Centaurus Group Twitter Analysis Winter 2017

Travis Robinson
Rohan Gokhale
Chase Hu

Introduction

The Centaurus group created a website that allows an end user to see how Twitter users from across the United States feel about assorted TV shows and movies. While originally our intent was to display this data in real time, that turned out to be impractical. Our project works by ingesting tweets, filtering out tweets that aren't usable for our intended purpose, cleaning them to make processing faster, and then loading them into a Hadoop database that we can process via Hive, which uses a MapReduce algorithm to process said data. When processed, this data is placed into a CSV file, which can then be used by a software package called Tableau to generate graphs and maps used on the website. In the time we had before our cutoff when the graphs and maps needed to be generated, we acquired approximately 100,000 usable tweets.

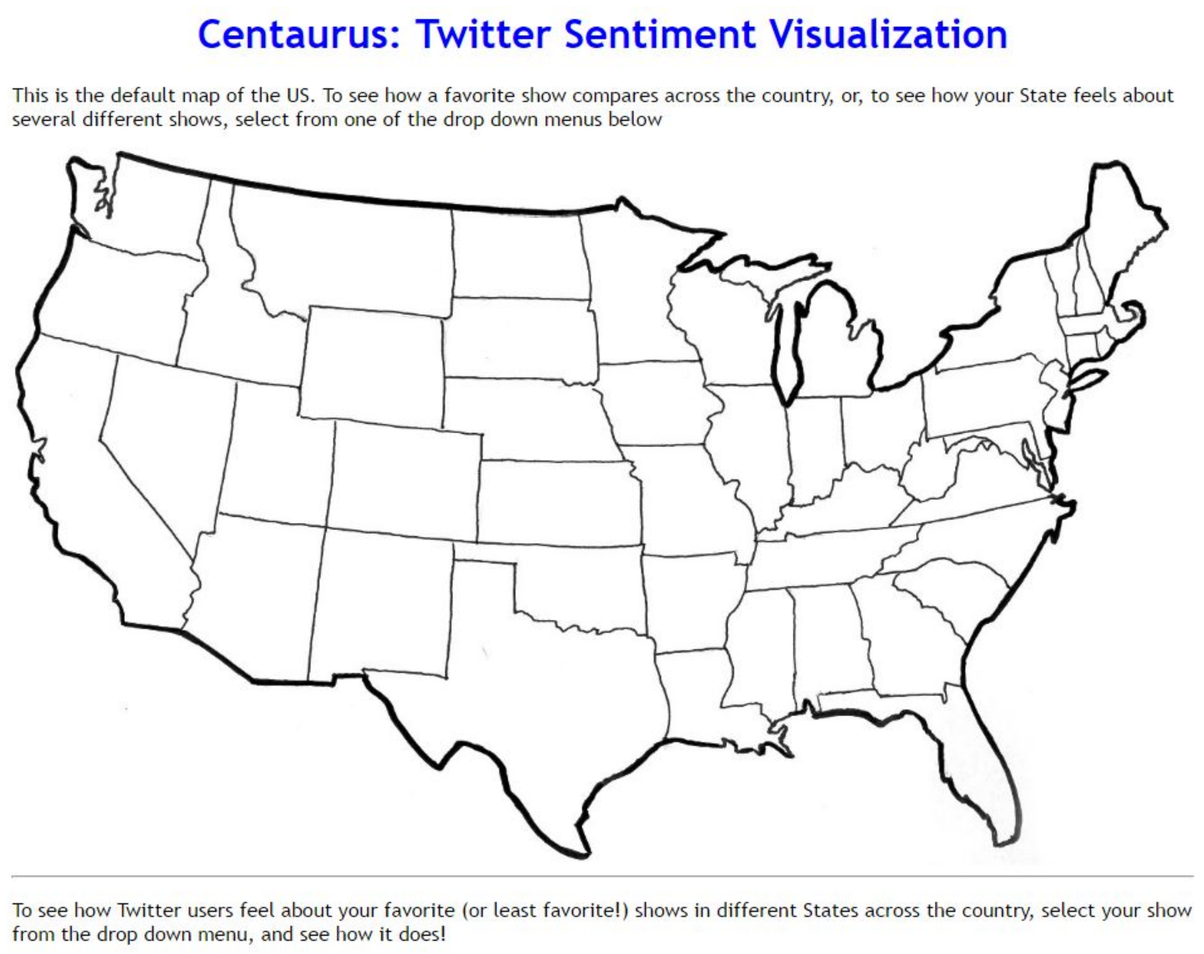
User's Perspective

From the end user perspective, the project consists of a web page, located here (http://web.engr.oregonstate.edu/~robitrav/capstone_test_location/mainpage.php). On this page, the user may select from a list of popular TV shows and movies, and get a visualization of how Twitter users across the country feel about that show or movie. Results are broken down by state, so that a user can see how people in their state feel verse people in other states. The end user may also select a particular state, and see how that state feels about all shows and movies that are in the list.

