Econ 224 Project Report

Jessica Bachner

December 18, 2018

Literature Review

On September 3, 2018, Nike released its 30th year anniversary "Just Do It" ad campaign, featuring former N.F.L. quarterback Colin Kaepernick. Nike's choice of endorsing Kaepernick was deemed controversial, as he is notorious for having kneeled during the National Anthem before N.F.L games as a form of protest against racial injustice. When Kaepernick first declared his endorsement by Nike in a tweet captioned "Believe in something. Even if it means sacrificing everything.", the Twitter world was taken by storm. Immediately, " 'Just Do It' and 'Nike' became top trending terms on Twitter in the United States, and by [the following] morning the hashtag #NikeBoycott was one of the most used on the social media service" (Belson and Draper 2018). Even President Donald Trump tweeted his thoughts on the campaign– an act that only enhanced the prevalence of the topic. The impact that this campaign had, especially on Twitter, made me particularly interested in having it be the centerpiece of this project.

Question

The purpose of my project was twofold, and was thus governed by two key questions. First, I tried to understand the typical profiles of people in favor of and opposed to Nike's 30th anniversary "Just Do It" campaign that features Colin Kaepernick. I approached this question by summarizing trends in positive and negative tweeters' US residence, Twitter influence, and age. To conduct this multistep analysis, I employed a range of techniques including text mining and heat mapping.

The second question I explored in this project was how to create a model that best predicts tweeters' true attitudes toward the campaign. This question was propelled by the seemingly faulty nature of computational sentiment analysis techniques in R. Specifically, this method of classification could not detect sarcasm or who/what a given tweet was referencing. For example, many tweets that were being computationally classified as negative, and thus against the campaign, were actually tweets that were in favor of the campaign– these tweets, though having a negative nature, often times were simply a user in favor of the campaign chastising someone against it.[1] To see if computational sentiment analysis could be improved, I employed the "Bag-of-Words" and "Term Frenquency-Inverse Document Frequency" techniques to train models that were then tested for prediction accuracy using cross validation.

Data Description

The original dataset I began working with was an existing dataset I obtained from Kaggle entitled "5,000 #JustDoIt! Tweets Dataset", which can be accessed at: https://www.kaggle.com/eliasdabbas/5000-justdoit-tweets-dataset. This dataset included 72 variables and 5089 observations. The key variables in this dataset included: the tweet itself (tweet_full_text), the tweet favorite count (tweet_favorite_count), the tweet retweet count (tweet_retweet_count), when the user created his/her Twitter account (user_created_at), how many followers the user has (user_followers_count), and where the user is from (user_location). One of the most important variables that this dataset lacked, however, was a variable indicating whether or not the user was in favor of, neutral towards, or opposed to the campaign. This variable became increasingly important as I realized that packages in R that carried out sentiment analysis could not distinguish between, for example, a user whose negative tweet was directed towards the campaign and one whose negative tweet was directed to someone who is not in favor of the campaign. With this variable I could not only more accurately understand the profile of the typical responder, but I could also create a series of models to predict how accurate different techniques in R can predict sentiment.

To provide myself with a reliable sentiment variable, I used the "grep" function in the tm package to filter out all tweets that did not include the following terms: "Nike", "company", "Colin", and "Kaepernick".

[1] From here on out, to avoid wordiness, the following terms will be interchangible: "those in favor of the campaign" and "positive tweeters"; "those neutral towards the campaign" and "neutral tweeters"; and "those opposed to the campaign" and "negative tweeters"

I then selected the 6 key variables listed above, along with 6 other related, though less informative, variables and exported the resulting data to Excel. In Excel, I labeled the 1,390 tweets as being either relevant or irrelevant to the campaign and as being in favor of, neutral towards, or opposed to the campaign. I then imported this new dataset, entitled "labeled_subset" into R, filtered out the tweets that were categorized as not relevant, and was left with my final comprehensive dataset comprised of 1,139 observations. The key variables of "labeled_subset" are the same as the previous, plus the hand-labeled variables "relevant" and "sentiment".

```
## [[1]]
## [1] "sentiment"            "relevant"             "tweet_full_text"
## [4] "tweet_favorite_count" "tweet_retweet_count"  "user_created_at"
## [7] "user_followers_count" "user_location"
```

