A Study on US 2020 Election Tweets

Kyle Morris
Winter 2020
https://github.com/jkylemorris/DSC-Portfolio

The 2020 United States election garnered an incredible share of the zeitgeist and, as many Americans could attest, was interminable as despite a clear victor emerging in the days following the election, a concession was never made.

One of the ways that we can trace that is by looking at Twitter data in the time up to, during, and after the election. By concentrating on certain hashtags, we can scrape data directly from the source and, combined with timestamps, paint a picture of the overarching sentiment of the Twitter zeitgeist at that moment in time.

Now, we are focused in this project on the 2020 election but there are many cases where this analysis could be useful for business. Imagine a keynote presentation whereas new products are introduced we could identify not only the volume or engagement regarding that product on Twitter correlated with when the announcement was made, but also what the sentiment of said Tweets are i.e., are users overwhelmingly positive about the announcement or do they have some reservations? With the data in hand, we could easily pull verbatims in real time as well.

Other trends could emerge that as well would allow us to make use of these sentiments as well. We could run ads in a certain market and then identify using the geocache on the Tweets from that time period to see if that market reacted positively or negatively to that ad.

As a proof of concept, this project will be utilizing the collection of tweets sent prior to, during, and just after the election focusing on the hashtags #biden and #trump. Sentiment analysis will be performed, which essentially assigns and positive or negative score to each word and then sums the text of the tweet to give the net polarity. A tweet with generally negative words (awful, terrible, the worst) would have a negative score, while a tweet with generally positive words (amazing, stupendous, magnificent) would result in a positive score.

Our dataset is a collection of tweets that was gathered on Kaggle by Manchun Hui. They consist of tweets that were written between October 15th through November 8th and are divided into two datasets. One dataset is tweets that included the hashtag #DonaldTrump or #Trump, and with #JoeBiden and #Biden. They were collected using the Twitter API statuses_lookup and snsscrape for the aforementioned keywords. The author continued to add to the original dataset, starting with version 3 comprising of 355,000 tweets between October 15th and October 22nd, to the version that was completed that consisted of approximately 1,727,000 tweets from October 15th through November 8th, 2020.

For each of the 1,727,000 tweets in the dataset, the following attributes were catalogued, if they existed:

created_at: Consists of the date and time the tweet was created.

tweet_id: The unique identifier of the tweet.

tweet: The full text of the tweet in original form.

likes: The number of likes the tweet received, at time of collection.

retweet_count: The number of retweets the tweet had gathered, at time of collection.

source: The utility used to post the tweet. There were an astounding 916 different utilities used to post Trump tweets, and 750 utilities were used to post Biden tweets.

user_id: The user ID of the tweet creator. These are the numerical user ID, not the username.

user_name: The user name of the tweet creator. This would be the display name of the creator as opposed to their @username. That one is seen next.

user_screen_name: This would be the screen name of the tweet creator (more commonly recognized as their @name).

user_description: The user description for the tweet creator. This would essentially be your Twitter bio, the byline after your name and username.

user_join_date: This is the date_timestamp of when the user joined Twitter. It is in month / day / Year hour : minute : second format and is a 12 hour clock timestamp.

user_followers_count: An integer that stores how many followers that user has. The range is from 0 followers all the way up to 82,417,099 followers!

user_location: The location listed on the user's profile. Note that this is not the actual geocached location – it is a user edited field and is not restricted to the user's actual location.

lat: Given the user's reported user_location, the latitude of that location

long: The longitude parsed from the user_location much like lat. If no location is provided, both long and lat are blank. If the location does not map to an actual location on Earth (i.e. Earth) then no location is provided. It appears that the script works only with English place names – the names of foreign cities seem to have not been mapped.

city: City name derived from user_location.

country: Country name derived from user_location.

state: State name derived from user_location.

state_code: 2-character state code from user location. These last 6 entries are all derived from the user_location and not necessarily the geotag on the tweet. As such, it relies entirely on user supplied data and is not necessarily truly indicative of where the tweet originated.

These are the data columns we must work with. In the first part of our exploration, we will be concentration on two columns in particular: the tweet column, which contains the plain text of the actual tweet, and the timestamp. The actual tweet will be analyzed and assigned a sentiment polarity, which uses the actual language of the tweet, and the timestamp will allow us to look at Tweet density on a chronological basis to see how sentiment changed, ebbed, and flowed over time.

Now that we have identified our dataset, we need to clean up and import our data. For this project, we will be working exclusively in Python for this portion of the project. A Pandas dataframe is useful for importing data like this, not only because we can perform a lot of cleanup functions on the dataframe but it also makes it quite easy to both import and export our data into a pandas dataframe. Our dataset was provided in the CSV or Comma-Separated Values format, which is a comma delimited

text file useful for storing data. The CSV format is somewhat standardized, and makes importing and exporting the data relatively painless, although care must be taken to make sure that the imported dataset was stored correctly.

