Predicting Political Party from Twitter Content

Jonah May
Department of Computer Science
Princeton University '17
jrmay@princeton.edu

Abstract

In this project, we attempt to predict political party using content from the Twitter feeds of members of the House of Representatives. The predictions used the entire Twitter history for 430 members of the House of Representatives. In order to extract features from this text, we utilized a bag of words representation. We used three techniques to classify these bags of words: Support Vector Machines, K-Nearest Neighbor, and Decision Trees. Across all Congresspeople, this yielded a feature space of roughly 1.1 million unique words. When using this entire feature set, all three algorithms achieved an area under the Receiver Operating Characteristic curve that was greater than .95. Next, we investigated how the classification techniques perform under a reduced feature space. When we reduced the feature space to include only hashtags, we were able to achieve better predictive performance than with the entire bag of words. With only the 5 most common hashtags, we were able to achieve and area under the ROC curve of .93. Additionally, we explored how the various algorithms performed as a function of training size. Both KNN and SVM suffered as the training set size decreased. However, Decision Trees were quite robust to reductions in training set size, achieving an area under the ROC curve of over .90 after being trained on only 35 Congresspeople. As we conducted experiments, we kept track of which Congress people were most often misclassified. We then compared them to ideology scores and found that misclassifications often occurred on politically moderate Congresspeople.

1 Introduction

As political rhetoric becomes more and more partisan, it is becoming important to understand how language shapes political party. The primary goal of this project is to predict the political party of members of the House of Representatives using the content of their Twitter feeds. Beyond simple classification, the secondary goal is to understand which words are the most important predictors of political party. We are not only concerned with making accurate predictions, but learning what features were important for making these predictions. We also want to explore how changes in the feature space and training set size impact the performance of our classifying algorithms.

2 Related Work

There has been other academic work that has explored political ideology on Twitter.They used volunteers to help label the twitter profiles. I will not need to label any users by hand because the political party of each politician is already known (Conover et.al.).this study, they were able to a chieve an accuracy rate of 91% using hashtag alone. They found that using hashtag data actually was a better predictor than the full text data. I plan on only using text data. If it turns out that my model predicts political party of politicians extremely well, I may extend it to attempting to classify non-political users on Twitter.

3 Methods

3.1 Raw Data Acquisition

Acquiring the data required significant interfacing with the Twitter API. In order to make simple calls to the API, we used the Python wrapper Tweepy. It allowed us to make calls to the API without having to worry about formatting urls and parameter properly. Data Acquisition required three major steps.

First, we had to procure a list of members of the House of Representatives and their Twitter usernames. Fortunately, Twitter maintains lists of of verified users of many categories, including branches of government. By simply querying this list, we were able to get a list of usernames and the name that the representative provided as their name. For most, this was simply their first and last name. However, many others would include titles such as Rep. or Congressman/woman.

Next, we had to gather tweets for each username on the list of members of the House of Representatives. Twitter imposes rate limiting on their API, of 450 calls every 15 minutes. Because public figures, like politicians, have high numbers of tweets, each call to the API can only gather a subset of their tweets. As a result, the tweets were gathered over a roughly 24 hour period in order to stay within the rate limit constraints.

The final step involved assigning a political label to each member of the House. In order to do this, we downloaded a table of members from the U.S. House website. The table included name and political party. One challenge we faced included that many politicians slightly altered their names on Twitter. Therefore, we had to do a fuzzy match to the official names used on the U.S. House website. This fuzzy matching was performed using python's fuzzywuzzy package.

After these steps, we had full 430 full Twitter histories along with political labels for each of them.

3.2 Bag of Words

In order to extract features from the raw tweets we used a bag of words representation. In a bag or words representation the features are the number of times a given word appeared in the text. In order to compute this bag of words, we used SciKit Learn's CountVectorizer class. The bag of words representation can be used to also track frequency of various length N-grams. For this project, however, we focused on individual words.

3.3 Feature Reduction

Feature reduction can be thought of as a horizontal shrinking of the data set. In order to perform feature reduction, we limited the features in two main ways. First, we created various frequency thresholds for features. In other words, if a word appeared very infrequently across all of the Congresspeople, it was not included in the feature space. Additionally, we limited the feature space to include only the top N most frequently occurring words. This technique was highly effective because it allowed us to have complete control over the exact feature space size.

We also reduced the feature space by limiting it to contain only hashtags. This was a particularly interesting way to reduce features because hashtags are, in a sense, summarizing words chosen by the tweets author.

For all classifications using feature reduction, we used a KFold cross validation technique. We used 10 folds, and the data was randomly selected.

