Natural Language Processing:
Understanding a Twitter Influencer

1. The Problem Statement

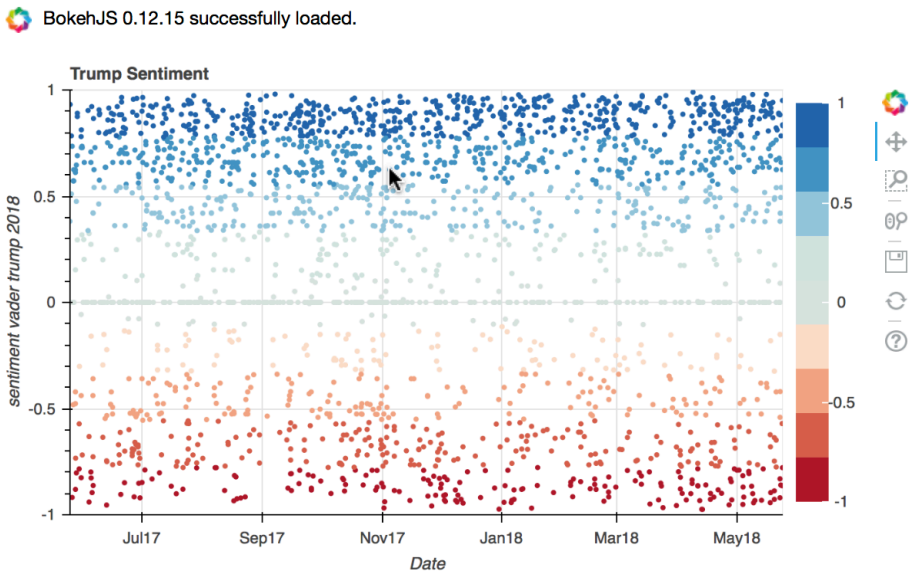
Sometimes a client, a celebrity, or a company can exert undue influence upon a product, a company, an industry, or a market. We will look at how a company can use Natural Language Processing (NLP) to understand an influencer. The president of the United States is such an influencer. We will look at Donald Trump's tweets to see if we can gain insight into his influence. We start with looking at the sentiment of his tweets. Knowing rather an influencer is in a good mood today (positive sentiment) can give insight into interactions with that influencer. We might pitch an idea on a good day, or if making an investment decision wait for a better day. Of course, these decisions are all made based on industry knowledge. We look to provide some tools using NLP to assist in making these influencer based decisions.

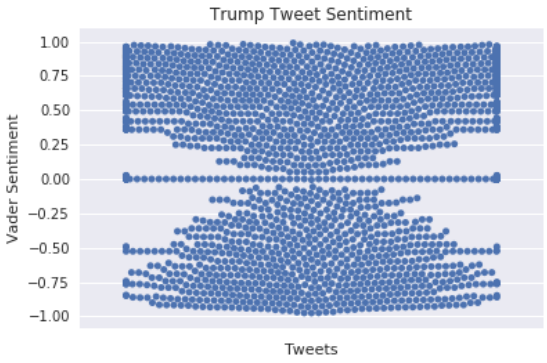
2. Data Collection and Processing

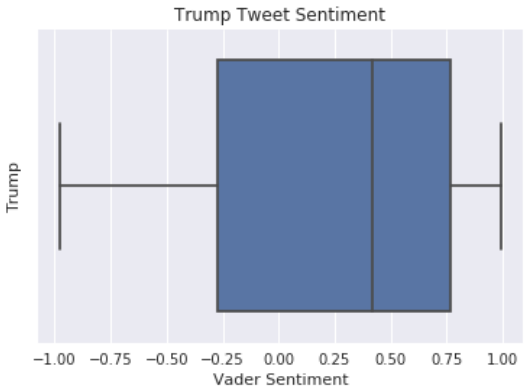
First we need to get some data. We scrap the web to pull in tweets from the last year. We use some Python code developed by Jeremy Cunningham to retrieve these tweets. This saves the tweets for the last year into a csv file. We read these in to a Juypter Notebook with Pandas. We change a few import setting about delimiters. Not all tweets have text (some have only pictures), so we drop all tweets without text. We reorganize the index, and convert the date and time text to datetime objects. We also clean the tweet text for NLP processing.

