**Twitter Sentiment on Clinton vs. Trump**

Manasa Hariharan

May 2019

**1. Abstract**

In this project, we try to estimate the sentiment on twitter towards the 2 candidates for the 2016 US Presidential election, Hillary Clinton and Donald Trump. Tweets related to the election are mined at different timepoints in the election campaign and a labeled dataset is also built with the help of campaign hashtags. The labeled dataset is used to tune a sentiment analysis model that is used to predict the overall sentiment at different timepoints. Finally, comparison between the model results and polling results from the same dates is done and a network of users and their interactions is presented for 1 timepoint

**2. Introduction**

Twitter is a social network and microblogging platform and was a platform for debate related to various topics and the major source of breaking news related to recent US elections with over 1 million tweets a day related to the election. Several studies have used Twitter data to analyze public sentiment in various domains including political analysis and have also attempted to predict the outcomes of elections all over the world [1][2]. A lot of the studies are lexical-based sentiment analysis. The lexicon-based approach involves calculating the sentiment orientation of a document from the individual orientations of the words and phrases in a document using manually created dictionaries such as SentiWordnet, VADER [3] etc. However, lexical analysis has been shown to perform poorly in the informal and unstructured language of twitter [4]. Recent research has also focused on supervised methods for classifying sentiments of political tweets using hand annotated tweets as the validation set. Moreover, research has suggested that it is possible to differentiate Republican and Democrat users based only on the words they use in tweets [5]. This project differs from traditional analysis because instead of classifying sentiment as positive or negative, we are trying to classify tweets as supporting one candidate or the other. Hence, this is sentiment towards entity rather than just the polarity of sentiment. Due to the lack of labeled data in this domain, we have innovated and collected tweets that contain hashtags that indicate a clear affinity towards one of the candidates (eg. #maga, #imwithher etc) and trained a supervised classifier on these tweets. Since the election has months of discussions, primary polls and debates preceding it, we have also collected data at 6 different timepoints to understand how the sentiment changes over the course of the campaign.

**3. Methodology**

**3.1 The Data**

The data was scraped from twitter’s website. Twitter has a very useful API and a python client that lets us extract tweets using parameters. However, it will not return tweets that older than 1 week/ 10 days. Since our data includes tweets that are around 2 years old, we made use of GetOldTweets, a python library that scrapes the twitter website based on a basic set of parameters passed (start and end dates of search, words and phrases in the tweets, sent by or mentioning users, geo location etc.). There were 2 stages to the data collection, for the labeled dataset, 4 lists of hashtags were used to collect tweets that had hashtags that indicated sentiment towards the candidates: Pro-Clinton, Pro-Trump, Anti-Clinton and Anti-Trump. Tweets were collected between the dates 15th July and 10th November of 2016. To reduce noise, another level of filtering was done with general phrases and hashtags referencing the candidates. The lists of hashtags were used from analysis that was done by Bovet A, et al [6] using label propagation to discover election-related hashtags. Once this data was collected, the data was labeled into 2 classes: 1) Anti-Trump or Pro-Clinton and 2) Pro-Trump or Anti-Clinton. Tweets that contained hashtags from both classes were removed to remove ambiguity. Retweets were also removed to reduce redundancy. For the second part of data collection, data from different time points were collected using hashtags and terms that referenced to either of the candidates. This data is essentially the testing set of our model. The filtering terms used were *donald AND trump, @realdonaldtrump, hillary AND clinton, @hillaryclinton, #donaldtrump, #hillaryclinton.*

**Number of Tweets per class - Labeled Dataset**

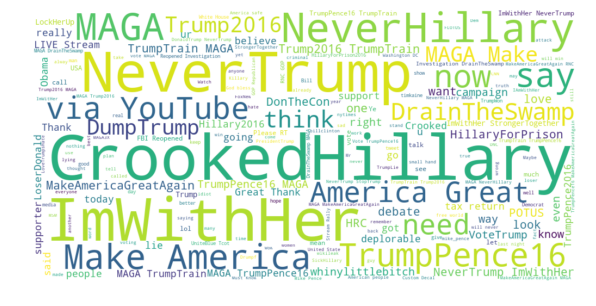
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Pro-Clinton/Anti-Trump (PC)** | **Anti-Clinton/Pro-Trump (AC)** | **Avg tweets per user-PC** | **Avg tweets per user-AC** | **No. of users-PC** | **No. of users-AC** |
| Without Filtering | 500,385 | 926,846 | 4.00263 | 4.874416 | 125,014 | 190,145 |
| With Filtering | 264,443 | 418,879 | 3.34471 | 4.31398 | 79,061 | 97,088 |