We made use of the Pandas read_csv method on both the Trump data and the Biden data. For the Trump data, 971157 rows by 21 columns were read. For the Biden data, 777,078 rows by 21 columns were read. Of these, approximately 100 or so rows did not import correctly – they were missing a few values, which caused for instance the tweet field to end up in the created at line. If we were going to use this column as a timestamp, plain text could most assuredly not remain in our dataset and were so omitted. These omitted rows were on the order of 0.0057% or for every row that was omitted, 17,270 tweets remained. We are confident that their omission did not affect the overall sentiment analysis.

Next, we needed a method that would strip the special characters out of the tweet. This includes punctuation, accents, and other special characters that could confuse our sentiment analysis and is often one of your first orders of business when doing any sort of text analysis. Instead of creating the method when we run it, we created a clean_tweet method that took a string (in this case, the tweet text itself) and returned a copy of the original tweet with all these characters stripped out. We then created a new column, cleantweet, and kept the original text. This decision was made in case we wanted to pull specific examples of tweets out after the fact – the cleaned version, while more useful to natural language processing methods, shifts the needle from human readable to machine readable.

We then needed to perform the actual sentiment analysis. We used a library called TextBlob, which essentially attempts to add natural language processing abilities to Python strings. TextBlob builds off the Natural Language ToolKit, and in fact we used the corpora from the NLTK to give context to our TextBlob installation. The function we created essentially allowed us to pass the string (in this case, the cleaned-up version of the tweet) to our TextBlob object and have it return the sentiment polarity. The sentiment polarity is essentially a normalized value between -1.0 and +1.0 that attempts to classify the overall sentiment of the body of text it was loosed upon. The magnitude of the polarity indicates how strongly positive or negative the text is. A positive value indicates that the tweet is generally positive, a negative value indicates that the tweet leans negative, and a value of 0 denotes a neutral tweet.

We created a new column in our Pandas dataframe that stored the results for each row. That way, we could access the sentiment value or polarity of the tweet instantly without having to run it through our TextBlob repeatedly. To save time and to give us more data to work with, we also created a sentiment column that classified each tweet as Positive, Negative, or Neutral. This lets us identify not only the overall score for each moment in time but knowing where the needle pointed for each tweet can provide a wealth of data. Were there actually more negative tweets than positive that hour, but the negatives were only slightly negative while the positives were tremendously positive and that tipped the scale?

With the dataset cleaned up and a few new columns created, we exported the data in a format that would work with Tableau. Although we initially stuck with the .CSV format, Tableau had difficulty importing that much data in this form and so we exported in the Microsoft Excel format instead. This was a much cleaner import into Tableau and resulted in much more accurate visualizations.

The data points were grouped by hour over the course of the dataset. Though we initially had it grouped by the second, we found that our picture told a better story of the peaks and valleys when grouped over a larger time period. Here are the results of our sentiment analysis graphs:

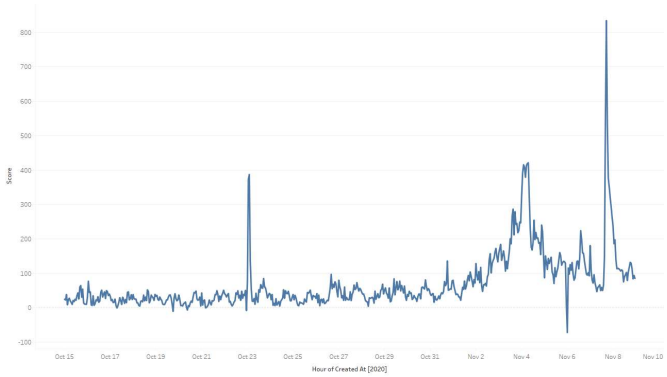

Figure 1: Trump Sentiment Score, Hourly

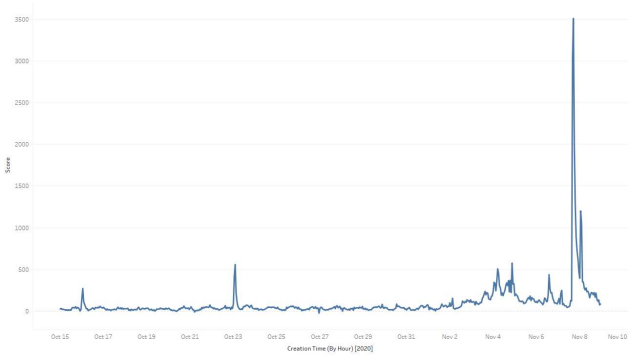

Figure 2: Biden Sentiment Score, Hourly

For comparison's sake, here is the Trump Sentiment Score on a minute-by-minute basis.