3. Trump Sentiment Analysis

We use the NLTK package, which has some pre-trained algorithms for sentiment recognition. One is the Vader Sentiment analyzer, which we download. We write a function to score each tweet for the sentiment based on a pre-trained word lexicon. The algorithm then calculates the sentiment based on the word weights. It scores on a scale ranging from negative (-1) to neutral, (0) to positive (1). We plot this using Bokeh for an interactive graph. We use this to explore the data; zooming into a specific time span, looking at positive tweets, or isolating a tweet.



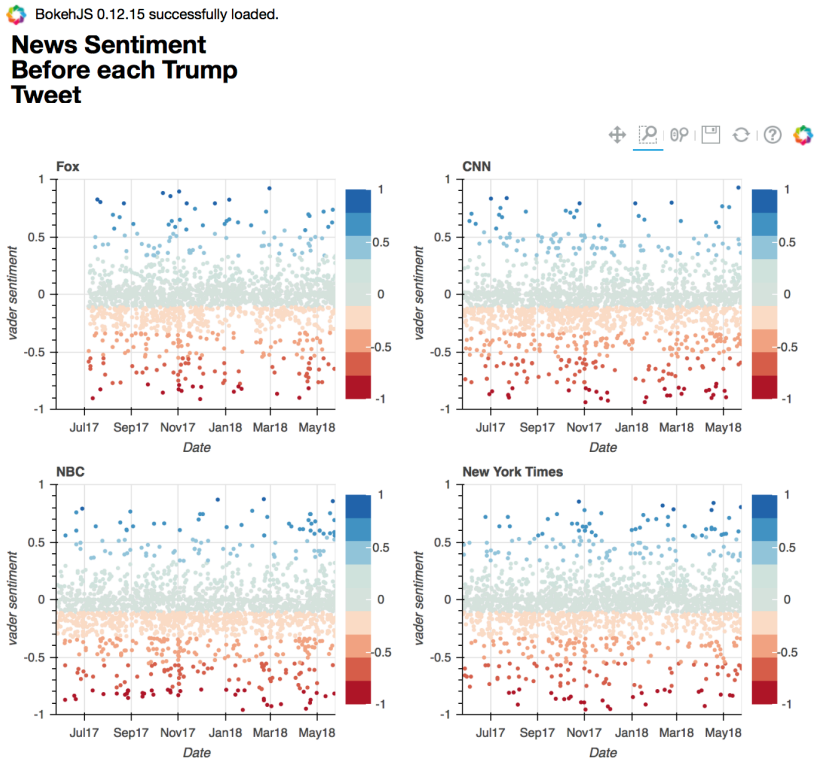




We focus our exploration on the sentiment analysis for the moment. We look at some basic statistics and see that Trump has a positive twitter feed for the last year. The mean of his sentiments is 0.231. The median values is 0.4199 not super positive, but positive.

4. News Sentiment Analysis

Next, we look at the sentiment of news broadcasters before Trump tweets to see if the news broadcasters influence Trump. The goal is to better understand an influencer, so understanding who influences him could help. We look at CNN, FOX, NBC, and the New York Times. We use the same scraper to get csv files for their tweets. We clean the data and keep only tweets with text, as we did with Trump. We get the sentiment for each tweet. We convert the sentiments from an object type to a float type. Next, we call a function that collects all the tweets from the news broadcaster that occurred between Trump tweets. We calculate the average sentiment of these tweets and plot with Bokeh.