While the above is for the more common end user, there are other user types that may use our system. These users have access to the code, and may run the system for their own purposes, for example if they wanted to create a different list of objects to search for, such as a different list of shows, political candidates, or hotels. These developer users would be able to do so with a minimal of effort, simply placing the desired list into the appropriate code location, and then running the given commands. From there the developer user would then be able to place outputs into their desired web page, report, etc. **The instructions for developers can be found in our Demonstrate Project submission.**

End User Usage Instructions

When the end user first loads the page, (located at http://web.engr.oregonstate.edu/~robitrav/capstone_test_location/mainpage.php) the first thing they will see is a default view of the United States, along with some intro text containing directions on how to use the site.

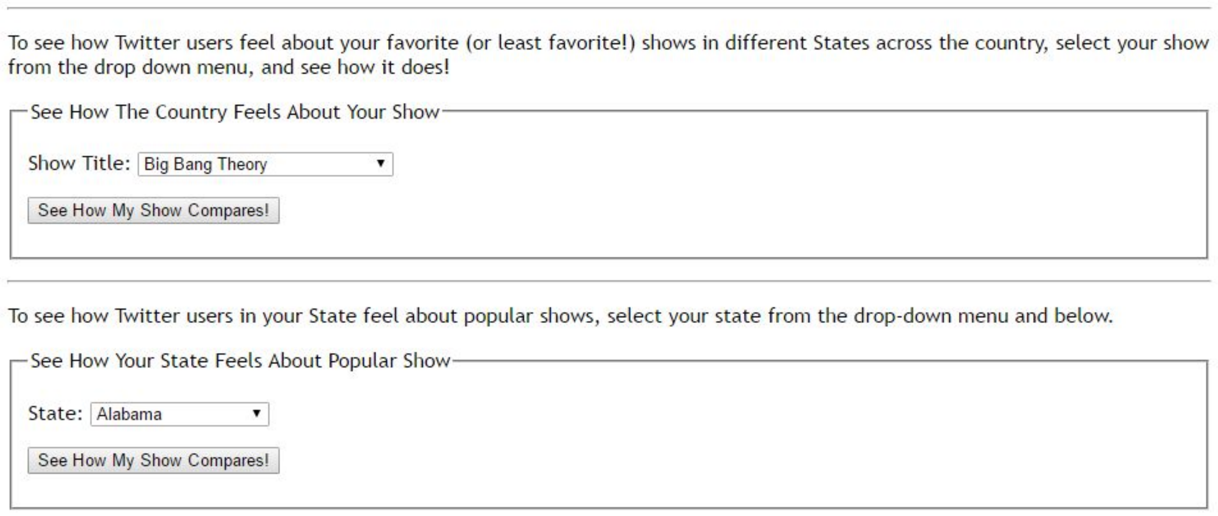


Centaurus: Twitter Sentiment Visualization

This is the default map of the US. To see how a favorite show compares across the country, or, to see how your State feels about several different shows, select from one of the drop down menus below

To see how Twitter users feel about your favorite (or least favorite!) shows in different States across the country, select your show from the drop down menu, and see how it does!

