Study of Political Abuse on Twitter

Divya Rajendran
Data Science
School of Informatics, Computing, and Engineering
Bloomington, Indiana
divrajen@iu.edu

Pramod Duvvuri
Data Science
School of Informatics, Computing, and Engineering
Bloomington, Indiana
vduvvuri@iu.edu

Samanavitha Pradhan
Data Science
School of Informatics, Computing, and Engineering
Bloomington, Indiana
sspradhan@iu.edu

ABSTRACT

This paper will describe the authors efforts to study the activity on social media platforms such as twitter during 2016 Presidential Election in the United States. To analyze the tweets by the candidates, perform sentiment analysis, identify if the polarity of either Trump's or Clinton's tweets lead to their win/loss in the elections.

KEYWORDS

Social Media Mining, Twitter, Sentiment Analysis, Machine Learning, Regression, Linear Model, Feature Extraction, Natural Language Processing, Topic Modeling

ACM Reference Format:
Divya Rajendran, Pramod Duvvuri, and Samanavitha Pradhan. 2018. Study of Political Abuse on Twitter. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

INTRODUCTION:

Social Media became an increasingly popular platform to share information to reach a wider audience. With social media holding this immense power of information sharing, one tends to be obligated to get connected to a network of individuals to keep a track of things happening around them.

While the concept of social media started as an innocent and novel idea of keeping in touch with your friends and family, keeping track with the real time events happening around you, lately social media is being abused with a lot of targeted information which might or might not be true. Can this abuse of social media information affect the outcome of an electoral result?

In this project, we would like to study the political aspect of abuse on social media, Twitter. We specifically want to focus on political abuse, specially during the election period - immediately before, during and after an election period. We want to identify if an abusive tweet has gained greater support which in turn led to the electoral win of a candidate.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

LITERATURE REVIEW:

We would like to talk about the effect of political abuse on political outcomes. What is Abuse? According to Merriam-WebsterâĂŹs dictionary [1], Abuse is defined as an improper use of an entity often for an improper benefit. There are various kinds of abuse like physical, mental, verbal, psychological, emotional, sexual, financial, cultural or identity abuse [2]. In this project, we focus on political abuse, which can be again divided into the below:

1. Actual offensive tweets which tend to malign some other individual, thus abusing the freedom of expression via verbal abuse and,
2. Sharing information which leads to unnecessary provocation and thus abusing the power of information sharing.

Here we worked on Twitter data to effectively gather information on tweets immediately prior, during and after election period. This period is the best time frame where the actual information via tweet sharing happens, which can be used to study the effect of such tweets on electoral outcomes.

Social media like Twitter tends to share information in limited character lengths. Currently the limit has been increased to double its previous limit, i.e., from 140 to 280-character limit. The politicians or the politically motivated express their opinions on their own political parties or rave and rant about their opposition parties, more often belittling individuals, either the political candidates or their supporters. During this process information sharing or raving and ranting, some tend to abuse this immense power to their unjust gains like maligning a fellow social media user or a fellow human.

In this project we would like to find out how politicians abuse the social media to their benefit, through provocation rather than through information sharing. The main idea to study the amount of abuse on social media during a political campaign is very popular these days given the results of the 2016 Presidential elections in the USA.

The paper on *Detecting and Tracking Political Abuse on Social Media by J. Ratkiewicz, M. D. Conover, M. Meiss, B. Goncalves, A. Flammini, F. Menczer* [3], our own researchers at Indiana University, provides an excellent introduction to this idea where

the authors use the data leading up to the 2010 Midterm elections in USA. The authors discuss on how when given the vast number of users that are on social media these days, people find ways to exploit it. The authors discuss a specific type of abuse called *astroturf* which means a substantial number of users are paid to post or say good things about a candidate and how such type of spamming activities has large consequences. The authors go on to discuss about the detrimental effects such activities have on elections. The authors propose a way to detect such orchestrated information and how it spreads on the social media platform âĂIJTwitterâĂİ.

In the paper *Predicting the 2011 Dutch Senate Election Results with Twitter by Sang E and Bose J* [4] proposes an approach to use twitter data to predict election results. They used it as an alternative to the traditional opinion-based polls. The paper also discusses about how the election data was collected and worked on to implement a simple concept of counting the number of tweets with a positive connotation by employing sentiment analysis and predicting if a political candidate would win the election. Our idea was similar to this paper, but we were studying the effect of abuse of information sharing on the election results.