The sentiments are gathered around the neutral line, and are not as negative or positive as Trump. We can once again interactively explore these tweets; zooming in and panning around. Moving on to statistics, we see that the mean sentiment for each broadcaster is slightly negative.

	cnn_sentiment_before_tweet
count	2146.000000
mean	-0.084623
std	0.242029
min	-0.939300
25%	-0.178769
50%	-0.067305
75%	0.008474
max	0.927400

	fox_sentiment_before_tweet
count	1846.000000
mean	-0.046573
std	0.233220
min	-0.911800
25%	-0.132844
50%	-0.029957
75%	0.049975
max	0.921100

	nbc_sentiment_before_tweet
count	1974.000000
mean	-0.092030
std	0.257940
min	-0.959500
25%	-0.198590
50%	-0.082350
75%	0.008679
max	0.873300

	nyt_sentiment_before_tweet
count	2073.000000
mean	-0.052120
std	0.246312
min	-0.956500
25%	-0.148000
50%	-0.033667
75%	0.044368
max	0.851900

How much does the tweets before affect Trump's sentiment? We look at the correlation. The numbers don't look encouraging. Investigating the tweets before each trump, the topics vary widely. We might be able to find a better correlation if we classify tweets by topic. This is what we will attempt next.

	sentiment_vader_trump_18
sentiment_vader_trump_18	1.000000
fox_sentiment_before_tweet	0.000538
cnn_sentiment_before_tweet	0.066650
nbc_sentiment_before_tweet	-0.001874
nyt_sentiment_before_tweet	0.060123

5. Topic Extraction with Latent Dirichelt Allocation

We used a Latent Dirichelt Allocation transformation to extract topics. This process uses a probability distribution over words. The topic distribution is taken to be a sparse Dirichlet prior (the probability distribution topics exhibit before evidence is taken into account). The idea is that tweets cover only a small set of topics and that topics use a small set of words frequently. A topic is based off of the likelihood of term co-occurrence (how often words appear together in a tweet). Words can occur in several topics with different probability, but with different neighboring words for each topic. For example: "America" can have two probabilities for separate topics like "economy" or "war". The probability that "America" is talking about economy is based off of the words in a tweet that occur near "America" and words in the economy topic.

5.a.) Overview:

A quick overview of the process. We first tokenize the tweets in order to form a “Bag of Words” representation. Then the “Bag of Words” is transformed by a TF-IDF (Term Frequency-Inverse Document Frequency), which weights the word frequency by the inverse of its popularity in all tweets. We do this to add a measure of term importance. In essence, this now discounts words that appear in all tweets and increases rare words. Afterward, we apply the LDA transformation to extract topics. We then tune the parameters and reapply LDA. Finally, we visualize the data.

5.b.) Pre-Processing for Topic Extraction

Here, we need tokens. We take the tweet text and using NLTK’s tokenizer we both lemmatize and tokenize the tweet text into a list of words. We also remove any English ‘stopwords’, words that appear frequently but have little topic information such as “the”, and “is”. We save the list of words into a list of lists. (If the data was larger, we could output to a file and later read in line by line.) We now switch to the Gensim package to finish topic extraction. We transform the nested list of words into a "bag of words" using Gensim dictionary and corpus functions. We then transform this vector space (the bag of words) into the TF-IDF vector space followed by a chained transformation into the LDA vector space.

5.c) LDA Model Tuning

Gensim LDA transformer requires the bag of words corpus, the dictionary, and the number of topics. We start with 10. The problem of picking the best topic number is similar to trying to pick the right number of clusters in unsupervised clustering. The most common techniques in clustering is the elbow method, the silhouette, and the gap. We explore some of these techniques later in optimizing the LDA model. We set passes to 20 and iterations to 400. This allows the expectation propagation, a Bayesian machine learning technique, to approximate the probability distribution by convergence. Alpha is the smoothness hyper parameter, which ‘auto’ allows asymmetric learning from the data. A small alpha makes each tweet more specific (fewer topics). A big alpha allows topics to cross pollinate (tweets can be in multiple topics). When alpha is to big then tweets appear in to many topics..

```
LdaModel(corpus_lda, id2word=dictionary, num_topics=10, passes=20, iterations = 400, APLHA= ‘Auto’, ETA= ‘AUTO, EVal_EVERY= None)
```

