Political Tweet Analysis: Trump Tweet Generator and Political Party Classification

Albert Tan (A12997595) and Lucas Lin (A12063086)
Electrical and Computer Engineering, University of California, San Diego, CA
(Dated: December 16, 2019)

Twitter continues to play a large influence in our social-media driven culture and the open abundance of tweets make it a great tool for data analysis and modeling. Our project uses tweets from politicians to explore two different areas of machine learning. For the first task, we created a fake-trump tweet generator. Markov Chains were used to determine the pattern of existing Trump tweets for generation of new tweets. For the second task we classified whether a tweet was made by a Liberal or a Conservative. We used a logistic regression model to perform binary classification based on thousands of labeled tweets from politicians.

INTRODUCTION

Whether or not President Donald Trump gets reelected in 2020, one of his legacies will be his use of social media in office. He is known for being very active on twitter, posting his unfiltered opinions on politics, celebrities, and news. For example, most recently, he has gotten attention for mocking Greta Thunberg, a 16 year old environmental activist.[1] Given the wealth of President Trump's tweet data, we thought it would be interesting to analyze and train a text generator to create fake tweets that resemble what the *@realDonaldTrump* could have posted.

The second part of our project addresses the tweet tendencies of U.S. politicians as a whole. Many people believe that the division between the two parties provides a major obstacle to improving the government. Sometimes it can be easy to determine the political leanings of a person based on a single tweet, other times it is not as obvious. We would like to see if the party polarization is strong enough for a logistic regression model to classify with high accuracy.

TRUMP TWEET GENERATOR

Dataset

The first dataset consists solely of Donald Trumps tweets sent from his iPhone. We believe that limiting the tweets to those from his phone and non-retweets would give a better illustration of Trump himself. The 13,716 tweets were taken from the Trump Twitter Archive [2]

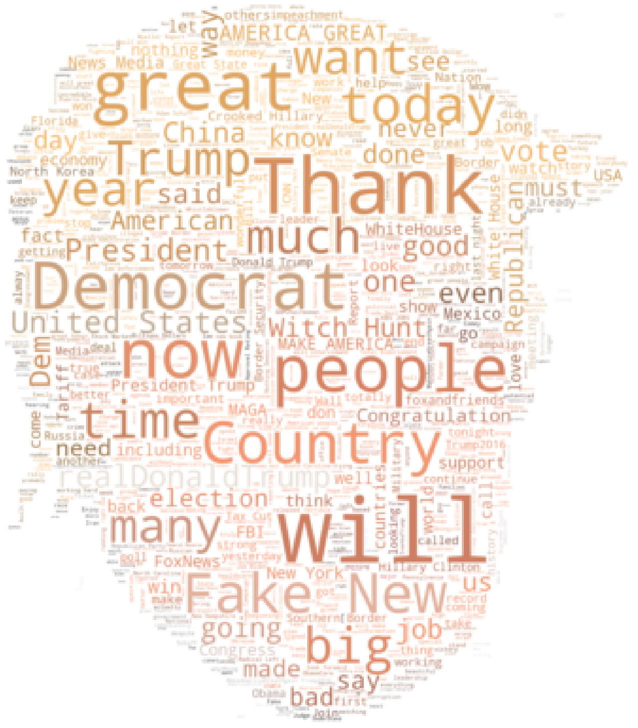


FIG. 1. Word frequencies of Trump's tweets presented in a word cloud (common words omitted).

and include a timestamp, number of retweets, and number of favorites for each post. The tweets date back to December 2012 and continue up to December 2019. His top favorited tweets overlap with his most retweeted tweets. The list below shows his three most popular posts based on retweets.

1. *Posted on 2017-07-02 (369,530 retweets):* **#FraudNewsCNN #FNN https://t.co/WYUnHjjUjg**

2. *Posted on 2019-08-02 (251,530 retweets):* **A$AP Rocky released from prison and on his way home to the United States from Sweden. It was a Rocky Week, get home ASAP A$AP!**

3. *Posted on 2019-07-19 (210,186 retweets):* **Just spoke to @KanyeWest about his friend A$AP Rocky's incarceration. I will be calling the very talented Prime Minister of Sweden to see what we can do about helping**

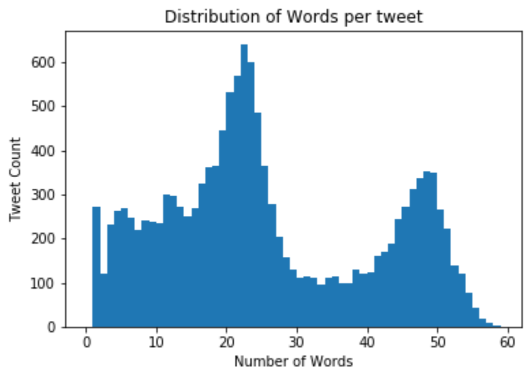


FIG. 2. Distribution of the number of words per tweet can be represented as a mixture of gaussians.