3.4 Training Set Size Reduction

In addition to shrinking our data horizontally, we also experimented with shrinking the training data vertically. In order to do this, we iteratively trained our model on increasing training set sizes. For each iteration, we would record a ”score”. For this experiment, the score was the accuracy of the predictions from that trained model. With these accuracy scores, we were able to create a learning curve that shows the relationship between training set size and accuracy.

3.5 Evaluation Metrics

Our primary metric for testing our models was the area under the ROC curve. For these curves, any values around .5 are indicative are of non-predictive model. As values increase towards 1, this indicates a more powerful model. For the learning curve plots, we chose to use accuracy as our scoring metric rather than area under the ROC curve.

3.6 Misclassification

As we conducted these experiments, we kept a tally of which politicians were most often misclassified. This data was later used to perform additionally evaluation and diagnostics beyond ROC curve. It helped us investigate questions such as: who does our model misclassify and why?

4 Results

4.1 Feature Reduction

For the table below, we performed feature reduction by limiting the minimum frequency that a word had to appear to be included in the feature set. With the Min. Freq. = 1, this is equivalent to every word getting its own feature. As Min Freq. increases, the feature set size decreases dramatically. When Min. Freq. = 1, there are over 1 million features. When Min. Freq = 250, there are less than 5,000 features. The values in the table are the average areas under the ROC curve created during cross validation. As you can see, as the size of the feature space decreases, SVM classification begins to perform poorly. On the other hand, both KNN and Decision tree still perform quite well as the feature set size is reduced.

We can see that for the large feature set size, all of the algorithms perform their best. However, there is a time efficiency trade off. For the largest feature size, the average run time for single fold was 52.68 seconds. As we reduced the feature size with a minimum frequency of 250, the runtime was cut to 23.88 seconds.

Table 1: Reduction by Min. Word Frequency

	SVM	KNN(N=5)	Decision Tree
Min Freq = 1	.97	.97	.96
Min Freq = 50	.68	.94	.96
Min Freq = 100	.50	.94	.95
Min Freq = 250	.49	.95	.92

In this table, we reduced the feature size to include only the N most commonly used features. At a feature set size of 5, both KNN and Decision Tree begin to trend down towards area under the ROC curve of .5.

Table 2: Reduction by feature maximum

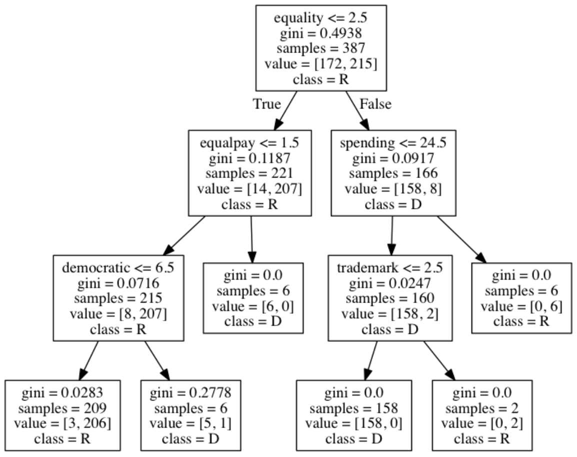
	SVM	KNN	Decision Tree
Feat. Max = 25	.5	.825	.71
Feat. Max = 5	.49	.744	.64

In the table below, we first subset our data to include only hashtags. Then, we limited the feature set size to the N most common predictors. Compared to the performance of all words, the algorithms are all more successful classifiers. With the same number of features, the hashtag-only predictor is much more successful.

Table 3: Reduction by feature maximum (hashtag)

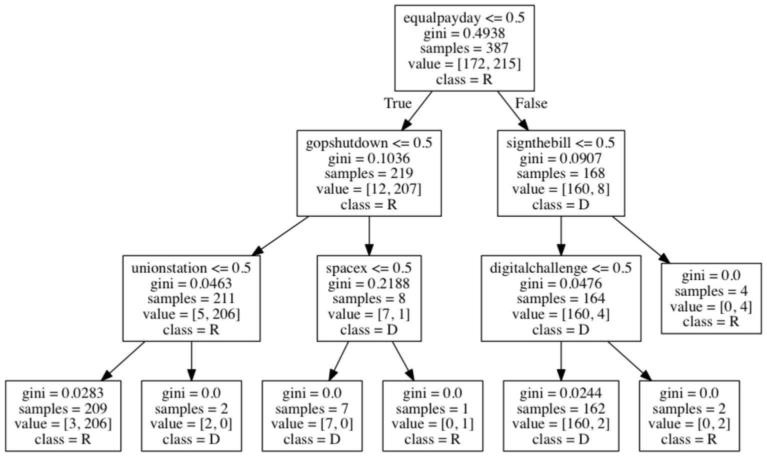
	SVM	KNN	Decision Tree
Feat.Max = None	.97	.94	.94
Feat.Max = 250	.79	.96	.97
Feat.Max = 100	.74	.97	.94
Feat.Max = 25	.74	.96	.94
Feat.Max = 5	.78	.93	.86