It is also important to note that when determining the locations of the positive and negative tweeters, I again performed some data cleaning by hand. After creating the subsets of positive/negative tweeters, I created a subset containing only the user's location and the sentiment of the tweet, and filtered out users who did not list a location in their Twitter profile. I then exported this subset of 680 locations and sentiments for positive tweeters and the subset of 89 locations and sentiments for negative tweeters, and hand-labeled which US state the tweeter resided in. The reasoning behind hand-labeling these locations was that many of the locations users entered were either non-existing places such as "wherever God takes me", or nicknames for one's residence, such as "The (215)" or "Cornhusker State". Once I labeled these locations by US state, I imported the datasets into R with the titles "locationspos" and "locationsneg".

Methods

In the first part of my project, I analyzed the profiles of positive and negative tweeters. I started this profiling by analyzing the location distribution of both positive and negative tweeters in the continental US. In R, one of the most efficient ways to plot a map of the US is by using the map_data function from the "ggplot2" package, along with specifying the type of map, in this case "state", which can be drawn from the "maps" package. When combining these two commands (map_data("state")), we are given a data frame that includes, among other variables, the longitudes and latitudes of the continental US states and a variable called "region" that lists the state corresponding to the given longitude and latitude. In order to create a heat map of the United States based on data from another dataset, we must first rename the states variable in our seperate dataset to be called "region". We can then merge these two data frames by the variable "region", and can use the "ggplot" function to create a map that fills and color codes the states by the frequency of the given value (in this case, number of positive or negative tweeters from that state).

Next, I profiled tweeters by their range of influence. I specifically compared user follower counts, tweet favorite counts, and tweet retweet counts across positive and negative tweeters. This did not require any complicated statistical or coding computation, as I simply looked at quartiles, means, maximums, and minimums of the variables in tables created by "stargazer".

My final step in analyzing the profiles of the positive and negative tweeters was identifying trends in when the user created his or her Twitter account. The assumption underlying this analysis was that those who have had Twitter accounts for longer are older. The original format of the "user_created_at" variable was a string of characters that included the day of the week, the month, the day, the exact time (to the seconds), and the year. In order to extract just the year portion of the variable strings, I used the "substr" function from the "stringr" package, which accepts the following arguments: x (a character vector), start (the first element to be extracted), and last (the last element to be extracted). I then plotted the years users created their accounts against the number of positive or negative tweeters using the basic "plot" function.

The next half of my project was devoted to training models to predict how a user felt about the campaign. As a preliminary step, I checked the proportions of positive, neutral, and negative tweeters in my "labeled_subset" dataset, and filtered out any URLS, hashtags, and (@) signs from the tweets using the "gsub" function from the package "tm". I then dove into my first prediction model that used was built-in sentiment analysis techniques in R from the "syuzhet package". I used the "get_sentiment" command from "syuzhet" to numerically classify my variable "sentiment_computed" as "<0" (negative/ tweeter opposed to the campaign), "=0" (neutral/ tweeter neutral towards the campaign), and ">1" (positive/ tweeter in favor of the campaign). I then binded these categorizations to my pre-existing dataset that included the true attitudes towards the campaign and used the “count” function to determine how many times the “syuzhet” package technique accurately predicted the actual sentiments towards the campaign.

The second modelling method I explored was the “Bag-Of-Words” model. The objective of “Bag-Of-Words” is to individually break up, or tokenize, every word in each tweet that is labeled as “positive”, “negative”, or “neutral”, and to use the presence of certain tokens in each type of tweet to predict whether a test tweet composed of similar tokens is positive, negative, or neutral. I began my modelling by setting the seed and stratifying my data using the “createDataPartition” function from the package “caret”. Stratification is the process of splitting one’s data into subsets whose data match the proportions of the original dataset. In this case, it was very important that I stratify my samples before creating my testing and training sets for cross validation, since my dataset had far more positive tweets than negative and neutral tweets, creating a massive class inbalance. I then verified that my training and testing sets accurately represented the proportions I had previously determined.

