"A Nation Divided": Classifying Presidential Speeches

Ambika Acharya
aacharya

Nicole Crawford
nicolecr

Michael Maduabum
rmaduabu

1 Introduction

Given the polarization of political parties, the variance of political views across states and the changing political climate over time in the United States, we ask how presidential candidates' speeches reflect these changes. Do candidates' speeches match the rhetoric of his or her party? Do candidates tune their speeches dramatically based on where they are delivering it? How much of the political change seen in the US over the past 20 years is reflected in the content of candidates' speeches? Particularly, we defined three tasks. For each task we input cleaned text speeches into logistic regression, SVM and Naive Bayes models, extract features tuned for the task and output a prediction based on the task:

1. **Party:** Democrat vs. Republic (binary classification)
2. **Region speech was delivered in:** South, Coasts, Midwest (Multiclass classification)
3. **Election year speech was given in:** 1996, 2004, 2008, 2012, and 2016. (Multiclass classification)

Additionally, we look at topic clustering across parties, regions and time to observe which words and topics are most predictive of each. We use weights learned from a Naive Bayes model to select phrases most predictive of each party, region and election period. Through these experiments we gained insight into how polarized the two major American parties are, how candidates tailor speeches based on geographic location and how their choice of topics to speak about vary across different cross sections of the American population.

2 Related Work

In their paper "Classifying Party Affiliation from Political Speech," Yu, Kaufmann, and Diermeier analyzed congressional speeches and classified them as being Democrat or Republican. They used standard NLP algorithms such as Naive Bayes and SVM with 5-fold cross validation on both House and Senate speeches, training on House speech and testing on the Senate and vice versa. They also tested these classifiers on with different weighting methods, including term-frequency and document-frequency weighting. They found that the classifiers trained on House speeches performed better on Senate speeches than vice versa, and that the overall best classifier was SVM with equally weighted features. However, Yu et. al. did not test these algorithms on a single dataset: either just House speeches or just Senate speeches. We apply these techniques to our dataset to see accuracy of party prediction based solely on a candidates speech.

In "Lexical Cohesion Analysis of Political Speech," Klebanov, Diermeier, and Beigman studied the lexical cohesion of Margaret Thatchers discourse by designing an algorithm to find the major semantic domains of her speeches. They used a stemmer and part-of-speech tagger, and then used a decision tree to decide whether two words were contextually related. This research was a first step towards being able to analyze whether two people have the same opinion on an issue but phrase it differently or if they simply have differing opinions. We apply these general ideas to our tasks by using classification to identify popular topics in speeches.

In "Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts," Grimmer et. al. discussed different methods for analyzing political texts. They used both supervised and unsupervised methods, specifically k-means clustering, as well as human analysis, to find similar traits in political texts. We will use different classification techniques to analyze our data, but due to the size of our dataset, we will

analyze just the topic clustering results to see important words in speeches, instead of manually analyzing the actual speeches.

Figure 1: Using the probabilities from the NB model, we generate the words most predictive of each region of the United States.

3 Data

Our corpus is a set of presidential campaign speeches from the 1996-2016 elections. This data was obtained from the University of California Santa Barbara Presidential Project dataset. In total, it includes 246 speeches evenly split between the two major parties (129 Democrat and 117 Republican). Each speech includes metadata describing the candidate, party, year and state in which the speech was delivered. Recent elections are more heavily represented; about half of the speeches are from the 2016 and 2012 elections. The breakdown by year can be seen in Table 1.

Year	Number of Speeches
1996	46
2000	3
2004	69
2008	61
2012	35
2016	32

Table 1: Breakdown of Number of Speeches By Election Year

4 Methods

4.1 Models

For the classification tasks we compare the performance of logistic regression and SVM models. For the task of party prediction, we use the binary version of these models. For the multiclass tasks, predicting election year and predicting which region of the country a speech was given in, we used the "One vs. Rest" versions of these models. This method trains a binary classifier for each potential class that distinguishes it from all the other labels. Additionally, we used L2 regularization and 5-fold cross validation to prevent overfitting.

4.1.1 Logistic Regression

Logistic regression is a binary classification algorithm which uses the logistic function as the hypothesis:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

The model then finds the optimal θ which minimizes the associated cost function $J(\theta)$ which will then determine a separating sigmoid curve between the two classes. Additionally, we use L2 regularization to prevent overfitting which adds a $||\theta||_2$ term to the cost function. We use sklearn's optimized implementation of logistic regression and specify the use of L2 regularization.