However, while hashtags may be a better predictor, they reveal less about how language divides the two political parties. In these decision trees, you can see the feature word, along with the splitting rule associated with it. The text only tree reveals that the most important predicting word is ”equality”. It also reveals other divisive words such as ”spending”. In the hashtag decision tree, we can see splitting rules that include highly partisan hashtags. A Republican Congressperson would never use #gopshutdown. As a result, the text features are more revealing than the hashtag features, even though hashtags are more predictive.



equality <= 2.5 gini = 0.4938 samples = 387 value = [172, 215] class = R
True
False
equalpay <= 1.5 gini = 0.1187 samples = 221 value = [14, 207] class = R
spending <= 24.5 gini = 0.0917 samples = 166 value = [158, 8] class = D
democratic <= 6.5 gini = 0.0716 samples = 215 value = [8, 207] class = R
gini = 0.0 samples = 6 value = [6, 0] class = D
trademark <= 2.5 gini = 0.0247 samples = 160 value = [158, 2] class = D
gini = 0.0 samples = 6 value = [0, 6] class = R
gini = 0.0283 samples = 209 value = [3, 206] class = R
gini = 0.2778 samples = 6 value = [5, 1] class = D
gini = 0.0 samples = 158 value = [158, 0] class = D
gini = 0.0 samples = 2 value = [0, 2] class = R

Figure 1: Full Set Decision Tree



equalpayday <= 0.5 gini = 0.4938 samples = 387 value = [172, 215] class = R
True
False
gopshutdown <= 0.5 gini = 0.1036 samples = 219 value = [12, 207] class = R
signthebill <= 0.5 gini = 0.0907 samples = 168 value = [160, 8] class = D
unionstation <= 0.5 gini = 0.0463 samples = 211 value = [5, 206] class = R
spacex <= 0.5 gini = 0.2188 samples = 8 value = [7, 1] class = D
digitalchallenge <= 0.5 gini = 0.0476 samples = 164 value = [160, 4] class = D
gini = 0.0 samples = 4 value = [0, 4] class = R
gini = 0.0283 samples = 209 value = [3, 206] class = R
gini = 0.0 samples = 2 value = [2, 0] class = D
gini = 0.0 samples = 7 value = [7, 0] class = D
gini = 0.0 samples = 1 value = [0, 1] class = R
gini = 0.0244 samples = 162 value = [160, 2] class = D
gini = 0.0 samples = 2 value = [0, 2] class = R

Figure 2: Hashtag Decision Tree

4.2 Training Size Reduction

In addition to shrinking the data set horizontally with feature reduction, we also investigate how the algorithms could handle being trained on very small training sets. In order to do this, we created learning curves. The training data for these curves was a hashtag-only feature set.

As you can see, Decision Tree algorithms perform quite well even with a small training set. On the other hand, SVM and KNN require larger datasets to achieve their maximum potential.