Next I used the “tokens” command from the “quanteda” package to tokenize my subset of training tweets and removed any special characters from the tweets that had not been cleaned during prior cleaning using the “gsub” function from the “tm” package. I continued to clean my tweets by using various functions from the quanteda package to lowercase, stem, and remove stopwords from the training tweets. In text analysis, stemming words involves cutting down words to their roots and removing all unimportant prefixes and suffixes– an example of stemming would be to convert the word “famously” to the would “fame”. Stopwords are considered words that do not carry much weight, and would thus not have much predictive power. Examples of stopwords include articles such as “the” or “a”. After this thorough data clensing, I converted my training tokens into a document-feature matrix (dfm) using the “dfm” command from the “quanteda” package. A document-feature matrix is a matrix that lists the documents, or tweets in rows, the tokens in columns, and the frequency of each token in each document as the observations. I then converted the dfm into a data frame and merged it with my dataset containing the sentiments of each tweet.

Finally it was time to conduct single tree cross validation. To do so, I used the “createMultiFolds” function from “caret” to make 10 stratified folds from my training set. Before having my computer run the cross validation, however, I implemented the functions “makeCluster” and “registerDoSNOW” from the “doSNOW” package to run my cross validation in parallel in order to speed up the computing time. I then used the “train” function from “caret” to conduct the cross validation, stopped running in parallel, and analyzed my results. The “train” function in “caret” eventually produces a results table that lists the accuracy levels of the model for several cp tuning parameters– this was how I determined my tuning parameters and accuracies for this and the following model.

The third and final modelling method I carried out was the “Term-Frequency- Inverse-Document Frequency” method (TF-IDF). When conducting the “Bag-Of-Words” technique, we must consider two factors: longer documents will have higher term counts and terms that appear frequently across the whole set of documents, or the corpus, may not be as predictive. The TF-IDF model tries to improve upon these considerations by normalizing documents based on their length (calculating term-frequency) and penalizing terms that occur frequently across the corpus (calculating inverse-document frequency).

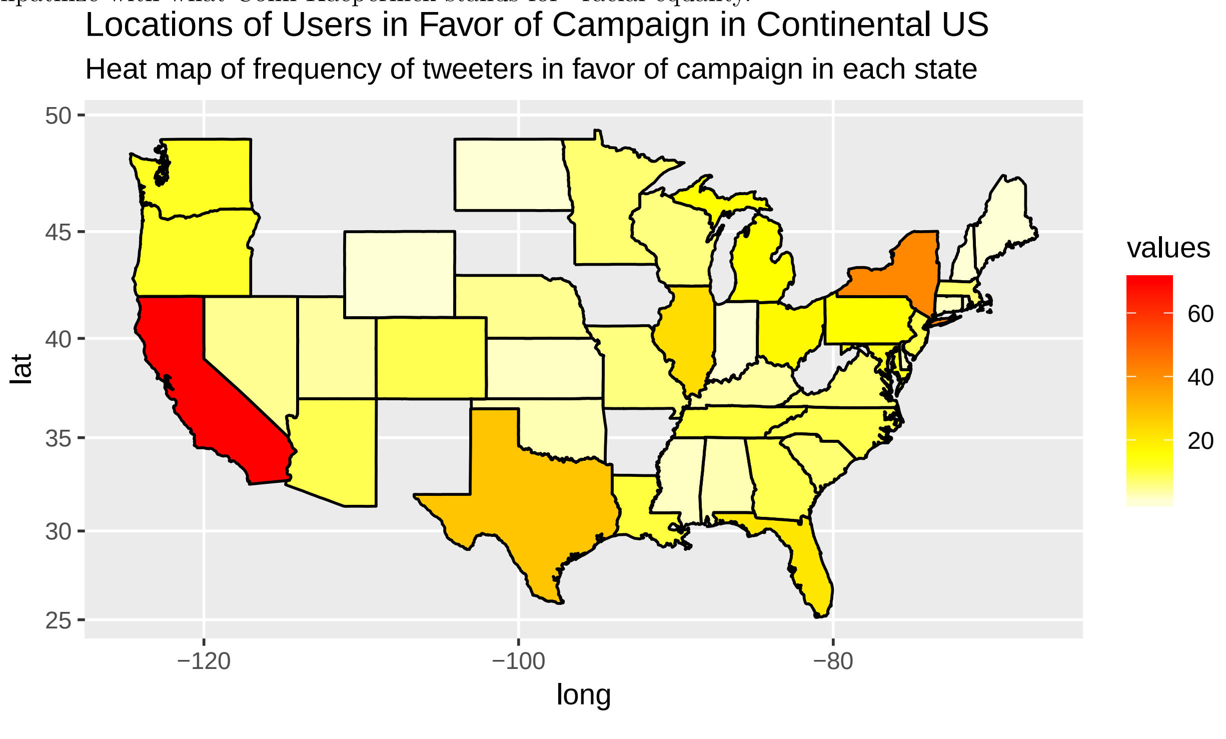
My first step in conducting “TF-IDF” was to create functions for term-frequency and inverse-document frequency. Term frequency, or the number of instances a term appears in a document, can be calculated by dividing the row (the unique terms in a tweet) by the sum of the row (total number of terms in the tweet). The inverse-document frequency is calculated by taking the log of the quotient of the count of distinct tweets in the dataset by the count of the tweets in the dataset that have a certain term. After creating these functions, I multiplied the two functions together. I then used the “as.matrix” command from the “Matrix” package to convert my training tokens to a matrix, and used the “apply” function, which eventually transposed the matrix, to apply the term frequency function to the rows of the matrix (full tweets). After repeating a similar process with the IDF to calculate the IDF vector that would be used for the training and test data, I applied the TF-IDF function onto the tweets. In order to prepare for cross validation, I had to transpose the TF-IDF matrix back to its original format, using the function “t”. I then carried out single-tree cross validation in the same steps as the previous model.

