A Low-Maintenance, Possibly Generalizable

Political Tweet Classifier – *Final Report*

By: Brendan Manning

I. Introduction

In 2016, campaign spending on the presidential and congressional races totaled $6.5 billion.[1] Despite all that spending, the presidential election was ultimately decided by a combined margin of approximately 107,000 votes in Pennsylvania, Michigan, and Wisconsin.[2] With elections this narrow, Karl Rove, former Senior Advisor to President George Bush, writes for the Wall Street Journal that both parties have turned to data analytics to "identify individual voters up for grabs and determine what motivates them, or which voters need extra encouragement to turn out and what energizes them."[3]

In 2012, the Obama campaign took advantage of Facebook's loose API restrictions to map connections between millions of volunteers and their friends on Facebook.[3] In 2016, Cambridge Analytica used a similar loophole to combine Facebook data with personality profiles to better target campaign advertising for the Trump and Brexit campaigns.[4] As a result of the Cambridge scandal, Facebook locked down its API.[5] The platform is no longer the easy source of political data it once was.

This project seeks to explore whether valuable data about political opinions can be extracted from Twitter. This "Political Tweet Classifier" rates a tweet as "Republican" or "Democrat"[6] using only the tweet's content. This technology has applications in real-time public opinion modeling, engagement analytics, and more.

II. Approach

Constraints

A prior project by Indiana University achieved high accuracy (91%) classifying political tweets, but all these tweets came from a large, manually tagged dataset.[7] This is both time consuming and makes the model less relevant as political issues and terms change. I wanted to ensure my model could easily be retrained to capture to these changes, even at the cost of some overall accuracy.

This process could be improved if there were a way to automatically label tweets in the dataset with their party. To do this, it helped to narrow the problem from classifying *any* political tweet to classifying tweets *specifically from* U.S. congressmen. This allows each tweet to be labeled with the politician's political party (which is well known). Afterwards, it is possible to measure how well this model generalizes to a small (~250 sample) set of (manually labeled) tweets from non-politicians.

Methodology

1. Build dataset: Scrape tweets, link with tweeter's party.
    a. Tweets from politicians can be obtained by running `twint -u <Username> -o tweets/<Username>.csv --csv --limit 300`
    b. The jupyter notebook will combine these csvs and join them with party information.

C. Combine with biographical information

```
tweets = pd.read_csv("tweets.csv")
tweets = tweets.merge(congressmen[['username', 'State', 'Party' ]], on=['username'])

tweets['tweet'] = tweets['tweet'].apply(lambda t: transform_tweet(t))
```

Figure 1 - Merge tweets with politician's party

---

[1] https://www.washingtonpost.com/news/wonk/wp/2017/04/14/somebody-just-put-a-price-tag-on-the-2016-election-its-a-doozy/

[2] https://www.washingtonpost.com/graphics/politics/2016-election/swing-state-margins/

[3] https://www.wsj.com/articles/the-campaign-data-arms-race-11574294077

[4] https://www.cnet.com/news/facebook-cambridge-analytica-data-mining-and-trump-what-you-need-to-know/

[5] https://theconversation.com/facebooks-data-lockdown-is-a-disaster-for-academic-researchers-94533

[6] I ignored so-called Independent voters. In addition to making the model simpler, it could be argued that there are no true "independents" among those who tweet about politics. Read more at: cnn.com/2019/03/14/politics/independents-pew/index.html

[7] https://cnets.indiana.edu/wp-content/uploads/conover_prediction_socialcom_pdfexpress_ok_version.pdf

c. Tweets for non-politicians are obtained by scraping popular hashtags (command shown). See Figure 3 for the list of hashtags I used to fetch non-politician tweets.
`twint -s "#<Hashtag>" -o <Hashtag>.csv --csv --limit 250 --lang en`
d. The tweets for non-politicians will, unfortunately, have to be manually joined and labeled in Excel.
    i. Labels go in the 'Party' column (label 0 for Democrat, 1 for Republican).

