Analysis of the Relationship between U.S. Congressmen's Tweets and Their Political Party

Amro Ashmeik, Daniel McGrory
Northwestern University EECS 349 Final Report

1 Task

Our task for this machine learning project was to predict a congressman's political party (Democrat or Republican) based on their tweets. This subject is particularly timely because, as the trends of Russian bots and fake news have demonstrated in the past few years, social media is a powerful medium for impacting the minds of voters. Therefore, the use of social media is essential for today's politicians to gain an edge in elections by creating a medium in which they can connect with their constituents and receive feedback, demonstrate their commitment to making a difference for their voter base, and to give their views on current events, bills, elections, etc. Twitter is also a unique medium when it comes to expression. The words used in each tweet are likely to have more meaning and power behind them and possibly be more polarizing or inflammatory than those in other mediums because the limited word count in each tweet forces the user to make each word count; therefore, we believe the words that the politician use in each tweet are more likely to be more partisan to appeal to their voter base; this project is designed to test this hypothesis.

2 Dataset

Our initial data set was composed of the most recent 500 tweets (as of May 22, 2018 and before) of the 535 members of Congress and 8 accounts of congressmen's press offices, meaning our initial data set included 271500 tweets. Each member of Congress was labeled as Democrat, Republican, or Independent. After removing the 2 Independents in Congress, we were left with 541 accounts which we could analyse; therefore, the total number of tweets in our final dataset was 270500. After preprocessing (removing stop words, punctuation, links, etc.) the tweets, we found there are roughly ~2.2 million total words and ~98,000 unique words in our final dataset. In addition to basic preprocessing, we removed words such as "Democrat", "Republican", "Dems," "GOP," etc. because we believed it would make the classification of examples trivial.

3 Approach

In the following sections, we discuss how we selected our models, features, and the breakdown of our dataset into training and testing sets.

3.1 Model Selection

Our chosen task is a text classification problem, more specifically a sentiment analysis problem, so through research, we determined that Recurrent Neural Networks (RNN), Support Vector Machines (SVM), and Naive Bayes were all good options for machine learning models for sentiment analysis. After further study, we decided to use the Naive Bayes and Support Vector Machines models because the RNN take longer to train but would only provide negligibly better accuracy. After preprocessing our data, both models performed extremely well and only took a few minutes to run on our huge dataset; therefore, we decided to use both. We acquired, trained, and tested the models using the sklearn Python package and used the default parameters for each model.

3.2 Feature Selection

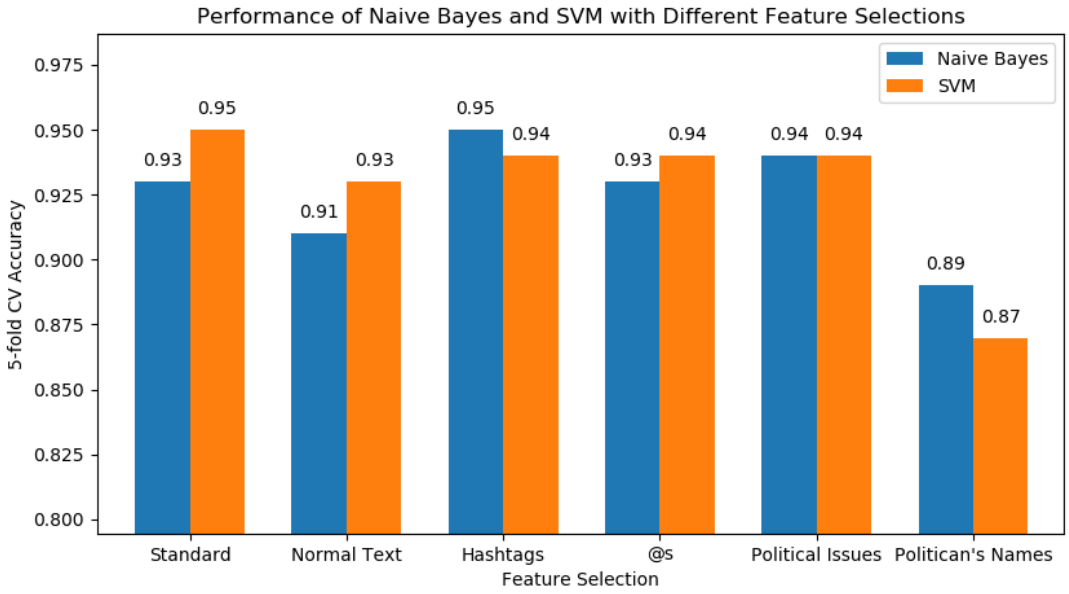
Our baseline features consisted of the words in each tweet; more specifically, we focused on the the content of hashtags and @'s, common political issues, politician's names and normal text (all text featured in the tweets that does not fall under the category of any of the previous features). The list of political issues we searched for in the tweets was scraped and compiled from the Internet.[1] The list of politician's names is composed of all members of Congress, the president, his cabinet, all heads of major federal bureaus and notable public figures and was also scraped from the Internet.[2,3,4] In all tables and figures, the data set containing all features is referred to as Standard and each dataset containing only one of these individual features are referred to by the name of that feature.