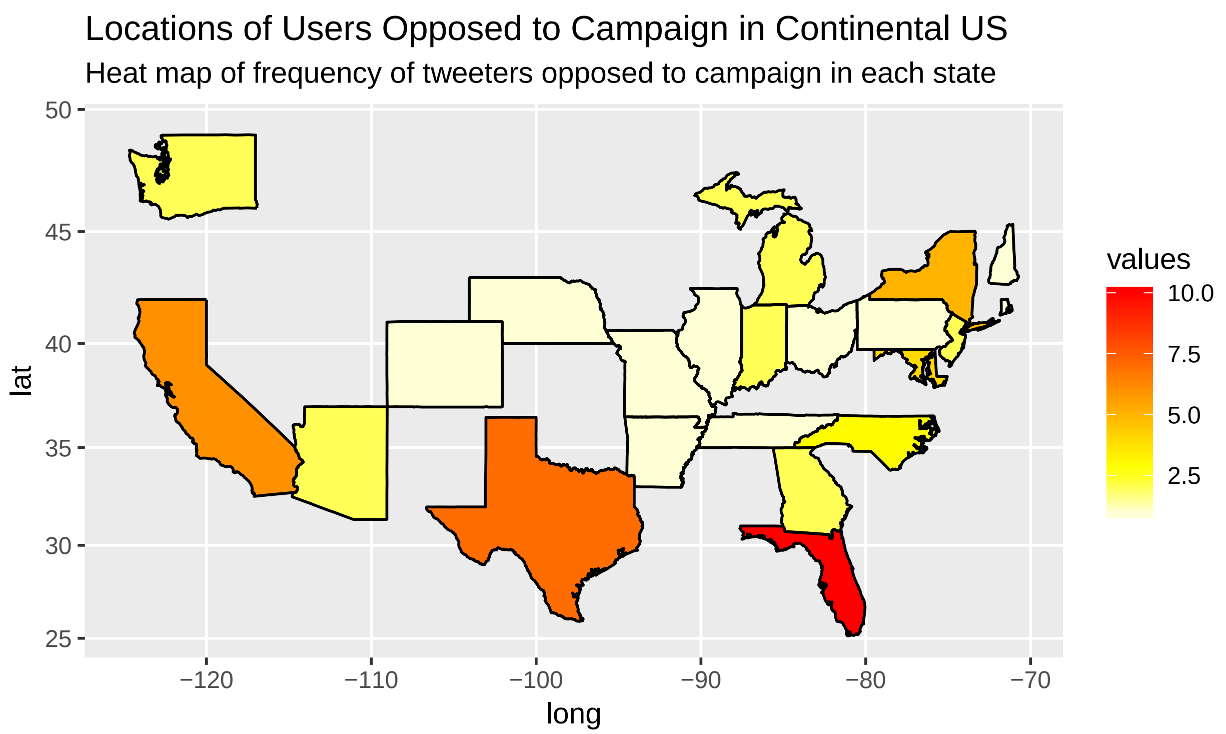
A fourth model I considered carrying out, “The N-Gram Model”, builds upon the “Bag-Of-Words” model by taking word ordering into consideration. This model usually improves modelling to some degree by creating tokens of 1 through N adjacent terms instead of creating tokens word-by-word. Since this method more than doubles the already huge dfm matrix, I decided to forgo the opportunity to explore it, as my computer does not have enough available memory to sift through this data.