2. Preprocess: Lowercase, clean URLs, remove stop-words, perform word stemming.

Original	Processed
"Our Soldiers, Sailors, Airmen, and Marines willingly place their lives on the line to protect America. It is our duty in Congress to support our troops. Please read my newsletter. http://ow.ly/CuXV50v2jxE #MS01 pic.twitter.com/SAoWyr2Gvo"	"soldier sailor airmen marin willingli place live line protect america duti congress support troop pleas read newslett owli ms01"

3. "Feature Extraction: Identify important n-grams using a TF-IDF-inspired formula.
    a. The importance *I* of a term *i* is given by:
    $$I_i = |RUsers_i - DUsers_i|^2 \cdot |RUsages_i - DUsages_i|^{2/3}$$
    where $XUsers_i$ is the number of users in Party X who have used Term *i* and $XUsages_i$ is the number of times anyone from Party X used Term *i.*
    b. This helps down-weight words that one "rogue" tweeter may have used an extreme number of times, which would otherwise be considered a hyper partisan term. A good example of this is #ms01 or #fl02 which were used exclusively by the congressmen from Mississippi's First Congressional District and Florida's Second, respectively.
    c. Using these scores, I identified the top 500 1-grams, 1000 2-grams, 2500 3-grams.
    d. Using these n-grams, I transformed the tweets into bag-of-word vectors.
4. Train Models: Neural Network / Decision Tree / KNN
    a. Test/Train split: 70/30
5. Evaluate: Politician Tweets, Non-Politician Tweets

```
nn = MLPClassifier(
    solver='adam',
    activation='relu',
    alpha=1e-5,
    hidden_layer_sizes=(2000,),
    max_iter=200
)
nn.fit(XTrain, YTrain)
print(nn.score(XTest, YTest))
```

Figure 2 - Code used to create the Neural Network

III. Results

Successes

N-Grams The model was able to identify relevant features to train on. Some of the 4,000 n-grams identified include

1. 1-grams: '@potu', 'discrimin', 'west', 'pelosi', 'god'
2. 2-grams: 'town hall', 'im work', 'hous democrat', 'mental health', 'new job', 'gun violenc'
3. 3-grams: 'keep us safe', 'health care worker', 'mobil offic hour', 'paid sick leav'

Performance The model performed well at identifying tweets from the testing split.

- Neural Network: **80.65%**
- Decision Tree: 71.71%
- KNN: 70.00% (k=9)

SVD The model's accuracy did not benefit from the use SVD, so that step was removed. The model's vocabulary contained 4000 terms and most tweets used 2-3 at most. Thus, it made sense to preserve every feature possible. As has been discussed before, Neural Networks avoid some of the worst effects of the curse of dimensionality. This is evident in my results where the Neural Network outperformed the Decision Tree and KNN by 8-9%. (Anecdotally I will add that the performance gap narrowed significantly since the beginning of the project).

Shortfalls

The model did not do well classifying the manually tagged tweets. From this we can conclude that this model, trained on tweets from politicians, *does not* (yet) generalize to predicting normal political tweets. We obtained **62.72%** accuracy on the 278 non-politician tweets. To better understand why it did not generalize well, consider the following tweets:

- The top two tweets are from politicians. They (1) use perfect grammar (2) adopt a professional tone of voice and (3) may be more administrative than partisan (see Sen. Casey's Facebook link).
- The bottom two tweets are from regular Twitter users. They (1) use different hashtags than politicians (like #EatTheRich) and (2) adopt a much less professional tone.

It seems that politicians and citizens tweet in different "dialects" of the same language. Looking at the accuracy breakdowns, there is an even fuller story to be told:

- Both models overfit towards one party or the other.
- The Neural Network gets 70-80% on all the hashtags which were Democrat-only. Conversely, it got 30-40% accuracy on all-Republican hashtags. We can see that the Neural Network returned "Democrat" by default.
- The Decision Tree is the opposite. It gets 60-80% accuracy on Republican hashtags and 16-50% accuracy on Democratic hashtags. Here, "Republican" was the default.
- It is important to note that these hashtags represent a very small subset of political discourse on twitter. Likewise, the testing set is small (~275 tweets). The accuracy obtained here should be taken with a "grain of salt."

	Overall	Republicans	Democrats
Neural Network	**62.7%**	**52.8%**	**70.7%**
Decision Tree	**53.4%**	**80.0%**	**31.8%**

	Neural Network	Decision Tree	Sample Partisanship
#VoteRed	48.3%	80.6%	R
#MAGA	66.6%	58.3%	R & D
#GunControl	83.3%	50.0%	D
#SleepyJoe	61.5%	76.9%	R
#VoteBlue	70.0%	33.3%	D
#Trump2020Landslide	41.6%	83.3%	R
#2A	33.3%	66.6%	R
#ProLife	63.4%	48.1%	R & D
#BlueNoMatterWho	70.8%	16.6%	D
#GreenNewDeal	73.0%	53.8%	D

Figure 3 - Accuracy of model on "non-politician" tweets by classifier, party, and hashtag

Note: These numbers may be different in the attached Notebook. They are adjusted slightly every time the model is re-trained. In rare cases, the bias of the Neural Network and Decision Tree has changed after retraining. However, it is always heavily biased towards one party or another.