4.1.2 SVM

We used a Support Vector Machine with L2 regularization and an RBF(gaussian) kernel. SVM separates classes by building a margin such that the distance from each class to that margin is minimized. Specifically we wish to do the following:

$$minimize \frac{1}{n} \sum_{i=1}^{n} C_i + \lambda ||w||^2$$

$$s.t. y_i(w * x_i + b) \geq 1 - C_i, c_i \geq 0 \text{ for all } i$$

4.1.3 Naive Bayes

The NB model learns probabilities based on the prior distribution across classes from the training data under the assumption that all features are independent. Specifically, when predicting a class based on training features:

$$p(y = 1|x) = \frac{\prod_{i=1}^{n} p(x_i|y = 1)p(y = 1)}{\prod_{i=1}^{n} p(x_i|y = 1)p(y = 1) + \prod_{i=1}^{n} p(x_i|y = 0)p(y = 0)}$$

For topic clustering, we used the log probabilities from the Naive Bayes model to find words which were highly associated with each party. To do this, we find the top words k which maximize the log probabilities: $|\phi_{k|y=1} - \phi_{k|y=0}|$for Democrats, and $|\phi_{k|y=0} - \phi_{k|y=1}|$ for Republicans. We repeat this for regions and election years doing pair-wise comparisons of these probabilities across the classes.

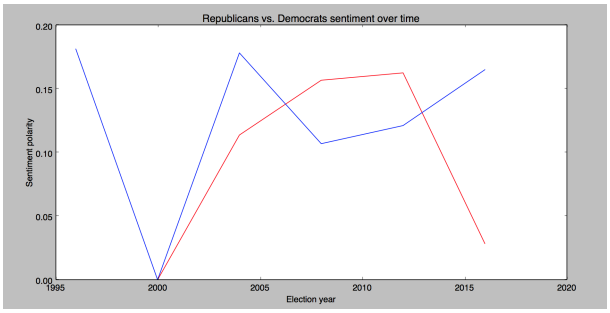

Figure 2: To see whether or not sentiment is an indicator of party or time, we plot the sentiment of both parties across time. We see that both parties fluctuate across time and compared to one and other, indicating sentiment may not be the best predictive feature.

4.2 Features

For all tasks we started by using unigrams (Bag-of-Words) as our feature vectors. We use a unigram model to create a feature vector the size of the vocabulary for each speech. A row in the feature vector indicates the count of the corresponding word in that speech. We generate the vocabulary from all of the speeches we train on and use Laplace smoothing to account for new words we see in the test set. Additionally, we remove stop words using the NLTK corpus (Bird, 2009), only use the stems of words using NLTKs Porter-Stemmer, remove mentions of the names of presidential candidates, and strip out punctuation. In addition to the counts of words, we used TF-IDF as a weighting scheme for the unigram feature vector. The TF-IDF weighting schema uses the content of all the documents to give weight to a word depending on its prevalence and use across all documents in the data. TF-IDF(term frequency-inverse document frequency) is defined as followed:

$$f_{t,d} * log(1 + \frac{N}{n_t})$$

Where $f_{t,d}$ is the frequency of the word in the current speech, N is the total number of documents in the training set and n_t is the number of documents in the training set the word is found in. Additionally, we extracted bigrams and evaluated the models' performance using the combination of these two features.

As our results show in Section 5, party prediction and election-year prediction were relatively easy tasks since using just unigrams and bigrams as features resulted in very high performance (greater than 90% F-1 score.) Therefore we focused on doing feature engineering for the region classification task.

4.2.1 Region Classification

Based on data analysis looking at trends in speeches across different regions of the United States, we generate features. We added binary features to this model such as the election-year the speech was given during, the party of the candidate, whether the state is a swing state, region-specific features and sentiment features. For the region specific features we check for the number of times a speech contains a word from a region-curated set, since we presume that candidates try to tune their speeches to reflect the issues people in the specific region care about. For example in the midwest we used these keywords that we determined to be significant by language modeling: {manufacturing, oil, industry, machinery, car} For sentiment features, we add a feature denoting the difference of the number of positive words and negative words in a speech, as determined by an NLTK corpus. We supplemented the unigram and bigram models with these new features and evaluated performance. For this task we includes several other features and combinations to improve accuracy. The most useful were the length of a speech and information about whether or not the speech was given in a swing state.

