Natural Language Processing and Machine Learning in American Politics

Aaron Kerns, Nick Kelly

Principles of Data Science
Professor Vucetic
7 May 2017

Introduction

The health of democracy in America depends on how well ideas are being thought out, publicly debated, and vigorously tested. In order to hold those in power accountable to the citizens, we have freedoms of speech and press, allowing us to uncover and publish details of what our government is doing without fear of repercussion. Thanks to these freedoms, there are massive stockpiles of data, going back to the early days of our republic, containing many details about exactly how each president spent their time in office. The American Presidency Project is a non-profit organization that has many different documents for each president; including their campaign speeches, executive orders, and even appearances in public. This accumulated data inspired our team to apply data science techniques to the office of the president, in an attempt to more closely understand how presidents behaved and how they compare to each other. Over the course of this project, we utilized natural language, machine learning, and even probability theory to dissect, pull meaning from, and create imitation data, all to attempt to understand the similarities and differences between our public officials.

Approach

For our project we wanted to see how we could analyze various corpora of US politicians to yield valuable information. We pondered different options for analysis; from Markov chain sentence generation, to authorship attribution. The ensemble of techniques would allow us to get an understanding of the speech and writing patterns for each politician.

We began by scraping the Presidency Project site with the Requests library in python to get the page that contains the presidential candidates. We then used this page with BeautifulSoup to pull the table of links pointing to each candidate's speeches from the page, then proceeded to build a dictionary of candidate names to their respective links. We iterated through the dictionary and scraped the content of each page into a corpus for each candidate. We ended up with a function, that when passed the url of a given election year, downloads all speeches from each of the presidential candidates in that given year into separate text files.

After we scraped the speeches from the pages, we preprocessed them so we could begin data analysis. We looked through the corpora for each candidate to find any text inconsistencies (i.e. utf-8 inconsistencies) and format text (i.e. interview names), then removed them with a regex preprocessing script we developed.

After scraping all of the campaign speeches for the most recent election, we became interested in how each candidate uses language as a way to differentiate the candidates from each other. When we first looked at the speeches in their raw text files, we noticed that most of the candidates started their speeches the same way, and many of them followed observable patterns in the way they talk. Donald Trump particularly interested us because we knew him to speak english in a very simple and consistent way. His repeated word use and excessive

superlatives made us think that we could use machine learning techniques to emulate the way he and other candidates speak.

We used a python library called Markovify to create a markov chain for a given corpus of text, which involves building a probabilistic distribution for each word in the corpus. It could then generate sentences by picking a random word to start on and then picking the next word using the probabilities established during the training. We used these chains to generate countless sentences in the style of presidential candidates, many of which sounded like they could actually be spoken by the actual person!

We found a library for this purpose called MarkovBot, which contained both a Twitter interface and a markov chain maker. Since we already had our markov chains generated, we reworked the MarkovBot library to use Markovify. We registered new Twitter accounts for the three popular candidates in the 2016 election: Bernie Sanders, Donald Trump, and Hillary Clinton. In order to interface our python application with Twitter, we had to make a Twitter app for each account on Twitter's application interface. We then used the modified MarkovBot to log in to Twitter and make a randomly generated tweet every 10 minutes. After around a week of using this tactic, we made the decision to move to markov chains trained on the candidate's' Twitter usage rather than their campaign speeches. Using the tweepy python library, we were able to interface with the Twitter API and scrape our desired user's' tweets 200 at a time. We stored the tweets in a text file, similar to how we stored the campaign speeches. Now that we had finished scraping our desired sources, we were ready to further analyze our data.

Results

We generated hundreds of markov chains over the course of this project, which revealed interesting patterns about many of the candidates. In particular, the markov chains generated from the campaign speeches gave interesting insight into what each candidate talks about most. Below are some of the sentences that our markov chains generated:

Donald Trump:

> “There is tremendous potential, folks, in the first 100 days.”
> “Now she is cashing in.”
> “The reason Hillary Clinton handled Haiti the last time you'll get that apology right around the world, we have the money.”
> “Well I have been left behind.”