Results

Part 1: Profiling Tweeters

My first step in profiling the tweeters included analyzing the geographical distribution of both positive and negative tweeters in the continental US. Upon creating the heat map of positive users, I noted the extreme diversity of the tweeters– as a matter of fact, 42 of the 50 states were represented among positive tweeters. The state that hailed the most positive tweeters was California, with over 60 tweeters. New York and Texas followed California in second and third place amongst positive tweeters. It is difficult to determine how informative this knowledge is, however, because of the fact that California, New York, and Texas are three of the four most populated US states. Making the geographical heat map for the negative tweeters perhaps proved even less informative. Since only 89 negative tweeters included a US location on their Twitter account, frequencies as a whole were much less than those of positive tweeters, of whom hundreds included a US location on their account. Despite this fact, the state that garnered the most negative tweeters was Florida, followed by Texas, California, and New York in second, third, and fourth place. It is difficult to pinpoint causality here due to the lack of sample size, but it can be the case that the reason why Florida and Texas were the most represented states among negative tweeters is because these states are known to have a much less representation of diversity/presence of minorities than New York and California. It would make sense that those who do not exemplify diversity are against the campaign, as they may not be able to sympathize with what Colin Kaepernick stands for– racial equality.


Locations of Users in Favor of Campaign in Continental US
Heat map of frequency of tweeters in favor of campaign in each state
lat
long
values
60
40
20
50
45
40
35
30
25
−120
−100
−80


Locations of Users Opposed to Campaign in Continental US
Heat map of frequency of tweeters opposed to campaign in each state
lat
long
values
10.0
7.5
5.0
2.5
50
45
40
35
30
25
−120
−110
−100
−90
−80
−70

The next variable I observed amongst positive and negative tweeters was their influence, derived as a function from the variables indicating user follower count, tweet favorite count, and tweet retweet count. Though the maximum user follower count amongst positive tweeters is almost 1.6 times greater than that of the negative tweeters, the negative tweeters, on average, have more followers than positive tweeters. Though the percentile breakdowns of tweet favorite and tweet retweet count were not fruitful, it was particularly interesting to note that the maximum tweet favorite counts and retweet counts were over 7 times and over 4 times greater, respectively, for positive tweeters than for negative tweeters. Thus, we can infer, from how the statistics varied across positive and negative tweeters, that positive tweeters have a larger social media influence than their negative tweeter counterparts.