**Unlabeled Datasets**

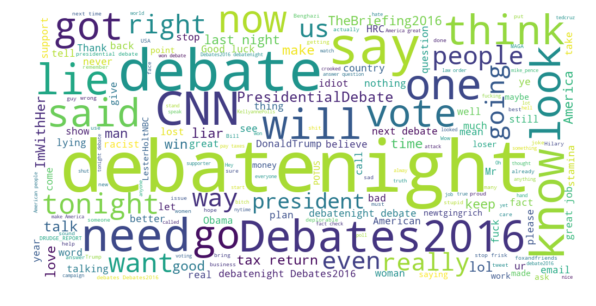
|  |  |
| --- | --- |
|  | **Number of tweets** |
| Republican convention (18-21 July) | 225639 |
| Democratic Convention (26-28 July) | 196740 |
| 1st Presidential Debate (26th Sep- 27th Sep) | 248697 |
| 2nd Presidential Debate(9th Oct- 10th Oct) | 310634 |
| 3rd Presidential Debate (19- 20 Oct) | 241004 |
| Election Day | 165302 |

**3.2 Text Pre-Processing**

Before using the labeled tweets to build a classifier, the text needs to be cleaned and converted to a matrix of features. Tweets are highly informal, unstructured with many spelling mistakes, html links and hashtags. Hence, cleaning for twitter requires more steps. First, the data is tokenized, hashtags and html links are also included as features. The hashtags that were used to scrape the data were also removed from the text to ensure unbiasedness in the final model. Spellcheck algorithms were attempted, but they didn’t lead to a decrease in final scores and hence were not used. Tweets have slang and often contained emphasized words such as ‘wayyy’, ‘tooo’, ‘yippeee’, multiple exclamation points etc. These were shortened into their proper English words. Tokenization is splitting up tweets into lists of words. Eg: *“Donald Trump is a liar #imwithher”* will be tokenized as *(“donald”, “trump”, “is”, “a”, “liar”, “#imwithher”)*. Finally, these tokenized lists will be used to build a corpus by taking all the words used in the entire dataset. Any word that occurs a minimum of 1 time in any of the tweets is included in the corpus. The feature matrix is built with the help of the corpus.



***Fig 1****: Word cloud from labeled dataset samples Obvious words like names of candidates were removed*



***Fig 2****: Word cloud from tweets on the night of 1st Presidential debate.*

We use a bag-of-words model to represent the text features. In the bag-of-words model, a text is represented as a bag of its words, disregarding structure or grammar, but keeping multiplicity. N-grams are used as features in the bag-of-words models with n = 1,2, 3,… and so on. N=1 would just be the individual words in the bag, n=2 will be all the consecutive pairs of words in the text. Continuing the example from above, the unigrams from the tweet would be the same as the tokenized list, while the bigrams would be *[(‘donald’,’trump), (‘trump’,’is’),(‘is’,’a’),(‘a’,’liar’),(‘liar’,’#imwithher’)]*. Bigrams can increase context and introduce more information to the model. In the final model, both unigrams and bigrams were used in the bag-of-words model. For the data cleaning, the nltk library in Python as a tweet specific tokenizer which implemented most of the above tasks. sklearn has a vectorizer that was used to combine the n-gram dictionaries into a sparse matrix. The total number of features in the training dataset was over 2.4 million.

**Number of features**

|  |  |
| --- | --- |
| Using only Unigrams | 439708 |
| Using only bigrams | 1977894 |
| Using both Unigrams and Bigrams | 2.4 million |

**3.3 Stochastic Gradient Descent Algorithm**

The Gradient Descent Algorithm is used to optimize any convex, differentiable objective function. To do so, it picks a starting guess of the parameters of the function and calculates the gradient of the function with the chosen parameters. It then searches the values in the surrounding parameter space to find parameters that give a lower gradient for the function. This way, it iteratively finds parameters that give the lowest gradient for the function, and hence the minima of the function. The Stochastic Gradient Descent Algorithm (SGD) is a variant of gradient descent where instead of using the entire dataset to calculate the gradient every iteration, it randomly chooses samples to use to calculate the gradient.