A$AP Rocky. So many people would like to see this quickly resolved!

Trump's top tweets are dominated by an non-partisan event occurring during the summer of 2019, in which President Trump came to aid the release of the American musician A$AP Rocky from Swedish prison.=

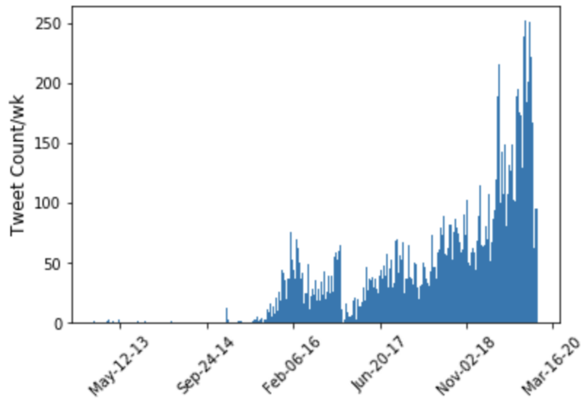


FIG. 3. There has been an increase in tweets over time. This may be from increased activity or simply increased use of iPhone/deletion of past tweets.

Preprocessing

Each tweet was separated into words and symbols using the natural language toolkit. A nested dictionary was then used to record the distinct words and a corresponding dictionary the count of different words that follow. If the word happens to be at the end of a tweet, the next word is "eot". Additionally, the first word of each tweet is added to a separate dictionary as a starting point for the generator.

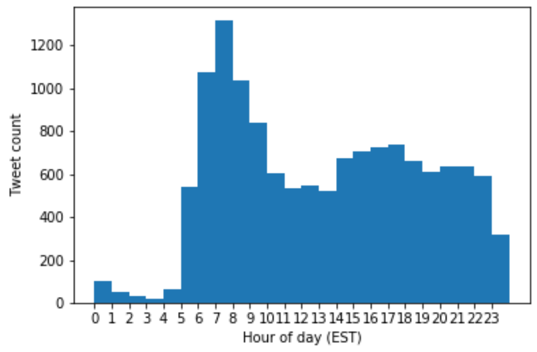


FIG. 4. Tweet distribution over time. Spike in tweets around 7am. Dormancy between midnight and 5am could indicate sleep patterns.

Model and Methodology

We used a Markov Chain model to generate Fake Trump tweets. This is a memoryless model because the next word only depends on the current word and is independent of all other words before it. The model starts by randomly picking a word from the dictionary of starting points with probabilities based on the number of times the word has been used to start a tweet. The start word becomes the current word. The dictionary of the current word is then used to determine which possible terms follow. The next word is picked randomly from this dictionary with probabilities weighted by the number of times the next word has followed the current word. The process is then repeated until 'eot' is chosen, which signals the end of tweet. This string is then measured to stay under the 255 character limit.

Results

From the fig 3, we can see that Trump's tweet output has been increasing since the start of his presidency at the beginning of 2017. Much of the recent spike can be attributed to posts regarding impeachment. Additionally information can be gathered from the time of day that Trump sends his tweets (fig 4). He is most likely to post in the morning around 7am. Conversely, he is very unlikely to tweet between midnight and 5am, indicating his possible sleeping patterns.

We used word count and character count to compare the model to tweet data. With regards to the character count, both followed a roughly uniform distribution. The word count of the training data resembles a mixture of at least three Gaussians with two peaks around 22 and 49 words. We suspected that a good model would follow the same distribution. However, over 10,000 generated tweets, the word count distribution of the Markov Chain model decays exponentially. There was a high representation of single word tweets consisting mostly of links to images or videos. This may be influenced by the tweet initialization.

The generated text correctly followed the common tweet syntax of message followed by hashtags (optional) and link (optional). However, the messages were often grammatically incorrect. A common error of parenthesis usage is illustrated in the second text below. Often the model will use only one parenthesis rather than a pair. This is due to the nature of Markov Chains being memoryless. The model does not remember whether or not an open parenthesis has been used earlier. A sample of the generated tweets are presented below:

1. **Just like to be funding ... This is acting swiftly on @FoxNews weekend — Hillary has gone!**
2. **And the likes to come out of consequence! #maga #AmericaFirst #RNCinCLE https://t.co/bdox6JcrAp**
3. **Deepest sympathies and studying, while ignoring Bidens, I'm 100%!**

PARTY CLASSIFICATION

Dataset

Building off of our twitter driven statistical modeling, our next goal was to perform binary classification for two classes: Democrat and Republican. That is, given an arbitrary tweet, our model aimed to generalize from what it learned to determine whether an arbitrary tweet was made by a conservative or liberal. With the help of Kaggle, we were able to find a .csv dataset that comprised close to 50,000 labeled tweets for each of those two classes.

