Natural Language Processing:
Understanding a Tweet Influencer

1. The Problem Statement

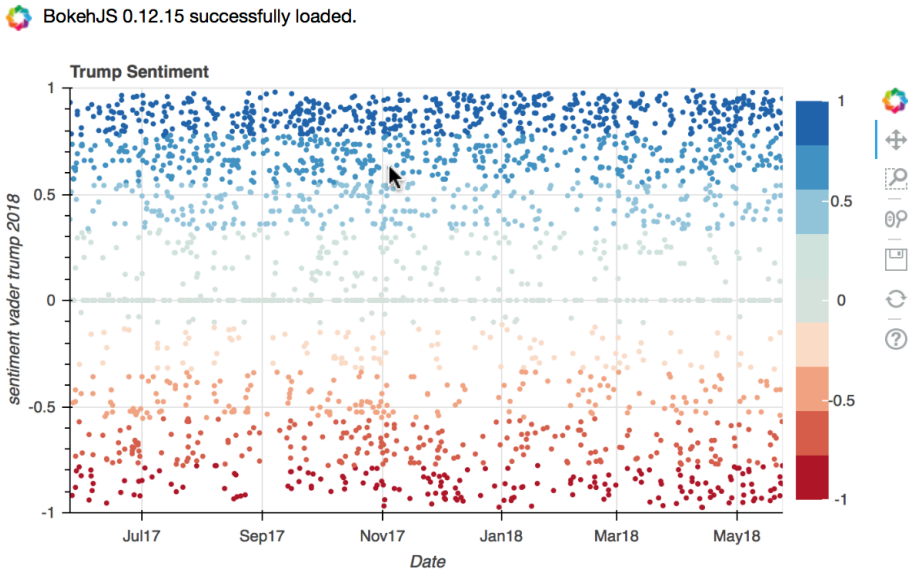
Sometimes a client, a celebrity, or a company can exert an undue about of influence upon a product, a company, an industry, or a market. We will look at how a company can use Natural Language Processing (NLP) to understand an influencer. The president of the United States is such an influencer. We will look at Donald Trump's tweets to see if we can gain insight into his influence. We will 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 any interactions with that influencer. We might want to pitch an idea on a good day, or if making an investment decision, we might 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 the decision maker in making these influencer based decisions.

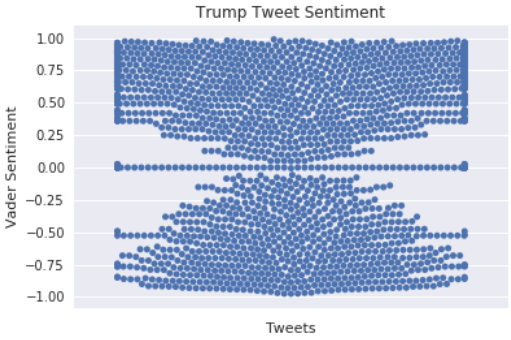
2. Data Collection and Processing

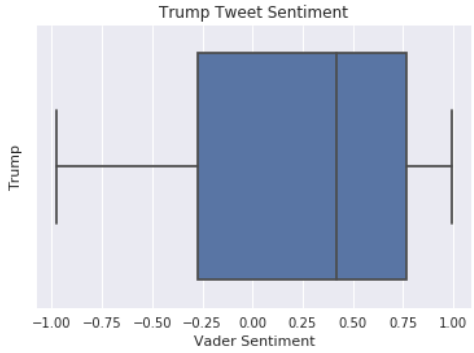
First we need to get some data. We scrap the web to pull in tweets from the last year. We use some Python code developed by Jeremy Cunningham to retrieve these tweets. This saves the tweets for the last year to a csv file. We read these in to a Juypter Notebook with Pandas. Change a few import setting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

3. Trump Sentiment Analysis

We will 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 overall sentiment based on a pre-trained word lexicon. The algorithm then calculates the overall sentiment based of the word weights for sentiment. 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 can use this to explore the data zooming into a specific time span, looking at just positive tweets, or isolating a tweet.



