Data Pipeline for Analyzing Tweet Content: How to Win Twitter Followers and Influence People

Contributors: Mikaela Hoffman-Stapleton, Erin Chinn, Arda Aysu, Kelsey MacMillan
Date: 10/13/2016
Class: MSAN 692, Final Project

1. Introduction

Many types of organizations, including clothes brands, food brands, and political campaigns use Twitter to interact with customers (or voters) and gain publicity. To be effective in attracting consumers and creating a positive brand image, they need to understand what content Twitter users are most likely to engage with. For instance, what blogs or media sources do they most frequently link to? What topics and keywords see the most activity? And what are their feelings on those topics? We can look at the recent tweets of groups of Twitter users and extract patterns in tweet content. Knowledge of these patterns can help organizations decide on the best content for their own tweets.

As an illustrative example of tweet content analysis, we have analyzed two groups of Twitter users based on the 2016 presidential campaigns. We selected users that recently tweeted with the hashtag #NeverTrump as a proxy for probable Hillary Clinton supporters, and also selected users that recently tweeted with the hashtag #MakeAmericaGreatAgain as a proxy for probable Donald Trump supporters. We make the assumption that users are not using these hashtags ironically, although it is inevitable that this will happen. In a real-world implementation, we might instead choose to get a list of the Twitter accounts of campaign donors, and base our tweet content analysis off of those accounts. That said, one advantage of using a hashtag instead of a more static list to create groups of users is that the dataset is more reactive to current events. For instance, after emails regarding Hillary Clinton's speeches to Goldman Sachs were revealed, we saw that "wikileaks" became an important term for users in the #MakeAmericaGreatAgain group.

We used Twitter's API to gather data on the most recent tweets by users from each of these groups, and then analyzed the content of those tweets to find common patterns. Although the specific user data and analysis can be easily augmented, we have chosen to perform the following analyses for our illustrative example:

- Create a word cloud of the most "important" terms in tweets from users in the group.
- Generate a list of the most commonly linked domains in the group.
- Generate a histogram of tweet sentiments for a group.
- Analyze the average group sentiment for specific topics.

This analysis yielded many insights, which are summarized in greater detail below. We discovered for instance, that #NeverTrump users often linked to traditional news sites, especially those that are sometimes characterized as more-left leaning, such as the New York Times, Huffington Post, and CNN. On the other hand, #MakeAmericaGreat users very often linked to Donald Trump's campaign website, as well as politically conservative opinion sites like hannity.com and Breitbart News. The word cloud revealed that while both "Hillary" and "Trump" were some of the most frequently tweeted words for both groups of users, "Trump" was typically more frequent than "Hillary" in both groups. And while "woman" was very important in the #NeverTrump group, it was virtually non-existent in the #MakeAmericaGreatAgain group. Based on these insights, a social media manager for Hillary Clinton, for instance, might choose to compose tweets linking to New York Time editorials on Trump's behavior regarding women.

2. Data Collection Technical Details

For the purposes of discussion, we will divide the data into two types:

1) First-level: data that comes directly from Twitter content and metadata, such as tweet text, user location, and tweet URLs.
2) Second-level: data that is produced by running an analysis on the first-level data.

2.1 First-Level Data

All first-level data was produced by querying Twitter's Public REST API (https://dev.twitter.com/rest/public) via a Python wrapper called Tweepy (https://dev.twitter.com/rest/public).

Three API calls were used:

- search
- user/show
- statuses/user_timline

Search is used to get a list of tweets with a given hashtag. From this list of tweets, we generate a list of users. This API call is limited to 300 tweets per search, so we had to do multiple queries until we got a sufficient number of users. By default, the Twitter API returns the most recent tweets. To make sure we got new tweets in every query we used a feature of the API that allows you to set a value for the max tweet ID (i.e. the most recent tweet the query should return). To ensure that we did not get duplicate users in the final list, we stored the list of users as a set.

User/show and statuses/user_timeline are used, respectively, to get data for each user and to get their most recent tweets (up to 300 per user). The key issue here is that Twitter limits the user/show request to 180 requests per 15 minutes. To obviate this limitation, we passed all API requests through an exception handler function that would wait 15 minutes when it encountered a rate limit exception, and then try the request again.

