Datamining and Sentiment Analysis Using Twitter Data

Shengye Chen
200354388
University Of Regina
chen248s@uregina.ca

Group Members
Alexander Biezenski -200315435
Elizabeth Rayner -200365470

Abstract

The project focuses on machine learning in the form of data mining and natural language processing. It takes real tweets, filtered by a particular concept, and uses natural language processing to extract sentiment from it. This is useful for getting a real-world reaction to ideas, policies, politicians and marketing campaigns, as well as many other topics. The project compares sentiment between two concepts over time and their average sentiments in a particular geographic region. It also compares mentions over time to get an idea of the comparative volume of tweets instead of just sentiment. The tweets are harvested from Twitter using a package to get a sample of real time tweets made by real users. The natural language processor derives meaning from text by looking for positive and negative words as well as differing capitalizations and emoji usage. This sentiment can be used to analyze the general sentiment of a region towards a particular concept and compare it to the average sentiment towards a different concept, accurately gauging public perception for marketing purposes. The sentiment output from the language processor is then passed to a webpage, which uses other packages to process the sentiment information, and render it in a clear, easy to understand way.

Key Words
Sentiment Analysis
Datamining
Twitter
Natural Language Processing

Introduction

The project is under the umbrella of machine learning with a specific focus on data mining. Specifically, it focuses on mining and processing Twitter data. The program, in the most general case, has the ability to harvest tweets from twitter and produce maps and line graphs showing the sentiment portrayed by the tweets. The sentiment analysis is the major part of data mining and will be discussed in the sections on data mining and sentiment analysis.

This project goes beyond the general case and harvests Twitter data relating to a specific topic. The topics chosen are United States President Donald Trump and American Speaker of the House Nancy Pelosi, identified in tweets by the keywords "Trump" and "Pelosi". There are a couple of reasons for choosing these as the topics for mining. The first is that a lot of data is generated every minute talking about them. This matters because in order to effectively perform data mining, a lot of data is needed, and these topics provide more than enough. Information about the means to gather that information will be discussed in harvesting Twitter data section. Also, most tweets concerning Trump and Pelosi originate in the United States, which is the area chosen to map. We chose the US because it allows us to demonstrate the differences in opinions over a relatively large area, that also contains many data points; 50 states, and the District of Columbia (D.C.) (although D.C. is very hard to see on the map given its size). Finally, people have strong opinions, both positive and negative about Trump and Pelosi. Unlike something like the color blue, almost everybody who tweets about Trump and Pelosi has some opinion on them that can be analyzed for sentiment and mapped. With only small modifications to the code the topics could be swapped out, so Trump and Pelosi serve more as a placeholder than anything, used to effectively demonstrate the program. There are also tweets harvested that are filtered by the State of the Union. This is a stand in for any major event such as an election and is used to demonstrate volumes of tweets mentioning a specific topic. Throughout the report Trump will be used as the major example as the most data is generated about him, but the tactics described apply to all potential tweets and filters.

The project is an implementation using an already created library that was accessed and used in order to get information. It is a small example of how a real-life system could be used to do research on brands or, specifically in this case, a political campaign. This project shows how artificial intelligence can be used outside of an academic setting in order to affect the world as code is run on real life data sets.

Please note the example tweets talked about in the report are not ones used in the final project but the code remains the same. This file was chosen as an example as it contains many of the variations that illustrate points made. This json file is tweets filtered by the State of the Union speech and was harvested a few hours after the State of the Union was made. Since the data is updated continuously the choice to use one file was made so that points could be more clearly illustrated.

Data Mining

At its most basic level data mining is a computer searching through data that is stored electronically (Chakrabarti), but more specifically, it can be defined as "the process of discovering patterns in data" (Chakrabarti), with the goal of using these patterns in some meaningful way. Depending on how data mining is done, such as its use in this project, it can be included in the umbrella of machine learning; machines working to acquire and process large volumes of data, which in our case, is machines acquiring large volumes of tweets and passing them through a natural language processor. The natural language processing all happens in sentiment analysis, which is talked about more in the section on sentiment analysis.

The majority of the machine learning in the project comes from the extraction and utilization of sentiment. The sentiment analysis when comparing two topic uses machine learning techniques such as natural language processing to take the first step in prescriptive modeling. It looks at variables to say what the more popular concept or course of action would be according to the sentiment of the public. It also utilizes anomaly detection to an extent, identifying states that have strong feelings, either positive or negative, towards the concept by the colors being particularly vibrant. The most obvious form of data mining occurring in the project is text mining. Two forms of text mining take place, described in the section on the totality of the code. The first is showing which of two sub-concepts is the most popular to get an idea of the range a concept has. The second type is sentiment analysis which is debatably the more useful version of text mining as it finds the average sentiment related to a concept which could be used to target particular regional markets. The machine identifies the regions in which the marketing is working and the regions where it is failing and to what extent compared to other regions.

Basics of Harvesting Twitter Data

Twitter data is harvested though a library called Tweepy. This library allows the user to gather a certain number of current tweets from Twitter. The first step is for the user to be authenticated by Twitter as a developer with permission to harvest tweets. Look under problems and notes for more information on the Twitter development account and receiving the keys. Then an API must be set up by the user using the codes given by Twitter. Twitter has multiple API options. The first is the search API, which "allows access to tweets from the past week" (Hanna). The next is the ads API, which "focuses on Twitter's ads" (Hanna). And finally, the streaming API, which is what is used in this project.

The streaming API only allows access to real time tweets, and gets only a sample of those. For more information look under problems. There are two different endpoints that the streaming API uses. The filter endpoint gets data based on keywords, usernames and location ranges, although this is limited to "a few hundred keywords, a few thousand usernames and twenty-five location ranges" (Hanna). The sample endpoint returns a 1% sample of all of the tweets available is a short time range. This project uses the filter endpoint on keywords, filtering on the keywords "Trump" and "Pelosi", as well as the example event "SOTU".

The next step is to set up the keywords that are being tracked. This step just creates a list of keywords that tweets will be filtered by. If a tweet contains all of the words on the list it will be mined and then added to the file created to hold all the tweets. This process produced a json file with all the twitter data mined, 1000 tweets that contain the keywords and all the information about that tweet. This project creates three different files based off of the keywords being searched.

The created file is then loaded. This step is to allow different files to be opened at any time, not just the file that was just created in the mining process. All tweets must then be flattened. This is a process designed to make their data easier to access, especially because some data is nested. A list for tweets is created, then each tweet that was harvested is gone through. Data pertaining to the user's personal information such as name, screen name and location are all stored in a subcategory 'user'. For eases sake when evaluating and analyzing tweets the users screen name and location are saved under a new heading, 'user-screen-name' and 'user-location', instead of under the subheading. The time the tweet was made is also changed to a date-time format and re-saved under the same heading. Then comes the flattening of nested tweets. There are three types of nested tweets. The first is the extended tweet, for when a tweet is over twitters normal 140-character limit. This just requires concatenation of the additional text onto the original tweets text. The second type is a retweeted tweet. Here the tweet being flattened retweeted a different tweet so that tweets username, text and location all need to be stored under new a new tweet. A set of tweet id is built to check if a new tweet has already been harvested. This is to prevent multiple retweets of the same tweet from skewing the sentiment. The retweeted tweet is then flattened like the regular tweets and added to the list. The same thing happens for quoted tweets. The flattened tweet is then added to the list that forms all of the tweet data that will be processed. The tweet list is then transferred to a data frame so that the analysis can be run on all the tweets.

Totality of Code

This section goes over how the code works in its totality. The first step is to set up a connection with Twitter and harvest the data. Then flatten the data and convert it to a data frame. All of this is gone over in more detail in the section on the basics of harvesting Twitter data. After all of that is complete the data is ready to be analyzed. The next step is to set up a function that will be frequently called. This is the check_word_in_tweet function that goes through each tweet and checks to see if a specific word is present in that tweet. It is important to note that this function should be performed on a word other than the words that were used as a filter for mining the data. The examples in this section will be based off tweets that were mined with the phrases "State of the Union" and "SOTU". For checking if a word is in the tweet one would want to use different words such as "Trump" or "Pelosi". The function only works on one word or phrase at a time and returns a logical series of tweets that contain the relevant words. Testing for words is not case sensitive as generally on the internet, proper grammar is not taken into consideration. The function checks the text of the tweet for the word and returns true or false for each tweet. Whether the word is present in the tweet or not is saved into a new column upon the return of the function. This can be repeated for any number of words, creating a new column in the data frame each time.

There are two functions that perform analysis and output results within the python code. The first is plotMapTime. This function works to analyze the sheer number of mentions two words or phrases receive and compares them in a graph. This is useful as it looks at the volume of times a word or phrase was mentioned as that can provide influence over the success of a product or person. An example of this happened in the 2016 election, which Donald Trump won. Whether positive or negative, Trump received significantly more mentions than any other candidate (Presidential Campaign 2016: Candidate Television Tracker), demonstrating how volume of mentions can have an impact. The function takes two words and, for each, checks every tweet for the word, placing the results in new columns, 'word1' and 'word2'. The index for these columns is set to date-time with the unit seconds. The columns are then resampled by the minute, getting the mean for each minute, creating two data frames that are then plotted. The corresponding graph shows over a range of minutes how often the word is mentioned. An example was run with the words Trump and Pelosi that appears in the appendix Graph 1.

The next function used is plotSentiment. It utilizes the sentiment analysis tool that is discussed more thoroughly in the section on sentiment analysis. This section will just discuss the code and the resulting graph produced. This function acts on two words, given by the function call. First the average sentiment is analyzed for each of the words. When the sentiment is analyzed it is saved into two new columns, 'sentiment1' and 'sentiment2'. The index for these columns is then set to date-time with the unit seconds and resampled over one minute, and saved in a dataframe. The corresponding graph is then produced. This graph is similar to the graph produced for mentions but instead plots the average sentiment that appears in those mentions. An example of this graph, also using the words Trump and Pelosi is shown in the appendix under Graph 2. The graph produced would have the y-axis going all the way to negative one if necessary to map negative sentiment or up to positive one to map positive sentiment. The graph produced however only has positive sentiment due to the data used.

Both the produced graphs are relatively rough, with more refined ones created for user viewing later, but they do show some interesting features. They show why data mining can be valuable information as demonstrated in the example used. Graph 1 shows how Trump is consistently mentioned more than Pelosi, at a rate of about 0.5 of the words in tweets versus 0.1. This would indicate that Trump was much more popular than Pelosi, an interpretation that would be called into question by Graph 2. Graph 2 is much more volatile than Graph 1 but in general shows Pelosi with a higher, or more positive, sentiment associated with her than Trump. These graphs are meant to provide a quick look at the data, with more detailed graphs shown in the final output, but they do provide a good starting point for looking at the data.

The final part of the code is used to output sentiment for a single word or phrase for use in the final user output. First, two files are opened, one a new file to export data to and the second to read in states names and their abbreviations. Both the state and the abbreviation are read in, as otherwise only half of the data would be caught as users can either identify their states by the full name or the abbreviation. All of the states names and abbreviations are read in, one pair at a time, with the states information then being sent to the sortByState function along with the concept that is being sorted by and the outfile. First a dataframes is built, that is sorted by the state, either relying on the users entered location or where available the users geographical coordinates. Then the state and the average are written to the outfile. This produced the average sentiment score for each state, which can be positive, negative or nan. A positive score means that the state feels generally positive about the phrase, a negative means the state feels generally negative about the phrase and nan means that the state did not tweet about the phrase so there is no data on it.

Output

For the output, two packages were used, as well as two new languages. The packages are Highcharts and Leaflet. The languages used are Javascript and HTML for the website and output hosted on it. The output from Python is displayed using a website rather than directly from the python code in order to present a clean, easy to understand representation of the data, as the raw text is very difficult to parse. There are three charts or maps produced in the output. The first is a map of the United States. Each state is given a color based on the average sentiment for that state. The first graph is one that tracks sentiment for a concept over time. This is also done in python but the graph displayed on the website is more interactive and easier to view. The final graph is different. The other two displays show the sentiment for larger concepts such as Trump or Pelosi, and the tweets are filtered by those words. The second graph uses tweets filtered by the

American State of the Union speech. This filter is filling in for any major political event such as an election where the two parties, here represented by Trump and Pelosi, are involved. It shows the mentions over time which demonstrates the more popular number wise of the two concepts, as well as the sentiment. This is also similar to the graph produced in python but is once again more interactive and cleaner.

The two line graphs are produced using Highchart. The first step is to collect the data from the python code, as described in the totality of the code section. This data is stored in a json file in the format of: subject[[datetime, value],…]. This is so it can be accurately read into Highcharts. The next step is the construct the configurations for the graph. This involves choosing the chart type, a spline, the x-axis as datetime and formatting the y-axis. The names of the data and the data series for all lines is then read in. Then the chart is drawn using the Highcharts.chart function within the HTML code. This is used to produce the sentiment graph and the mentions over time graph. Examples of these graphs are Graph 3 and Graph 4 respectively.

Leaflet is used to create the map seen for output. This map shows the sentiment related to an object on a statewide basis. It is set up to view the United States automatically by setting the view on leaflets to the United States coordinates and modifying the zoom. The next step is to set up the state lines. This is done with a GeoJSON file that gives coordinates. These coordinates are connected to form state lines. The map uses colors to show differing sentiment. Red colors are a positive sentiment while blue colors are a negative sentiment. Grey means that no sentiment was found for that state. Because of the way sentiment analysis works the colors are set up to be sensitive to slight changes in the range of -0.1 to 0.1 due to the number of filler words that rank as neutral. There are then layers added to the map for each set of data, both Pelosi and Trump. These layers can be turned on or off depending on what information a user wants to look at. An example of the map is given under Map 1.

Sentiment Analysis

The sentiment analysis is the most important part of the project and where the major tool is utilized. The tool in question is the SentimentIntensityAnalyzer, which is imported from the package nltk.sentiment.vader. The goal of sentiment analysis is to attempt to derive what a large number of people feel about a topic by analyzing their average sentiment. Sentiment can be positive, negative or neutral. A neutral sentiment would be a statement of fact with no emotion attached to it such as “cats exist”. There is no way to tell if the poster feels positively or negatively about the fact that cats exist, only that they do. When applied over a large number of people, or in the case of this project tweets, one can understand if the topic has a positive or negative reception. This essentially come down to natural languages processing as it is interpreting the meaning behind words (Hanna). Sentiment analysis works by counting the positive and negative words in a set of text, in this case a single tweet. These words are taken as a portion of the rest of the document (Hanna) and the text is given the positive, negative or neutral score. This is useful for figuring out if a concept is viewed positively versus just being talked about a lot and is useful for gauging the reaction towards a product, policy or, as it this project, a politician.

The vader language processor, which processes natural language is good to use for sentiment analysis as it not only analyzes sentiment associated with keywords such as ‘bad’ or ‘happy’ but also emojis and different forms of capitalization. For example, ‘NICE’ is seen as more positive than ‘Nice’. This relies on qualitative data which is converted to a number to make it more understandable and give values over time.

The first step for sentiment analysis after all the packages have been imported and the data is mined and flattened is to instantiate the sentiment analyzer. The sentiment is generated with the following code:

```
sentiment_scores = ds_tweets['text'].apply(sid.polarity_scores)
```

where ds_tweets is the tweets data structure and sid is the sentiment intensity analyzer. That generates sentiment scores by applying polarity scores to every tweet in the data structures text. The polarity score for each tweet are then returned. There are four values returned: negative, neutral, positive and compound. Positive measures the percentage of words that have a positive sentiment, negative measures words that have a negative sentiment and neutral measures words that do not contribute to the overall sentiment. The compound is the combination of the positive and negative parts. It ranges from between negative one for completely negative text to positive one for completely positive text. It is the compound that is used to analyze sentiment so it is separated into its own data structure. The average sentiment can then be found for tweets containing a particular word.

Here are some examples of how the sentiment analysis works. The tweets text will be included in the appendix and will be reference here. The first tweet, Tweet 1, produces a positive result and it should. The user is clearly, to a human reading, demonstrating that they are in favor of Pelosi and the tweet gets a sentiment score of 0.7865, or feeling mostly positive. Tweet 2 is a completely neutral tweet, getting a score of 0.0. It is an example of a tweet with no emotion associated with it as it is just a statement of fact. When the tweet was manually edited to include the word ‘LOVE’ in all capital letters at the beginning it scored 0.7125, compared to a 0.6369 when spelled ‘Love’ or ‘love’. Repeating the process with the word hate scored -0.6633 for all capitals and -0.5719 for both ‘Hate’ and ‘hate’, showing the difference that one word can make. A fake tweet created for testing, Tweet 3, that had only negative words produced a score of -0.9375, and a fake tweet, Tweet 4, containing only positive words produced a score of 0.9488. This shows the sentiment analyzer to be fairly accurate in detecting the emotion behind the tweet. The system is not perfect however. The flaw being that the context of the word being searched for is not taken into consideration. To the human eye Tweet 1 is clearly disparaging to Trump, however his name does appear in the text meaning that the computer would consider it a positive tweet about Trump. A continuation of the project would be to find a way to take context into account, perhaps by breaking the tweet down into sentences or other logical parts to analyze the sentiment of each part separately, possibly getting a more accurate result. This however could also decrease the accuracy if a tweet was two sentences but had the relevant word in only one of those sentences. Then if a strong sentiment is analyzed in the second sentence it would not be properly factored in. An example of this is the made-up Tweet 5. When the whole tweet is analyzed it produces a score of -0.5106. However, if broken down by sentence the first part receives a score of 0.0 and the second part, which would not be associated with the keyword “Trump” receives a score of -0.5106. This would cause more inaccuracies as there is a risk of the same error as before occurring due to sentence structure in addition to the keyword not being associated with the sentiment. This problem would be a good future step to look into.

Problems and Future Steps

The main library used in this project is called Tweepy. In order to access tweets and use the library one needs to register as a developer with Twitter. This involves filling out a form stating what you will use the data for. Twitter will then approve or deny applications as they see fit. When describing the project, it was made clear it was for educational purposes and so was approved for a free account. This free account has some limits though. The most problematic of these limits is the sampling limit. Twitter limits the amount of data you are allowed to collect at one time. As a result, each of the samples gets 1000 tweets each time. There is also a limit to how often sampling can occur. We found we could get a sample approximately 15 times every 15 minutes, which still limits the data collected. This is significant as it limited certain types of analysis. An example of this is running sentiment analysis for a specific region. By default, a Twitter user's current location is turned off, so the number of users with a location listed is severely limited. A way to work around this is to analyze the text of the tweets for locations as well but this strategy could cause a major problem. It relies on users stating their accurate location in the tweet such as "Loving New York City" while in New York City. This type of analysis would cripple the mapping system when dealing with political data as someone could be tweeting about Trump's foreign policy, say about Mexico, and be in New York. As a result, this solution was discarded. Instead two alternate strategies were employed. The first one is just collect a lot of data and combine it, which was accomplished by running the code many times over a long period of time. The second was to take the users home location, a location users enter themselves, as their location. This presents two problems. The first is users flat out lying about their home location. This was seen as an almost negligible risk as usually when a user lies about their location they are pretending they are at Hogwarts, not Wyoming. The number of users lying about actual states was predicted to be small enough that it would not overly impact the data. The second problem would be users tweeting while on vacation. It was decided this would not be a problem as most users would not update their location for a vacation and their emotions still represent that of their home state. Overall Twitter's limits on collecting data were pretty easily circumvented as more data could be collected fairly easily.

A problem without many work arounds was the problem of creating a network graph. This would be a graph connecting users who retweet and quote tweet other users. The issue with this is that the graph would be highly centralized or highly sparse and be very limited in scope and the amount of information that could be gained from it. The graph would possibly center completely around a node, such as Trump himself, with the only significant number of connections coming out of that one node, giving no other significant information. The other option would be a series of small node networks with basically no connection to each other as a small group of two or three people interact. The only way to effectively prevent this would be to be able to continuously collect large amounts of data so that all connections could be caught and graphs would not be overly centralized or completely decentralized, which was impossible given Twitters collection limitations.

If the project were to be expanded upon there is a couple of aspects that would be improved. Firstly, the machine learning portion related to data mining would be taken to the next level with some predictions over time. Secondly the sentiment analyzer would be improved upon so that it could read context better. Another improvement, provided that more storage space was available, would be more continuous mining of data in order to get it over a longer period of time.

Notes of Running the Code

As previously mentioned one needs to apply for a developer account in order to run Tweepy. Once this account is approved a user is given four values necessary to running the code.

These are the consumer key, consumer secret, access token and access secret. They are values given by Twitter to ensure data is only being harvested by approved entities. We have included the values liked to Elizabeth Rayner's account in our submission so the code should correctly run. If not please contact us as it is most likely an update on Twitters side and the credentials have gone out of date.

Also when harvesting data please note that all the tweets are stored in a json file. Unfortunately, they are all stored on one line of a json file so the data looks terrible. If you would like to reformat it for human eyes we suggest using the website https://beautifier.io to clean up the data. This is only if you want to view the file, the code will run the same no matter what format the json file is in.

In order for the python code to correctly harvest data one needs to create a folder named "twitter_data". This was so that the code would not be overwhelmed by output files.

Conclusion

The project focuses mainly on sentiment analysis as a way to extract meaning from text. It falls under machine learning in two ways, data mining and natural language processing. The datamining portion is done with the assistance of tweepy, a package designed to gather data from Twitter. There is then a comparison of prevalence and sentiment associated with concepts. This sort of data mining could be useful for campaigns or marketing as shows where a concept receives a positive reaction and where it receives a negative reaction. The type of data mining that is most obviously used is text mining, which uses machine learning and natural language processing to derive meaning from thousands of tweets. The sentiment analysis, which relies heavily on natural language processing, uses machine learning to decide if a tweet is positive or negative. If the project was to be expanded upon, the natural language processor would be improved upon in order to derive more context. The project is a good start towards a machine that could accurately, and in real time, give updates as to the popularity of a concept. As it is it gives fairly accurate updates in not quite real time, but does provide a good general idea of a concepts popularity.

Acknowledgements

This project was inspired by a tutorial seen on Datacamp. This tutorial was used as the inspiration for harvesting and analysing Twitter data and as guidance as to how to use some of the libraries. Twitter was very useful for this project, giving access to the data that was used for the project. Leaflet, Moment.js, and Highcharts provided useful packages that were used for output. This project was written in python and used many of its packages including json, os, tweepy, pandas, glob, datetime, matplotlib, and nltk. A section of code was taken from GitHub at the recommendation of Datacamp. It is the code for SListener and allows the harvesting of data more than once every 15 minutes.

References

Agafonkin, Vladimir. “Quick Start Guide - Leaflet - a JavaScript Library for Interactive Maps.” *Leaflet*, 2017, leafletjs.com/examples/quick-start/.

Agafonkin, Vladimir. “Using GeoJSON with Leaflet - Leaflet - a JavaScript Library for Interactive Maps.” *Leaflet*, 2017, leafletjs.com/examples/geojson/.

Agafonkin, Vladimir. “Interactive Choropleth Map - Leaflet - a JavaScript Library for Interactive Maps.” *Leaflet*, 2017, leafletjs.com/examples/choropleth/.

Agafonkin, Vladimir. “Layer Groups and Layers Control - Leaflet - a JavaScript Library for Interactive Maps.” *Leaflet*, 2017, leafletjs.com/examples/layers-control/.

Chakrabarti, Soumen. *Data Mining: Know It All*. M. Kaufmann, 2009, *O'Reilly*, www.safaribooksonline.com/library/view/data-mining-know/9780123746290/content/kindle_split_0.html.

“Convert Numpy, List or Float to String in Python.” *Stack Overflow*, stackoverflow.com/questions/24914735/convert-numpy-list-or-float-to-string-in-python.

Ferro, Rodolfo. “Sentiment Analysis on Trump's Tweets Using Python .” *The Practical Dev*, 24 Nov. 2018, dev.to/rodolfoferro/sentiment-analysis-on-trumpss-tweets-using-python-.

Hanna, Alex. “SocialDataAnalytics-Winter2018/lab04.” *GitHub*, github.com/SocialDataAnalytics-Winter2018/lab04/blob/master/slistener.py.

Hanna, Alex. “Analyzing Social Media Data in Python.” *DataCamp*, 2018, www.datacamp.com/courses/analyzing-social-media-data-in-python.

“Highcharts Javascript Charting Library.” *Highcharts*, www.highcharts.com/blog/products/highcharts/?fbclid=IwAR2HjjnGCPhldF8fZbxW1SvmyLLFyAA59BGcbHdw9cwWTO-aSeefbHyzibo.

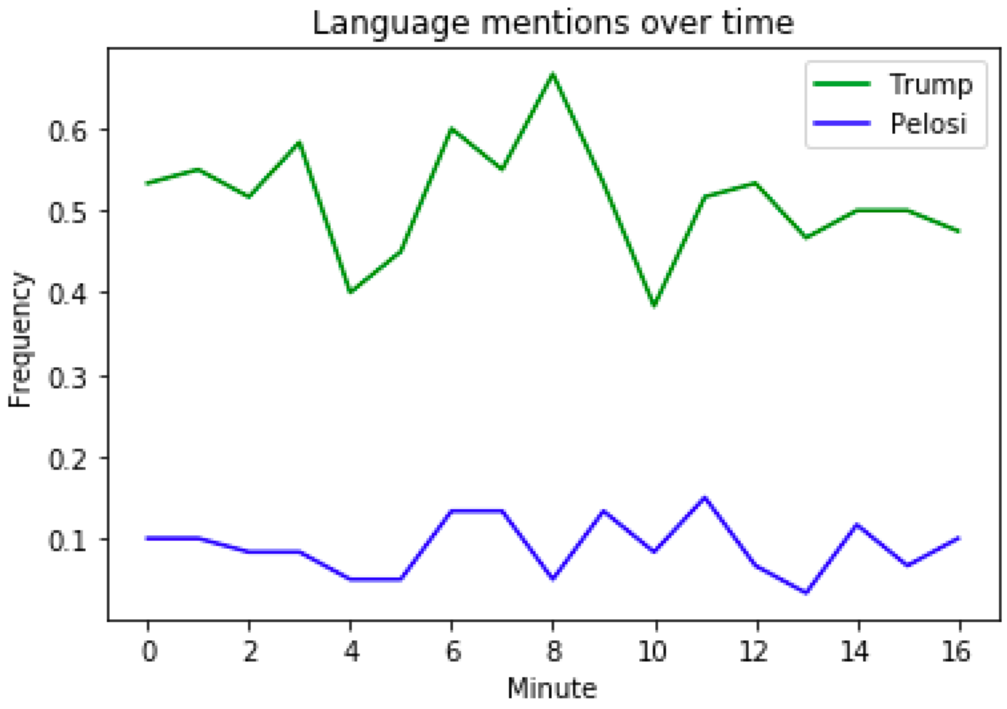
“Online JavaScript Beautifier.” *Online JavaScript Beautifier*, beautifier.io/.

“Presidential Campaign 2016: Candidate Television Tracker.” *US Presidential Campaign 2016: Candidate Television Tracker*, television.gdeltproject.org/cgi-bin/iatv_campaign2016/iatv_campaign2016?filter_candidate=&filter_network=NATIONAL&filter_timespan=ALLTILLELECTION&filter_displayas=RAW.

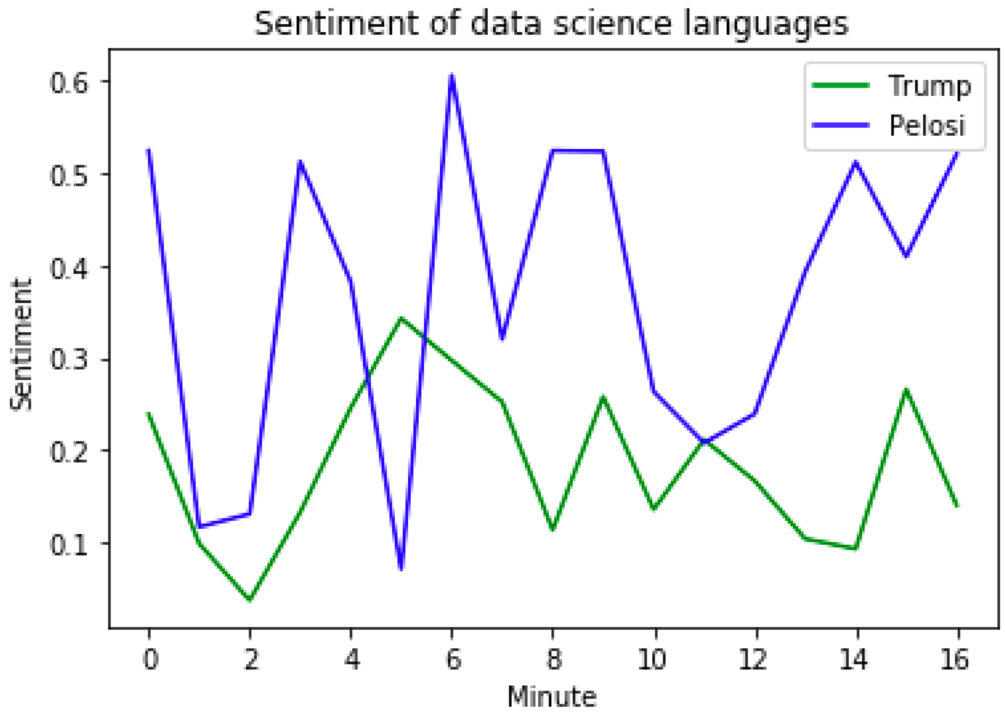
Whalen, Sean. “Creating Beautiful Twitter Graphs with Python.” *Towards Data Science*, Towards Data Science, 15 Dec. 2018, towardsdatascience.com/creating-beautiful-twitter-graphs-with-python-c9b73bd6f887.

Appendix

Graph 1:



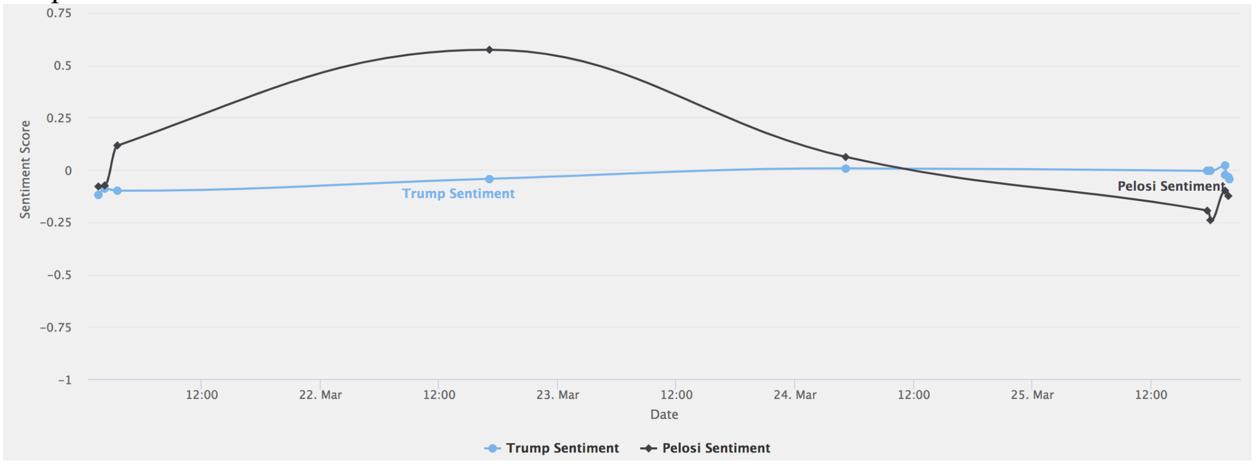
Graph 2:



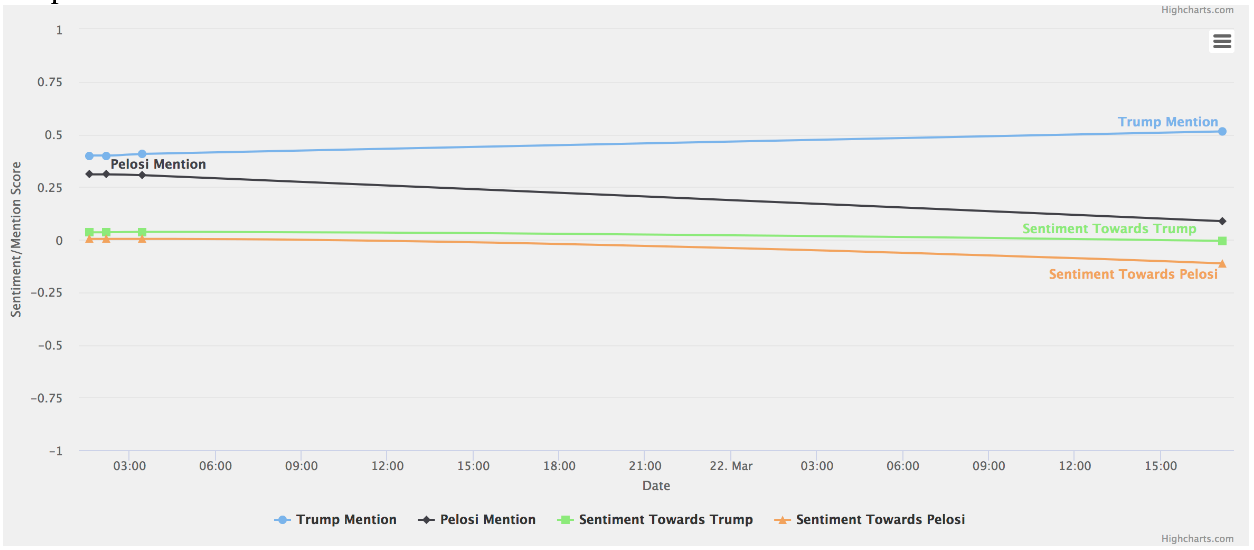
NOTE:

All screenshots of the website were taken on March 26th. As we continually update results in the time before the presentation the images might change. This is just us updating the values to reflect the current state.

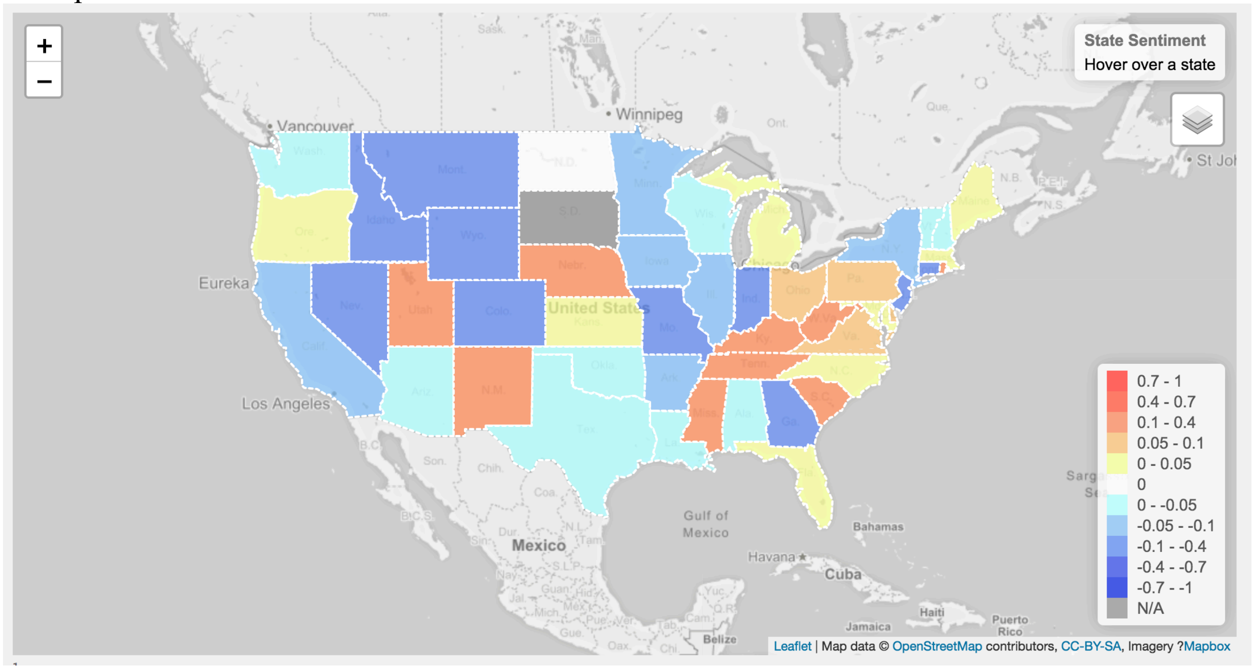
Graph 3:



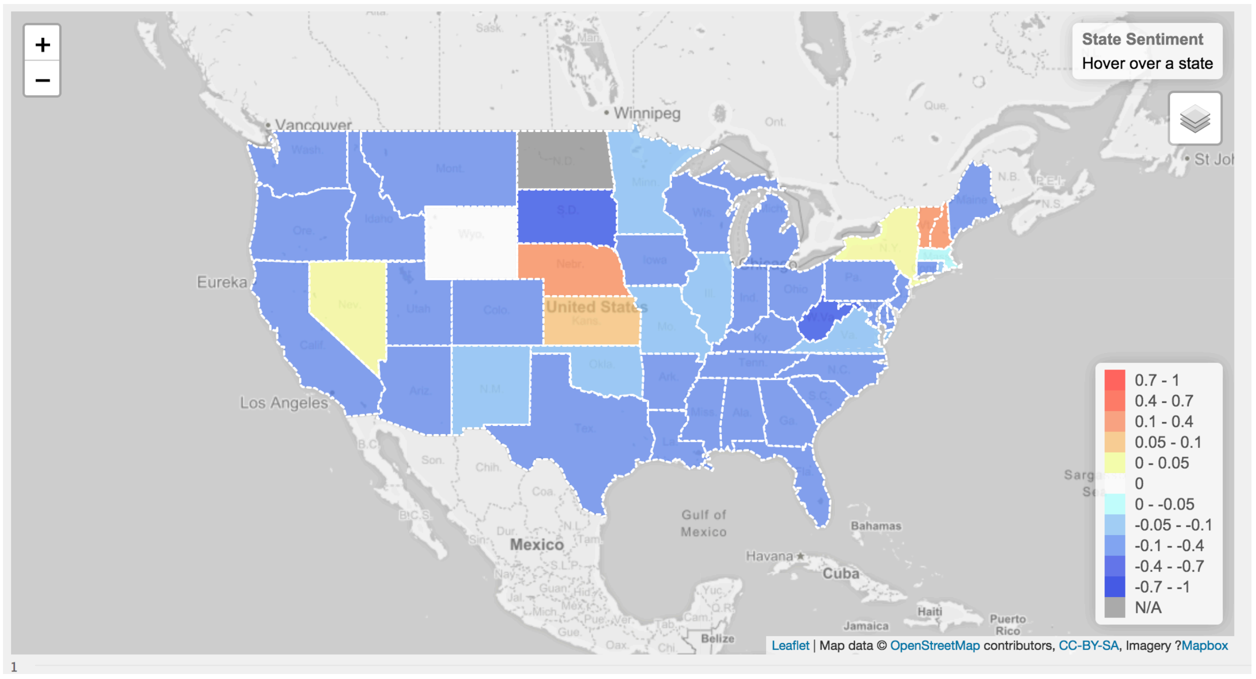
Graph 4:



Map 1:
Trump sentiment:



Pelosi sentiment:



Tweet 1: “I. LOVE. THIS. Speaker Nancy Pelosi jokes that "it will take days to fact-check" Trump 2019s State of the Union speech. "

Tweet 2: "President Trump discussed immigration in his State of the Union address. #Immigration #SOTU #GreenCard #Deportation2026 https://t.co/GHvo0CyNVV"

Tweet 3: "HATE, anger, sad, mad"

Tweet 4: "HAPPY, love, good, great"

Tweet 5: "President Trump discussed immigration in his State of the Union address. He is an idiot”

Link to the website for output: http://www2.cs.uregina.ca/~biezensa/index.html
Lint to the Github for code: https://github.com/jacky1c/Twitter-Data-Anaysis/tree/v1.0