**Presidential Data Analysis Progress Report I**

We began this week by rewriting our proposal with the focus on scraping historic presidential data. We found a website called The American Presidency project which has an archive of all the campaign speeches for each candidate for the past several elections. We began the web scraping by using the Requests library in python to get the page that contains all of the 2016 presidential candidates from both political parties. We then used this information with Beautiful Soup to pull the table of candidates off the page and iterate through each candidate. We used this data to build a dictionary of candidate names and the links to their campaign speeches page. Again using Beautiful Soup and Requests, we were able to get the content of each candidate’s page. We then pulled the text of the campaign speeches out of each link on each candidate’s page and put their speeches into a text file organized by election year.

We ran into a few problems when we were scraping speech data from the website, some of which we fixed, others still needing to be fixed. One of the major problems was that it takes a lot of time to visit each page and retrieve the data for the various speeches which makes testing very inefficient. We had to find a way to store the speech data in a way which makes it easier to develop logic for our program to not continue scraping data it had already retrieved.

Now that we had all of the most recent candidate’s speeches in a folder ready for analysis, we were able to start using the Markovify library in python to train a Markov chain for each candidate on their respective campaign speeches. We used these Markov chains to generate short sentences from each candidate using the style and vocabulary of their campaign speeches. These Markov chains often made impressively coherent sentences, but also made a lot of completely nonsense sentences. The Donald Trump Markov chain was by far the most coherent one, making statements that seem like they could be actual tweets from the man himself such as, “There is tremendous potential, folks, in the first 100 days”. We think the Trump Markov chain works so well because he has a very simple sentence structure containing many catch phrases that he uses all the time and can be used interchangeably. We plan on extending the Markovify library that we used to pay attention to sentence structure a little better, and we hope that this will give us more insight into how each candidate structures their sentences.

We found a library called word2vec which uses Google’s TensorFlow machine learning library as well as a corpus of text to build a skip-gram model of the text, which looks at the context that each word appears in and models the similarity of each word in context to other words in similar contexts. This model took a significant amount of time to train using an i7 and an Nvidia Geforce 940MX. Once the model was complete, we used the TSNE function from Scikit learn to visually represent the distance between words. TSNE broke the data down into 3 principle components using PCA and graphed the Euclidian distances between words in a 2d space. The resulting graph is very interesting to look at as it does a nice job of clustering similar meaning words together without ever having known the definitions of any of them. We are not finished with this part of the project yet, and it shows promise to reveal a lot of latent information in the way each candidate uses the english language.

With the progress we have made, there is still a lot more work to be done. The EDA that we completed using Markov chains was a good start to learning more about the presidential speech data. We plan to use this analysis, in conjunction with more EDA techniques to provide a complete analysis of our data set. In the coming weeks we will be furthering our research of this data and finding out different ways to perform EDA on it.

Along with our EDA, we will be training a classifier to predict some aspect of our data. The classifier and what we will be classifying is not clear yet, but with further research into the data we will begin to get ideas about what can be predicted. Our idea right now is to train classifiers for various data sets that could somehow predict the presidential elections. For instance, which newspapers endorsed each candidate, or how many speeches they made. This data would be put into a data frame for each election year and the newly trained classifier could attempt to predict the winner of the election. The small sample size of candidates for each election would make it difficult to make an accurate predictor, but the number of elections we have data on could make it fairly accurate.