often comes with being online. The phenomena of echo chambers can especially exacerbate the presence of hate speech. Hence, it is of the utmost importance that social media networks cut down on its presence on their sites. This led us to develop our direction of investigation: how can we classify and distinguish hate speech online? Are there particular models that perform better at this task? This question is difficult because detecting hate speech is a complex language task that can't simply be modelled as a function of social media metrics like engagement, follow count, likes, or other similar pieces of data. Furthermore, some people may disagree on the hard line that demarcates regular language from hate speech. Due to the fact that we require labeled data to train our models on, if the classifications of the dataset are only given by a single individual, it is possible that their personal biases may leak into the performance of our models, which presents another difficulty.

2 Data Description

We were unable to use the Twitter API to source our own data, so we looked online for a Tweet dataset satisfying our conditions. Consequently, we found a dataset with 24,783 total observations (before cleaning) where each data entry represented a Tweet. The dataset classified each tweet either as hate speech, offensive language, or neither. To do this, they employed a group of multiple people who would each vote on the category they felt the tweet fell into, and then classify it as the group that received the highest proportion of votes. The recorded features of the dataset included an index of the observation, the number of people who voted on the classification of the tweet, how many people voted it as hate speech, how many voted it as offensive language, how many people voted it as neither, the final classification of the tweet as an integer from 0 to 2 where 0

represents hate speech, 1 represents offensive language, and 2 represents neither, and finally, the raw text of the tweet. In our initial data cleaning, we simply removed all observations with empty tweet values, as those observations would not be helpful in training our models. The size of this dataset should be a very strong asset in terms of the development of our model/results, because with more data, outliers should play less of a role, and we should expect less variance in our results, helping us present more confident results. The fact that multiple people weigh in on the classification of each tweet also adds further credibility to the given labels, reducing any personal biases that may influence our models. However, one drawback of our data is the fact that since we did not directly gather it ourselves, we cannot be sure it is sampled without bias (i.e. randomly), or that it has been cherry-picked, which may lead us to true warped conclusions from our analysis. Furthermore, during our EDA, we noted that a disproportionate majority of the observations fell under the "offensive language" category, giving us an unbalanced dataset.

3 Exploratory Data Analysis

We begun our EDA by computing basic summary statistics for our data, which were fairly few in number, due to the relatively few features our dataset possessed. These statistics told us that for the majority of tweets, the voting team comprised of 3 individuals. In fact, the minimum number of people who voted on tweets was 3 (with a maximum of 9), so there were no tweets whose classifications were given solely by an individual, which lends further confidence towards the label of each observation. Next, as mentioned in the previous section, we computed the proportion that each class made up of the entire dataset. It turns out that 77.43% of the tweets were classified as offensive language, 16.80% were classified as neither offensive nor hate speech, with hate speech making up the remaining 5.77% of the dataset. We were a bit surprised at the disproportionate nature of the dataset, but overall, it seems like it reflects reality, as hate speech makes up a small minority out of all tweets. At this point, we also decided to compute 3 new features in the dataset, *hate_proportion*, *offensive_proportion*, and *neither_proportion* which each respectively represented the percentage of voters who classified the tweet as that category. We wanted to understand how "confident" the voting committee was in assigning the labels of each tweet. It turned out that for most tweets that were

classified as offensive or neither, there was a 100% consensus on the classification on the tweet (average consenses of ~92% and ~90% respectively, both with 100% medians). However, it seems that hate speech was much more ambiguous, with tweets in this group on average receiving a 72.8% consensus in their labeling. This suggests to us that even for humans, hate speech classification is a difficult and ambiguous task. At this point, we began the text processing for the tweets, following a similar approach to the one taken in Part I, by tokenizing, removing links and punctuation, lemmatizing, and removing stopwords/null tokens.