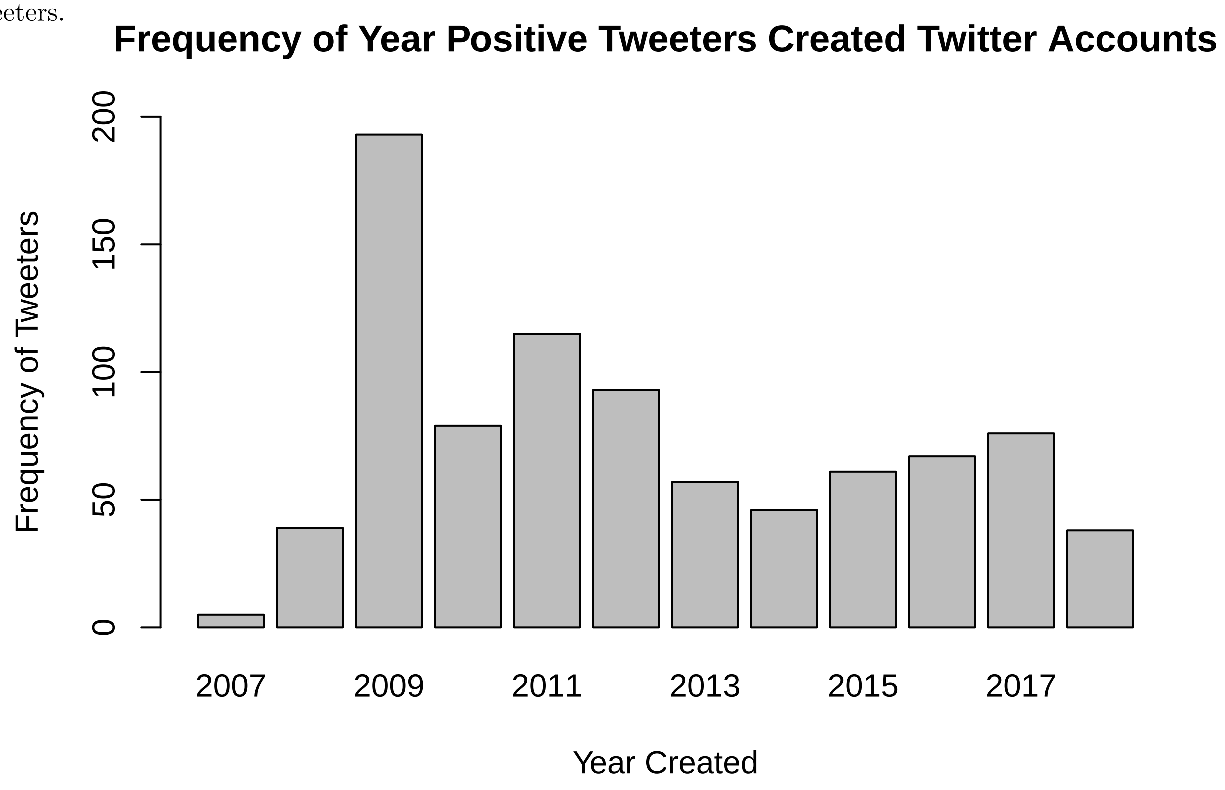
Table 1: Summary Statistics of Influence Variables for Tweeters in Favor of Campaign

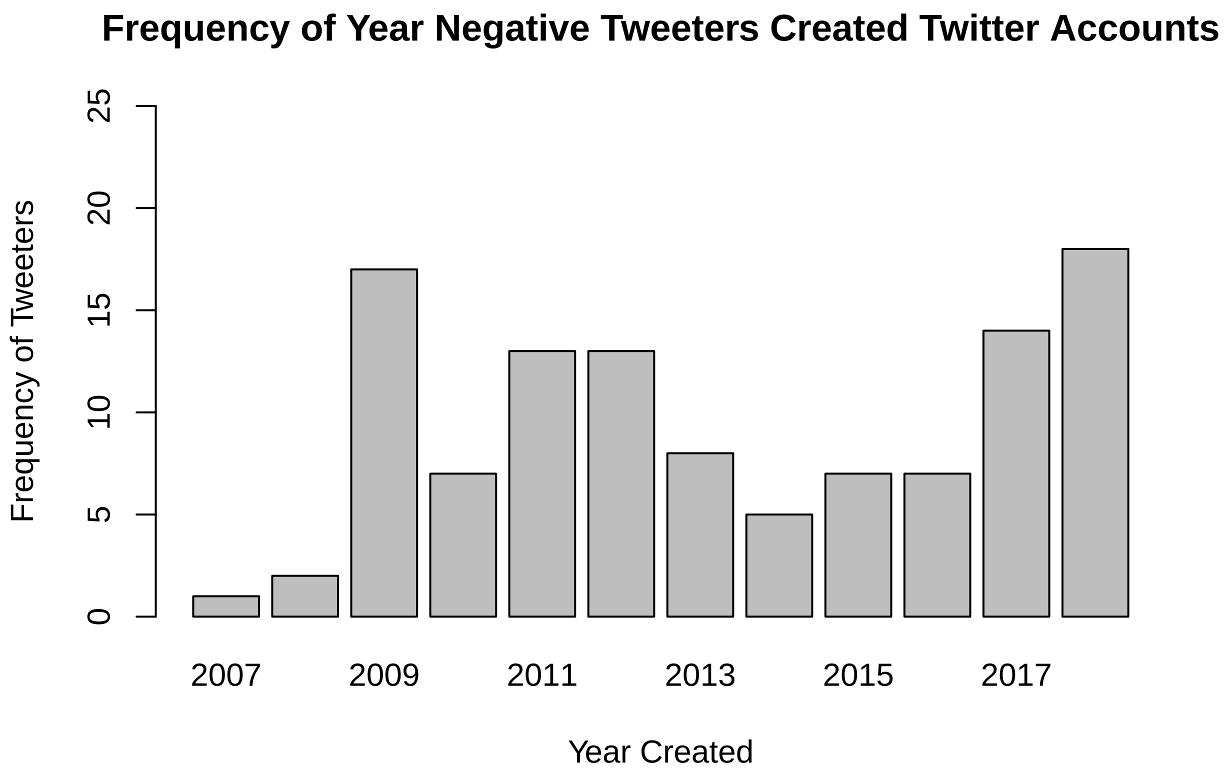
Statistic	Min	Pctl(25)	Median	Mean	Pctl(75)	Max
user_followers_count	0	81	347	2,242	1,230	114,411
tweet_favorite_count	0	0	0	7	1	744
tweet_retweet_count	0	0	0	3	0	306

