University of Utah

Final Report

CS 6140 Data Mining

Text Analysis - Political Debates

Author:
Han Ambrose
Soren Nelson

Instructor:
Jeff Phillips

April, 2020

Contents

1 Abstract

This report explores several data mining techniques to analyze text data using bags of words, dimensionality reduction and clustering to improve noise in the data collected. Specifically, Singular Value Decomposition (SVD) and Lloyd's clustering algorithm were focused to yield more accurate results.

2 Introduction/ Problem and Motivation

As social networks are growing fast, platforms like Twitter, Reddits and Facebook have become places where people can freely share their opinions. However, semantic analysis of such textual data is such a challenging task as from data collection to cutting noise and junk information. In this paper, we proposed a couple of techniques to address those types of problems.

In this paper we are hoping to extract some interesting information regarding the political debates, especially finding words or topics that are related to a particular candidate to see if public's opinion towards that candidate is either positive or negative in general. Then we compare all of the candidates together. We also compare across different debates to see trends over time if the public's opinion have changed from time to time. The idea is a candidate can do poorly in one debate but could do better in another debate.

3 Data Collection

3.1 Collecting Tweets

We started our data collection by getting the most popular political hashtags regarding the Democratic debate. Using Tweepy, we gathered 48 for first and 89 hashtags for the second debate. We then used these hashtags as search terms to get all tweets using these hashtags with more than 5 favorites.

In the next step, we used regular expressions to preprocess the data. We decide to break down the text into words. This is done by 'tokenization', specifically done by WhitespaceTokenizer so that '@' and '#' are attached to the words in the form of '@mention' and '#hashtag' instead of being separated. It is important to keep them together because when analyzing only the content of the speech, we want to remove words that starts with '#' and '@' to avoid repetition of hashtags and popular users. This removal does not take away important information because these hashtags were already been captured when we use them as 'search_term' to collects tweets regarding Demoncratic Debates.

In every language, some words are common words like 'a', 'the', 'that', etc. These words do not show any meaning or the context we are looking for to make sense of. In this case, we have to remove these words called 'stop words' and punctuations. In addition, the tweets may contain retweet 'rt' or 'via' or contains link 'https', these words have to be removed as well. Below is an example of a tweet before and after processing:

Before : 'Man that #Debatenight In Vegas was nuts well Scott got a lot things to talk about get ready!'

After: 'Man', 'nuts', 'well', 'Scott', 'got', 'lot', 'things', 'talk', 'get', 'ready!'

Another challeging task when it comes to analyzing tweets is that tweets have very short contents. It is difficulty to get any meaningful context if the tweets are not long enough to convey any polarity or subjectivity. Most tweets in facts only include hashtags and mentions without specifically expressing the author's opinions about the subject matter. After cleaning the tweets and break them down into 1-gram words, we further filtered out any tweet that has less than 20 words in lengths.

3.2 Collecting Transcripts

On top of collecting data from tweets, we also collected data from debate transcripts including the 'New Hampshire' and 'Nevada' debates, which were supposed to be the last two Democratic Debates before most candidates dropped out. These are recorded speech of all of the candidates during the debate. We want to compare the topics and analyze the differences between what the candidates said during the debates, particularly the positivity/negativity in their speech and how that effects the the public's response in the tweets.

Now, analyzing transcript is a little easier than analyzing tweets because of the nature of the cleaniness of such data as the speech is much longer and does not contain as much noise as Twitter's data. However, we still want to follow the same process of cleaning data to also remove stop words and non-alphabetical characters.

4 Experimental Methods/Key Ideas

4.1 K-grams Model

The first step to understand an idea about the data is to extract the frequency of important words used. This can be done using bag of words with n-grams. 1-gram word: Below is the term frequency for tweets talking about Trump.

[('Trump', 1237), ('amp;', 395), ('President', 327), ('like', 219), ('Trump.', 207), ('would', 178), ('people', 177), ('one', 175), ('get', 172), ('Donald', 165), ('Bloomberg', 155), ('Russia', 155), ('last', 149), ('Democrat', 130), ('know', 127), ('trump', 112), ('see', 111), ('Dems', 110), ('think', 104), ('Bernie', 104)]

We can see that words like 'President', 'Donald' are highly associated with Trump. Also, Bloomberg was popular because it was the first time he joined the stage. However, these words do not give us much meaning on its own.

2-gram word: While frequent words represent a clear popular topic, one term frequencies do not give us a deep explanation of what the context is about, which leads us to the next step of trying 2-gram words. Now we can see that words make more sense when they are together. In

this case we can see that people talking about Trump also talks about Russia, and that they want to find a candidate to beat Trump in the next presidential election.

[(('President', 'Trump'), 101), (('Donald', 'Trump'), 64), (('Russia', 'interfering'), 38), (('Elizabeth', 'Warren'), 33), (('last', 'night'), 32), (('Colorado', 'Springs'), 32), (('Donald', 'Trump.'), 30), (('President', 'Donald'), 28), (('Roger', 'Stone'), 24), (('beat', 'Trump.'), 24), (('get', 'Trump'), 23), (('Trump', 'amp;'), 22), (('God', 'bless'), 21), (('last', 'night.'), 21), (('President', 'Trump.'), 21), (('Mike', 'Bloomberg'), 21), (('Trump', 'supporters'), 20), (('White', 'House'), 18), (('America', 'Great'), 18), (('Las', 'Vegas'), 17)]

4.2 Bag of words with Inverse Document Frequency (IDF)

K-grams model is helpful to tell us the what words or group of words that have high frequency and repetitive. However, it really does not give us any extra or meaningful context. For example, the word "like", which appears a lot in our case has very high frequency but does not have any meaning.

But if we talk about the word "beat", it gives us a context. We can say with some certainty that the tweet is talking about the topic of defeating Donald Trump in the next presidential election. On the other hand, we also have words that is rare, could be like a group of words with no space like "thisDebateIsGettingReallyInteresting", which is common in tweets and social media's text. The problem with k-grams model is that all words are considered equally important when it comes to assessing relevancy on a query. In fact certain terms have little or no discriminating power in determining relevance. Hence, we need to assign the weight of the words in some way.

This can be done with Inverse Document Frequency (IDF), which assign the weight the the words using the formula below:

$$weight = log\left(\frac{\text{number of docs in your corpus}}{\text{number of docs in which this word appears}}\right) \tag{1}$$

Now, for each tweet, instead of having a vector of 0 and 1, we have a vector with weights.

4.3 Dimensionality Reduction

Inverse Document Frequency method did improve the the discriminating power, but there is still a lot of noise in tweets. This leads us to the next method of dimensionality reduction. In this case, we use Singular Value Decompositon method (SVD).

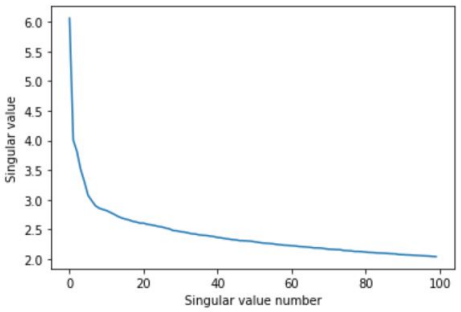
We created a term document matrix X with rows are represented by words and columns are represented by documents/tweets. Theses words are assign weight to the number of times each term appeared in each tweet as discussed in the IDF method above.

The SVD of term document matrix X can be defined as:

$$X = USV^T \tag{2}$$

We then reduced the dimension of to k. In this case, most of the data stays in the first $k = 10$ dimensions. We choose $k = 40$ to be safe.

$$X_k = U_k S_k V_k^T \tag{3}$$



Dimension reduction provides benefits that the terms that share substructure become more similar to each other and the terms that are dissimilar begin to become more dissimilar. This means that the tweets/documents about a specific topic become more similar even if exact term/word doesn't appear in all of them.

Now to find words that are related to a particular candidate, we compute the similarity between the candidate's name and the words in the reduced matrix. We decided to use cosine similarity as it seems appropriate with vectors calculations and it's widely used for textual data.

Cosine similarity measures the similarity between two vectors by taking the cosine of the angle the two vectors make in their dot product space. If the angle is zero, their similarity is one, the larger the angle is, the smaller their similarity.

$$sim(\mathbf{x}_a, \mathbf{x}_b) = cos(\theta) = \frac{\mathbf{x}_a \cdot \mathbf{x}_b}{\|\mathbf{x}_a\| \|\mathbf{x}_b\|} \tag{4}$$

The table below shows the top 10 words that are associated with "Trump" and "Biden" and its similarity score:

We can now see much more meaningful context after dimensions are reduced. The words are much more expressive and precise to describe the topics about a candidate. These top words also have pretty high correlation or similarity with the search word.

4.4 Sentiment Analysis

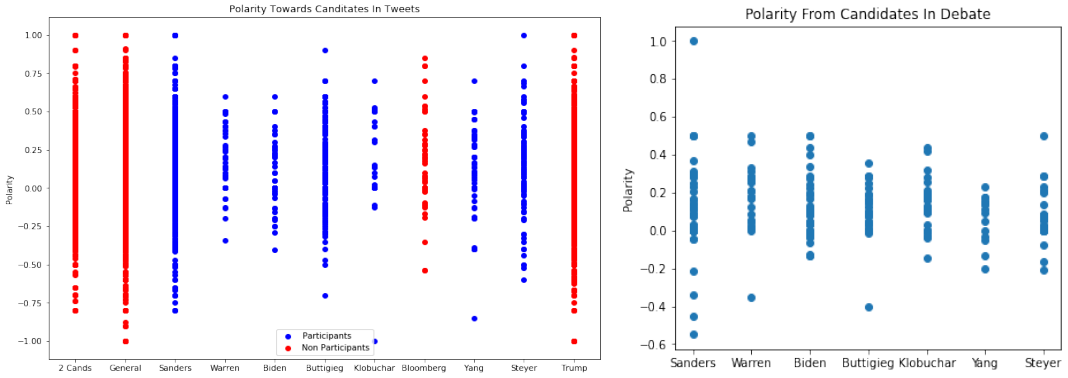
We then wanted to see if the sentiment of what people tweeted during the debates was correlated to the sentiment of what was said by the candidates during the debate. To start, we found the subject of a given tweet by checking whether a tweet used a single candidate's name. We separated out all tweets that referred to more than one candidate and any tweet that didn't

Trump	Biden
Similar Words — Similarity Scores	
beats — 0.8	klobuchar — 0.91
tops — 0.78	cab — 0.87
sworn — 0.76	incapacitating — 0.85
evasive — 0.75	bribes — 0.85
knowledgable — 0.75	straightforward — 0.85
lashes — 0.74	buttigieg — 0.85
uttered — 0.74	rankings — 0.85
hispanic — 0.74	yougov — 0.83
longest — 0.71	joe — 0.83
presumably — 0.7	liberace — 0.83

Table 1: Top 10 words associated with Trump and Biden

mention a candidate. The polarity, with 1 being positive and -1 being negative, on twitter towards each candidate and from each candidate during the first debate is seen below.

Debate 1



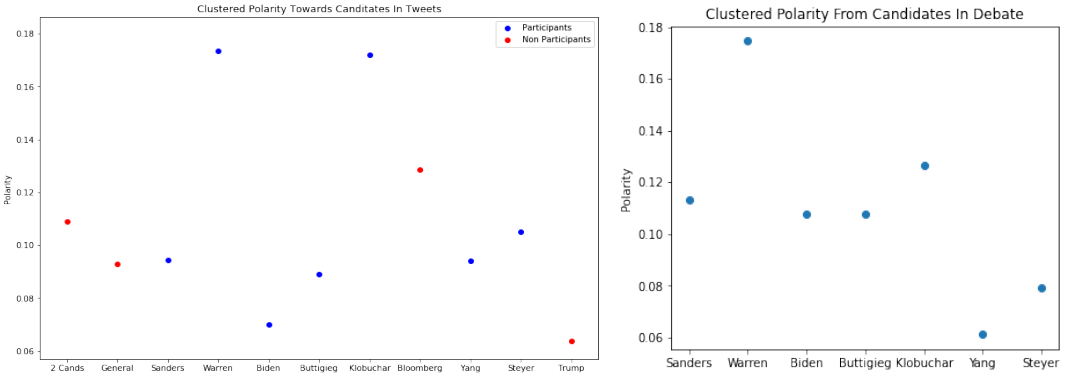
This is insightful in that you can see who people were more polar towards on twitter than others and shows Bernie Sanders was clearly the most polar on stage. All candidates were more positive than negative.

4.5 Clustering

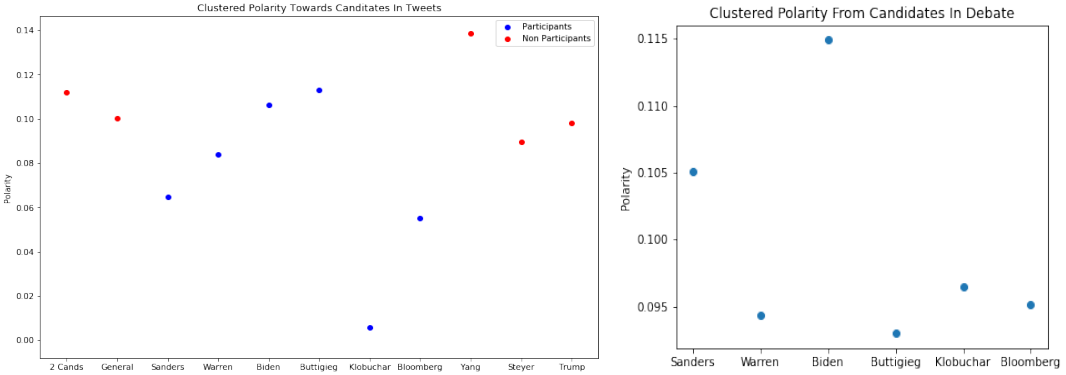
To gain more insight, we ran Lloyd's algorithm to cluster the sentiment analysis data. We initialized one cluster per candidate at 0 polarity. Both debates are shown below. This is much clearer picture of the polarity. Everyone is still positive but those candidates who were more positive in the second debate were claimed as having a better debate by experts and had positive discussion about them on twitter. Joe Biden and Bernie Sanders were clearly both the most

positive in the second debate. Interestingly after this debate is when most other candidates dropped out.

Debate 1



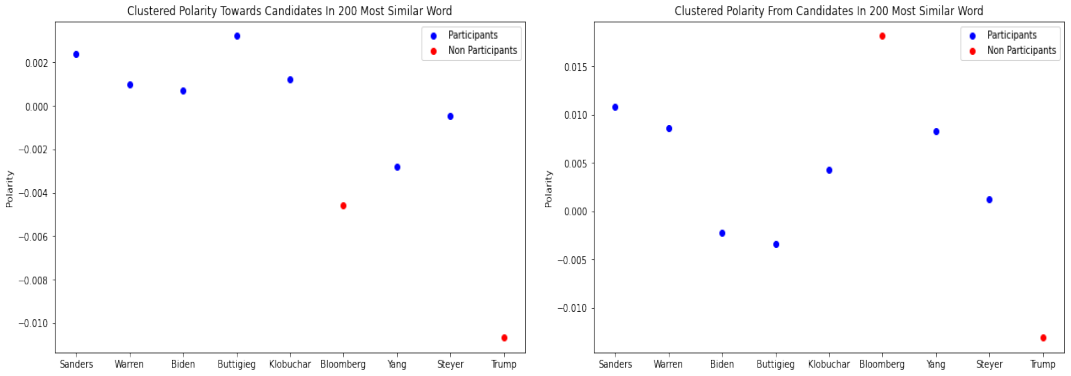
Debate 2



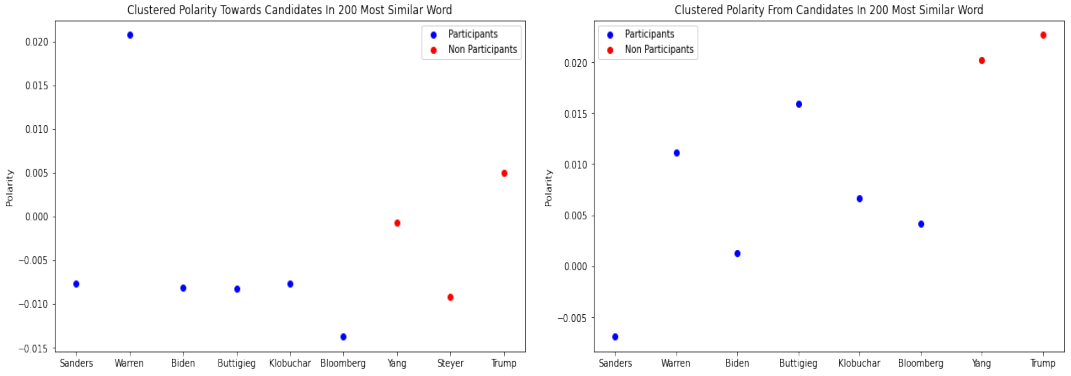
4.6 Clustering with Dimensionality Reduction

With the success of the dimensionality reduction, we wanted to see if that would help our semantic analysis. We took the most similar 200 words to each candidate from the tweets and the debate transcripts and then gathered the polarity of these words which was used as input into Lloyds algorithm. President Trump is most similar to the most negative words in the first debate on twitter and in the transcripts but among the most positive in the second. Elizabeth Warren is similar to the most positive words overall. This measure of polarity was more neutral than measuring the tweets directly as above. The results are shown here.