Interesting Takeaways

Political Polarization A 2019 Pew Research Center study asked Americans which of about 20 political issues (e.g. military, immigration, race relations, climate change, etc.) the Congress should prioritize. The only issues Pew found were of the same importance to each party were "Jobs" and "Social Security."[8] I was curious whether my dataset of political tweets would reflect this extreme polarization.

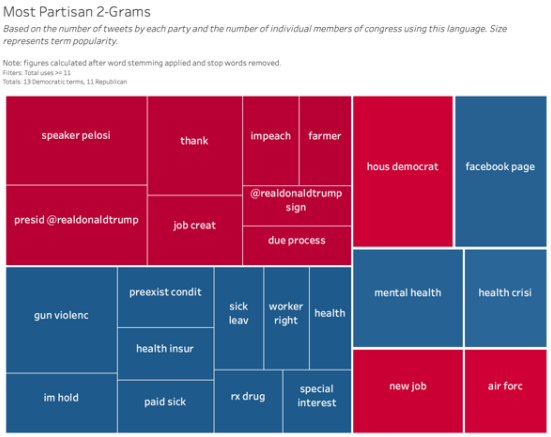
[8] https://www.pewresearch.org/fact-tank/2019/02/05/republicans-and-democrats-have-grown-further-apart-on-what-the-nations-top-priorities-should-be/

As part of feature extraction, I created a function "*most_important_ngrams"* that ranked a term's "uniqueness" to one party or another. It was based on the per-party difference in (A) the number times used and (B) the number of individual accounts using the term at least once. The function returns a higher number for more partisan terms. For terms which are very partisan, we can assume that the party that used it the most is the "direction" of the partisanship (more R or more D).[9]

Ngram	Republican	Democratic	% Republican	Total Uses	Total Partisanship
town hall	23	84	0.21495	107	0.785047
hous democrat	37	2	0.94872	39	0.948718
im work	2	30	0.06250	32	0.937500
mental health	1	27	0.03571	28	0.964286
speaker pelosi	32	0	1.00000	32	1.000000
new job	23	1	0.95833	24	0.958333
gun violenc	0	30	0.00000	30	1.000000
proud stand	1	14	0.06667	15	0.933333
pass bill	5	21	0.19231	26	0.807692

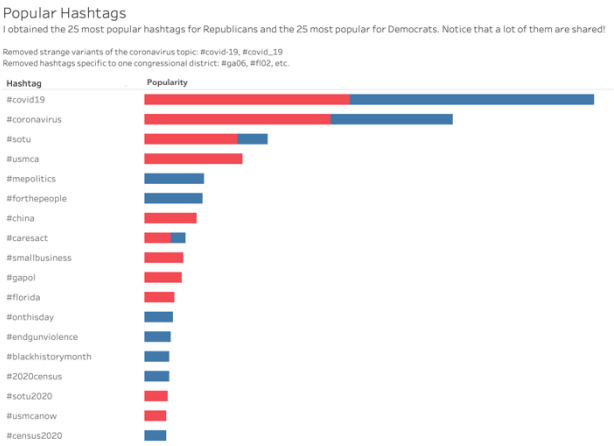
Figure 4 - A preview of the important_ngrams data and some additional calculated fields in Tableau

I chose to visualize the most partisan terms with a treemap. After some testing, I found that 2-grams were the most interesting. They are more unique than 1-grams, while many 3-grams are just these same 2-grams with an extra word added at the beginning or end. Note that the size of each block represents total mentions while the color represents party.



My hypothesis that the tree map would represent different priorities was correct. Democrats tweeted more about healthcare, gun violence, fighting special interests, and labor issues. Republican messaging focused on job creation, impeachment, farmers, and the military.

<u>Popular Hashtags</u> I made a similar graph for popular hashtags. These are the most popular hashtags by total number of uses. Note, this was done separate from the *most_important_ngram* function.



Mentioned Users Both parties mentioned @realDonaldTrump the most (although Republicans mentioned him 227 times versus the Democrats' 33 times). Afterwards, both parties mentioned government agencies and official party accounts the most.

- Top 5 Republican: *@realdonaldtrump, @potus ,@cdcgov, @speakerpelosi, @sbagov*
- Top 5 Democratic: *@realdonaldtrump, @cdcgov, @oversightdems, @housedemocrats, @sbagov*

[9] I acknowledge that there may be edge cases where a less-partisan term may be used more by Republicans, but actually be a Democrat-leaning term (or vice versa) according to the given formula for importance. Future versions of the *most_important_ngrams* function could address this by returning "signed" answers (where the sign indicates the direction of partisanship and the magnitude indicates the degree of partisanship). For now, though, we can safely assume that among the top 50 terms I selected, all where labeled correctly. If a term is very partisan, it is a safe assumption that the party given by the importance score also uses that term the most.