4 Models

As our dataset is labeled, and we are attempting to classify tweets into one of hate speech, offensive language, or neither, we will be undertaking a multiclass-classification supervised learning task. Due to the fact that one our directions of investigation was understanding how different models performed in hate speech recognition, we chose 4 models all capable of this: Multinomial Naive Bayes (referred to as MNB), Logistic Regression (referred to as LR), Decision Tree (referred to as DT), and Random Forest (referred to as RF). We will be utilizing the count-vectorization method to transform the tweets into inputs for the models. MNB is a generative model that models the probability of observing the data given the classification, then applies Bayes rules to get the reverse, classifying the input. It is effective when dealing with many features (i.e. a large variety of words), but also makes the assumption that all features - the words, are independent, which is not necessarily the case. LR, DT, and RF, on the other hand, are all discriminative models. LR, although normally a binary model, can be generalized to multiclass classification by using multiple binary LR models, whose results are combined together. It is simple, but also requires that the input features have low multicollinearity. DT generates a set of classification rules that maximizes accuracy on its training data. However, as a result, it often suffers from overfitting, which is why we also include RF, which essentially generates multiple DT models on bootstrapped training data. It classifies by running the input data on all of its trees and aggregating their decisions together, and as a result, is much more flexible than DT. However, this approach can be computationally expensive, especially if there are a large number of trees. Running an 80/20 train/test split, we will evaluate model performance

primarily based on how well our models are able to distinguish hate speech. In particular, we would like a high recall for the hate speech class, even at the cost of having a high false positive rate. Potentially classifying some offensive/neutral tweets as hate speech, but being able to catch (and hence remove) the majority of hate speech is much better than the alternative, where we let more hate speech through in hopes of increasing model accuracy.

5 Results and Conclusions

After training our models and running them on our testing dataset, we found that MNB, LR, DT, and RF had roughly 87%, 90%, 87%, and 89% accuracy - a seemingly good result. However, when we consider the confusion matrices, while we can see that the majority of truly offensive and neither tweets are accurately predicted into the correct categories by the models, the majority of truly hate speech tweets are actually predicted as offensive tweets, which is not what we want to see, based on our evaluation metrics. Looking closer at the statistics of the results, we notice that all the models perform very poorly when it comes to classifying hate speech. With precision hovering around the mid 20% range (with the exception of MNB, at 9%), and recall in the 35 - 45% range, none of the models seem to be effective in their task. The reason why the models were able to attain such high overall accuracy scores is due to the combination of facts that offensive tweets made up the majority of the data (and hence the testing set), and that the models performed well in their ability to distinguish offensive tweets, meaning that for most of the testing data, the models classified accurately. We can see this reflected in the fact that the weighted average performance scores are all fairly high, while the macro average scores, which take into account the performance of the models on each class equally, are much lower. One conjecture for why the models collectively performed so terribly is due to the fact that they are all bag-of-word models. This means that they model the meaning of a tweet solely based on which words were used, and how frequently, without taking into account any context, grammar, or semantic meaning. This is a very severe weakness, as we can see that for instance, the two sentences “I love my friend, but hate homework”, and “I hate my friend, but love homework” have very clearly different semantic meaning, yet appear as exactly the same input to the models, due to the fact that they have the same words, in the same numbers. Furthermore,

many words change meaning depending on context (e.g. bat being possibly an animal, object, or verb), but these models have no way to distill that information when they only receive information about whether a word existed in the tweet, and how many times it did so. Consequently, when the distribution of words from two separate classes are too similar (i.e. they use similar vocabulary), it becomes very difficult for bag-of-words models to distinguish the two groups apart. We went back to our data, and modelled how frequently some of the most common overall words appeared in each class, and noticed that many of the words which appeared frequently in hate speech also were present in significant amounts in offensive language, giving credence to our conjecture. Another possible reason for the poor performance of our models is due to the disproportionately balanced nature of our dataset. While this may be more reflective of the distribution of tweets in reality, it may actually encourage our models to tend to predict tweets as offensive language, simply because they observed it more often in the training data. This is particularly the case for MNB, which incorporates a prior. Overall, we can conclude that for a task as complex and semantically dependent as hate speech classification, bag-of-words models are far too simple, and discard far too much information, to possibly be effective. Instead, we should incorporate tools that take into account factors like order, perhaps through the use of n-grams, or word importance, through TF-IDF instead of simply count-vectorizing the text. It would also be a good idea to try and source more balanced data, even though it may be less reflective of reality, as it would help prevent the models from predicting "offensive language" simply due to the fact that it is very common. Other avenues of improvement are to increase the features we inference on. For instance, we could begin considering tweet sentiment, or engagement, in tandem with the tweet itself, which may help us better distinguish hate speech.