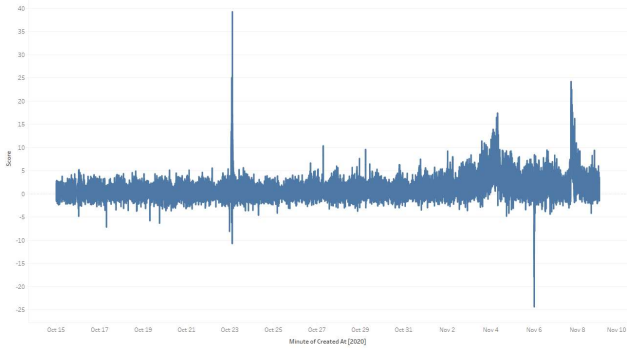

Figure 3: Trump Sentiment Score, Minutely

As you can see, we have a lot more variance in the minutely chart compared to the hourly ones. We felt that the hourly version does a better job of adequately capturing the snowball effect on Twitter. Our next task was to identify some of those peaks and see if we cannot discover what happened to cause those flurries of tweets.

For our Trump data, we identified 3 peaks and 1 valley. On October 23rd at during the 2 AM hour we reached a score of 387. Between 2AM and 7AM on November 4th we saw another peak between 418 and 421. On November 6th during the midnight hour support for Trump plummeted reaching a negative 71.8 polarity. Finally, November 7th at 5PM we saw support soar to an astounding 834.3.

What could have caused these shifts? The November 4th data point may be the easiest to identify as that happened in the wee hours of the morning the day after the election. At approximately 11:20PM Fox News called Arizona for Biden – leading to the 2:30AM address to his supporters by President Trump. This hour as well featured the Associated Press calling Arizona for Biden as well.

October 22nd was the date of the final presidential debate. When pulling a few of the tweets it does appear that the surge was caused by the debates as the overwhelmingly positive tweets reference debate performance.

President Trump held a press conference the evening of November 5th. The negative tweets in the midnight hour tend to revolve around this press conference and it seems safe to say that the Twitter population did not react overwhelmingly positive to it.

Finally, what about our enormous spike at around 5PM on November 7th? November 7th was the day that, with the Associate Press calling Pennsylvania for Biden. This was also the day the Four Seasons Total Landscaping press conference was held. In this case, one of the pitfalls of sentiment analysis is observed. Since we used tweets that used Donald Trump's name, we have a lot tweets that say "Goodbye Donald Trump you're fired, I'm optimistic for the future" which results in a tweet with a positive sentiment, just not in the way President Trump would prefer.

On the Biden side, we see no negative peaks. We see positive spikes on October 16th at 1 am (score 271), October 23rd at 2AM (559 score), November 4th at 5AM (506 score), November 4th at 9PM (576 score), November 8th at 1AM (1,202 score), and finally a massive spike at 5PM on November 7th (3,512 score)

Some of these times coordinate. Our massive spike on November 7th does correspond with the cementing of 270 electoral votes for the President-Elect. October 23rd was the same spike we saw for the final debates. November 4th was likely the post-election morning when the US awoke to see the tides shifting for Joe Biden. That evening was when Biden had amassed approximately 264 votes. The 1AM spike on November 8th seems to be in reaction to Joe Biden addressing the nation. Finally, the October 16th spike seems to correlate with the Town Halls held in lieu of the second Presidential Debate.

As an additional exercise, we prepared a blended sentiment that included both the Trump and Biden tweets to show the overall sentiment of the election. Given that the magnitude of Biden's sentiment scores outpaced those of President Trump's, it does very much resemble the Biden chart.