Coronavirus My dataset contains the last 300 tweets for about 500 congressmen (counts normalized so neither party was overrepresented). I wanted to see whether both parties' online messaging reacted to the pandemic in the same way. The chart (right) shows that both parties mentioned the coronavirus with similar frequency throughout the time period sampled.

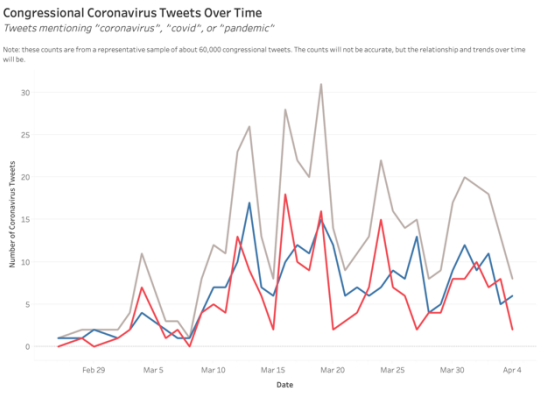


Figure 5 – Coronavirus tweets over time. Gray = overall mentions.

Tableau Dashboard You can view interactive versions of all three charts shown above on Tableau Public: https://public.tableau.com/profile/brendan.manning - !/vizhome/CongressionalTwitterAnalysis/Dashboard. Hover over any element for more information. You can also filter the tree map by number of uses.

IV. Conclusion

At the start of the project my goals were to (1) train a model that would predict the partisanship of a politician's tweet and (B) test whether such a model would work on *any* political tweet in general. I achieved great success in the first area, correctly labeling 80% of the testing split. The model's performance in the second area, however, was only marginally better than if it guessed at random. That does not mean this project was a failure, however.

Some aspects of this project were intended to address shortcomings in a previous political tweet classifier. The 80% accuracy on politicians' tweets is remarkably high considering the time constraints of a school project. There is room for hyperparameter tuning and process improvement that could lead to more accurate and generalizable outcomes if this project were continued. As an example, I would like to present you with one possible improvement:

The Indiana University team achieved their best performance using a vocabulary of only hashtags. I believe my weighted n-gram approach is helpful in identifying partisan phrases like "due process" (leans R) or "preexisting conditions" (leans D) that their model would miss. That said, it would be worth investigating whether giving hashtags additional weight would improve generality.

Future time invested into this project would be beneficial to the two stakeholders I identified in my "Lightning Talk". Academics and journalists could use tweet classification to monitor the activity and enthusiasm on both sides in real-time. The graphs I produced for the EDA step (hashtag popularity, partisan keywords, etc.) could become interactive graphics which update daily. Candidates and parties could use this to identify clusters of followers whose tweets are similar to their own.

This Political Tweet Classifier shows that politicians' tweets are a good starting point for more accurate and maintainable partisanship identification models. Future investment would yield even better results than what was already achieved in a mere 40 hours.

V. Acknowledgements

- A list of Congressional twitter handles can be found on GitHub[10]
- I used a WikiTable extractor[11] to get the parties of the members of the U.S. House[12] and U.S. Senate[13] from lists on Wikipedia.

[10] https://github.com/oduwsdl/US-Congress
[11] https://wikitable2csv.ggor.de/
[12] https://en.wikipedia.org/wiki/List_of_current_members_of_the_United_States_House_of_Representatives - Voting_members_by_state
[13] https://en.wikipedia.org/wiki/List_of_current_United_States_senators#List_of_senators

- I consulted an article[14] by Heaton Research for help picking the optimal hidden layer size for my Neural Network
- As mentioned before, I used a study[7] by Indiana University as a point of comparison for the performance and maintainability of my project.
- I scraped tweets using twint[15], a popular open-source project.

VI. Time Log

1. ~12 hours – Gather tweets, create most_important_ngrams function, train models
2. ~10 hours – Improved preprocessing, created manually tagged dataset of non-politician tweets, and calculated the model's accuracy at predicting these tweets.
3. ~10 hours – Created interesting visualizations about popular hashtags, mentioned users, coronavirus, and partisan keywords in Tableau.
4. ~8 hours – Finish Tableau statistics, combine into Tableau Dashboard, write report.

[14] https://www.heatonresearch.com/2017/06/01/hidden-layers.html
[15] https://github.com/twintproject/twint