Data from the API is stored in memory as a list of Python dictionaries. Each dictionary is a user with key fields for:

- User ID, “uid”
- Screenname, “screen_name”
- Language, “language”
- Location, “location”
- Tweets, “tweets”

The value associated with the “tweets” key is itself a list of dictionaries, with each dictionary representing a single tweet with key fields for:

- Tweet raw text, “text”
- Links in tweet, “urls”
- Hashtags in tweet, “hashtags”

The benefit of using a dictionary to represent each user, is that it is easy to augment the data associated with user by simply adding a new key-value pair to the dictionary.

2. 2 Second-Level Data

To be as efficient as possible with API queries and minimize rate limit issues, all first-level data is collected with exactly one query to the Twitter API of each type (User/show and statuses/user_timeline). Then all second-level data is generated by analyzing the resulting list of user dictionaries. No further requests are made to the API.

The results from the second-level data analysis are stored as new key-value pairs in the list of user dictionaries. Again, this is an extensible data storage model.

Domain Data

To generate data for the domains most commonly linked to it was necessary to make requests to all the links in each tweet because many links were generated by a shortening service like Bitly and redirected to different websites. Runtime proved to be a major challenge with the domain data collection. The runtime fluctuated dramatically from run-to-run, and could take up to 10 minutes for 1000 tweets. To speed up the process, we implemented a function to gather all URLs into a single list, make the URL requests in parallel, and then map the results back into the user dictionary objects using a url-to-domain hashmap. The asynchronous requests were done using a Python library called grerequests (https://github.com/kennethreitz/grequests). Although the use of parallel processing made the domain data generation four times faster, it still limits us to only a couple thousand tweets per group if we want to get data within a 15 minute time frame.

TFIDF Data

To generate tfidf data on the two user groups we aggregated all the tweets from each individual user. Each user's tweet aggregation was then compared to a corpus of tweets we generated from neutral, non-political Twitter accounts, specifically from @CocaCola and @Gap. The corpus included 6000 tweets, 3000 from each of the retailers. The corpus was generated once and stored to file to prevent rate limit issues with Twitter's API. However, it could easily be regenerated with new tweets and/or different Twitter accounts by re-running the script.

In our preliminary tfidf analysis we pulled the corpus from two large, theoretically unbiased Twitter accounts from popular news sources, @latimes and @nytimes. However we found that their Twitter coverage of political issues was quieting the political discourse of the two user groups in tfidf analysis. To obtain an accurate tfidf score it is important to compile a corpus that is unbiased in 2 ways: 1. unbiased in political stance, i.e. not favoring one candidate over another and 2. unbiased in rhetorical content, i.e. the topics discussed cover a variety of genres (politics, sports, entertainment, pop-culture, etc).

All tweets, including aggregates from the users as well as the corpus, were cleaned using regular expressions. We eliminated hashtags (e.g. #imwithher), direct Twitter mentions (e.g. @parrt), and URL's. Prior to removal, each produced nonsensical "words" that could potentially cloud the analysis. Words less than 3 letters long and common english stop words, from Sci-Kit Learn's library, were removed as well.

We used Sci-Kit Learn's tfidf vectorizer to compare user tweet aggregates to the corpus and generate tfidf scores on each word. Sci-Kit Learn's fit_transform function requires all input be in the same form. Because the corpus was previously generated and saved to file, each user's tweet aggregate had to be saved to file as well. To prevent generating an overwhelming collection of user files, this file was written and then subsequently overwritten with the content from each following user.

As a preliminary analysis of the differences in status content between the two user groups, we generated word clouds of the most popular tweeted words using the Python library wordcloud (https://github.com/amueller/word_cloud). A visual representation of the content can be more powerful than a numeric comparison in describing the differences or similarities between Twitter groups.

Sentiment Data

In order to analyze the sentiment of individual tweets, we implemented the textblob Python library (https://textblob.readthedocs.io/en/dev/). The raw text from each tweet was used as an argument to TextBlob(), which created a “blob” object that could then be used to extract useful information about the text. In our case, we called blob.sentiment.polarity to ascertain the sentiment of the text. This returned a number on a scale from -1 to 1, where -1 is considered the most negative sentiment, 1 is considered the most positive sentiment, and 0 is neutral.

Topic Data

To gather topic data, we first hard-coded a short list of political topics that could be used to identify a political tweet. Each individual tweet was cleaned, tokenized, and stemmed to see whether or not it shared any terms with the stemmed list of political topics. All shared terms were added to the “topic” field for that tweet. This allows us to go back and check whether or not a tweet was political (whether the length of the topic field is non-zero), and if so, what political terms it contained. With this information we can create further subsets of our users and their tweets.

