Predicting Political Affiliation of Congresspeople Based on Twitter Data

Sam Arnts[1] and Hanyang Zhang[2]

I. Introduction

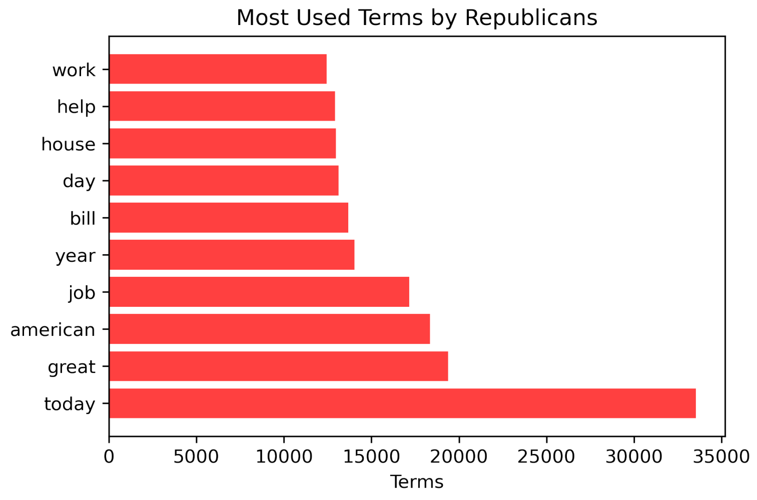
The goal of this assignment was to predict party affiliation based upon a twitter dataset which contains both training.csv (592803) and testing.csv (265000). We produced visuals of different types of methods, and compared the results among them. Both of the datasets contain six columns representing the id number of the tweets, the number of favorite tweets counted, the full tweets, lists of hashtags in the tweets, the number of times that the tweets have been retweeted, and the year of the tweets produced. There are some significant findings we found while doing the competition. First, after implementing "Logistic", "Gaussian", "Complement", "RandomForest", "ADABoost" methods, the method with highest accuracy is the random forest classification algorithm which is 0.861. Another finding is that the number of times specific words or hashtags are used in the test texts between Democrats and Republicans are significantly different. For example, Democrats used the word “family” much more often than Republicans, while Republicans use the word “great” much more often than Democrats, which suggests the fact that Republicans and Democrats each tend to discuss certain topics or issues more than the other group. The details of the finding will be found in the following sections.

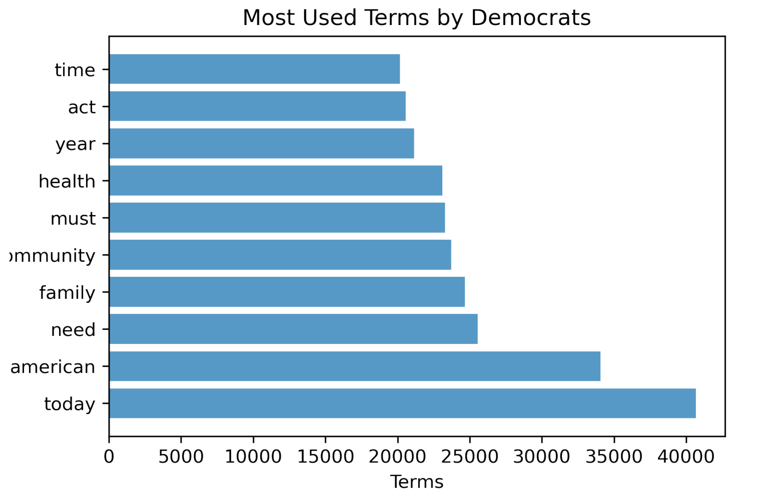
II. Descriptive Analysis

In our descriptive analysis, we tried to find differences in the way in which Democrats and Republicans tweet, and tried to create a model based on our findings.

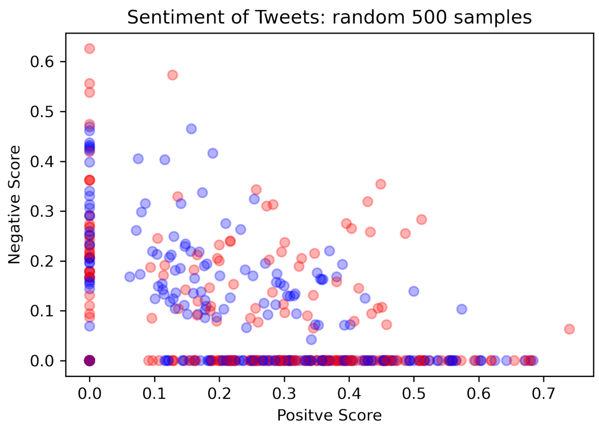
We thought the best way to go about this was to make two separate dictionaries from our training data; one containing the top 100,000 words used in tweets made by Republicans, and one containing the top 100,000 words used in tweets made by Democrats. To do this, we first had to remove stop words (stop words as defined in by the nltk package), remove links, and then lemmatize and tokenize the text. After cleaning the data, we sorted the text into two arrays based on party id.

Next, we used the CountVectorizer function to collect the frequency of each word that is used by the two parties. Figures 1 and 2 show the 10 most used words by each party. Both parties use the word “today” much more than any other word, so a tweet containing the word “today” isn’t very helpful in classifying the tweets. The data shows that Republicans use the words “great”, “job”, “bill”, and “help” at a much higher proportion than Democrats, who use the words “family”, “need”, “health”, and “act” at a much higher proportion than Republicans. As a side note, for those who are familiar with American politics, these results seem very reasonable, as Republicans are known to focus on job creation and wanting to “Make America Great”, while Democrats are known more for pushing for advances in healthcare and increased support for families in America. The difference in the number of times these words are used by each party is the main driver behind our classification method which we describe below.