3.3 Training and Testing Set Breakdown
We used 5-fold cross validation to evaluate our models; therefore, 80% of the dataset was used for training and 20% was used for testing.

4 Results
Using both Naive Bayes and SVM, we found the accuracy of our model's ability to classify each politician into their respective party based on all of the features in the dataset as well as by evaluating each feature in the data set individually (See Figure 1). To better understand the total accuracy that we calculated for each algorithm, we calculated the precision, recall, and various other metrics of the models run on all features in order to find the total accuracy (See Table 1). We also used SVM to find which words from the dataset are the most polarized (See Table 2a and 2b). The final metric we evaluated was the accuracy of each model after the top N number of these polarized words we removed.

4.1 Analysis
After training and testing on the (Standard) data set using 5 fold cross validation, we found the total accuracy of our ability to classify each politician based on all features of the dataset. The accuracy for Naive Bayes was 93% and the accuracy for SVM was 95%. ZeroR accuracy for our dataset is 54.5%, so Naive Bayes and SVM performed much better than the baseline.

Figure 1. *Comparing 5-fold CV accuracy of Naive Bayes and SVM with different features*



Using 5 fold cross validation, we were able to able to calculate the precision, recall, f1 and support metrics for each algorithm, as can be seen in Table 1. Naive Bayes seems to have a low recall for Democrats. In this case, recall tells us the proportion of actual Democrats that were classified correctly, meaning that that Naive Bayes incorrectly classifies Democrats as Republicans at a relatively high rate. Since Republicans are the majority label in our dataset, this aligns with our knowledge of Naive Bayes classifying an example with the majority label when it is unsure. Overall, our F1 scores are high for both Naive Bayes and SVM.

Table 1. *Summary of the precision, recall, F1, and support obtained using standard cleaning and 5-fold Cross Validation for Naive Bayes and SVM*

Algorithm	Party	Precision	Recall	F1	Support
Naive Bayes	**Democrat**	0.98	0.83	0.90	246
	Republican	0.92	0.91	0.93	295
SVM	**Democrat**	0.97	0.93	0.95	246
	Republican	0.95	0.97	0.96	295

Across each feature selection, the accuracy in the models did not vary much. For every feature selection, except using politicians' names only, the accuracies were within 91%-95%. The worst performing model was SVM, with an accuracy of 87% using politicians' names as the only features. Removing hashtags, @s, political issues, and politicians' names saw a slight decrease in performance, but it was still within 4% of the best model. It is interesting to see that hashtags alone and @'s alone both contain enough information to classify at 90%+ accuracy. It is a testament to how polarizing the hashtags and the act of tagging other users can be. For numerical data on these individual features, see Figure 1 earlier in the Analysis section.

We were also able to determine which words were the most polarizing (as in which word is the most frequently in one class over another). Features with the greatest negative coefficients were words that occurred more frequently among Democrats as opposed to Republicans and features with the greatest positive coefficients were words that occurred more frequently among Republicans as opposed to Democrats. The greater the absolute value of the coefficient, the more information it provides to the classifier. Table 2a and Table 2b show the top 10 most informative features for each label, categorized by the feature selection. Considering that our dataset consisted of the most recent 500 tweets (at that time), the most informative features appear to be words reflecting people or issues that have been recently prominent in the news. For instance, looking at the standard cleaning column, we can see that Trump, gun (likely referring to gun control), health, and net neutrality are the most talked about topics by Democrats as opposed to to tax reform and obamacare for Republicans, which were all popular topics in May.