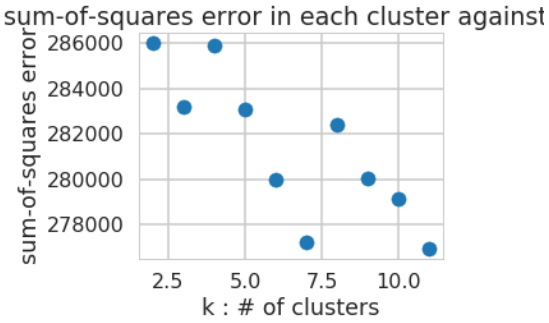
This model now can transform any tweet (or word) into our vector space of trump tweets. We see that the result is the probability of the tweet relating to one of our topics. We can use this to tag tweets by topic.

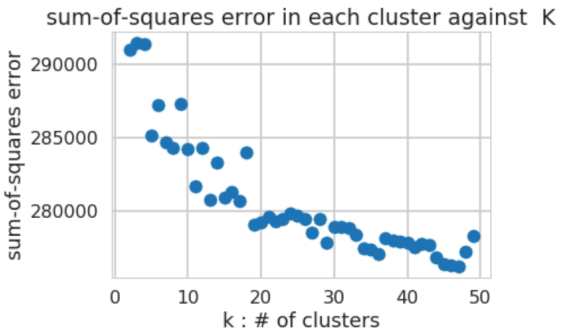
Before that, we first explore how well the topics are formed. The topics look like this:

```
[(0,
  '0.020*"great" + 0.011*"president" + 0.011*"thank" + 0.008*"honor" + 0.008*"thought" + 0.007*"prayer" + 0.007*"france" + 0.007*"melania" + 0.006*"paris"
+ 0.005*"people"'),
 (1,
  '0.011*"news" + 0.010*"fake" + 0.010*"great" + 0.009*"people" + 0.009*"thank" + 0.007*"join" + 0.007*"medium" + 0.007*"trump" + 0.006*"usa" +
0.006*"live"'),
 (2,
  '0.015*"hillary" + 0.014*"great" + 0.013*"job" + 0.013*"clinton" + 0.009*"america" + 0.008*"crooked" + 0.008*"comey" + 0.007*"fbi" + 0.007*"state" +
0.007*"dollar"'),
 (3,
  '0.012*"great" + 0.009*"election" + 0.009*"president" + 0.009*"news" + 0.008*"fake" + 0.008*"thank" + 0.007*"trump" + 0.007*"russian" + 0.006*"big" +
0.006*"thing"'),
 (4,
  '0.024*"great" + 0.024*"republican" + 0.020*"tax" + 0.016*"america" + 0.015*"cut" + 0.014*"healthcare" + 0.013*"obamacare" + 0.012*"vote" +
0.012*"senator" + 0.011*"democrat"'),
 (5,
  '0.022*"news" + 0.020*"fake" + 0.013*"great" + 0.012*"story" + 0.012*"medium" + 0.012*"trump" + 0.011*"job" + 0.009*"election" + 0.009*"russia" +
0.009*"president"'),
 (6,
  '0.019*"korea" + 0.015*"north" + 0.015*"great" + 0.013*"china" + 0.012*"trade" + 0.010*"deal" + 0.010*"country" + 0.009*"president" + 0.008*"year" +
0.006*"much"'),
 (7,
  '0.021*"american" + 0.016*"great" + 0.010*"america" + 0.008*"people" + 0.008*"time" + 0.008*"today" + 0.006*"country" + 0.006*"first" + 0.005*"thank" +
0.005*"day"'),
 (8,
  '0.017*"great" + 0.010*"people" + 0.008*"year" + 0.007*"th" + 0.007*"look" + 0.006*"thank" + 0.006*"today" + 0.006*"god" + 0.006*"woman" +
0.006*"country"'),
 (9,
```

```
 '0.019*"great" + 0.013*"border" + 0.011*"people" + 0.010*"country" + 0.010*"security" + 0.009*"military" + 0.009*"need" + 0.008*"big" + 0.007*"vote" +
0.006*"back"')]
```