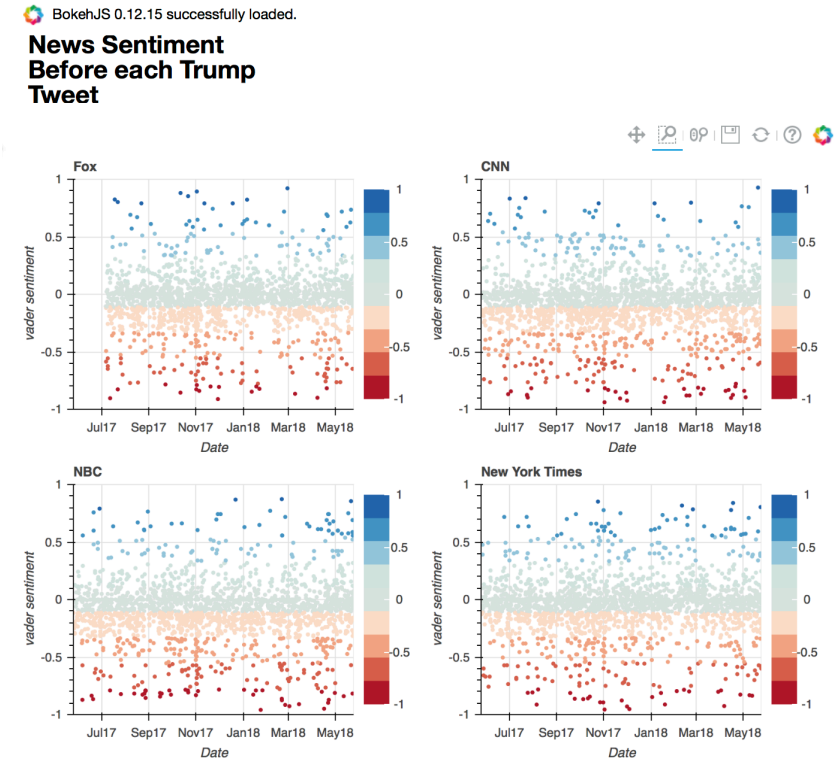




We focus our exploration on the sentiment analysis for the moment. We look at some basic statistics associated and see that overall Trump has a positive tweet feed for the last year. The mean of his sentiments is 0.231237 (on a scale from -1 to 1). The median values is 0.4199 not super positive, but positive.

4. News Sentiment Analysis

Next, we look at the sentiment of news tweets 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 will look at CNN, FOX, NBC, and the New York Times. We will use the same scraper to get csv files for the last year tweets. We clean the data to keep only tweets with text as we did with Trump. We similarly get the sentiment for each news tweet.



We need to convert these to float numbers for plotting since the NLTK module returns an object type. Next we call a function to collect all the tweets from the news broadcaster that occurred between a trump tweet and the one before it. We calculate the average sentiment of each broadcaster before each Trump Tweet and plot those here with Bokeh.

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

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

	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

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, we see that the topics vary widely compared to what Trump is tweeting about. We might be able to find a better correlation if we classify tweets by topic. This is what we will attempt next.

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 we belief topics exhibit before evidence is taken into account). The idea is that tweets cover only a small set of topics and that topics use only 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 other 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

The last algorithm did not need tokens, so 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 these list of words into a list of lists. (If the data was to large 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 to the LDA vector space using the Gensim algorithms for both respectively.