Table 2a. *Top 10 Democratic words found via measuring the descending regression coefficients of each word using SVM*

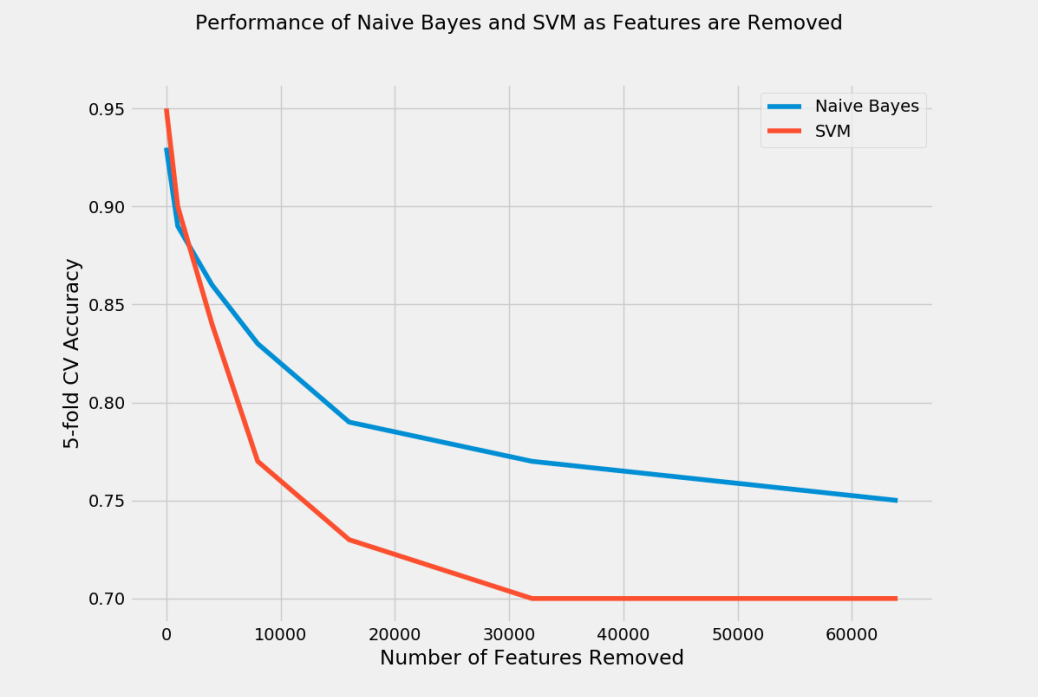
Standard	Normal Text	Hashtags	@s	Issues	Names
trump	would	netneutrality	fcc	gun	trump
gun	gop	goptaxscam	gop	trump	pruitt
health	delaware	getcovered	cfpb	health	young
netneutrality	communities	trumpshutdown	housedemocrats	wall	joe
joe	must	dreamers	housegop	vote	coons
netde	open	trump	epascottpruit	infrastructure	claire
dreamers	laredo	teamgottheimer	uscbo	climate	pete
loebsack	must	netde	msnbc	medicare	brady
goptaxscam	internet	teampeters	fvsu	campaign	delaney
aca	corporations	aca	senmarkey	change	lipinski

Table 2b. *Top 10 Republican words found via measuring the descending regression coefficients of each word using SVM*

Standard	Normal Text	Hashtags	@s	Issues	Names
taxreform	code	taxreform	housecommerce	reform	graham
tax	great	taxcutsandjobsact	foxbusiness	house	obama
great	passed	obamacare	houseappropsgop	obamacare	billy
reform	bonuses	schumersshutdown	stevescalise	obama	gorsuch
obamacare	mississippi	99countymeetings	vp	senate	clinton
mississippi	hearing	medicareforall	foxnews	clinton	neil
senate	act	endalz	waysandmeansgop	tax	schumer
taxcutsandjobsact	tune	tcot	financialcmte	conservative	mike
sofla	jobs	utpol	senronjohnson	afghanistan	pence
house	news	betterway	hascrepublicans	terrorism	hillary