How to read this? Look at topic 1 "news", "fake", "people" etc are words that contribute to topic 1. The other arrays show words that contribute to their topic. This is that word co-occurrence and probability mentioned earlier. We designated 10 topics but is there a better number would 11 or five or 100 topics be a better classification. There is no well defined solution. We will attempt to tune this with a similar unsupervised technique, clustering. We first convert our LDA model's cosine similarity into a compressed sparse matrix. We choose cosine similarity as the metric to evaluate topics on, because it uses the direction word vectors are pointing as its measure of similarity. We feed this matrix into a SciKit Learn's K-Means algorithm and visualize the plots for 50 and 10 clusters.

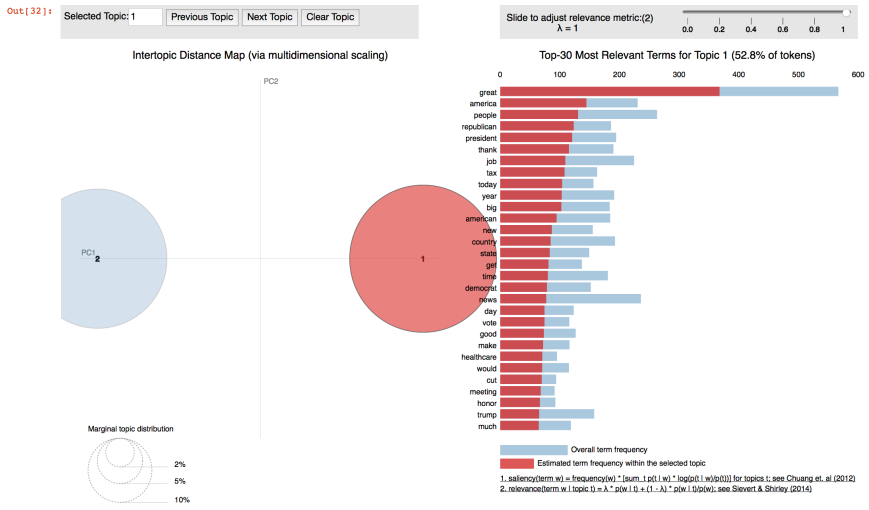


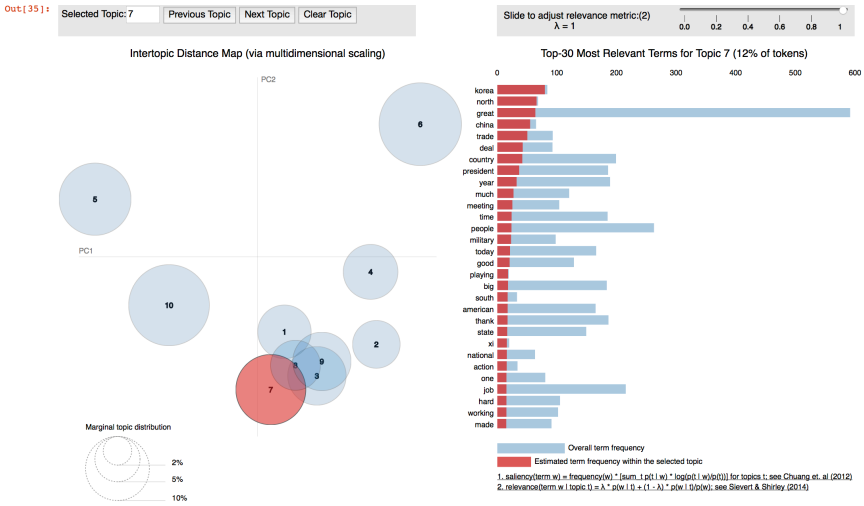


The best cluster numbers (number of topics) seem to be 4 or 10, using the elbow method with the plots. Next, we use the silhouette scoring method and find the best cluster is 2 with a score of 0.091. We take these results back to the LDA model in Gensim and run it with 2 topics to get:

```
[(0,
  '0.019*"great" + 0.008*"america" + 0.007*"people" + 0.007*"republican" + 0.006*"president" + 0.006*"thank" + 0.006*"job" + 0.006*"tax" + 0.005*"today" +
0.005*"year"'),
 (1,
  '0.012*"great" + 0.009*"news" + 0.008*"fake" + 0.008*"people" + 0.007*"job" + 0.006*"country" + 0.006*"time" + 0.005*"trump" + 0.005*"american" +
0.005*"year"')]
```

These topics look much less interpretable. Recalling the elbow method results suggested 2 or 10. We then compare these using pyLDAavis, a cool LDA visualization package.