3. Summary of Findings

To collect all the first-level and second-level data described in this report requires about 1 minute of analysis per 300 tweets. To allow faster analysis functions (like sentiment analysis) to collect more data, we turned off the slower functions (like getting domain data) and increased the number of users and number of tweets.

Top Domains

For the top domains analysis, we collected data on 30 users per group and 30 tweets per user for a total of 900 tweets per group. This quantity of data was sufficient to produce meaningful insights.

Collecting data on top domains linked in tweets revealed some interesting findings. Table 1, below, contains the top ten websites for each group of users. The #NeverTrump (proxy for Hillary supporters) more often linked to traditional news sites like the New York Times and MSNBC. CNN and the Washington Post often appear in this list as well. These are also news sites that in some circles are accused of having a liberal bias.

Users in the #MakeAmericaGreatAgain tended to link to more social media and punditry sites. They also frequently linked to Donald Trump’s campaign website. The converse was not true for the #NeverTrump group. We never saw Hillary Clinton’s campaign website on the top domains list.

<u>Table 1</u>: Top Ten Linked Domains by Group (generated 10/9/2016).

Rank	#NeverTrump	#MakeAmericaGreatAgain
1	www.nytimes.com	www.youtube.com
2	www.facebook.com	www.breitbart.com
3	www.youtube.com	www.donaldjtrump.com
4	www.msnbc.com	www.instagram.com
5	www.iontelevision.com	www.thegatewaypundit.com
6	www.huffingtonpost.com	www.facebook.com
7	bipartisanreport.com	dailycaller.com
8	www.washingtonpost.com	www.infowars.com
9	www.politico.com	www.foxnews.com
10	www.iwillvote.com	wikileaks.org

TFIDF for "Important" Words

The word clouds shown below were produced with 50 users and 50 tweets per user for each group. The images produced from each Twitter group were mildly surprising. While both groups mentioned their political rival frequently, Trump was discussed more than Hillary in both groups. One interpretation could be that the Hillary Twitter group focused more on the dislike of Trump as a candidate and less about support of her as a candidate.

The word clouds also demonstrated the differences in gender discussions between the two Twitter groups. The Hillary follower word cloud included gender specific terms such as "women" and "woman." The Trump follower word cloud contained "women" as well but at a much lower frequency. In addition the Trump group cloud contained derogatory gender terms. This speaks to similarities between Trump's own dialog during the campaign compared to the tweet dialog amongst his Twitter followers.

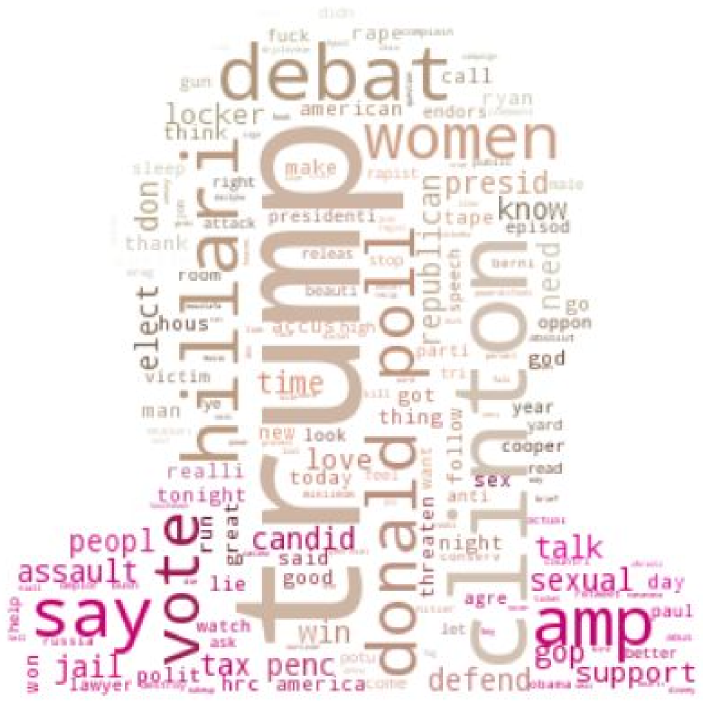


Figure 1: Word cloud based on tfidf for #NeverTrump (generated 10/10/2016).