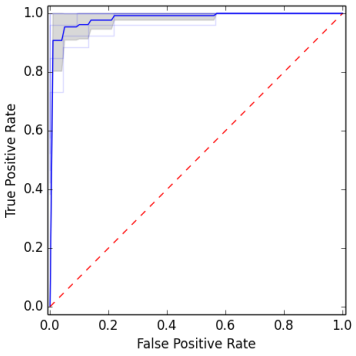

Figure 3: This is an ROC curve for the party prediction task, which we performed very highly on and got an AUC=0.98.

4.2.2 Word2Vec

As an alternate set of features, we used Google's open-source word embeddings project Word2Vec and evaluated the performance of these features against the previous set of features on the region classification task. The Word2Vec model is trained on all of the training data such that for each word in a training speech we can use Word2Vec to

transform that word into a vector space and represent that word as a 100-dimensional vector.

$$together \rightarrow [0.256, .01, .7035...0.214]$$

We then multiply this vector by the word's tf-idf weight as described in the previous section and then average the vectors of all words in the training speech which results in a feature vector of length 100. We then input this feature vector as the representation for a given speech into the classification models.

5 Results

5.1 Party Prediction

Classifier	Accuracy	Prec	Recall	F-1
Log. Reg.	95% ± 4%	0.96	0.96	0.96
SVM	85% ± 15%	0.87	0.82	0.82
NB	69% ± 6%	0.74	0.63	0.58

Table 2: ***Preliminary Results:*** *Accuracy, precision, recall, and F-1 score for each classifier. Results obtained using 5-fold cross-validation.*

Democrat	GOP
crime	cheers
welfare	appreciate
deficit	boo-o-o
tuition	September
wealthiest	lawsuits

Table 3: ***Key Party-indicator Words:*** *5 words most strongly associated with speeches given by each party.*

Our preliminary results were highly successful at the binary classification task. The SVM classifier was had the most variance, giving accuracies ranging from 70% to 90% and the Naive Bayes classifier performed significantly worse than the others.

5.2 Election Year Prediction

Feature	Log. Reg.	SVM
Unigrams	0.88	0.7
Bigrams	0.93	0.66

Table 4: F-1 scores for Election Year prediction

5.3 Region Prediction

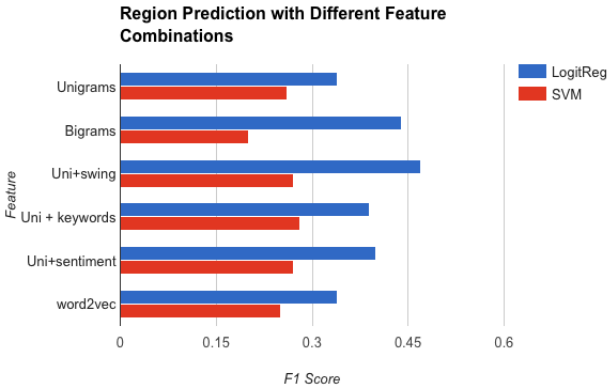

Figure 4: F-1 scores for region classification with different features

6 Analysis

6.1 Party Prediction

In general, these results in Table 2 indicate that classifying a presidential candidate's speech as Democratic or Republican is a relatively straightforward task. Logistic regression has the highest accuracy of our three methods, which shows that Democratic and Republican presidential discourse has very different features and thus can be separated linearly. From the ROC curve in Figure 3 we see how well this model performs, showing how different the rhetoric of the two parties is. One reason Naive Bayes may not do as well as SVM or logistic regression is because it assumes conditional independence; however, certain words are more or less likely to appear in a speech if another word has already been used. For example, if a candidate has used the word crime, the probability of welfare (given that the candidate is a Democrat) increases.

Table 3 shows the words most strongly correlated with each party. Our results indicate that Democratic presidential candidates tend to address safety ("crime") and economics ("welfare", "tuition", "wealthiest"), most likely speaking to people focused on rising prices and affordability of both college tuition and everyday items. On the Republican side, candidates are more likely to discuss terrorism ("saddam", "hussein", "september", "11^{th}"), which suggests Republican presidential candidates focus more on quelling the fear of being attacked.

6.2 Election Year Prediction

Similar to predicting political party, predicting the election year was a relatively straightforward task as gleaned from Table 4 However in this instance, logistic regression performed much better than our SVM classifier. It's possible that our data is linearly separable, and because we used a Gaussian kernel, which is not linear, our SVM classifier was not as robust.