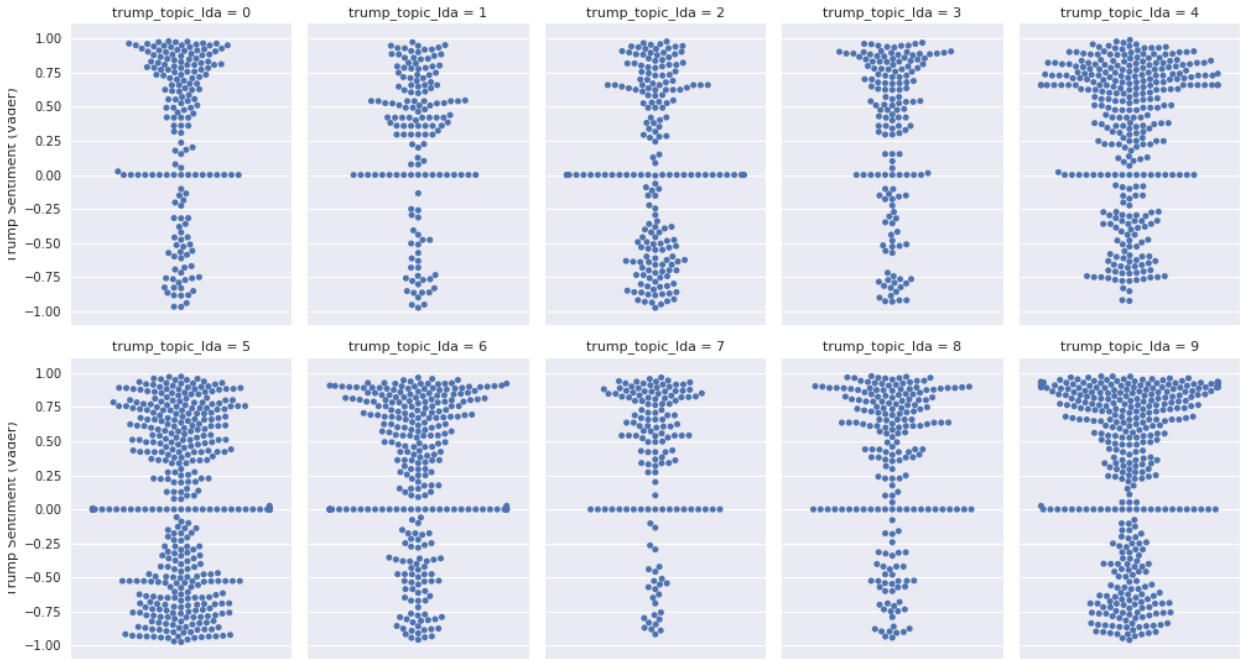




We see from the cool PCA visualize why the silhouette method preferred 2 over 10 topics. The clusters are nice and separated and very distinct (that's what the silhouette score measures). However, the elbow method gives us a much more human interpretable answers for topic inference. This makes sense if we think about topics being allowed to overlap, especially if we are basing topics off word co-occurrence. We are building this for humans, so we go with ten topics.