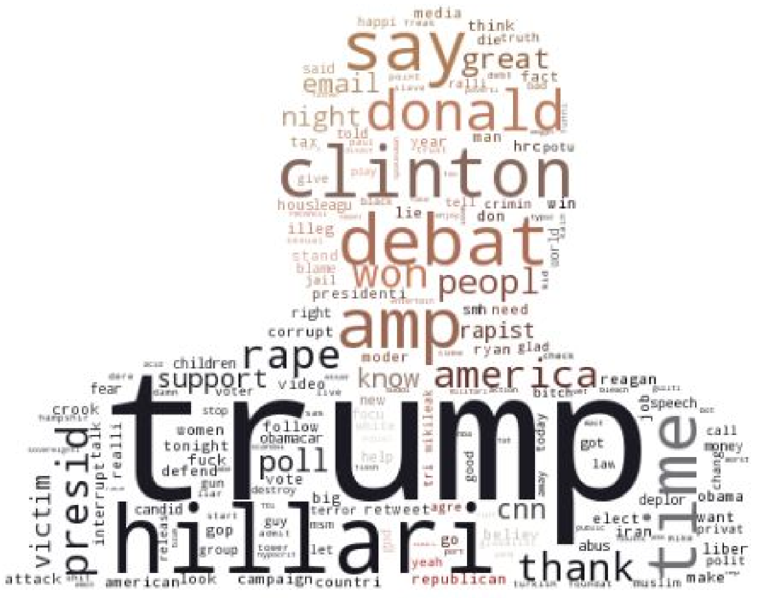
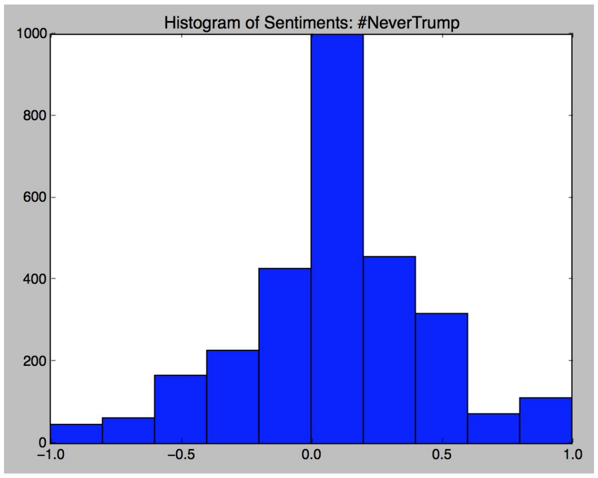


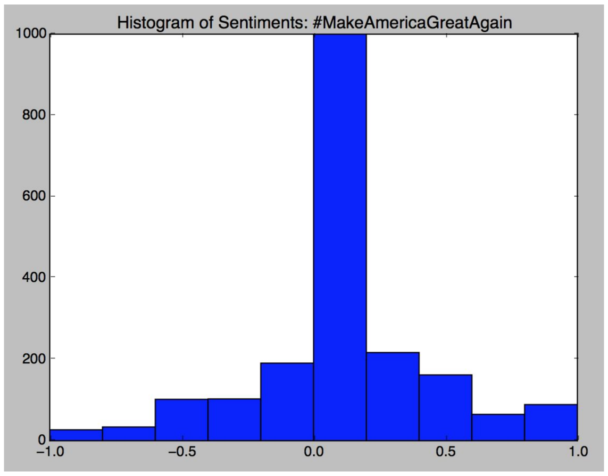
Figure 2: Word cloud based on tfidf for #MakeAmericaGreatAgain (generated 10/10/2016).

Sentiment Analysis

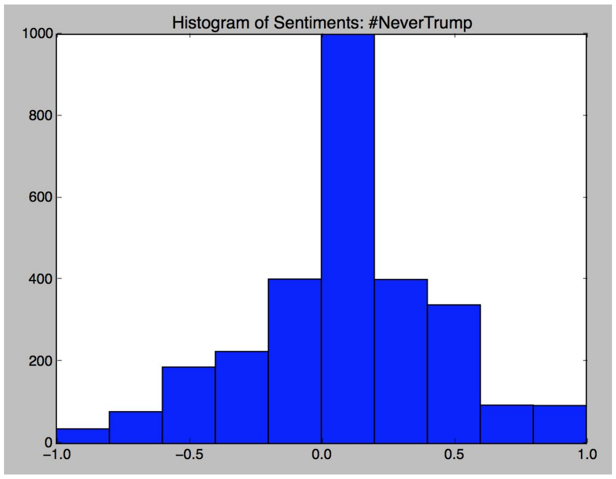
We collected 50 tweets per user for 50 users for both groups, and repeated this process after an hour had passed. We were interested to see how the average sentiments for each group compared. For the first cycle, we found that #NeverTrump users had an average sentiment of 0.051, while #MakeAmericaGreatAgain users had an average sentiment of 0.068. For the second cycle, we found that #NeverTrump users had an average sentiment of 0.050, while #MakeAmericaGreatAgain users had an average sentiment of 0.073. Histograms of the sentiment scores for each group can be found below.