A key feature of our dataset was it's extremely high dimensionality. Unlike computer vision tasks which simply involve lower dimensional binary images of 0's and 1's, NLP is a domain that dabbles with arguably more challenging problems, in particular needing to perform sentiment analysis of the data. In addition, textual data is known to be highly sparse.

Preprocessing

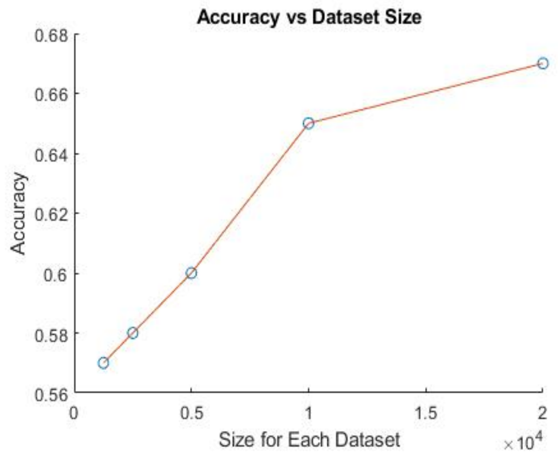
We then proceeded to use Pandas for data cleaning, filtering of unwanted characters, and structuring into a .txt format that was compatible with our model. Because our dataset contained many unreadable characters, we encoded all the text data to a UTF-8 format that could be directly interpreted by python. The result was 2 sets of training and 2 sets of testing sets for each class, all of the same size. We also varied that size to compare how it affected model performance. One drawback to our dataset was that many tweets were cut short by 30 percent, due to how the csv file was initially constructed from the Kaggle source. Finally, to build our data-set, we randomly sampled from the entire pool of each class, which ensured that the tweets we picked out were as unbiased as possible.

Model and Methodology

To classify tweets, we used the built-in logistic regression function the SK Learn Library to train our model off of our data-sets. Earlier work was done by [3], whose project on classifying movie reviews motivated our political party classification task.

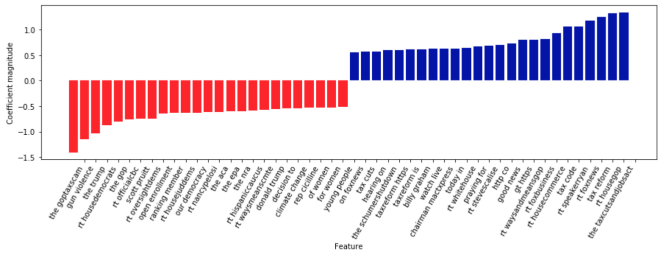
We decided to use logistic regression because although it wasn't as eye-catching as state of the art deep-learning algorithms, it was a simple technique that was robust enough for the task of NLP with high dimensional sparse data. This called for the use of Grid Search and Cross validation. Grid search served as a strategy for tuning and allowed us to adjust hyper-parameters to achieve the best results. Cross validation was used to minimize overfitting. Finally, just like for most NLP problems, we built a dictionary of words. The countVectorizer was used to transform the input documents into a matrix of semantic features.

Results



Our results were depicted in the accuracy vs data-set size graph above. The classification results peaked at 67 percent accuracy when 20000 tweets were used for each class (split between training and test) and increased steadily from the smallest data-set size of 2500 tweets, which performed at 57 percent accuracy. With additional parameter-tuning, more tweets, and better tweet quality, we believe that number could easily go up.

To combine our two projects together, we built a pipeline where we fed our fake Trump tweets that we generated into our binary classifier as inputs. The results were as expected: around 67 percent of the tweets were classified as Republican. For example, out of the 3 sample tweets we generated in the previous section, the first 2 out of those 3 were classified as Republican. Similar percentages were attained for greater sample sizes.



The image above is a visualization of the top 25 features for each class, with the words at the endpoints denoting the most discriminant phrases, whereas the words in the middle were less discriminative but still among the top 25 defining words in the model. The most significant two-word feature for the Republicans (Red) was ”the taxcutandjobsact” while the most significant for the Democrats were ”the goptaxscam”. This is as expected given that those phrases reveal the party and speak largely in favor of itself even without much context.

CONCLUSION

While the word count distributions did not match between the Trump Twitter Archive and fake tweet generator, the success of the model is primarily a qualitative measure. In this sense, the Markov Chain Model is successful at producing tweets with characteristics of Trump's twitter posts such as high frequency of exclamation points and capitalized words.

[1] V. Stracqualursi, CNN.
[2] B. Brown, “Trump twitter archive,” (2019).
[3] M. , ITNEXT.