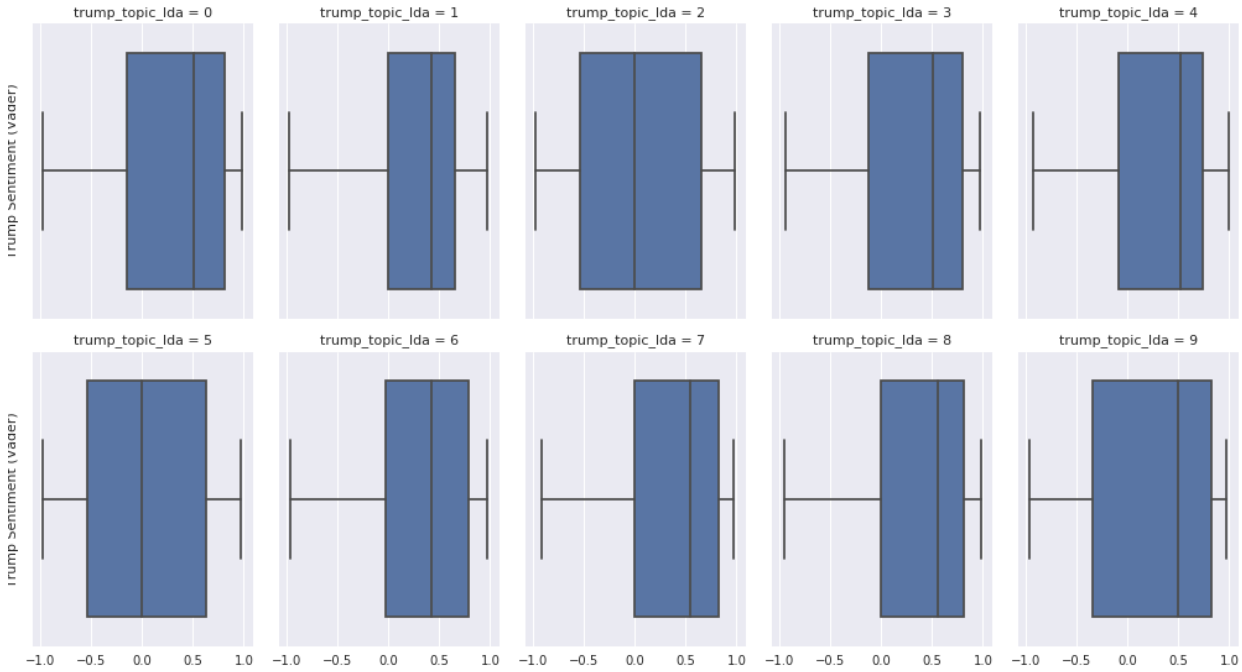
5.d.) LDA Model Application

Finally, we tag each tweet by topic and visualize. We plot the swarm plots and investigate.