5.c) LDA Model Tuning

Gensim LDA transformer requires the bag of words corpus, the dictionary, and the number of topics. We will start with 10. The problem of picking the best topics 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 will explore some of these techniques later in trying to optimize the LDA model. We set passes to 20 and iterations to 400. This allows 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 to be 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 into 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,
  '0.014*"great" + 0.012*"president" + 0.010*"trump" + 0.009*"today" + 0.008*"donald" + 0.008*"thank" + 0.006*"family" + 0.006*"america" + 0.006*"j" +
0.005*"one"'),
 (1,
  '0.012*"trade" + 0.010*"great" + 0.009*"country" + 0.009*"year" + 0.009*"deal" + 0.009*"new" + 0.009*"market" + 0.009*"people" + 0.008*"stock" +
0.007*"time"'),
 (2,
  '0.027*"republican" + 0.022*"great" + 0.015*"people" + 0.014*"senator" + 0.014*"obamacare" + 0.010*"vote" + 0.010*"healthcare" + 0.009*"big" +
0.009*"democrat" + 0.009*"repeal"'),
 (3,
  '0.016*"time" + 0.014*"medium" + 0.013*"fake" + 0.013*"news" + 0.008*"need" + 0.008*"people" + 0.008*"america" + 0.007*"great" + 0.006*"election" +
0.006*"country"'),
 (4,
  '0.021*"news" + 0.020*"fake" + 0.014*"story" + 0.012*"russia" + 0.010*"obama" + 0.008*"russian" + 0.008*"phony" + 0.008*"collusion" + 0.007*"election" +
0.007*"democrat"'),
 (5,
  '0.016*"american" + 0.012*"people" + 0.011*"great" + 0.009*"today" + 0.008*"country" + 0.007*"god" + 0.007*"thank" + 0.007*"keep" + 0.006*"america" +
0.006*"law"'),
 (6,
  '0.009*"join" + 0.008*"great" + 0.008*"live" + 0.007*"job" + 0.007*"bill" + 0.007*"america" + 0.006*"thank" + 0.006*"trump" + 0.006*"new" +
0.006*"back"'),
 (7,
  '0.013*"hillary" + 0.013*"clinton" + 0.009*"crooked" + 0.008*"news" + 0.008*"dems" + 0.008*"tax" + 0.008*"healthcare" + 0.008*"want" + 0.007*"fake" +
0.007*"cut"'),
 (8,
  '0.032*"great" + 0.017*"korea" + 0.016*"america" + 0.014*"north" + 0.013*"meeting" + 0.011*"make" + 0.011*"president" + 0.010*"china" + 0.007*"today" +
0.007*"day"'),
 (9,
```

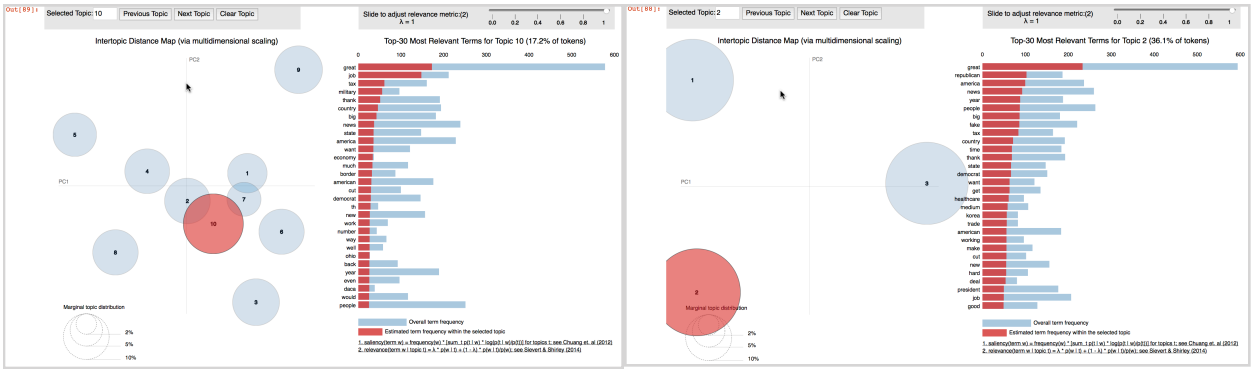
```
 '0.028*"great" + 0.024*"job" + 0.010*"tax" + 0.009*"military" + 0.008*"thank" + 0.007*"country" + 0.007*"big" + 0.006*"news" + 0.006*"state" +
0.006*"america"')]0.014*"great" + 0.012*"president" + 0.010*"trump" + 0.009*"today" + 0.008*"donald" + 0.008*"thank" + 0.006*"family" + 0.006*"america" +
0.006*"j" + 0.005*"one"'),
```

How to read this. Look at topic 1 "trade", "great", "deal" 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 had to designate 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 for this at the moment but 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 3 with a score of 0.659. We take these results back to the LDA model in Gensim and run it with 3 topics to get:

```
[(0,
  '0.015*"great" + 0.008*"people" + 0.007*"america" + 0.007*"news" + 0.007*"country" + 0.006*"american" + 0.006*"today" + 0.006*"fake" + 0.006*"time" +
0.005*"thank"'),
 (1,
  '0.018*"great" + 0.008*"republican" + 0.008*"america" + 0.007*"news" + 0.007*"year" + 0.007*"people" + 0.007*"big" + 0.007*"fake" + 0.006*"tax" +
0.006*"country"'),
 (2,
  '0.017*"great" + 0.009*"job" + 0.008*"president" + 0.008*"people" + 0.008*"news" + 0.006*"thank" + 0.006*"fake" + 0.006*"trump" + 0.005*"election" +
0.005*"new"')]
```

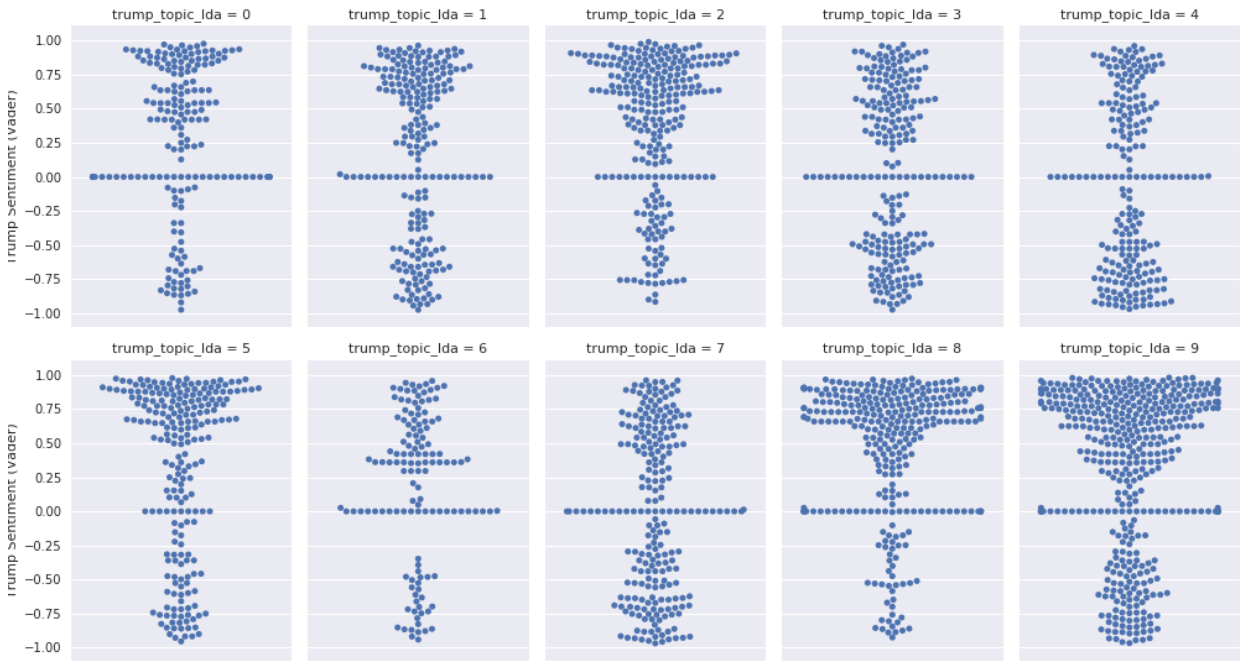
These topics look much less interpretable. Recalling the elbow method results suggested 3 or 10. We then compare these using pyLDAavis, a cool LDA visualization package.