Table 2: Summary Statistics of Influence Variables for Tweeters Opposed to Campaign

Statistic	Min	Pctl(25)	Median	Mean	Pctl(75)	Max
user_followers_count	0	53.2	331	3,077	1,948.2	70,154
tweet_favorite_count	0	0	0	3	1	97
tweet_retweet_count	0	0	0	1	0	70

My final step of profile analysis was to gain some insight into how old positive and negative tweeters are. To do this, I made the assumption that older tweeters have had their Twitter accounts for a longer period of time than younger tweeters. Upon graphing the distribution of the years that positive users created their Twitter accounts, I found that the distribution was skewed to the left, with the year 2009 being the most popular year that positive users created their accounts. 2011 and 2012 were the second and third most popular years that positive users created their accounts. 2018 had the lowest frequency of positive users who created their accounts other than the year 2007 (the year after that Twitter was founded). On the other hand, for negative tweeters, 2018 was the most popular year that users created their accounts, with 2009 coming in a close second and 2017 coming in third. If my assumption correlating age and account creation date has any validity, then it can be inferred that positive tweeters are, as a whole, older than negative tweeters.





Part 2: Making Models

The first model, using computed sentiment analysis techniques to determine true attitudes towards the campaign, was, as assumed from the outset, not as powerful as other techniques. As was mentioned earlier, a fault with this tool is that it cannot detect sarcasm or who/what a tweet is directed at. For these reasons, we saw that the computed sentiment only matched the true sentiment 54.3% of the time. Upon further investigation, it came to my attention that most of this error was derived from truly positive tweets that were misidentified as being negative. Specifically, many people who were in favor of the campaign composed nasty tweets directed at President Donald Trump for his condemnation of a campaign that glorified, in his view, someone who was a disgrace to the country. Since Trump's tweets always garner a large pool of responses, it is clear that this probably had a large impact on the predictive power of this tool.

The second model, which used the "Bag- Of- Words" method, was found to perform the best out of the 3 models, with a maximum accuracy of 77.5% at the cp tuning parameter of 0.0228. The reason why this model outperformed the first by over 20 percentage points was because it used the tweets, specifically their tokens, to try to match incoming test tweets with similar words. This model is probably more predictive than the first, because there are certain words in the English language that have different connotations and denotations, and being able to train a model to treat these words figuratively may be able to improve predictability.