Immediately under this default map, the end user will also see two drop down menus:



To see how Twitter users feel about your favorite (or least favorite!) shows in different States across the country, select your show from the drop down menu, and see how it does!

See How The Country Feels About Your Show

Show Title: Big Bang Theory

See How My Show Compares!

To see how Twitter users in your State feel about popular shows, select your state from the drop-down menu and below.

See How Your State Feels About Popular Show

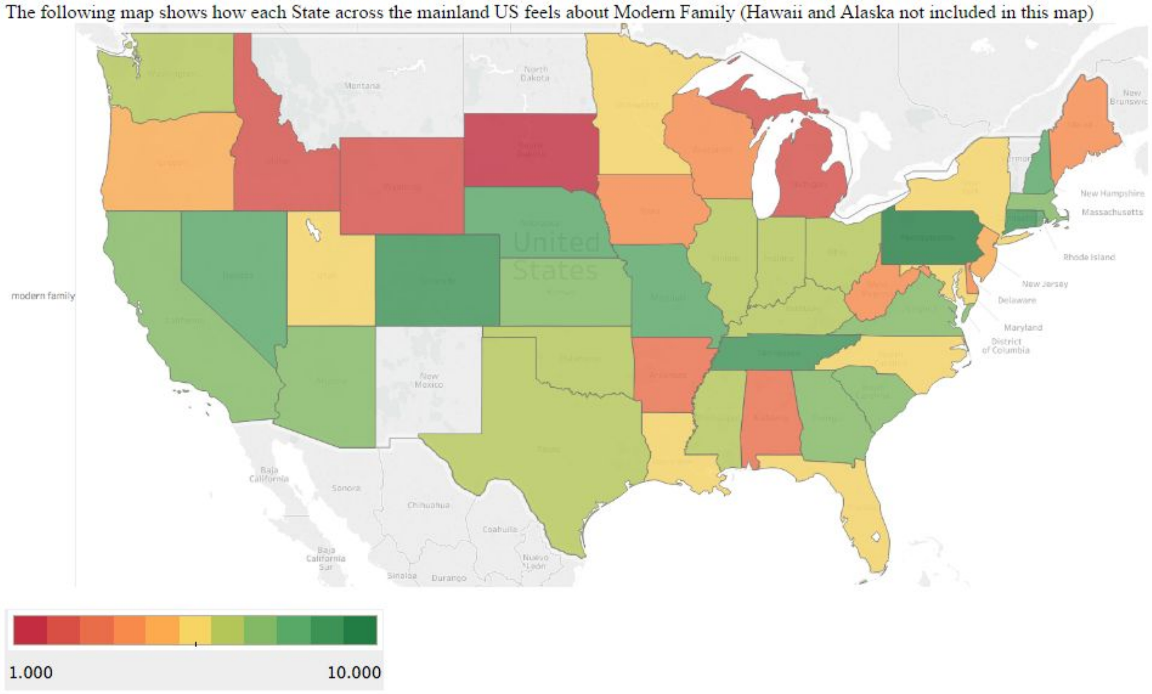
State: Alabama

See How My Show Compares!

The first drop down menu, under "See How The Country Feels About Your Show," will display another map of the United States, colored according to score: green being higher scores, red being lower, and yellow being neutral. Scores are graded on a scale from one to ten, based Twitter users feelings about that show.