Bernie Sanders:

> “We need to be made to the top one-tenth of 1 percent who have worked with him.”
> “And the truth is that we can make this into a drug store to refill your prescription, the price you pay for it.”
> “I'd just add one little thing. It is a working-class woman or a Catholic.”

Marco Rubio:

"I am our nominee, we are going to be done, the 21st century, too many leaders that are being made."

"They're the person trying to start a career, a business or a nuclear-armed Iran? And I will be another American century."

Ted Cruz:

"Freedom means the right to keep America safe."

"It is the power of the world is allowed so many others have had their lives because of their Egyptian citizenship."

Barack Obama:

"I will never be equal."

"Joblessness rose more last month he was gone, and my sister and me reach a little further than some in the global energy crisis."

"And you'll have the debate."

These sample sentences help show the generalizations that get revealed about each candidate speech patterns. Bernie can be seen expressing his concern about the wealth concentration of the top one-tenth of 1 percent. Many of his generated tweets talk about his 'political revolution' and his average campaign contribution of $27. The markov chains reveal that Donald Trump loves exaggeratory words like 'tremendous' and 'huge'. Almost any time he uses 'Hillary' or 'she' he ends up launching into a rant about how corrupt she is or how bad her judgement is. Republican candidates like Jeb Bush, Rand Paul, and Rick Santorum always seem to mention Obama's Iran deal, Isis and terrorism, and how burdensome regulations are, as would be expected from the Republican candidates.

The Twitter bots that we created using these markov-chain-generated sentences ended up getting some human interaction, as well as some followers. On the next page are the interaction statistics provided by Twitter that each of our bot accounts got during this project. We were not able to keep the bot running constantly, but the bumps in tweets indicate when the bots were active.

Hillary Clinton

Donald Trump

Bernie Sanders

The sentence structure and usage of our bots were very primitive and did not include much logic past the probabilistic functionality, so we began exploring ideas into developing a more sophisticated AI. This led us into working with the Natural Language Toolkit (NLTK) library. There were two main objectives for our work with NLTK: feature extraction and classification. Our motivation started with a blog about authorship attribution which asked the question, "*Can the tools and techniques from [Building Machine Learning Systems with Python] be used to identify who wrote each chapter?"* (Authorship Attribution, Yager). We applied this question to our corpora of speeches and tweets and rephrased the question to be, "*Can the vocabulary and style of politicians be used to identify who wrote a piece of text?*".

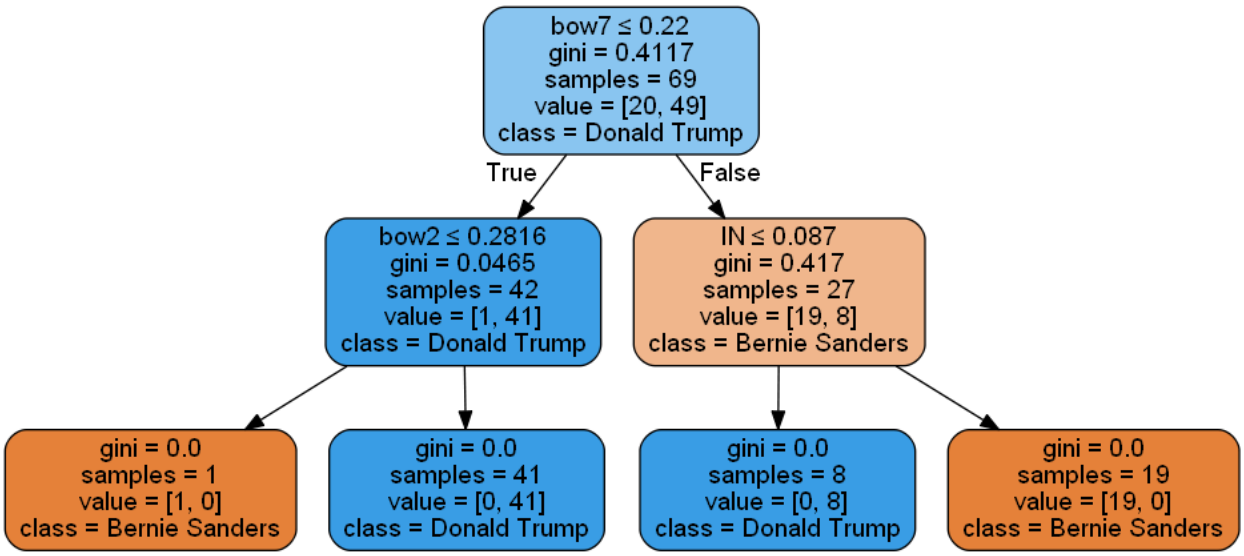
In order to answer our question, we had to examine what characteristics of speech could be used to identify the author. The blog laid out a set of features from various scholarly resources that aid in the analysis of speech patterns. There are four categories of features: lexical, punctuation, bag of words, and syntactic. Each category contains a set of features that describe it. Lexical features pertain to vocabulary, e.g. sentence length and vocabulary diversity; punctuation pertaining to the frequency of important punctuation in sentences; bag of words pertaining to the analysis of commonly used words; and syntactic pertaining to the patterns of syntax used.