The third model, the TF-IDF model, performed similarly, though slightly poorer than the "Bag-Of-Words" model. The maximum accuracy achieved by this model was 76.7% at the cp tuning parameter of 0.0158. As previously mentioned, the TF-IDF model attempts to improve upon the "Bag-Of-Words" model by normalizing tweet lengths and penalizing terms that occur frequently across the corpus. There are two likely reasons why this model did not improve "Bag-Of-Words". First, it is likely that document length may not be an important indicator of whether a tweet was in favor or opposed to the campaign. Rather, it is likely that positive and negative tweets are similar in character length. Perhaps if the model was trying to predict whether or not a given text was a spam or not, normalizing document length could have improved the model. Second, it may be the case that words that appear many times throughout the corpus are indeed important in predicting sentiment. Because the same word may be used by a positive tweeter who is condemning Trump, for example, and a negative tweeter, certain corpus-wide high frequency words may be important in indicating the true sentiment of a tweet.

Table 3: Comparing Accuracies Across Modelling Methods

	c(computing_accuracy, cv1accuracy, cv2accuracy)
Model 1	0.543
Model 2	0.775
Model 3	0.767

References

Bates, Douglas, and Martin Maechler. 2018. *Matrix: Sparse and Dense Matrix Classes and Methods.* https://cran.r-project.org/package=Matrix.

Belson, Ken, and Kevin Draper. 2018. "Colin Kaepernick's Nike Campaign Keeps N.F.L Anthem Kneeling in Spotlight." https://www.nytimes.com/2018/09/03/sports/kaepernick-nike.html.

Benoit, Kenneth, Kohei Watanabe, Haiyan Wang, Paul Nulty, Adam Obeng, Stefan Müller, and Akitaka Matsuo. 2018. "quanteda: An R package for the quantitative analysis of textual data." *Journal of Open Source Software* 3 (30): 774. doi:10.21105/joss.00774.

Boettiger, Carl. 2017. *knitcitations: Citations for 'Knitr' Markdown Files.* https://cran.r-project.org/package=knitcitations.

code by Richard A. Becker, Original S, Allan R Wilks. R version by Ray Brownrigg. Enhancements by Thomas P Minka, and Alex Deckmyn. 2018. *maps: Draw Geographical Maps.* https://cran.r-project.org/package=maps.

Corporation, Microsoft, and Stephen Weston. 2017. *doSNOW: Foreach Parallel Adaptor for the 'snow' Package.* https://cran.r-project.org/package=doSNOW.

Diez, David M, Christopher D Barr, and Mine Cetinkaya-Rundel. 2017. *openintro: Data Sets and Supplemental Functions from 'OpenIntro' Textbooks.* https://cran.r-project.org/package=openintro.

Feinerer, Ingo, and Kurt Hornik. 2018. *tm: Text Mining Package.* https://cran.r-project.org/package=tm.

for R by Ray Brownrigg, Doug McIlroy. Packaged, Thomas P Minka, and transition to Plan 9 codebase by Roger Bivand. 2018. *mapproj: Map Projections.* https://cran.r-project.org/package=mapproj.

from Jed Wing, Max Kuhn. Contributions, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, et al. 2018. *caret: Classification and Regression Training.* https://cran.r-project.org/package=caret.

Hlavac, Marek. 2018. *stargazer: Well-Formatted Regression and Summary Statistics Tables.* Bratislava, Slovakia: Central European Labour Studies Institute (CELSI). https://cran.r-project.org/package=stargazer.

Jockers, Matthew L. 2015. *Syuzhet: Extract Sentiment and Plot Arcs from Text.* https://github.com/mjockers/syuzhet.

Studio, R. 2012. "RStudio: integrated development environment for R."

Wickham, Hadley. 2016. *ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York. http://ggplot2.org.

———. 2017. *tidyverse: Easily Install and Load the 'Tidyverse'.* https://cran.r-project.org/package=tidyverse.

———. 2018. *stringr: Simple, Consistent Wrappers for Common String Operations.* https://cran.r-project.org/package=stringr.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2018. *dplyr: A Grammar of Data Manipulation.* https://cran.r-project.org/package=dplyr.

Xie, Yihui. 2018a. *knitr: A General-Purpose Package for Dynamic Report Generation in R.* https://yihui.name/knitr/.

———. 2018b. *tinytex: Helper Functions to Install and Maintain 'TeX Live', and Compile 'LaTeX' Documents.* https://cran.r-project.org/package=tinytex.