Overall, logistic regression did very well with few features because the topics within different election years varied greatly. We managed to consistently get over 90% accuracy on a 5-class classification task (baseline of 20%). When looking at the words from these speeches distinguished by year, unsurprisingly, the words that distinguish a speech from one year versus another typically come from issues occurring during that election cycle, like "emails" in 2016.

6.3 Region Prediction

For topic clustering in Figure 1, we find words that we expect to be heavily associated with industries in the regions to be those most predictive of region. This indicates candidates' focus on the industries specific to the state they're speaking in. In terms of classification, this task was much more difficult than the first two tasks we attempted, and our results as seen in Figure 4, even with different combinations of features, were much less accurate. Similarly to the task of predicting the election year, logistic regression performed better than SVM with RBF kernel, meaning that the data is linearly separable.

While we tried many different combinations of features for this task, our best performance was using unigrams and whether or not the speech was given in a swing state.

An interesting result from our topic clustering by region was that the Midwest is very distinctive from the coasts; the most unique word in speeches given in these states was "manufacturing." However, when we built a feature using this information, our classifier did not perform much better than just using unigrams.

When analyzing the sentiment of parties over time in Figure 2, we saw a few fluctuations indicating that sentiment might be influential in prediction tasks. However as seen in the results for region prediction, adding sentiment was not helpful. Additionally when using word2vec as a feature vector, we saw very low performance for both logistic regression and SVM. We suspect that the low performance across combinations of features is due to the region task being especially difficult since speeches don't vary too much from region to region. Even though a candidate might mention "cars" more in a speech in the midwest, he or she will still bring up these issues in speeches given in other regions.

Running feature selection to find the most important features indicated that the use of bigrams, unigrams and swing state was most useful suggesting we use a different model for this task. We believe that training a neural network could be especially useful for separating the three classes in this task.

7 Conclusion and Future Work

We used classification algorithms (logistic regression, SVM, and for the first task, Naive Bayes) to predict political party, region, and year of a given presidential candidate speech. Predicting political party and year were relatively easy and straightforward tasks, while predicting the location of a speech proved much more difficult. This may be because certain issues are only discussed in specific election years and Democratic and Republican candidates focus on specific topics during their speeches, making those tasks simpler to predict. Because predicting the region a speech was delivered in is so challenging, we suspect that candidates do not tailor their speeches as dramatically from one geographic region to another as we hypothesized.

Moving forward, we would expand our data sources to include other forms of presidential discourse, such as debates, interviews, and town halls. Additionally, our current dataset only has speeches from the past six elections. Gathering more speeches from older elections could give us better and more interesting results for how presidential candidates' discourse has changed over time. It's possible that the difference between two parties was not as stark as it is now, or that region prediction could be an easier task in previous election years due to a stark contrast between speeches given in different states. Given the importance of political discourse in our nation, gleaning information from textual political documents can give us great insight into how the country's government is changing.

References

Bird, Steven, Ewan Klein, and Edward Lope 2009. *Natural Language Processing with Python*. O'Reilly Media.

Gfu, Daniela, and Dan Cristea. *Computational techniques in political language processing.* 2011 Future Information Technology. Springer Berlin Heidelberg. 188-195.

Grimmer, Justin, and Brandon M. Stewart. *Text as data: The promise and pitfalls of automatic content analysis methods for political texts.* 2013 Political Analysis: mps028.

Klebanov, Beata Beigman, Daniel Diermeier, and Eyal Beigman. 2008 *Lexical cohesion analysis of political speech*. Political Analysis 16.4 (2008): 447-463.

Mayoraz, Eddy, and Ethem Alpaydin. 1999. *Support Vector Machines for Multi-class Classification*.

Mejova, Yelena, Padmini Srinivasan, and Bob Boynton. 2013 *GOP primary season on twitter: popular political sentiment in social media.* Proceedings of the sixth ACM international conference on Web search and data mining. ACM

Sim, Yanchuan, et al. 2013. *Measuring ideological proportions in political* speeches."

Yu, Bei, Stefan Kaufmann, and Daniel Diermeier. 2008. *Classifying party affiliation from political speech*. Journal of Information Technology & Politics 5.1 (2008): 33-48.

Link to code: https://github.com/aacharya14/bridgegate