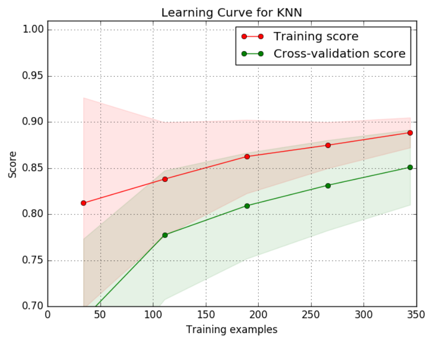


Figure 3: KNN learning curve

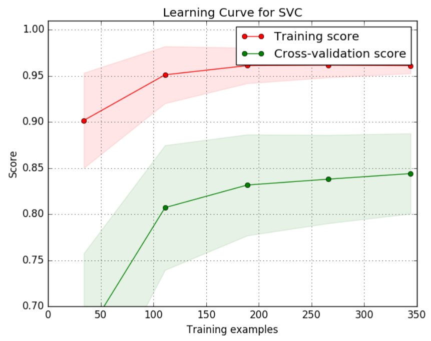


Figure 4: SVC learning curve

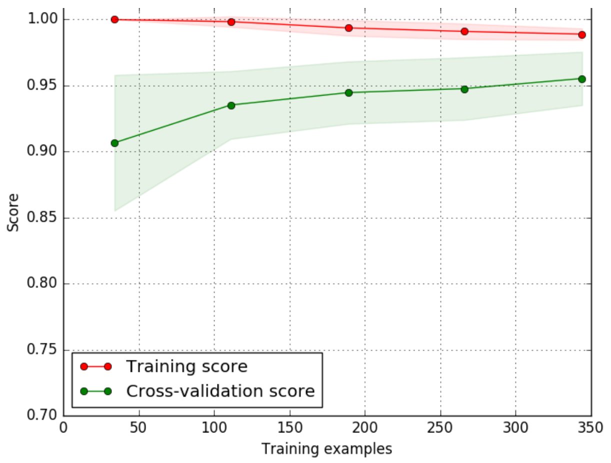


Figure 5: Decision Tree Learning Curve

4.3 Misclassifications

Below are the most often misclassified republican members of the house. The ideology score is a scale from 0-1. 0 is most liberal. 1 is most conservative. The ideology scores were sourced from YouGov.com. The average ideology score for a member of the house is .88. For these misclassifications, the average score is: .82. This indicates that it is more difficult to classify politicians who have more moderate ideologies.

Table 4: My caption

Name	Ideology Score
Burgess	.829
Meehan	.695
Harris	.906
Roe	.97
Gowdy	.79
Olson	.95
Ross	.89
Pearce	.92
Denham	.71
Yoder	.81
Brady	.82
Poliquin	.63
Kelly	.87
Barton	.77

5 Discussion and Conclusion

5.1 Feature Reduction

One of the most noticeable trends within the feature reduction results is how KNN and decision tree both maintain their predictive power as the feature space shrinks. This tells us that it does not take many features to generate a meaningful distance metric between Congress people. It is not very surprising that Decision Trees worked well with a small feature space. Decision trees already ignore the vast majority of features when they classify. As the depth of the tree increases, it if going to be splitting on unimportant factors. This indicates that there is a relationship between importance of a feature and its rank in terms of how common it is.

Another prominent trend in our data revolves around the different predictive powers of hashtags versus simple text alone. It is logical to think that hashtags serve as the best predictors. They are words that the tweet's author has chosen to summarize the tweet. Inherently, hashtags are usually important content words. If the goal is to make the best predictions in the shortest amount of time, then hashtags are the best option. However, if the goal is to uncover hidden differences in partisan rhetoric, then tweet text are the best features to use.

5.2 Training Size Reduction

The biggest trend in the training size reduction experiment is that decision trees require the smallest amount of training data to learn how to make good predictions. KNN and SVM on the other hand need more training data to ramp up their predictive power. This tells us that the splitting rules that the decision tree generates are powerful enough that they can be quickly detected in a small training set. It also tells us that the nearest neighbor relationships from a small subset of Congress are not very applicable to Congress at large.

5.3 Misclassifications

The fact that the average ideology values of the misclassified republicans is more moderate than the average republican tells us that our model is actually detecting information about partisan ideals and not just finding predictive, but meaningless features. It is true that there are some outliers to this trend, where misclassified congresspeople are either highly conservative or liberal.

Interestingly, it is more common for Republicans to be classified as Democrats, although I do not have a clear explanation for this.

5.4 Future Extensions

In terms of applications of this project, I would like to extend it to predicting the political party of general Twitter users. This could be used by political campaigns to target the general population by political affiliations or preference. However, it unclear if this model would be successful on the general population. The classifiers in this project were trained on highly partisan politician's Tweets. It is unclear if the predictive trends in highly political tweets will generalize to predicting political preference of the general population.

Acknowledgments

Conover, Michael D., Bruno Goncalves, Jacob Ratkiewicz, Alessandro Flammini, and Filippo Menczer. ?Predicting the Political Alignment of Twitter Users.? 2011 IEEE Third Int?l Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third Int?l Conference on Social Computing (2011): n. pag. Web.

"Directory of Representatives." House.gov. N.p., n.d. Web. 10 May 2016. ¡http://www.house.gov/representatives/¿.

"Analysis Methodology." GovTrack.us. N.p., n.d. Web. 10 May 2016. ¡https://www.govtrack.us/about/analysis¿.

"4.2. Feature Extraction." 4.2. Feature Extraction ? Scikit-learn 0.17.1 Documentation. N.p., n.d. Web. 10 May 2016. ¡http://scikit-learn.org/stable/modules/feature_extraction.html¿.