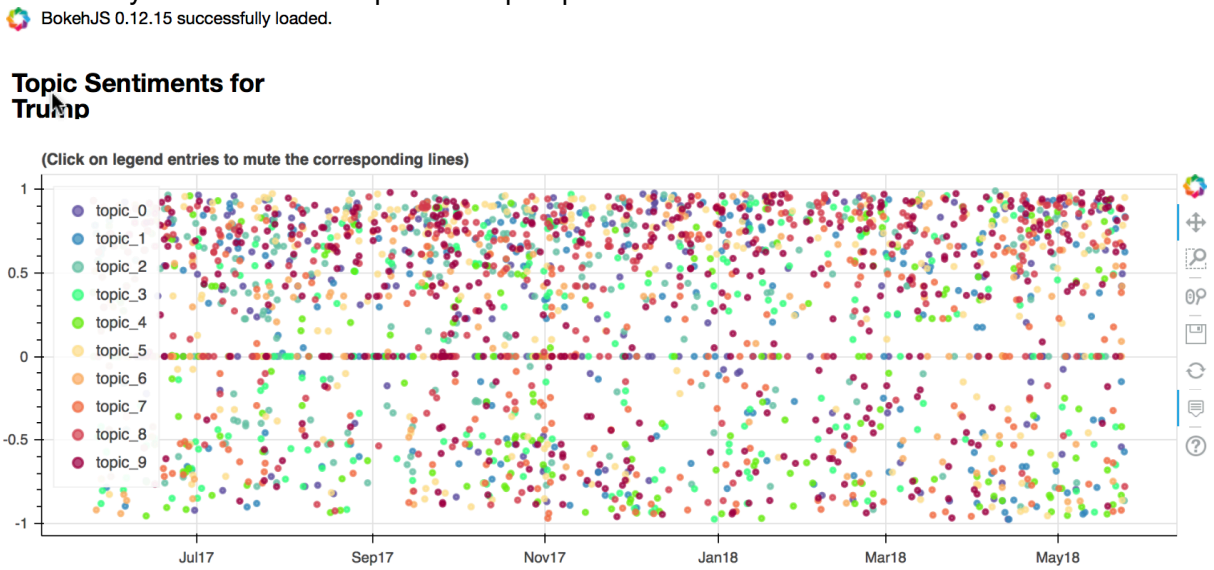


We get a nice look at Trump's Sentiment for each topic. We see that topic 9, 8, 4, and 6 seem to be pretty positive. Topics 5, and maybe 2 seem to be slightly negative. Next, we plot box

plots:



Looking at the box plot we get a much better feel for the quantitative side. Topic 5 and topic 2 are actually neutral. One last plot to help explore these tweets.

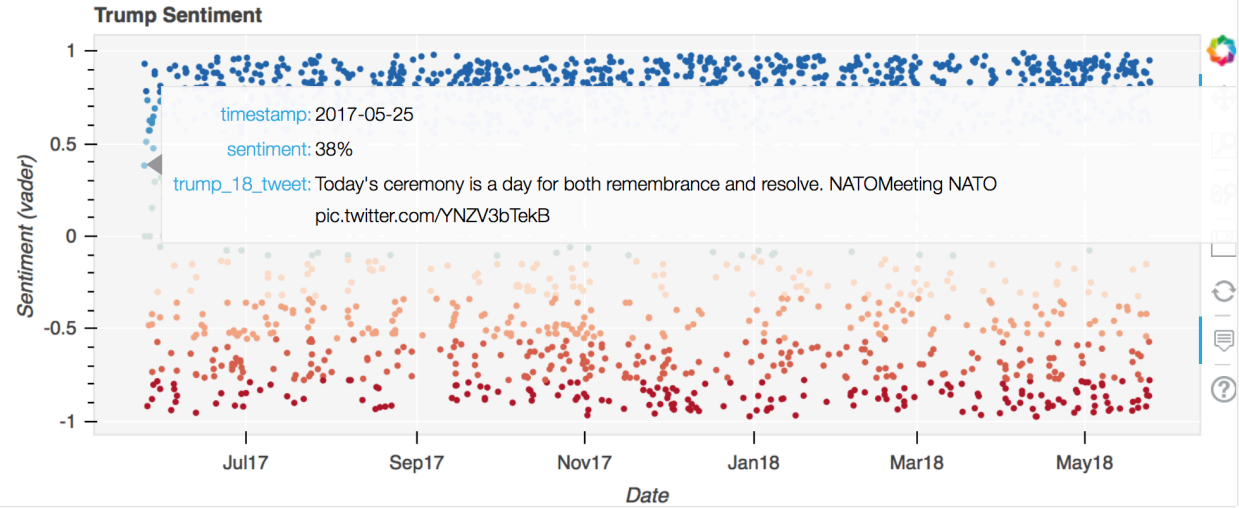


6. Summary

We have built another cool interactive tool, which we can add to a decision maker's or executive dashboard. Clicking on the topics in the legend toggles on and off each respectively. The pan and zoom functions on the right allow user exploration over the timeline. At this point, we have three cool interactive features to help understand an influencer's topic.

We revisit our first Bokeh graph of Trump sentiment. We add a hover tool which provides context to each data point. Now we have two user interactive graphs for exploration

and decision making.



Next steps, we would explore topics in the news tweets. We could use the Gensim LDA algorithm to classify which tweet fits best into a topic. In the future, I would also incorporate "named entity recognition" to tag tweets and explore sentiment based on mentioned entities followed by network analysis with tweet likes and retweets.