We extracted features of corpora from our two sources: Twitter accounts and campaign speeches. We found eccentricities with each source because of the nature of their vernaculars. For example, Twitter fosters a more concise pattern of speech due to the character limit, where campaign speeches allow the speaker to express articulated ideas and are more representative of their true speech patterns.

Twitter users may use a unique and concise pattern of speech; however, there is a lot to learn from analyzing tweets. Each individual user has their own style for producing tweets; so we created feature vectors using the aforementioned feature set to describe the styles of our politicians in a quantitative way. In order to build them, we created a table to store the tweets with their features as a vector. The tweets were assembled in sets of 10 to provide a stronger context for evaluating the features.

We then applied the same methodology to building feature vectors for US presidential campaign speeches. The data was vectorized in the same fashion as the tweets, but with each row being a speech rather than a set of tweets.

After establishing the vectorized corpora, we wanted to find the most efficacious group of features. We used six different scikit-learn classifiers (KNeighbors, DecisionTreeClassifier, RandomForestClassifier, MLPClassifier, AdaBoostClassifier, and GaussianNB) to test our feature sets and then returned the classifier's accuracy for each group of features. This yielded different results for the corpora which gave us insight into which group of features reveals more information about each corpus. For example, the Decision Tree Classifier is very informative to the usefulness of certain vectors when visualized, so we created a graphical form that can be seen below.



This decision tree was trained with candidates' speeches and was able to split most of the speeches simply based on the frequency of the word 'this'. This further validates the effectiveness of the bag of word vector model we incorporated into our overall vector representation of text. The split on the right manages to cleanly separate the two politicians speeches by asserting that Bernie's speeches at that node have fewer prepositions than Trump's speeches, demonstrating the value of our syntactic vectors.

More details of the feature analysis are shown on page of the attached appendix. It shows the accuracies of each feature set for each classifier we used. These classifier accuracies reveal the importance of each feature set in the context of our corpora and allow us to analyze how good of an indicator each one is for classifying text.
If the classifier accuracy logic is used, punctuation does not seem to be a very clear indicator for classification when analyzing campaign speeches. We found this to be the case because the speeches are transcribed by a third party, using punctuation as they see fit. Thus, differentiating between candidates punctuation habits in speeches is difficult. However, punctuation is a very clear indicator for the candidates' tweets. Which would indicate that they have distinct punctuation usage patterns when generating tweets. This analysis pattern allows us to determine which feature sets are working for classification; which, in turn, allows us to pick the most reliable feature sets to train our classifiers.

In order to truly test our classifiers, we decided to plot them on a Euclidean plane. We performed dimensionality reduction using Principal Component Analysis (PCA) and plotted the most correlative dimensions on our graph. The results we found were conclusive in that the classifiers were able to accurately predict the classes for both Twitter data and campaign speech data; although the Twitter data yielded a higher accuracy than the campaign speeches. The images of these plots are shown on pages 3 and 4 of the attached appendix with their accuracies listed in the bottom right corner of the plot.