Where w is the set of parameters, is the learning rate, is the value of the function Q associated with the i-th observation in the dataset.

It has proven to be just as accurate as Gradient Descent [7] while being much more efficient and fast. SGD has been applied successfully in many large-scale text-based problems due to their large number samples and sparse feature matrices. SGD can also incorporate a regularization term to penalize the number of features in the model.

)

Where R(w) is the regularization term and is the coefficient and a hyperparameter that needs to be tuned. R(w) can be L1 or L2 norm where L2 norm: , and L1 norm: .

L1 norm will result in sparse solutions and the final number of features used being possibly less than the total number of features. A number of loss functions are also possible to use with SGD including log loss, which corresponds to logistic regression, hinge loss, corresponding to linear Support Vector Machine and modified-huber loss which is a smoothed hinge function that is more tolerant to outliers. Python’s sklearn library has a SGDClassifier function that implements this algorithm and an easy to tune framework.

**4. Analysis**

The SGDClassifier has many parameters that need to be tuned. To find the best model possible, the labeled dataset was divided into test and train of 10/90 split. The train dataset was used to tune the hyperparameters of the model using Grid Search with 7-fold cross validation. 7-fold cross validation is done by splitting the shuffled dataset into seven parts. The model is then run 7 times with each fold acting as the validation set and the other 6 forming the training set. Grid Search tries all parameter combinations and picks the best set of parameters based on which combination yields the lowest validation score on the chosen metric. The validation score for each combination is calculated by taking the average of all validation. F1-score is used as the metric in Grid Search. To understand F1-score, we need to understand precision and recall, 2 very useful metrics to understand classification results.

Precision = and Recall =

True positives would be cases were tweets were correctly classified as PC, False Positives would be cases were AC tweets were incorrectly classified as PC. False Negatives are cases were PC cases were incorrectly classified as AC. F1- score is the harmonic mean of precision and recall. F1- score is better than Classification score in the cases of imbalanced datasets.

F1-Score =

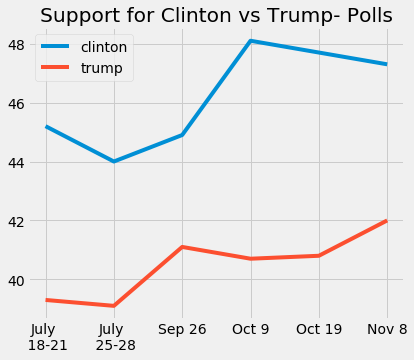
Since this dataset is imbalanced, Balanced Class weights would try to balance the overall probabilities of classes in the test dataset. However, they may end of overcompensating and biasing the final results sometimes. Different alpha values and loss functions were also tried. A summary of all the parameter combinations is given below:

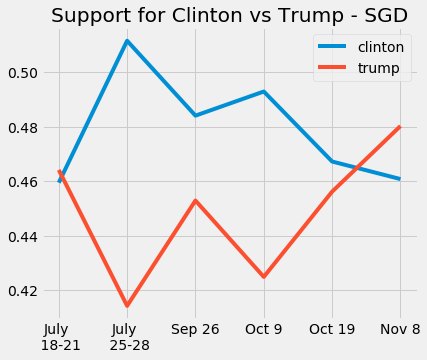
|  |  |
| --- | --- |
| Loss functions | Log, hinge, modified huber |
| Class weights | None, ‘balanced’ |
| Alpha | 10 values between 10^-1 to 10^-7 |
| Penalty | ‘l1’, ‘l2’ |

The best results were given by the parameter set class weight = None, Alpha = 4.641588833612782e-05, penalty = ‘l1’and loss = ‘log’ using both unigrams and bigrams. Some other well performing results ae summarized below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class weight | Alpha | Penalty | Loss | Features | F1-score (hold-out) |
| None | 4.6415888336e-05 | L1 | Log | Unigrams and Bigrams | 0.996 |
| balanced | 4.6415888336e-05 | L1 | Log | Unigrams and Bigrams | 0.75 |
| Balanced | 1e-7 | L2 | Modified huber | Bigrams only | 0.77 |
| Balanced | 0.0021544347 | L2 | hinge | Bigrams only | 0.70 |
| None | 1e-6 | L2 | Log | Unigrams and Bigrams | 0.85 |