We also wanted to look at the sentiment of the tweets, and see whether or not one party’s tweets had a more negative or positive connotation as compared to the other party. We achieved this through utilizing the SentimentIntensityAnalyzer (SIA) tool in the nltk library. SIA and its pretrained sentiment analyzer VADER (Valence Aware Dictionary and sEntiment Reasoner) can assign a tweet a negativity and positivity score (scores range from 0 to 1) based on its content. For each tweet, we subtracted its negativity score from its positivity score, meaning that tweets with a heavy positive connotation had a score closer to positive 1, and tweets with a heavy negative connotation had a score closer to negative 1. As Figure 3 shows, however, there is no indication that one party is necessarily more positive or negative on twitter as compared to the other, so this effort was primarily in vain. One interesting observation that doesn't necessarily pertain to the project, however, is that it seems congressmen and congresswomen of both parties send out more tweets with a positive connotation than with a negative connotation. (Maybe this is surprising to us because only the most negative of tweets make headlines?)

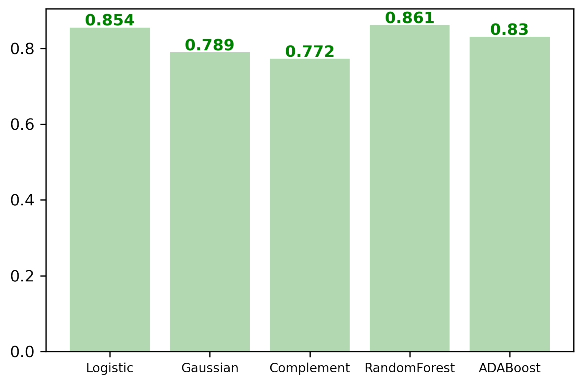


III. Classification Methods

We tried a wide array of methods to improve our classification model. The main driver of our classification strategy is located in the party-score function. Here, the parameters are the dictionary of words the Democrats used, the dictionary of words the republicans used, the text, and how large we wanted to make new dictionaries with different levels of exclusivity. The purpose of this function is to assign each tweet a value that expresses how well the tweet falls into the Democrat or Republican archetype. If a word in a tweet is in the Democrat dictionary, it gets the amount of times the word is used by Democrats added to its Democrat score, and if a word is in the Republican dictionary, it gets the amount of times the word is used by Republicans added to its Republican score. More importantly, however, is if a word is in one party's dictionary and not in another. In this case, the points are determined by how exclusive the word is to one particular party. For example, in our current code, if a word is in the Democrat's main dictionary, but not in the dictionary representing the Republican's top 25,000 words, the Democrat score is multiplied by 10,000. If the word is not in the Republican's top 5,000 words, the score is multiplied by 1,000. We spent a lot of time both tweaking the size of the exclusive dictionaries, and tweaking the amount we would multiply the scores by when one of the above conditions are met.

We also experimented with training our models with different amounts of data, and at first we thought training with a smaller amount of data (300,000 samples) was producing better results. This is because when we were utilizing tools such as train-test-split to test our model, we were getting higher accuracies with lower amounts of data and smaller dictionary sizes, which can be explained by more words that aren't really important to classification being left out of our dictionaries. However, when we switched to testing our models with the actual test set, we found that our results were significantly worse with a smaller amount of samples as compared to if we just used the full train set. This can be attributed to the fact that the words in the test set cannot be added to the party specific dictionaries (as obviously no labels are provided), so the best course of action is to use the most amount of data to train the model as we could, as it will be a better representation of all tweets from congresspeople in general.

Finally, we utilized many different models to test which was best for our data set. As Figure 4 shows, the models that performed the best were Logistic Regression and Random Forest Models. As Professor mentioned in class, Ensemble Learning methods are typically the ones that are used by winning Kaggle competition teams, so we weren't surprised that it performed well. We experimented with the max-depth parameter for the Random Forest Classifier, and found that a depth around 20 would produce the best results.



IV. Classification Results

Overall our classification results are quite poor when compared to the rest of the class. Our accuracy scores hover around 83-84 percent, about 5 percentage points lower than the majority of the pack. We don't think the problem is with the type of classification algorithms we used, especially because many of our classmates on Slack discussed using the same algorithms that we did, and because many resources online suggest logistic regression / random forest classification for nlp. We think that the low score is due to the method in which we assigned "Democrat" or "Republican" scores to tweets. First, we only considered looking for individual words that are indicative of being a Republican or Democrat tweet, when we should probably have looked for n-grams of size 2 or 3 that are common among the parties. This idea occurred to us very soon before the deadline, and we are discovering it is more computationally intensive to look for n-grams of size 2 or 3 within tweets rather than individual words. Also, we didn't really experiment with the TfidfVectorizer until the very end, and after observing our classmates' discussions on Slack and doing some research of our own, we are finding the TfidfVectorization may have been more appropriate for this problem. If we were to do this project again, we would spend more time experimenting with determining n-grams of size 2 or 3 that are associated with a specific party, and spend less time adjusting the parameters in our party-score function. We live and learn, I suppose.