Conclusion

At the beginning of this project, we aimed to provide clarity to the massive amount of data that has been collected about US presidents and candidates. We worked toward that goal and through the use of natural language processing, machine learning, and exploratory data analysis, we managed to shed some light on the characteristics of recent presidents as well as presidential candidates.

We created 3 twitter bots that can regularly generate semi-coherent and relevant tweets using markov chains. Using nltk we constructed a vectorizer that could reveal differentiating details about candidates that casual observers would not typically pick up on. We ended up with a classifier that when fed tweets was able to predict the politician that authored the tweets with high accuracy. We even created a classifier trained on campaign speech data that can identify the candidate who gave a given speech.

There were several other data analysis methods that we pursued during the course of this project that we had to abandon so we could bring the project's scope under control. In our quest to uncover meaning from the massive amounts of data available, we occasionally found ourselves lost in the vastness of the data. It was only through trial and error that we were able reveal and utilize the different characteristics of American politicians.

Works Cited

"A Beginner's Guide to Collecting Twitter Data (and a Bit of Web Scraping)." *Knight Lab*. N.p., 30 July 2015. Web. 08 May 2017.

"Tutorial: Creating a Twitterbot." *PyGaze*. N.p., n.d. Web. 08 May 2017.

Yager, Neil. "Authorship Attribution with Python." *AICBT*. N.p., n.d. Web. 08 May 2017.

Appendix

Graphical Representations of Natural Language Classification

Feature Set Accuracies per Classifier

Twitter

```
kNN :
Lexical: 0.92 (+/- 0.00)
Punctuation: 1.00 (+/- 0.00)
Bag of Words: 0.95 (+/- 0.00)
Syntactic: 0.95 (+/- 0.00)

Decision Tree :
Lexical: 0.85 (+/- 0.00)
Punctuation: 1.00 (+/- 0.00)
Bag of Words: 0.84 (+/- 0.00)
Syntactic: 0.94 (+/- 0.00)

Random Forest :
Lexical: 0.93 (+/- 0.00)
Punctuation: 0.99 (+/- 0.00)
Bag of Words: 0.85 (+/- 0.00)
Syntactic: 0.97 (+/- 0.00)

MLP :
Lexical: 0.48 (+/- 0.00)
Punctuation: 1.00 (+/- 0.00)
Bag of Words: 0.85 (+/- 0.00)
Syntactic: 0.96 (+/- 0.00)

AdaBoost :
Lexical: 0.96 (+/- 0.00)
Punctuation: 1.00 (+/- 0.00)
Bag of Words: 0.85 (+/- 0.00)
Syntactic: 0.96 (+/- 0.00)

Gaussian :
Lexical: 0.90 (+/- 0.00)
Punctuation: 1.00 (+/- 0.00)
Bag of Words: 0.86 (+/- 0.00)
Syntactic: 0.98 (+/- 0.00)
```