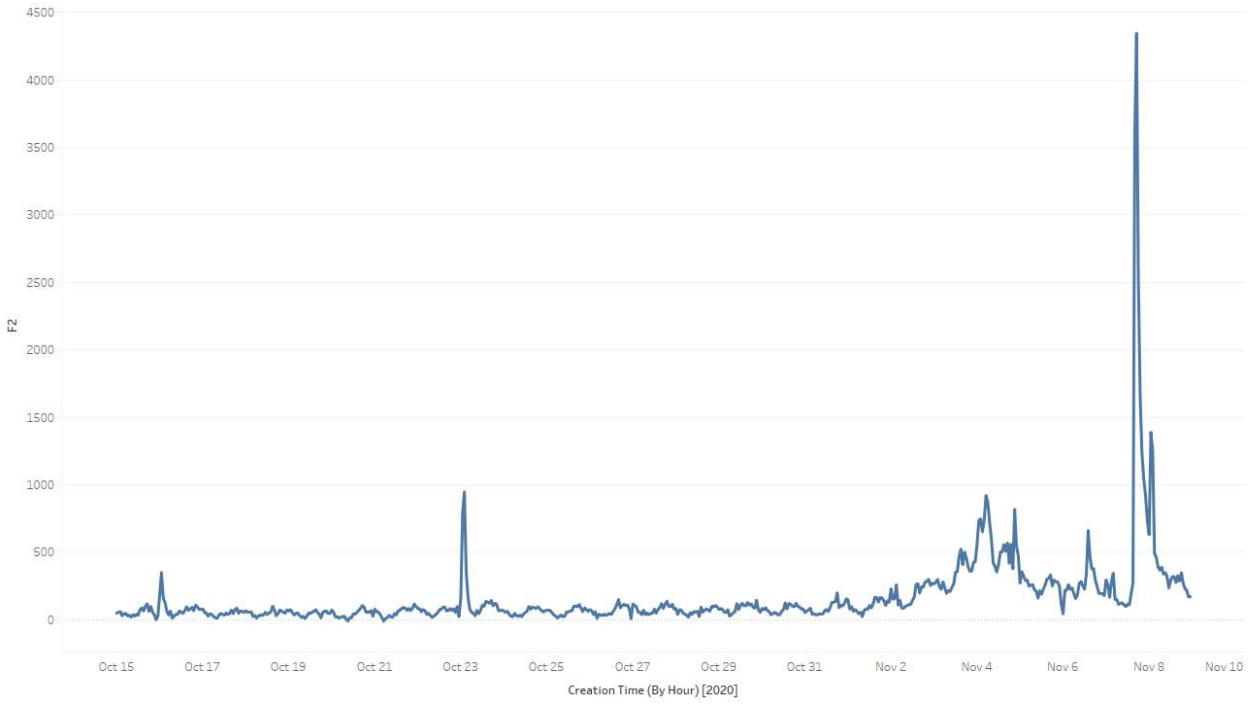

Figure 4: Election Sentiment Score, Hourly

As you can see, sentiment analysis is a useful (and quickly deployed) way of quickly getting a feel for both the volume and sentiment of user opinion. This could prove useful for tracking customer response to new events, keynote presentations, and new product releases. Analysis can be dialed in to the individual tweet level or cover several months' worth of tweets.

Technology and real time analysis of data has allowed us to conduct analysis far faster than could be accomplished in the past. A study of polls, statements by voters, and analysis of the zeitgeist previously could have taken months. In this case, our data was ready for us the day after it happened.

Given that the data is in a raw, unrefined form the cleanup step took significantly more effort than a study designed from the beginning to source required data. However, the tradeoffs will be worth it. We hope to have a sense of different factions of the hivemind that is Twitter reacted in the run up to the election and how that shifted over time.

Appendix

References

China tweet that enraged Australia propelled by 'unusual' accounts, say experts. The Japan Times. (2020, December 5). https://www.japantimes.co.jp/news/2020/12/05/asia-pacific/politics-diplomacy-asia-pacific/china-tweet-australia/.

Cunningham, M., Karson, K., & Scanlan, Q. (2020, November 11). *Key post-election certification deadlines and dates*. ABC News. https://abcnews.go.com/Politics/key-post-election-certification-deadlines-dates/story?id=74157237.

Fryling, K. (2020, November 10). *5 mistakes people make when sharing COVID-19 data on Twitter*. News at IU. https://news.iu.edu/stories/2020/11/iupui/releases/10-covid-19-data-visualization-errors-study.html.

Heiduk, J. (2018, March 22). *Twitter Analysis with Python*. DataScience+. https://datascienceplus.com/twitter-analysis-with-python/.

Jacob, T. (2019, May 21). *Australian Election 2019 Tweets*. Kaggle. https://www.kaggle.com/taniaj/australian-election-2019-tweets.

Kim, R. (2017, December 8). *Another Twitter sentiment analysis with Python - Part 1*. https://towardsdatascience.com/another-twitter-sentiment-analysis-bb5b01ebad90.

Kiss, M. (2020, August 26). *10 Tips for Presenting Data*. ObservePoint. https://resources.observepoint.com/blog/10-tips-for-presenting-data.

Kumar, N. (2020, July 16). *Twitter Sentiment Analysis using Python*. GeeksforGeeks. https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/.

Sistilli, A. (2017, June 7). *Twitter Data Mining: A Guide to Big Data Analytics Using Python*. Toptal Engineering Blog. https://www.toptal.com/python/twitter-data-mining-using-python.

Sleeper, R. (2020, August 15). *How to Make a Timeline in Tableau*. Playfair Data. https://playfairdata.com/how-to-make-a-timeline-in-tableau/.