The actual model is then fit on the entire labeled dataset with the final set of parameters. This fit model is then used to predict the sentiment of tweets in the 6 different test datasets. Since different users have different average tweet rates, the sentiments of tweets are grouped by their users. For each user in a date, the number of tweets counted as AC and PC are counted and if they have more tweets supporting Clinton, the user is classified as PC, otherwise AC. For a small fraction of users, the number of tweets supporting and opposing Clinton were equal and hence they were classified as undecided. This process was repeated for all 6 timepoints and the results from the model is compared with the polling results below.





***Fig 3, 4****: Polling estimations of support for trump vs Clinton and the model results for the same dates. Percentage supporting 3rd party candidates (in fig 3) and percentage undecided (fig 3 and 4) have been left out*

From the results above, we see that while the results of the analysis do not follow the polls, it does a better job of predicting opinion on election day as Trump did end up winning the presidency (He only won through the electoral college and not the actual vote share, but the actual vote share of Trump was still much higher than most polling estimates). The polling estimates were taken from the HuffPost Polling API which had an overall polling estimate from all major polls conducted around the respective days [8]. During the republican the online chatter was oriented more towards to the republican party and more tweets against Clinton were being sent and hence Trump had a slightly higher percentage of users supporting him. Similarly, during the democratic convention, Clinton has a comfortable lead over Trump due to the positive wave following the convention and the choosing of Clinton as the Democratic nominee. The 3 presidential debates, where Clinton arguably did much better than Trump see Clinton in the lead. However, on the day before the election, we see that Trump has a small lead over Clinton. This might be because of the last-minute investigations into Clinton’s emails but whatever the reason, it is indicative of the voter sentiment at the ballots as well.

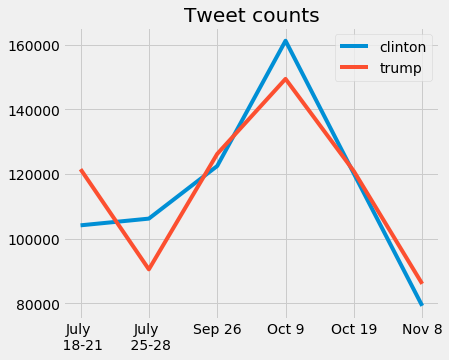
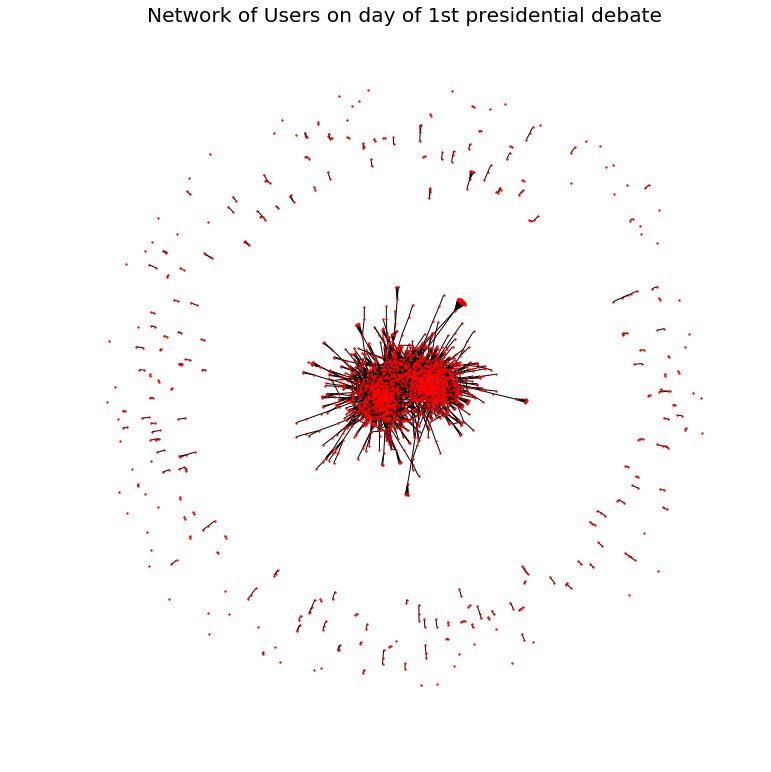
Here we also see the number of tweets that were classified as supporting Trump or Clinton. Unlike user percentage it seems to be mirroring each other’s trends more closely. The 2nd debate had the greatest number of relevant tweets. Only tweets in the day before the election were used to mirror final polls and, in this case, most tweets were only about voting and not particularly about the candidates, so after filtering the noninformative tweets, the final set was used.

Figure 5: Tweet counts, Fig 6 (below) User networks

Finally, we attempted to also observe the User Networks and the way users communicate with each other regarding the elections. However, it couldn’t be done by simple web scraping and we had to use Twarc library[10] to use twitter API and access tweets by id. And then find ids of users who followed or replied to the tweets. In this case, twitter’s rate limits[11] were very severe (75 hits per 15 minutes) and it took an extremely long amount of time[>12 hours] to get just the nodes and edges information. Hence, we have only created the network for 1 of the 6 timepoints, the 1st presidential debate. users that didn’t communicate with any other users were removed. From the network we see that a lot of users communicate to 2 or three other users (possibly family or friends) while many politically active users form the corona an communicate widely to a lot of other members. We also see that there is a slight separation in the corona that could represent the different camps of political parties.

**5. Conclusion**

In conclusion, this project is an exploration of the methods that can be used to incorporate modern sentiment and natural language processing into the process of polling to better understand the electorate and to better predict elections. In today’s world, most leaders are accused of being out-of-touch with the country and the electoral process and the way people engage with it has radically changed. These changing times call for better techniques for polling as well to ensure that the political parties move with the times. Due to the large-scale nature of the data, many more techniques can be used and compared along with the ones I’ve performed here. A few techniques I would try in the future include more network analysis including label propagation to ascertain user orientation from the political orientations of their twitter friends and followers. I will also try to do some more data collection to more time points and better filter of the tweets. Finally, I would like to say that this project would not have been possible without that very supportive and encouraging guidance of Prof. Qu, the TA Yujia Deng and my classmates. I would like to that you for the opportunity to undertake this project in the field I am most interested in and learn in the process. Please feel free to give me any feedback and ideas for further work in this area.

**6. References**

[1] O’Connor, B., Balasubramanyan, R., Routledge, B. R. & Smith, N. a. From tweets to polls: Linking text sentiment to public opinion time series. 122–129, DOI:citeulike-article-id:7044833 (2010)

[2] Tumasjan A, Sprenger TO, Sandner PG, Welpe IM. Election forecasts with Twitter: how 140 characters reflect the political landscape. Soc. Sci. Comput. Rev. 2011;29:402–418. doi: 10.1177/0894439310386557.

[3] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

[4] Gonzalez-Bailon S, Paltoglou G. Signals of public opinion in online communication: a comparison of methods and data sources. Ann. Am. Acad. Pol. Soc. Sci. 2015;659:95–107. doi: 10.1177/0002716215569192.

[5] Sylwester K, Purver M. Twitter language use reflects psychological differences between democrats and republicans. PLoS One. 2015;10:1–18. doi: 10.1371/journal.pone.0137422.

[6] Bovet A, Morone F, Makse HA. Validation of Twitter opinion trends with national polling aggregates: Hillary Clinton vs Donald Trump. Sci Rep. 2018;8(1):8673. Published 2018 Jun 6. doi:10.1038/s41598-018-26951-y

[7] Leon Bottou and Frank E. Curtis and Jorge Nocedal Optimization Methods for Large-Scale Machine Learning, Technical Report, arXiv:1606.04838

[8] <https://app.swaggerhub.com/apis/huffpostdata/pollster-api/2.0.0>

[9] <https://elections.huffingtonpost.com/pollster/2016-general-election-trump-vs-clinton>

[10] <https://github.com/DocNow/twarc>

[11] Twitter Documentation. <https://developer.twitter.com/en/docs>

[12] data: https://drive.google.com/open?id=1jBr5L3GJ88bTE\_GYXVi06GuFqqgzChvf