The paper *Predicting elections with twitter: What 140 characters reveal about political sentiment by Tumasjan A, Sprenger T, Sandner P, Welpe I* [5], gives us an insight into how the actual political statements reflect on twitter and the ripple effect it carries among the group of a political candidateâĂŹs followers. This paper gives us a much-needed understanding on how a fake news will likely affect the public sentiment on a political party.

DATA SET DESCRIPTION:

Twitter is a type of micro blogging social media where a lot of registered users freely share the opinions and sentiments via tweets in real time on a varied set of topics like politics, movies, environment, games and so on. Earlier the character length of a tweet was limited to 140 characters and now increased to 280 characters in September 2017, giving a freedom of using more words.

A tweet is a message, or a short post written by the Twitter user expressing ideas or quoting someone. Every tweet has a set of attributes like the username, location, gps coordinates, tweet text, retweeted, retweeted count, hastag, video url, meme tags, date and time of the tweet, replies, reply counts, loved counts and so on. We would like to use these all the attributes available for a tweet and divide the obtained tweets into a set of relevant and irrelevant tweets, from which we work on the relevant tweets which are more politically aligned.

User Tweets Collection

We extracted tweets from Twitter corresponding to political candidates during the election period through the official Twitter API. We collected individual tweets from users, *HillaryClinton* and *realDonaldTrump* username's for each of the days starting Apr-2015 to 31-Dec-2017 to get all the tweets posted by Hillary Clinton and Donald Trump. We used a code or method named *GetOldTweets-python developed by Jefferson-Henrique* [21] as a base to get all the old tweets for the usernames. We modified this code to include the replies count as well to know the reach of a user's tweet among twitter users. Since Twitter did not allow us to extract all the tweets at the same time, we had to extract tweets of each of the users a month a time and it took a week for us to gather all the tweets without exceeding the tweet limit set by twitter.

Poll Data Collection

We searched for an opinion poll data for the period of elections for the year 2015-2016 and 2016-2017 and we have found an existing poll data set from the data sets from Kaggle *2016 Election Polls Data Set* [22]. This data set had opinion polls for every month from users all over U.S, it is divided into state wise poll rating, as well. We wanted an overall opinion poll rating, so we considered the data for U.S as a whole directly. The poll rating for Trump and Clinton are shown in the attributes, *adjpoll_trump and adjpoll_clinton*. we have aggregated these values based on month, start and end date of the poll, so that we had a mean value of poll % rating for both Trump and Clinton for each month starting from January to December, based on which our analysis is done.

APPROACH

Data Pre-processing

The main step before we fed our data to a predictive model is to clean it. The data mined from social media would always be messy, people tend to use social media platform to express their emotions, opinions and given that you can only send 140 character tweets on twitter means people use short-forms. We aim to remove all such hyperlink, emojis from the tweets. We wrote a function *preprocess_input* which would take our tweet text and remove numbers, special characters from each of the words in a tweet text, tokenize it with the help of *nltk* library in python and later combine each of the words of length greater than 2 into the text back again.

Afinn Score for Tweets

We calculated Afinn score for each of the tweet text we obtained for both the users, which we aggregated according to each month of every tweet, so that we had an aggregated total afinn score for tweets per month. Since we did not have enough poll data, we considered the available data from the month January 2016 to December 2016. We plotted the Afinn Score and retweets per month, Afinn score and poll ratings and poll ratings and retweets per month to analyse the relationship between each of them.

The plots in the **Jupyter Notebook** show the relationship between each of the combinations of attributes.

Topic Modelling Data

The authors have manually labeled over 700 tweets for multi-class classification in the final step of the project. The above mentioned 700 tweets have been classified into 11 categories/labels.

Label
Advice
Campaign
Democrats
Fake News
Fox News
Golf
Hillary
Investigation
Obama
Real Estate
Trump Attack

Predictive Modeling

We have used Python various packages like *AFINN, Scikit Learn, TextBlob, linear model selection, Logistic Regression, train test split, accuracy score, svc, nltk, wordcloud* for performing various tasks in our project.

Sentiment Analysis: For the task of sentiment analysis we have used the *TextBlob* package. After we have cleaned each tweet it is then classified as *Positive, Neutral, Negative* basing on the polarity score between {-1 to +1}. The textblob method also provides additional information such as subjectivity, subjectivity assessment about the input text.

Transformations

The input text needs to be transformed before it can be used for training and testing, this is because most machine learning algorithms expect the input data to be in a numerical format. Hence, we need to transform our raw input text. This is done by using a *text feature extraction* method called *"The Bag of Words"* representation. We use *sklearn.CountVectorizer* implementation of the bag of words model in our project. Most input text are not of the same fixed length, it would be difficult for an algorithm to handle this variable length problem, the bag of words representation solves this issue for us.

Working of Bag of Words.

- **Tokenizing** the input text is converted into tokens by using punctuation's and white-spaces as the delimiters.
- **Counting** the occurrences of each token is counted in each line of text
- **Normalizing** and weighting with diminishing importance tokens that occur in the each line.

A corpus of the text can thus be represented by a matrix with one row per line and one column per token (e.g. word) occurring in the corpus.

We call vectorization the general process of turning a collection of text into numerical feature vectors. This specific strategy (tokenization, counting and normalization) is called the Bag of Words or *Bag of n-grams* representation. The matrix represents the word occurrences while completely ignoring the relative position information of the words in the line of text.

We typically use a small subset of the words in the corpus, the resultant output matrix will have a lot of zeroes. This output matrix is a sparse matrix of type *scipy.sparse.csr_matrix*. This sparse matrix is used for faster algebraic operations.

Another method we employ to normalize the tokens is called *Term Frequency Inverse Document Frequency - (TFIDF)* . In this method, the number of occurrences of a word is multiplied by it's idf component, so that words with lesser significance such as the articles "a", "an", "the" do not shadow the frequencies of some rarer and more interesting words. This is achieved by a tf-idf transform. The below equations give a brief overview of math involved the TFIDF transformation

$$tf - idf(t, l) = tf(t, l) * idf(l)$$
$$idf(l) = log(\frac{1+n}{1+df(l,t}})$$

Where n is equal to the number of lines of text and $df(l, t)$ is the number of lines that contain the term t. The resultant tf-idf vector's are normalized using the euclidean norm. We use the *sklearnfeature_extraction.text.TfidfTransformer* implementation to transform our text for suitable usage by a classifier. We also tried multiple transformations on the input text data. That is we implemented the Bag of Words and TFIDF using a *Pipeline* in our project. We have used the *sklearn.pipeline.Pipeline* implementation to combine the above transformation on our data.

Latent Dirichlet Allocation: We also perform Latent Dirichlet Allocation using scikit-learn implementation of it on the tweets of Donal Trump to find words frequently used by him.

Natural Language Processing. Natural language processing (NLP)is a field of computer science, artificial intelligence concerned with the interactions between computers and human languages, and, in particular, concerned with programming computers to fruitfully process large natural language data. It is very common to perform NLP before we train our predictive model. We use the built in NLTK [3] for our preprocessing [5] to remove some commonly used words as mentioned in TFIDF process, these common words are usually referred to as *stop words*. Another common method as a part of NLP preprocessing is to perform *stemming* . Stemming refers to the process of reducing derived words. Example: "running" is reduced to "run".

After we have transformed our input text, we have to choose an appropriate machine learning algorithm for our model. There are many classifiers available for us in machine learning. The next section of this document discusses the various methods we have used to predict the label using the tweet.

Methods

The below flow diagram best describes the process we have adopted in this project.

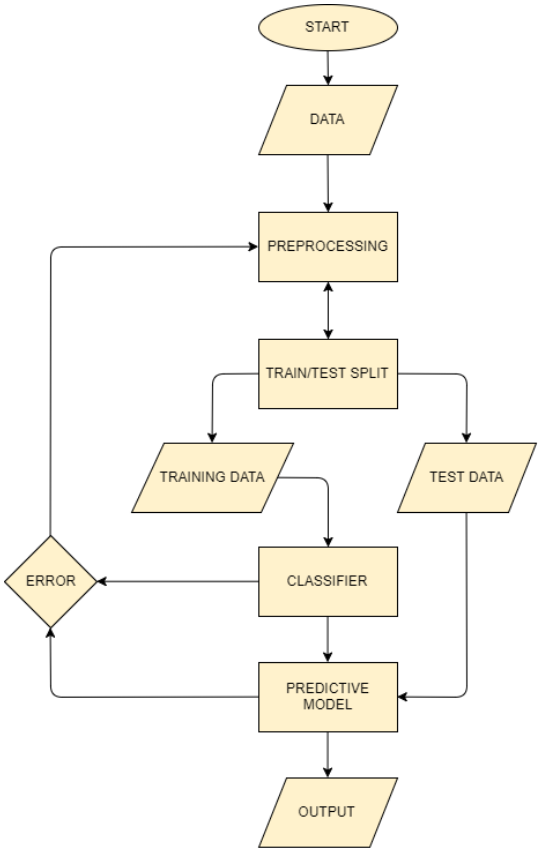

Figure 1: Process Flow

These are the various predictive machine learning models we have used for the task of or prediction after we have transformed the data.