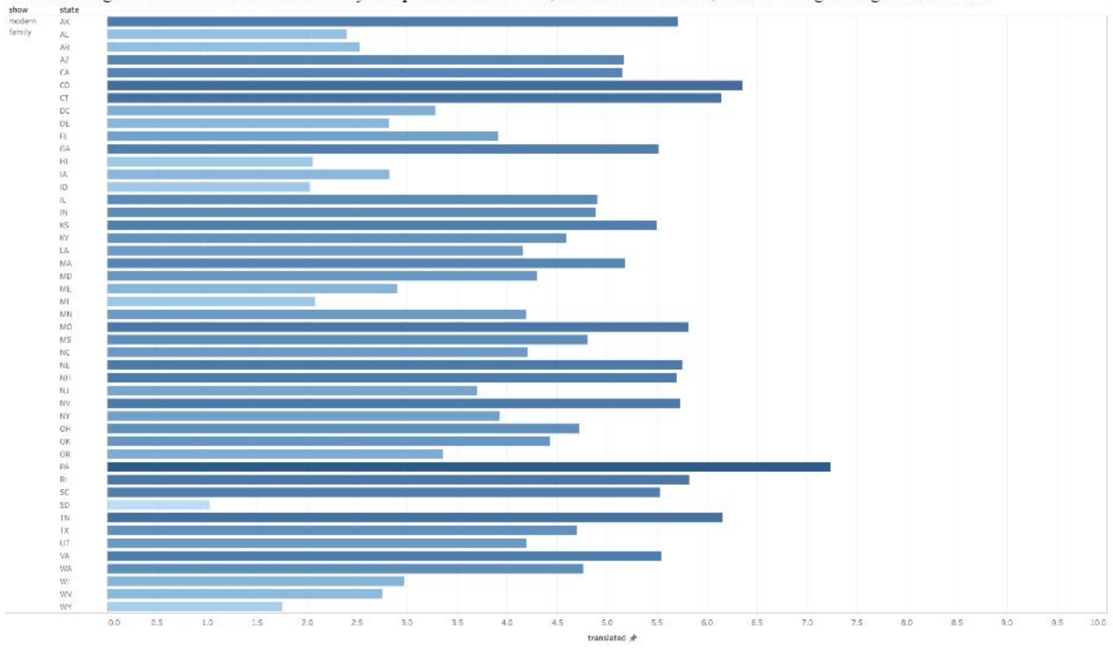
Centaurus: Twitter Sentiment Visualization

The following map shows how each State across the mainland US feels about Modern Family (Hawaii and Alaska not included in this map)

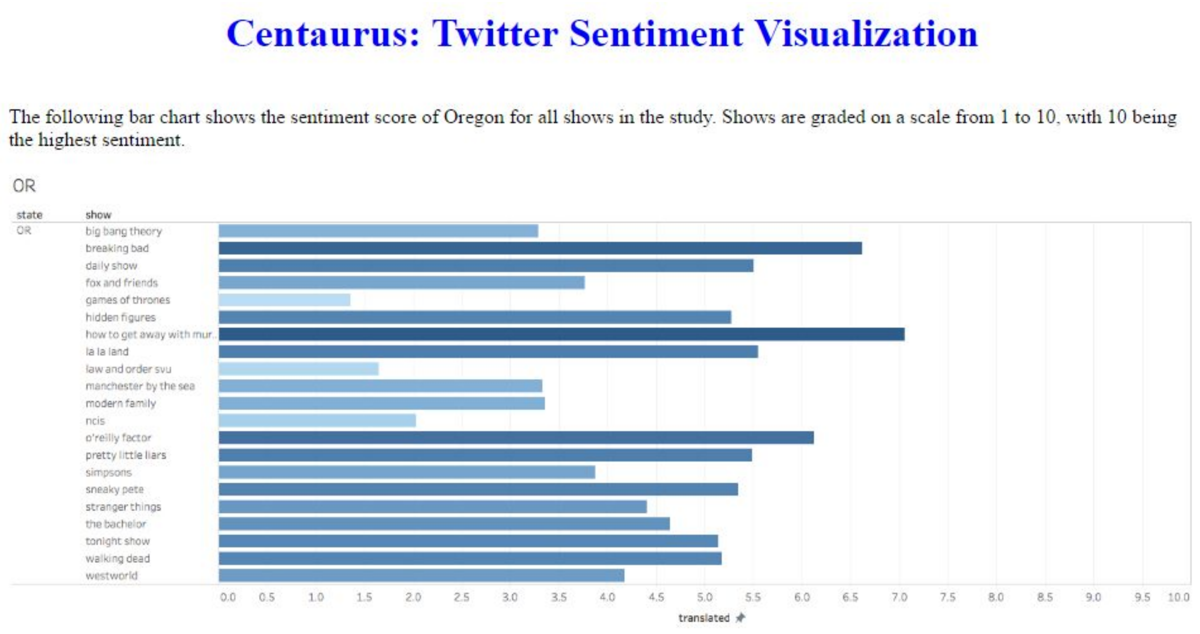


Immediately under the country map is a bar chart, containing the numerical value of the selected show's score in each state.

The following chart shows how Modern Family compares across the US, on a scale of 1 to 10, with 10 being the highest sentiment



The other drop down menu allows a user to select a state and see how Twitter users in that state feel about all shows and movies used in the study. On selecting a state, a bar chart is displayed containing the numerical scores of all shows in that state, graded on a scale of 1 to 10, with 10 being the highest sentiment.



Centaurus: Twitter Sentiment Visualization

The following bar chart shows the sentiment score of Oregon for all shows in the study. Shows are graded on a scale from 1 to 10, with 10 being the highest sentiment.

Technologies Used

Libraries: json, apscheduler, tweepy, dataset, atexit
Languages: Python
Other Systems: Naive Bayesian Algorithm, Tableau, Hadoop, Hive

Apscheduler, Atexit: These are two Python libraries that we used for our project to automate much of the data handling. Apscheduler is a Python library that can be used to schedule tasks to be performed at certain times, or following a particular interval. This was used so that we could convert saved tweets to a JSON format, and reformat and clean the JSON formatted tweets into something that could be used by our sentiment analyzer. Atexit works similarly, allowing tasks to be scheduled on program exit. This allowed us to ensure that data was saved on program exit or in the event of a crash.

Hadoop: Hadoop is a framework originally created by Facebook and now run by Apache. It's a framework used for running applications across a computer cluster, and is made up of two primary parts: the map-reduce part, which allows any node in the cluster to process a portion of data, allowing parallelization of data processing (allowing data to be processed twice as fast, or even faster, depending on the number of nodes in the cluster), and the HDFS, or Hadoop Distributed File System, which is how the data is stored that allows MapReduce to achieve the parallelized processing improvements.

Hive: Hive is a system for storing data within the Hadoop framework. Hive allows for the use of parallelization via map-reduce algorithms with SQL-like queries; this greatly speeds up data processing for large data sets. In other words, by using Hive, we can process our data faster than we otherwise could, because Hive allows more than one row of data to be accessed at the same time, while without Hive only one row could be accessed at a time.

Naive Bayesian Theorem: The Naive Bayes algorithm is a technique to classify pieces of text based on Bayes's Theorem. The equation for Bayes' Theorem is below:

$$\underbrace{P(c \mid x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x \mid c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

(Source of Picture: https://www.analyticsvidhya.com/blog/2015/09/naive-bayes-explained/)

The algorithm works by first being 'trained' with a data set (a set of tweets in our case) that have predetermined scores or sentiment. The training involves constructing a frequency table from the training set. The word counts are converted into probabilities, which represents how likely tweets containing each word are to be positive or negative. This is done by taking the number of times each word occurs and dividing it by the total number words in its respective positive or negative subset, effectively assigning a degree of 'positiveness' or 'negativeness' to each word. Then, for each tweet, the sum of the degrees of probability for each word, arriving at total positiveness and negativeness for each tweet. If one of those sums is higher, then the tweet is appropriately labeled positive or negative. A score between -1 and 1 is obtained by dividing the difference between those sums by the addition of those sums. This ensures that a very positive tweet will have a score close to 1, and a negative tweet will have a score close to -1. Tweets that have a similar positive and negative sum will have a score closer to 0.