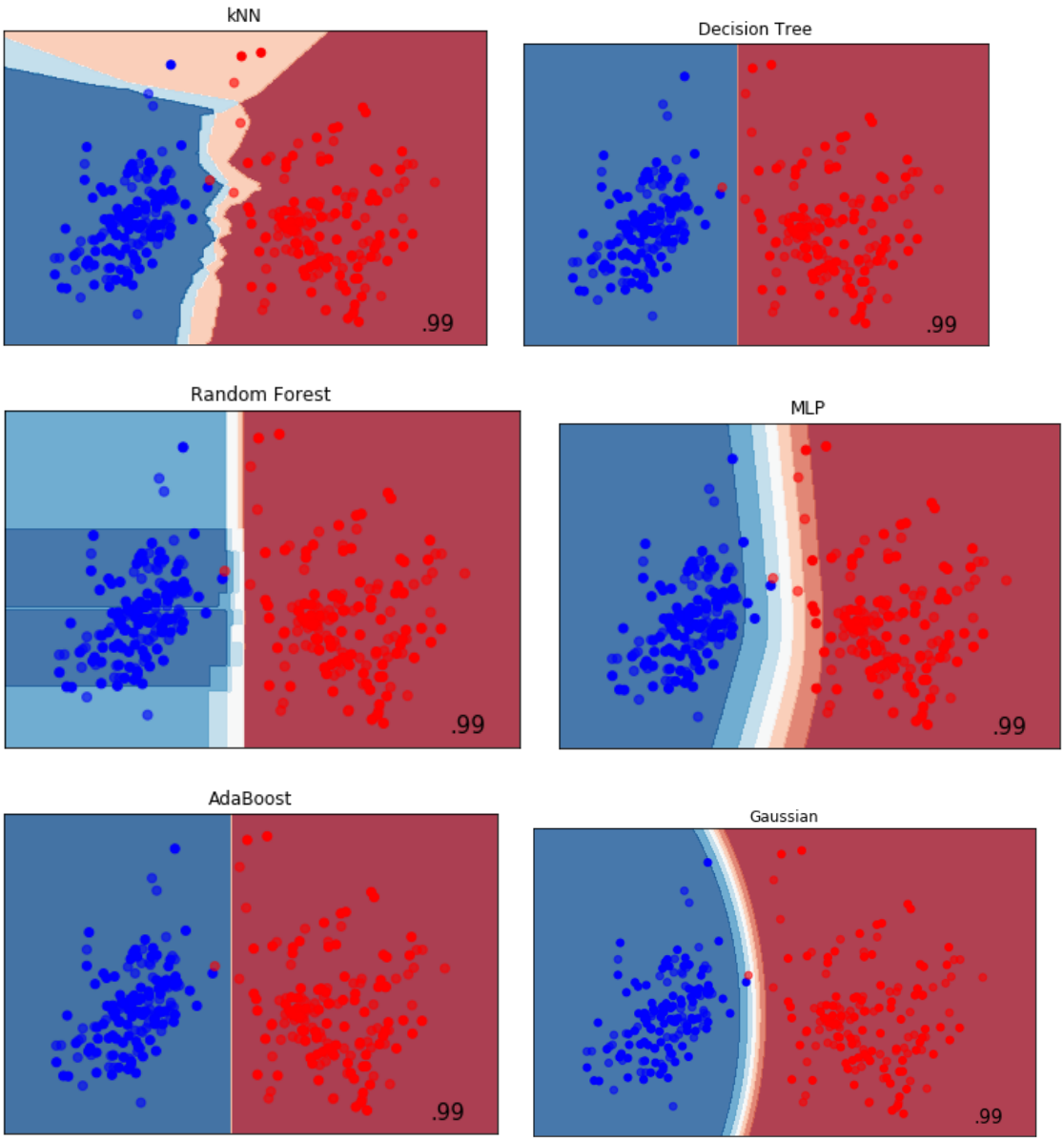
Speeches

```
kNN :
Lexical: 0.86 (+/- 0.00)
Punctuation: 0.79 (+/- 0.00)
Bag of Words: 0.86 (+/- 0.00)
Syntactic: 0.93 (+/- 0.00)

Decision Tree :
Lexical: 0.83 (+/- 0.00)
Punctuation: 0.71 (+/- 0.00)
Bag of Words: 0.76 (+/- 0.00)
Syntactic: 0.83 (+/- 0.00)

Random Forest :
Lexical: 0.88 (+/- 0.00)
Punctuation: 0.62 (+/- 0.00)
Bag of Words: 0.83 (+/- 0.00)
Syntactic: 0.90 (+/- 0.00)

MLP :
Lexical: 0.83 (+/- 0.00)
Punctuation: 0.81 (+/- 0.00)
Bag of Words: 0.93 (+/- 0.00)
Syntactic: 0.79 (+/- 0.00)

AdaBoost :
Lexical: 0.83 (+/- 0.00)
Punctuation: 0.74 (+/- 0.00)
Bag of Words: 0.93 (+/- 0.00)
Syntactic: 0.86 (+/- 0.00)

Gaussian :
Lexical: 0.86 (+/- 0.00)
Punctuation: 0.79 (+/- 0.00)
Bag of Words: 0.88 (+/- 0.00)
Syntactic: 0.93 (+/- 0.00)
```

Clustering Algorithms

Bernie Sanders and Donald Trump

Twitter:



Clustering Algorithms cont.

Campaign Speeches: