DATASCI W205: Storing & Retrieving Data - Final Report

Analysis of Twitter Sentiment for Presidential Candidates

Manish Sannat & Will Oldfather

Motivation and Project Goals

During an election year, a great deal of energy is spent covering the presidential race, particularly when it comes to trying to predict the outcome of the election. While these prediction models have been fairly accurate in the past, it has become clear that this presidential race is quite unlike any other election before it.

Over the years, the United States have become increasingly polarized, and we see from the figures below that many states have been consistent in leaning Republican or Democrat for the past few election cycles. This often leaves only a few true "battleground" states for candidates to contest.

Figure 1: Electoral Map in 2000

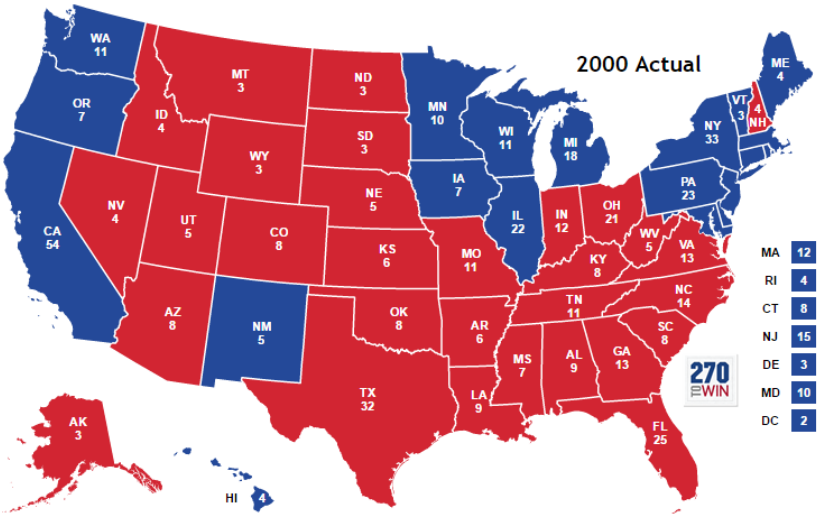
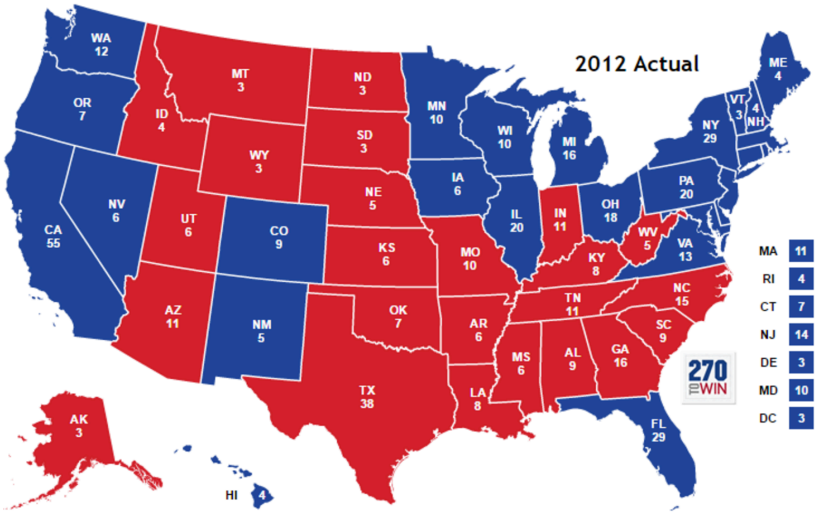


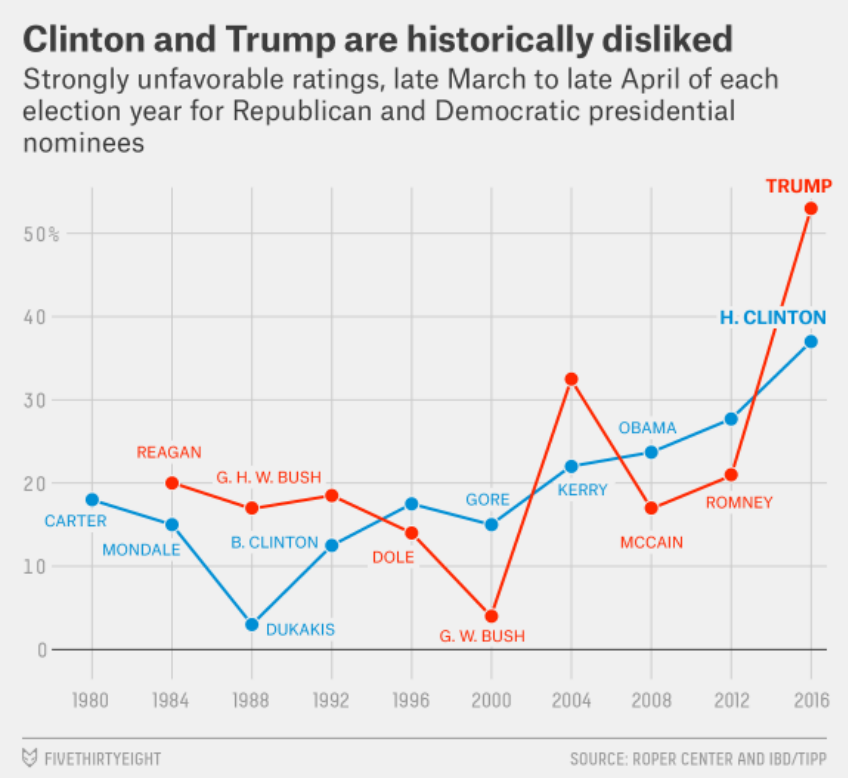
Figure 2: Electoral Map in 2012



Under more normal circumstances, traditional projection models would predict a close election, as the majority of the states would be expected to remain consistent in their overall party preference. However, 2016 cannot be

classified as a "normal" election. In addition to having the first-ever female presidential nominee, Trump and Clinton can both be considered somewhat contentious, due to Clinton's email controversy and Trump's negative rhetoric involving immigrants and Muslims, among other topics. This is also reflected in the approval numbers, as both candidates have historically-high unfavorable views from the public.[1]

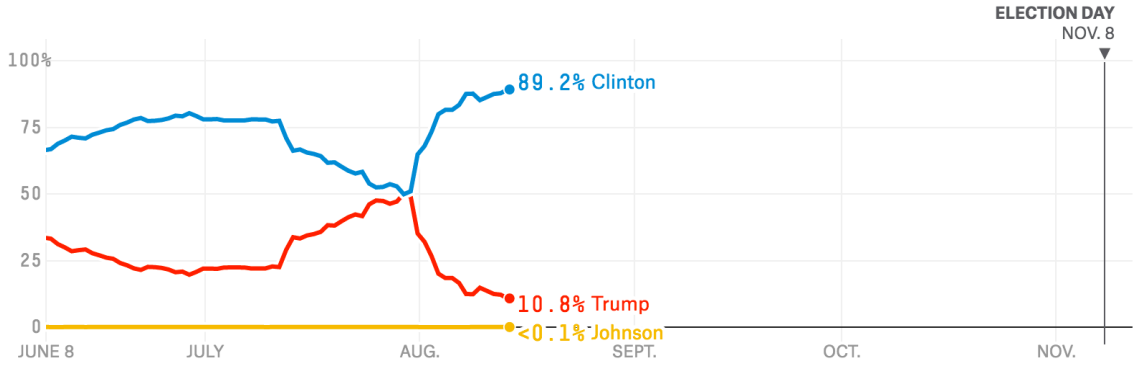
Figure 3: Clinton and Trump Unfavorable Ratings



In this political climate, it becomes more difficult to predict the outcome of the election compared to a more conventional race. Furthermore, polling data this far ahead of the general election can fluctuate significantly, particularly following National Conventions or newsworthy events. As we can see from FiveThirtyEight.com's election prediction graphic below, the odds for each candidate based on polling data have changed dramatically over time, and feature sharp movements around the National Conventions.[2]

[1] http://fivethirtyeight.com/features/americans-distaste-for-both-trump-and-clinton-is-record-breaking/
[2] http://projects.fivethirtyeight.com/2016-election-forecast/?ex_cid=rrpromo

Figure 4: Forecasted Chance of Winning the Presidential Election



Under these conditions, it can be difficult to predict how the race will unfold using traditional polling methods, which can be separated by several days or even weeks. Thus, accurate forecasting, particularly in this unpredictable race, requires analysis of data in real time.

Fortunately, social media platforms, such as Twitter, provide this opportunity. In recent years, social media has become increasingly intertwined with the political process. Indeed, social media can make politics more relevant for a younger demographic, and even influence voter turnout.[3] Social media platforms also offer a method for candidates to interact more directly with voters, provide a wider audience for campaigning, and give candidates a way to gauge public opinion more quickly and more accurately than ever before.[4] Furthermore, these social media platforms provide a forum for political discussion and debate amongst everyday citizens, and allow people to express their emotions towards candidates more candidly and more openly.

For the abovementioned reasons, we seek to use Twitter data to analyze public sentiment about the presidential candidates in near-real time, to supplement the traditional forecasting methods. Specifically, we track the top words, hashtags and phrases that public Twitter users are saying about each candidate, and use sentiment analysis to explore positive/negative reactions to each candidate. Finally, we utilize geographic tagging of Tweets to map these words and sentiment scores across the country, allowing us to compare the content that we see in Twitter to the traditional prediction models.

Data Source

For our project, we collected and stored the text strings and geographic tag information (when available) from Tweets acquired from the Twitter API.[5] This allows us to connect to the Twitter feed of publicly available Tweets and extract their content to be stored in JSON format. Tweets are the core

[3] http://www.nytimes.com/2012/09/13/us/politics/social-networks-affect-voter-turnout-study-finds.html
[4] http://uspolitics.about.com/od/CampaignsElections/tp/How-Social-Media-Has-Changed-Politics.htm
[5] https://dev.twitter.com/overview/api

component of Twitter and represent the method by which users provide "status updates" and interact with one another within the platform.

For our project, we used the Tweepy Python library to extract the Tweet content in JSON format. Specifically, we collected the UTF-8 text string content of each Tweet, as well as the geographic coordinates of the Tweet as reported by the user or client application. The original JSON formatting made it easier to collect and store the Twitter data in MongoDB (in BSON format), which strongly contributed to our decision to use MongoDB as part of our project architecture.

Data Acquisition and Pre-Processing

Our data collection process begins by using the Tweepy Python library to access the Twitter API and extract Tweet text strings and geographic coordinates. In order to limit our search to Tweets containing substance relevant to our project topic of the presidential candidates, we filtered our search to look for Tweets containing specific mentions of the presidential candidates (e.g. "Trump", "Hillary", "@realDonaldTrump", etc.).

In addition, the text of each Tweet was pre-processed for analysis by parsing the text string into individual words. We also removed special characters, URLs, stop words, emojis, etc., which would not be relevant for our analysis. This allowed us to isolate the relevant content of the Tweets for our text analysis and eliminate some of the noise from the data. We also performed additional pre-processing using the Natural Language Toolkit (NLTK), which tagged and tokenized the Tweet text for further sentiment analysis later.

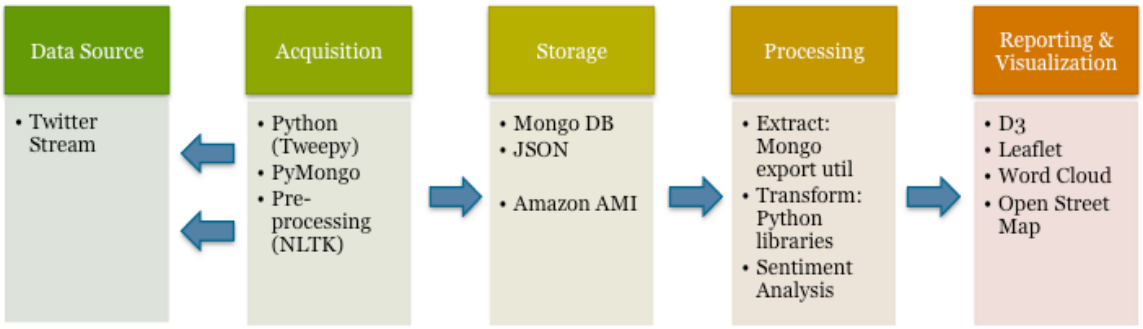
We collected our data by streaming Twitter data from the Twitter API in 60-minute intervals, which were performed multiple times over the course of several days. Each batch of Tweets was then pushed to and stored in MongoDB on our Amazon EC2 instance. We interacted with the MongoDB database using the PyMongo Python library.

In total, we collected about 300 minutes worth of Tweet streams. This provided us with a total of approximately 400,000 unique Tweets and a combined data volume between 2 and 3 GB.

Process Flow and Implementation

The figure below provides a high-level overview of our project's process flow. The following sections provide a description of the process flow in greater detail.

Figure 5: Process Flow Diagram



Data Acquisition

1. Stream the text and geographic coordinates contained in Tweets sourced from the Twitter API's Tweet stream. We connected to the Twitter API using the Tweepy Python library.
2. Apply filters to identify and collect Tweets specifically related to the presidential candidates.
3. Parse Tweet text strings into individual words and remove non-useful information (i.e. special characters, URLs, emojis, stop words, etc.).
4. Tag and tokenize Tweet text using NLTK, for performing sentiment analysis later.

Data Storage

1. Data acquired from the Twitter API was written in bulk to the MongoDB database, which was established on a single Amazon EC2 instance. The Tweet text and geographic coordinates were also stored in JSON format.
2. We stored the raw Twitter data in the MongoDB database, with fields for words and coordinates. PyMongo Python library was used to interact with the MongoDB database and further clean the data.

Data Processing

1. Extract the data from MongoDB database using the Mongo export utility.
2. Use Python libraries to transform and explore the Tweet data further, such as searching for frequent terms and hashtags, and performing topic modeling.
3. Apply IBM's Alchemy API to perform sentiment analysis on the Tweet text data. Identifies which Tweets contain overall positive, negative or neutral sentiment.

Reporting and Visualization

1. Used Jupyter Notebook to run reporting and visualization code.

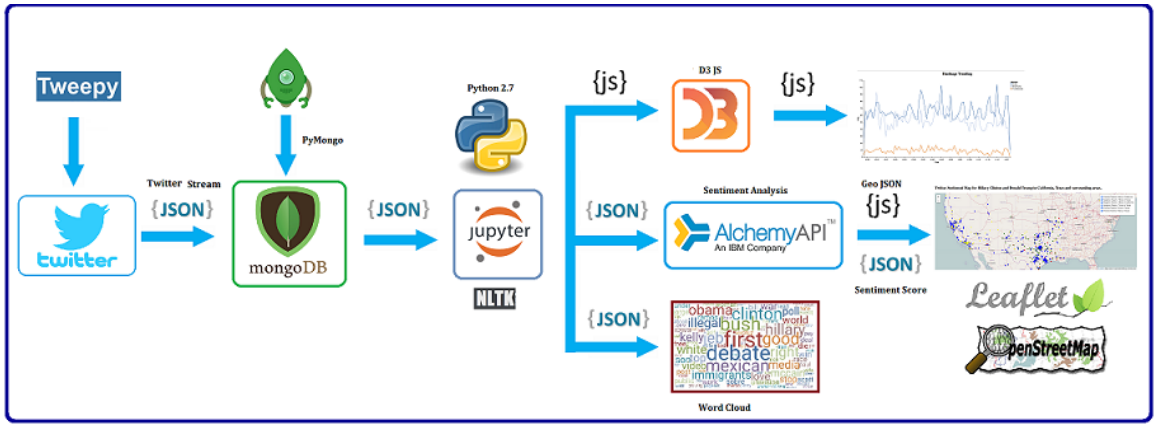
2. Presented the results of our analysis using D3.js for topic trending, hashtags trends, and words occurring together in a tweet.
3. Used Python WordCloud library for word cloud visualization.
4. Open Street Map and Leaflet for geographic results mapping and sentiment map.

Architecture, Tools and Third Party Libraries

System Architecture

The figure below presents the final architecture for our project, and illustrates the technologies used at each step in the process.

Figure 6: System Architecture



For our project, we used a single Amazon EC2 instance, as our project involved small to mid-sized data processing and not more than 5 GB of data volume. We determined that the Amazon m3.medium instance would be best for performing these processing and storage tasks.

Tools, Technologies and Third Party Libraries

Our project utilizes the following tools, technologies and third party libraries.

1. Data Acquisition Layer
    a. Tweepy Python library for connecting to the Twitter API and collecting Tweet data
    b. Twitter API for providing Tweet stream to access
2. Data Storage Layer
    a. MongoDB for storing raw Tweet data in BSON format
    b. PyMongo to interact with the MongoDB database
    c. Mongo DB utilities to dump DB data and convert into JSON format.
3. Processing Layer
    a. Python libraries with JSON and MongoDB
    b. Tokenization and tagging using NLTK

    c. IBM's Alchemy API for sentiment analysis
4. Reporting and Visualization Layer
    a. D3.js for trending analysis and topic modeling
    b. WordCloud Python library for word cloud visualizations
    c. Open Street Map and Leaflet for sentiment mapping

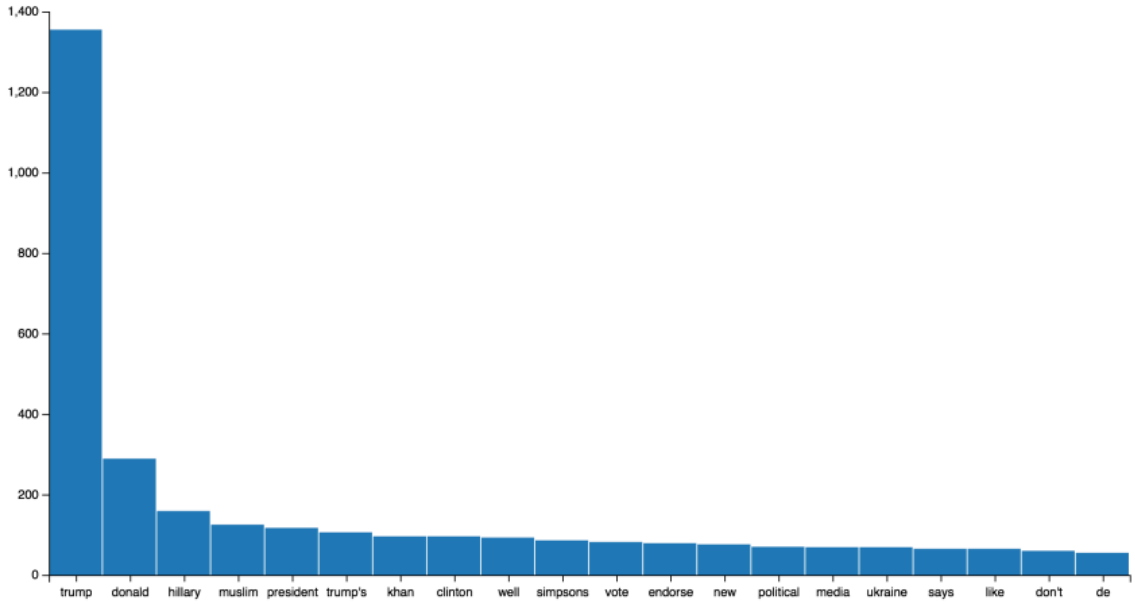
Analysis Results

We performed several types of exploratory analysis on the Tweet data to assess the content and sentiment of Tweets, as well as performing some trending and geographic mapping analysis. The following sections present examples of our project results.

Topic Modeling

First, we processed the Tweet data to collect the most frequent words, pairs of words and hashtags occurring in the identified Tweets. Then we used D3.js to create a histogram of the top 20 most frequently-occurring words, as illustrated in the figure below.

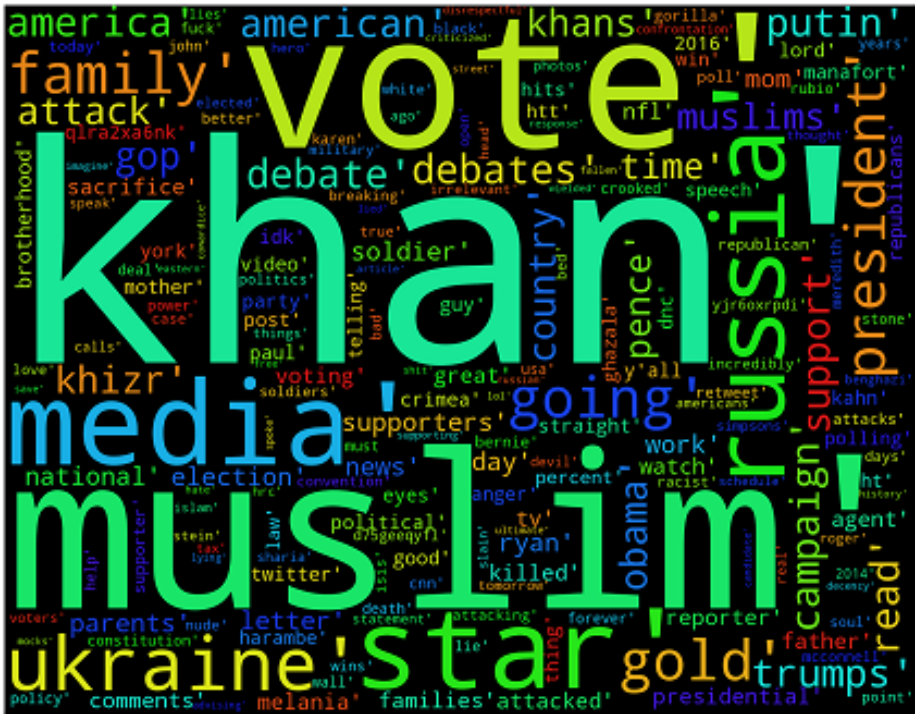
Figure 7: Most Frequent Words Identified for both Candidates



Word Cloud

Next, we created word clouds of the most common terms for each candidate using the Python WordCloud package. As seen in the figure below, the recent controversy surrounding Trump's comments about the parents of the Muslim U.S. soldier, Captain Humayun Khan, who was killed in action, generated a lot of attention on Twitter.

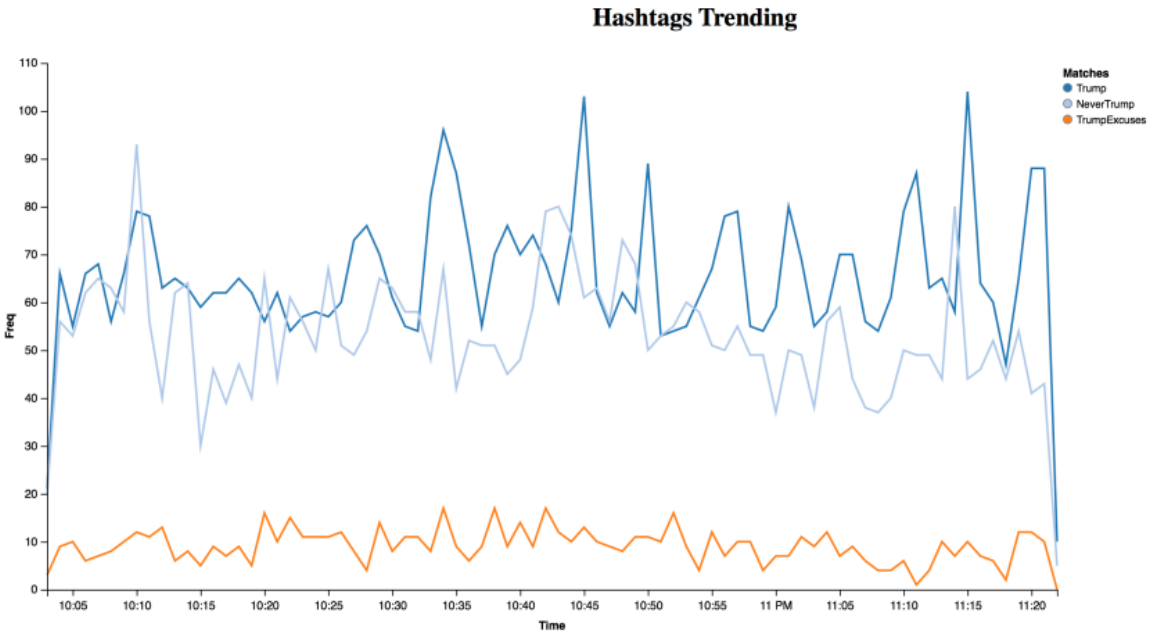
Figure 8: Word Cloud for Tweets Associated with Trump



Trending Analysis

We also analyzed the distribution of certain hashtags over time, and compared the trends of up to three different hashtags over a certain period, to see which topics became more widely discussed over time. In the figure below, we compare “#Trump”, “#NeverTrump” and “#TrumpExcuses” over the course of 85 minutes.

Figure 9: Hashtag Trending Analysis



Sentiment Analysis

In order to evaluate whether Twitter sentiment mirrored the predicted polling results we saw on sites such as FiveThirtyEight.com and 270toWin.com, we used IBM's Alchemy API to perform sentiment analysis on the Tweets, which we pre-processed using NLTK.

We selected California and Texas for this analysis, as these represented states with large populations that also featured voting forecasts that were dominated by Clinton and Trump, respectively. For future improvements, we would like to perform sentiment analysis for the entire United States with more Twitter data, in order to completely cover the map.

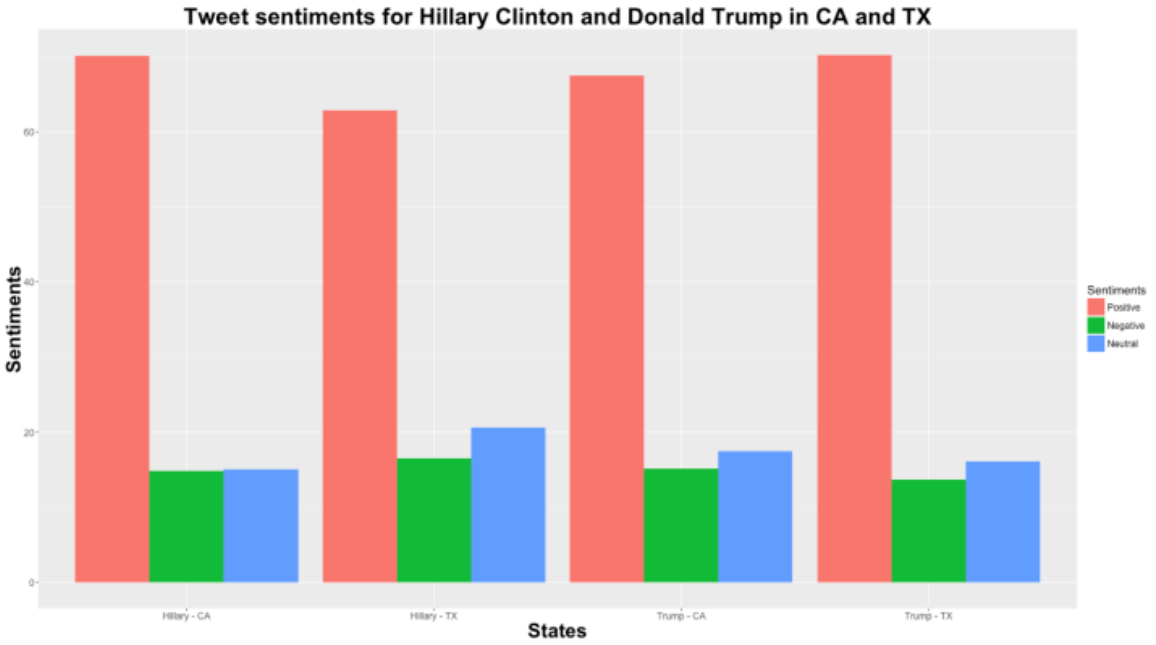
Figure 10: Election Prediction of California

Figure 11: Election Prediction of Texas

As shown in the figure below, the results of our sentiment analysis were quite surprising. Our analysis did not show significant differences in sentiment for

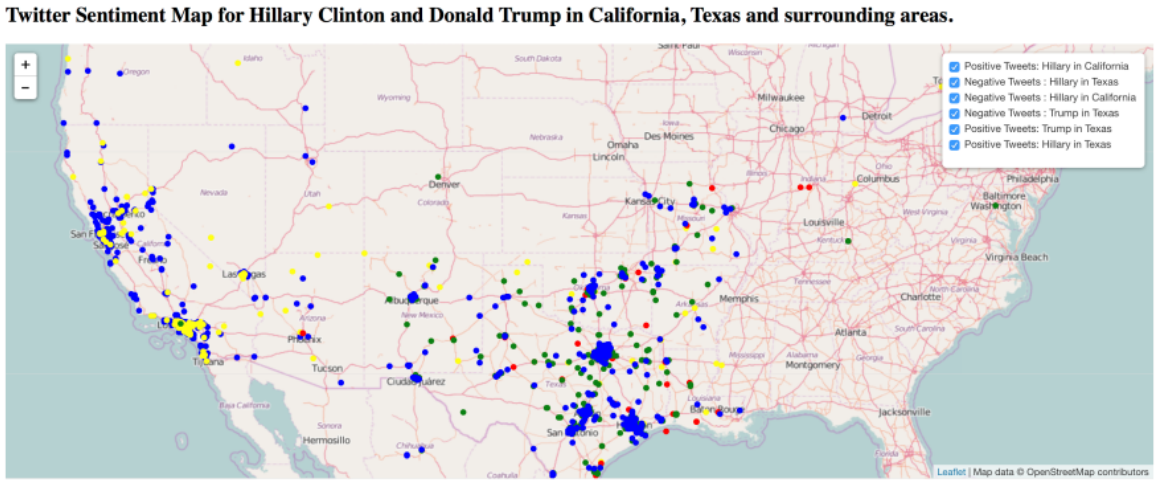
Trump and Clinton between California and Texas, despite our knowledge that the states tend to lean heavily in opposite party directions.

Figure 12: Sentiment Analysis for Trump and Clinton in CA and TX



For further analysis, we mapped the results of our sentiment analysis over a map of the United States based on the geographic coordinates of the relevant Tweets.

Figure 13: Geographic Mapping of Tweet Sentiment



Scalability, Performance Enhancements and Limitations

The Tweepy Python library is used for collecting Tweet data and with code enhancements it can be multi-threaded to capture tweets from every state in the United States simultaneously. However, Twitter streaming is known to limit/interrupt connections to a data consumer if it sees unusual load.

MongoDB is used to store Tweet data and sentiment scores, and is capable of scaling up to meet large storage requirements with supporting hardware.

IBM's Alchemy API is a paid service and future enhancement to conduct sentiment analysis of a significant number of Tweets may require building out an algorithm to conduct the sentiment analysis.

Project Takeaways

Through performing our final project, we did encounter certain challenges and learned key takeaways for future work. First, our approach to the project came with certain limitations for rate of extraction of Tweets and the amount of available storage on a single EC2 instance. We also encountered problems with data cleaning regarding certain characters that were not eliminated during the initial filtering and parsing stages.

Also, sentiment analysis can present significant challenges, which were reflected in our analysis results. For example, our sentiment analysis did not do well in addressing positive words made negative by being paired with "no" or "not" (e.g. "good" vs. "not good"). Sentiment analysis also cannot effectively scale the degree of positive or negative sentiment inherent in a word (e.g. "good" vs. "awesome"). This leads to sentiment score results that may not accurately represent the true sentiment of the text, which could be influencing the results of our analysis.

Furthermore, Tweets at any given time are highly reactive, providing an instant or knee-jerk reaction to current events. Therefore, the content of public Tweets may not accurately reflect the broader trend of how a candidate will ultimately fare in an election.

In addition, the active Twitter base represents a small percentage of the overall voter base, and is biased towards a younger, lower-income demographic. This has potential to skew our results, particularly in a pro-Democrat direction.

Finally, there were also a limited number of public Tweets with geographic coordinate information. This reduces the overall effectiveness of our geographic analysis, and prevents us from being able to draw strong conclusions about sentiment towards the candidates across the United States as a whole.

Future Improvements

For future work, we have identified a few main improvements that we would like to build-in to our project.

First, it would be useful to aggregate multiple sources of data, including data from other social media platforms, such as Facebook, or search engine data from Google, in order to broaden the scope of our analysis and mitigate the biases we faced. Furthermore, it would be interesting to join the text data with detailed polling data, in order to merge the sentiment analysis with existing prediction models and improve the meaningfulness of our results.

In the future, we would like to perform more in-depth sentiment analysis with a greater amount of training data, in order to drive deeper emotional insights. This would involve more/longer collection periods for Twitter streams, and we would also likely need to increase our storage and processing capacity using multiple Amazon instances.

Finally, it would be valuable to add an alert mechanism to notify the user when positive sentiment dips or negative sentiment spikes, or sentiment is persistently elevated or depressed over a long period of time, in order to quantify and provide feedback about what issues are resonating most with social media users and generating the largest reaction.

GitHub Repository

https://github.com/MSannat/W205-5-MSannat

S3 Data Backup

The batch programs provided to collect Twitter stream and run sentiment analysis would provide required data to run the analysis as mentioned in this document.

However, if due to some issue you are unable to get the required data, please let us know and we can provide S3 links for the data used to conduct the analysis in this project.