Figure 1 shows that Naive Bayes and SVM perform similarly across every feature selection. Figure 2 shows the importance of the most highly weighted (most polarizing) features in regards to the accuracy of our models. Removing 1000 of the most highly weighted features (500 from each side of the spectrum) showed the greatest rate of decrease in the performance of the classifiers. The model performance decreases as each additional feature is removed and the rate of decrease lessens, which is why we see the performance gradually decrease and appear to begin plateauing as the more insignificant features are removed.

Figure 2. *5-fold CV accuracies of Naive Bayes and SVM as the top N number of features with the greatest regression coefficients are removed for each label.*



4.2 Conclusions

It is clear from our data and analysis that it is possible to identify with high accuracy the political party of an individual politician based on the tweets on their Twitter profile. When using both of our algorithms for analysis on the full data set, we were able to predict the party of a politician with at least 93% accuracy. Even when we removed highly polarized words, the words most useful in making our decision on a politician's party, from the data set, we still maintained a relatively high accuracy of at least 70%. Additionally, when we focused on individual features, such as hashtags or politician's names, we achieved at least 87% accuracy. Clearly, there is a strong relationship between the words a politician uses in their tweets and their political identity, even in small samples of their tweets or when evaluating individual features such as hashtags or the accounts they tag. The data we gathered on the most polarizing words is also very useful; the amount of unique words that are highly polarizing is incredibly large because even when the highest weighted section are removed from the data set, both models still achieve relatively high accuracies, as can be seen in Figure 2. Furthermore, the actual content described by these highly polarizing words demonstrates that many of the polarizing words correspond with the most polarizing news topics or events occurring at the time of the tweet. Using all of this information, we, as citizens, can be better informed on how to receive and process the information our politicians convey to us over Twitter so as to be more objective in our evaluation of this information and avoid being influenced by their biases, partisanship and rhetorical strategies.

5 Future Work

There are a lot of opportunities to expand this research in the future. We could expand our sample pool to include the Twitter profiles of politicians from other branches of the federal government, bureaucrats, and state politicians. We could also develop a similar model by training on the profiles of politicians from other countries with multiple parties to see if our model is able to handle a system with more than two parties. We could even go so far as to try to train a model on regular Twitter users to see if we are able to predict the political orientation of any Twitter user using machine learning. Additionally, we could also attempt to develop models which give a numerical output as to where a politician lies on the continuous political spectrum (left, right, authoritarian, libertarian, etc) as opposed to our current discrete, binary output.

6 Attributions

Amro: Modified code of a Twitter scraper[5] from online, used the scraper to collect initial dataset, composed the initial data cleaning file, extensively rewroted and reworked the cleaning file until it worked as desired. Rewrote and edited website code from a Bootstrap template online. Composed figures from data. Wrote and edited the final report.

Dan: Wrote a file to handle substrings for cleaning, wrote several files to scrape and store data from websites and text sources in order to compile the issues and names datasets, extensively rewrote and reworked the cleaning file until it worked as desired. Rewrote and edited website code from a Bootstrap template online. Composed tables from data. Wrote and edited the final report.

7 References

[1] Political Issues of 2018." *ISideWith*, ISideWith.com, www.isidewith.com/polls.

[2] "A-Z Index of U.S. Government Departments and Agencies | USAGov." *U.S. Data and Statistics | USAGov*, www.usa.gov/federal-agencies/.

[3] "Members of the U.S. Congress." *H.R.1238 - 115th Congress (2017-2018): Congress.gov | Library of Congress*, Library of Congress, congress.gov/members?q=%7B%22congress%22%3A%22115%22%7D.

[4] "The Cabinet." *The White House*, The United States Government, www.whitehouse.gov/the-trump-administration/the-cabinet/.

[5] Henrique, Jefferson, GetOldTweets-python (2016), GitHub repository, https://github.com/Jefferson-Henrique/GetOldTweets-python