Tableau: Tableau is a data visualization software that we used to create charts and maps from different data storage types; we used CSV files, though Tableau can also create them through database connections. Initially, we had thought that Tableau could be used to create the images we wanted in real time, by automating the map and graph images creation whenever a call was made to it, and then saving the image into the desired directory. The website would then be able to access the images and post them when they were requested by an end user. Unfortunately, that didn't prove possible, and we had to pursue other routes detailed in the section of project plan changes.

Tweepy: Tweepy is a Python library that allows for easy accessing of the Twitter API. Without use of this library, there would be more work required to access tweets, due to the need to maintain an open HTTP connection that can constantly receive data (the Twitter API operates as a stream, and so a constant connection must be maintained, similar to reading from a file of infinite length). The Tweepy library handles all of that for us, as well as allowing us to only receive tweets that make it through a filter (this way we only need to handle tweets that contain the shows/movies we desired, instead of having to go through all tweets ourselves and filter out the ones we don't want. Of course, there was still filtering that we needed to do, due to Tweepy only allowing us a single set of filters.)

How Everything Functions Together

The whole system starts off with the scraping of tweets: this is done by the ‘tweet_scraper’ Python script, which is run by the ‘run_me_to_get_tweets’ Python script. While the tweets are being scraped, once an hour all collected tweets are dumped into a file using a JSON format, and stored in the ‘CS467/get_tweets/acquire_tweets/unclean_tweets’ directory. These files are read by the ‘tweet_cleaner’ Python script, also run by ‘run_me_to_get_tweets,’ which further filters these tweets for ones that we are able to use (for example, not all tweets include a US location). The ‘tweet_cleaner’ then stores these cleaned tweets in a tab delimited file contained in the ‘CS467/get_tweets/clean_tweets’ directory, and keeps a list of all JSON files it cleaned in the same directory as the uncleaned tweets.

From here, the tweets have their sentiments scored by the ‘ngAlg’ Python script, which stores them in the ‘nbScores.txt’ file in the directory ‘CS467/sentiment_analyzer’. After tweets have been scored, they are loaded into an HDFS database, which allows for faster processing due to the parallelization of data, since the HDFS is able to make use of the entire cluster for processing.

After this, a Hive script (Hive being the SQL-like system used with Hadoop) makes a query to find the average sentiment scores for all the shows analyzed across all the states, and outputs them into a CSV file. This CSV is then processed by Tableau to produce graphs and maps as images, which are loaded into the website. The website is then hosted in the public_html folder of one of our team members (really it can be hosted anywhere, but in this case it saves us money on hosting fees) where it is available for viewing by anyone interested.

What Each Member Accomplished

Rohan Gokhale: Rohans primary accomplishments were in getting the project environment created, by setting up the Google cluster and Hadoop framework that we needed to use for our project and to do the more effective parallelization of our analysis. He also wrote the script that did the the naive Bayes analysis that our project eventually settled on for our sentiment analysis, and wrote the program that would acquire tweets from Twitter.

Travis Robinson: Travis main contributions were in building the website that would display the data for the end user to see what sentiment results were. He also built the script that would filter, clean and store tweets that had been acquired by Rohans scraper, and integrated them so that they could be run and then left to their own devices. He also implemented the initial sentiment analyzer, which had been written by the group as a whole.

Chase Hu: Chase's contributions were to build the databases that so much of the original project would have relied on, as well as creating all of the charts, graphs, and maps that were used by our project. He also created the initial sentiment dictionary that was used for our first analyzer, and did virtually all of the testing that our project required.

In addition to each member's individual contribution, all team members were involved in writing the algorithm that was used for our first sentiment analyzer. After that one failed to produce results we considered accurate enough, all team members wrote another separate analyzer, so that we could more efficiently divide our work (Rohan's method ended up being the most accurate). Finally, all team members were involved in the research our project required, which was a lot, since this was a field that none of us had much familiarity or experience in.