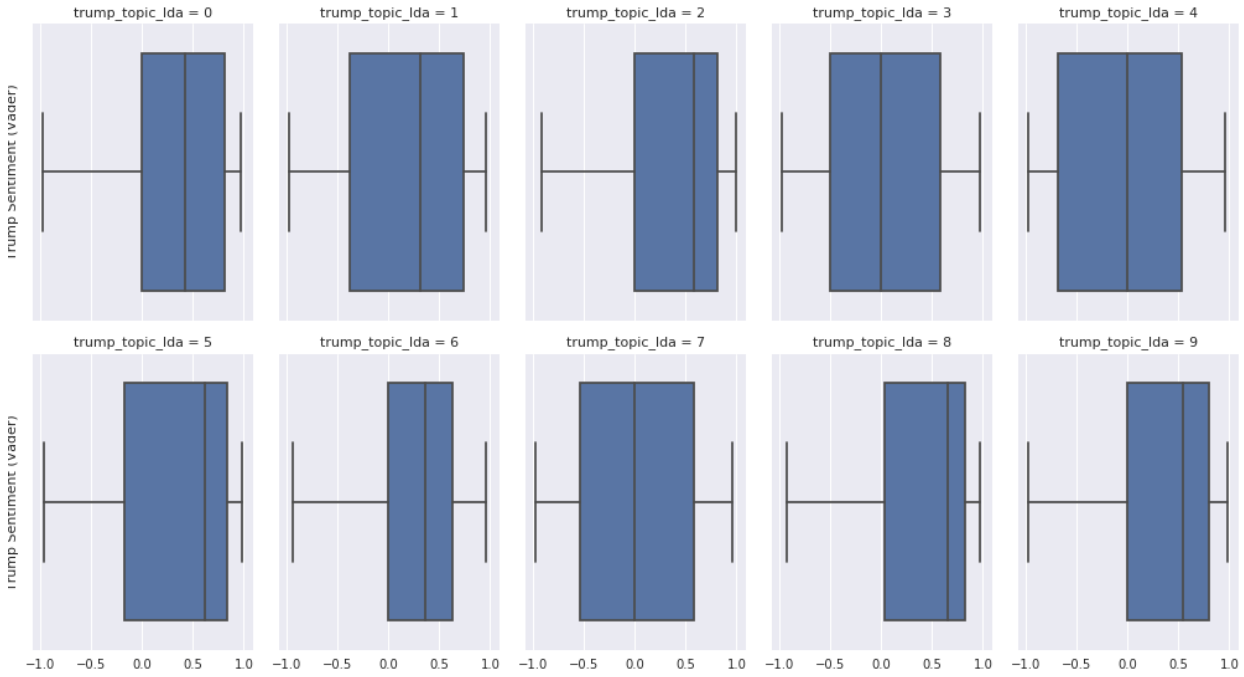
We see from the cool PCA visualize why the silhouette method preferred 3 over 10 topics. The clusters are nice and separated and very distinct (that's what the silhouette score measures). However, the elbow method gives us a much more human interpretable answers for topic inference. This makes sense 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

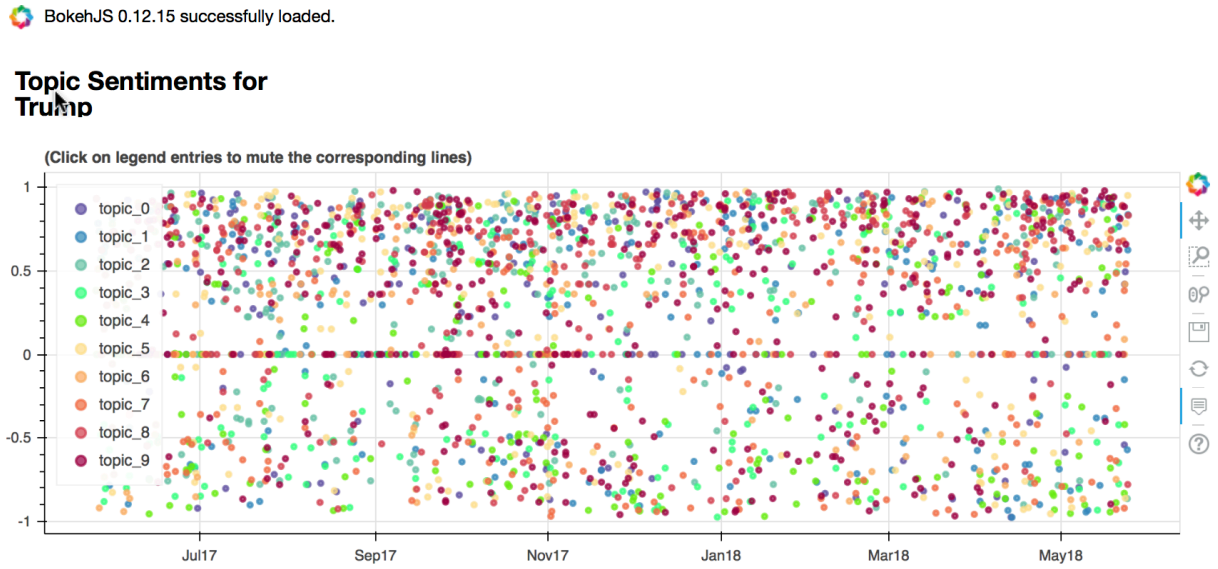
Finally, we tag each tweet by topic and visualize. We plot the swarm plots and investigate.



We get a nice look at Trump's Sentiment for each topic. We see that topic 9, 8, 5, and 2 seem to be pretty positive when trump talks about these. Topics 4, and maybe 7 seem to be slightly negative. Next, we plot box plots:



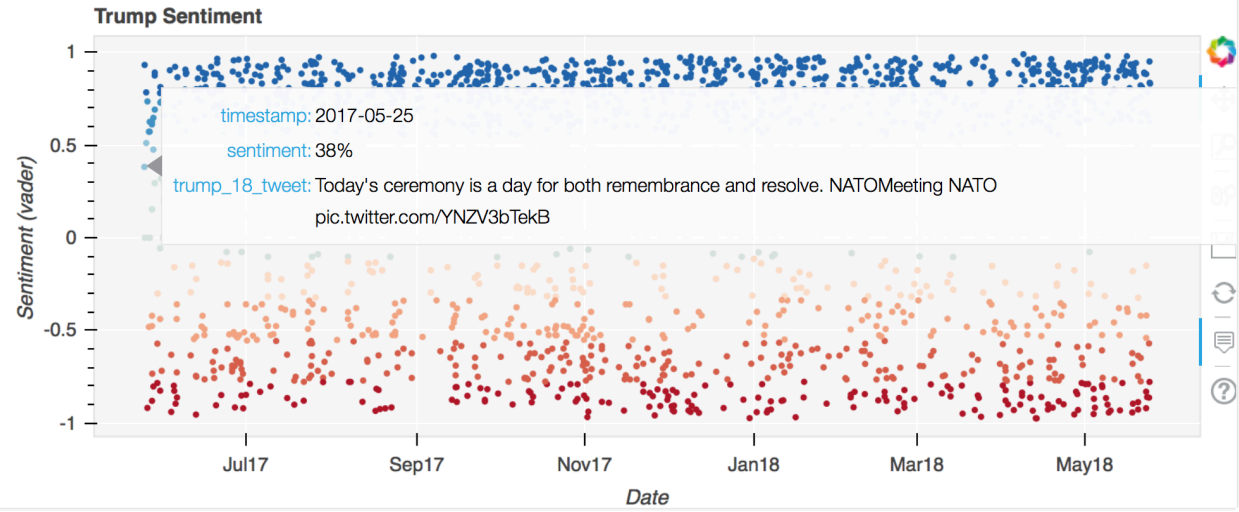
Looking at the box plot we get a much better feel for the quantitative side. Topic 4 and topic 7 are actually right near neutral. One last plot to help explore these tweets.



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. A start to an amazing interactive dashboard.

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 these topics in the news tweets. We could use the Gensim LDA algorithm to classify which tweet's fit best into a topic. In the future, I would also incorporate "named entity recognition" to tag tweets and explore sentiment based on mentioned entities, and look at network analysis with tweet likes and retweets.