We see consistency among the histograms and averages for each group, and also notice that #MakeAmericaGreatAgain users have a slightly more positive average. However, this difference is minimal; we do not suggest that these results imply that Trump supporters are more positive than Hillary supporters. We note that the majority of tweets were scored as neutral. It would be interesting to explore libraries other than textblob that may be better at recognizing more nuanced reactions and tones.





<u>Figure 3</u>: Histograms showing sentiment scores for #NeverTrump and #MakeAmericaGreatAgain (first cycle, generated 10/10/2016). Note that the largest bin is cut off so that the scale is reasonable.



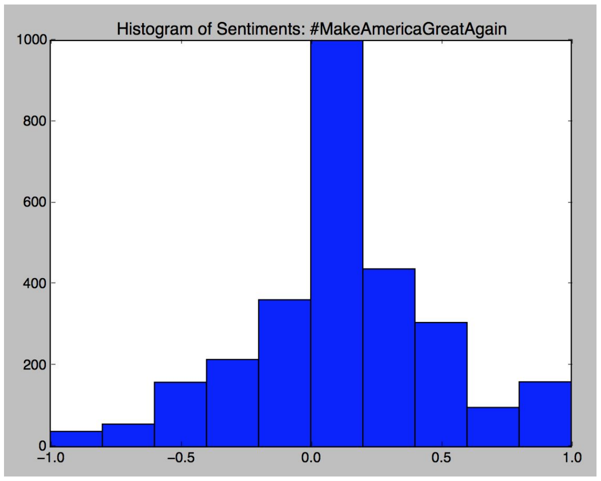


Figure 4: Histograms showing sentiment scores for #NeverTrump and #MakeAmericaGreatAgain (second cycle, generated 10/10/2016). Note that the largest bin is cut off so that the scale is reasonable.

Topic Analysis

We questioned if there was a difference in rates of political content between users that had tweeted with #NeverTrump or #MakeAmericaGreatAgain. Looking through 100 users per group and 100 tweets per user, our limited list of political terms was able to identify over 98% of users as having made at least one political tweet. While theoretically we should be reaching 100%, this is still a high enough percentage for us to assume reasonable accuracy in finding political tweets.

Initial findings show that the #MakeAmericaGreatAgain group has a lower percentage of tweets that were identified as political according to our list of political terms, which suggests they may be slightly less politically-engaged than those in the #NeverTrump group. Another interesting result is that compared to the sentiment scores on all tweets, both groups see a decrease when looking only at their political tweets. The political sentiment score is still positive and the two groups have much more similar scores. This decrease in sentiment for both Hillary and Trump supporters did not come as a surprise, as this election has not generated much positive coverage on either side.

Table 2: Frequency and Sentiment of Political Tweets (generated 10/10/2016).

	#NeverTrump	**#MakeAmericaGreatAgain**
Avg % of Political Tweets	0.373639	0.337835
Avg Sentiment of Political Tweets	0.041625	0.042609

4. Further Work

For future analysis we suggest storing the lists of Twitter user dictionaries as JSON files. JSON files are well suited to nested dictionaries and would enable more data collection over the course of multiple code runs. This would also solve a problem we encountered where the Twitter API has a fatal error during our data collection, which halts and essentially breaks the program. While this issue could partially be solved by more sophisticated error handling, saving data periodically over a series of runs would provide a failsafe that protects the program from losing all the data collected before the Twitter API error.

Another consideration for the future would be to generate the list of topic keywords using a more sophisticated and dynamic data pipeline. One possibility would be to scrape Buzzfeed or New York Times top stories for high frequency words, and then stem those words and feed them into our tweet topic classification function. Additionally, future analysis might include how much Twitter users engage with a specific tweet, e.g. number of retweets, or number of replies. Potentially we could then use that information to weight particular Twitter users and their tweets higher in our analysis.