Major Deviations

The most significant deviation that came up in our project was in how much integration and automation we were able to introduce. This was due to our having a misunderstanding of what we could accomplish with Tableau, one of the tools that we had decided to use; we had been under the misunderstanding that automation and integration with Tableau was possible. That is, we'd thought that we could set up Tableau to automatically generate the graphs and maps that we desired whenever a call was made, so that every time a user wanted to check out a show on our page, the page could make a call to Tableau, which would then generate the graph (essentially, an API). Unfortunately this was not possible without purchasing licenses. Instead we used static images, generating them via Tableau with the data that we had, and then saving the images in the proper location. This did produce very nice images, far better than we were able to produce through other methods, but it did mean the website data is static and had to be manually updated.

Another change that we had to make was to our sentiment analysis algorithm. While not expressly laid out in the project plan, our initial plan was to acquire a sentiment dictionary, which we then used to score tweets. The method we originally used was to add to the tweets score based on whether a word was positive or negative, and apply any modifiers to it (for example, "very good" is worth more than "good", "not bad" is considered a positive sentiment while bad is negative, etc.), and sum up the score from these words. The end result of this though was unsatisfactory; this method typically produced correct results around half the time.

An alternative method that we tried was related to using a dictionary that was generated based on whether prior tweets were positive or negative, and then comparing the ratio of and negative words to the total number of words. If a positive ratio was higher, a positive score was returned, and if the negative ratio was higher, a negative score was returned, provided that the ratio of positive or negative words was at least a quarter of the total words in the tweet. This method gave us a score around 60% correct, which we determined wasn't high enough. Since this method was not terribly different from the original method, the difference could have been from random variation in the test sample or from the fact that the sentiment dictionary was more geared towards analyzing tweets rather than longer text.

Another method for implementing sentiment analysis was using a combination of Python and the Naive Bayes classifier (from the NLTK library). We obtained these training tweets from https://inclass.kaggle.com/c/si650winter11/data and http://alias-i.com/lingpipe/demos/tutorial/sentiment/read-me.html. We will discuss the significance of the differences between these sources below. The training involved constructing a frequency table from the training set. For the subset of positively scored tweets, all of the words in those tweets were counted. The corresponding frequency table then consists of every word found in those tweets, along with the number of times that word occurred in the set. The same was done for the subset of negative tweets. Next, the word counts are converted into probabilities, which represents how likely tweets containing each word are to be positive or negative. This is done by taking the number of times each word occurs and dividing it by the total number words in its respective positive or negative subset, effectively assigning a degree of 'positiveness' or 'negativeness' to each word.

One thing that made a big difference in the results we obtained from using the naive Bayes classifier and algorithm was the training set that was used. One set came from pre-scored tweets while another set came from pre-scored sentences from movie scripts. We found that the training set using tweets were more accurate than using the training set that used the movie scripts. This made sense since the syntax of a tweet is much different than how people talk in real life.

Final Statements

Our initial desire was to give users the ability to get real-time updates about user preferences about TV shows and movies. This didn't exactly turn out the way we intended, but the core of our project, to allow users to visually see how people in their state, and compared to other states, felt about their prefered shows, was still intact. If we hadn't gone down the path we did, for example using a tool other than Tableau or if we'd had more time after going the Tableau route to correct our error, we may have been able to develop the full functionality that we desired.

We also discovered that using a sentiment dictionary with the algorithm that we'd implemented, was not a very effective method for doing analysis, as it gave us poor accuracy. We hypothesize that the way people tweet has something to do with the accuracy. Given only 140 characters in each tweet, users had to be selective for what they wrote. We had much better results using a naive Bayes method that used previously scored tweets as its training set.

In summary, we completed the main goals we set from the onset of the project. We could have designed all project tasks more accurately if we'd had a better understanding of the tools that were needed and their limitations at the beginning of the project. We are still satisfied with the state of the project due to the highly accurate results of tweet sentiment scoring using the naive Bayes method as well as the core functionality of the website, allowing users to visualize sentiment scores in a variety of ways, which in the end was our main goal even if we had to follow a different than expected route to get there.