Debate 1



Debate 2



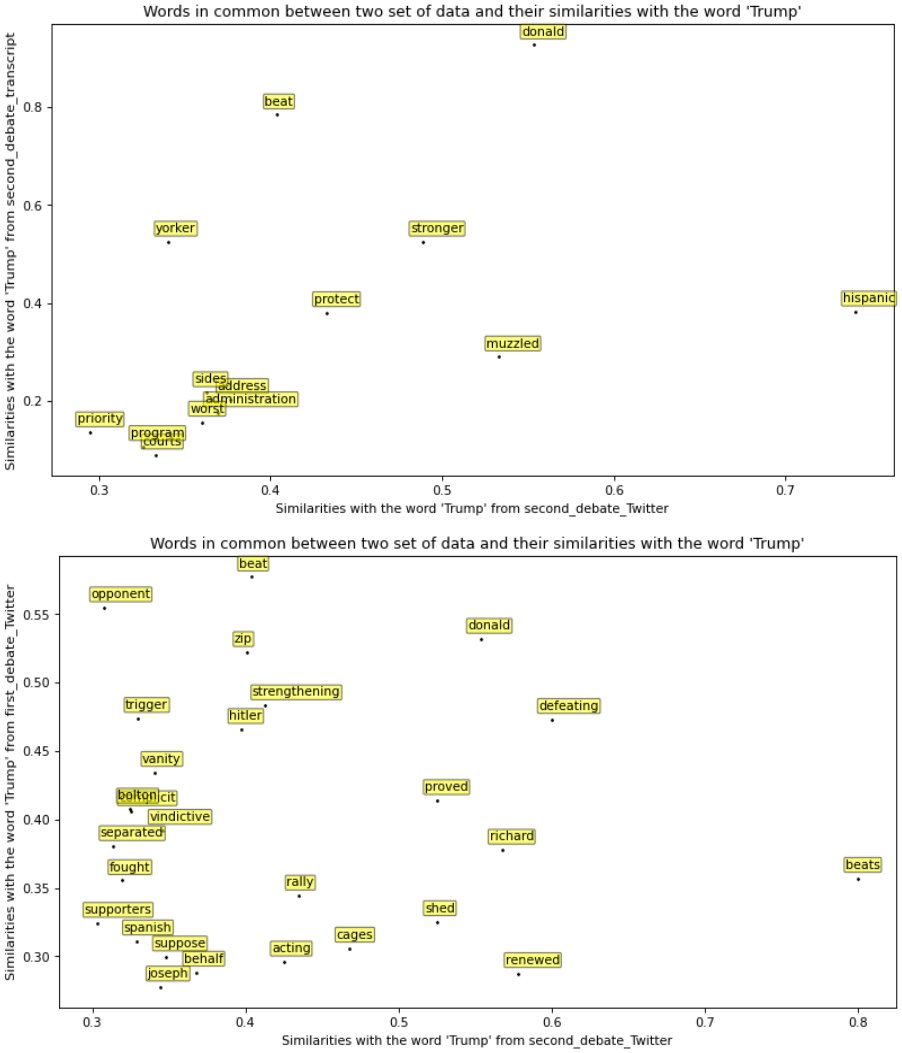
5 Discussion of Results

Using the above techniques, we were able to do the some analysis to see if there is any topics in common across different debates and datasets. Using output from SVD model, we collected 200 words assosiated with a candidate's name from 2 datasets and perform the intersection to find common words in both dataset and their similarities with the candidate's name. An example below shows how topics associated with the word "Trump" are explored.

The first chart shows any words in common associated with "Trump" from the both Twitter's data (public's opinion) and transcript's data (candidates' recorded speech). We can see that the topic of beating Donald Trump was heavily focused from both the candidates and the general public.

The second chart shows any words in common associated with "Trump" from the New Hampshire's debate and the Nevada's debates collected from Twitter. Again, the words "beat", "de-

feating" are highly associated when talked about Trump, showing that it has been the consistent focused topic.



6 Conclusion

From experimentation, it is found that SVD works very well with noisy data such as tweets. Choosing the right k dimension to reduce to is also debatable. This project could be extended in explore different k dimension to see how sensitive the output is when k is changed.

On the other hand, semantic Analysis led to various insights into what people thought about candidates and this slightly correlated with how polar the candidates were during the debate but less so than we anticipated. We were able to remove some of the noise in polarity to get a

clearer picture via clustering and dimensionality reduction.

In addition, it is also quite important to clean up text data such as removing stop words, punctuation's and symbols, getting rid of retweets to avoid duplication to yield better results. We could potentially stem the words as well to further reduce similar words.

7 Appendices

7.1 Distribution of work

1. Han:
    - Data Collection: Assisted with creating Tweepy accounts to pull Twitter's data. Created filters/search terms to narrow down the search in data collection process
    - Analysis: K-grams, Bags of Words through IDF, SVD
2. Soren:
    - Data Collection: Used search terms to gather popular tweets from twitter during the debates.
    - Analysis: Semantic Analysis, K-means with Lloyd's algorithm

7.2 References

[1] https://www.ijettcs.org/Volume5Issue4/IJETTCS-2016-07-22-21.pdf

[2] https://www.ieeexplore.ieee.org/document/7415168