Multinomial Naive Bayes. Multinomial Naive Bayes (MNB) [9] is a simple classifier that uses the Bayes Theorem and assumes independence between the features and is used for multinomially distributed data, since the data is represented in sparse matrix of word vectors this would be ideal for such scenarios. We shall use *sklearn.naive_bayes.MultinomialNB* implementation in our project. We have used various train/test splits to analyze the accuracy of our model. The same has been represented in terms of error in the plot below.

Support Vector Machines (SVM). Support Vector Machines [13] are predictive models in machine learning that are used for *Supervised Learning*. When the output variable is labelled it is called supervised learning, in our scenario we are the topic name which is the label. SVMs are highly efficient in performing linear classification. SVMs are also very helpful in text classification, hence we have chosen to use them in our project.

An SVM uses a hyperplane or a set of hyperplanes to perform the task of classification. We have used two different implementations of SVMs in our project, *sklearn.svm.SVC* and *sklearn.linear_model.SGDClassifier* for topic prediction. We shall also implement SVMs with Stochastic Gradient Boosting in our project, this method is discussed in the next sub-section.

Stochastic Gradient Boosting. Stochastic Gradient Descent (SGD) [14] is a simple technique for using linear classifiers under convex loss functions such as (linear) SVMs and Logistic Regression. The efficient application of SGD in large-scale and sparse machine learning problems in recent times has made it popular. SGD has been widely used in NLP and text classification. This is reason why we have chosen SGD for this text classificatauthion problem. Also SGD is very easy to implement. For the parameters we use *elasticnet* for the penalty as it might bring sparsity to the model. We also use 100 as the value for the parameter max_iter.

Logistic Regression. Logistic Regression [12] is a machine learning algorithm that is used for regression and classification, it uses the natural logarithm function to find the relationship between the independent variable and the target variable, it uses test data to find the coefficients. The function can then predict the future results using these coefficients in the logistic equation. We use the *sklearn.linear_model.LogisticRegression* [12] implementation in our project.

Random Forest. Decision Trees are versatile Machine Learning algorithms that can perform both classification and regression tasks, and even multi-output tasks. They are very powerful algorithms, capable of fitting complex datasets. A Decision Tree is the fundamental concept behind a *Random Forest* [15].

In general, aggregation of answers tends to give a better result, same technique used in machine learning is called ensemble learning. Instead of using a single classifier we use a group of classifiers or predictors, these group of classifiers usually tend to give a better prediction. When a group of decision trees are trained on random subset of the training data we get better results. Such groups of decision trees or ensemble of decision trees are known as a "Random Forest". We use *sklearn.ensemble.RandomForestClassifier* [15] implementation in our project.

Bagging. We can always use a group of diverse classifiers in our predictive model. Another approach would be to just use one training algorithm and train it on different random subsets of the training dataset. When such different samples of the training data is used with replacement to train our model the process is known as *Bagging* [16]. The bagging process is very popular and frequently used in ensemble learning.

Voting Classifiers. This is another application of ensemble learning. The outputs of individual classifiers are aggregated and the class which gets the most votes is the prediction. When it is a majority vote, it is known as a *hard voting*. In most scenarios, the voting classifier always achieves a higher accuracy. We use *sklearn.ensemble.VotingClassifier* [17] implementation in our project.

Among the different voting classifiers we have used, Voting Classifier 3 has given us the least error rate, so we use the Voting Classifier 3 and perform cross validation to check the foldwise error rate and average error rate. The results of

the cross validation shall be discussed in the **Results** section. BaggedDecisionTree error rate has been plotted to compare it with the group of classifiers in Voting Classifier 3. The group of classifiers in Voting Classifier 3 are : **LinearRegression Classifier, SGD Classifier and MultinomialNaiveBayes Classifier**.

RESULTS & FUTURE WORK

We plotted the Afinn Score and retweets per month, Afinn score and poll ratings and poll ratings and retweets per month to analyse the relationship between each of them. The plots in the **Jupyter Notebook** show the relationship between each of the combinations of attributes. We understood from the plots that Trump's opinion ratings actually decreased during the election period whereas Clinton's increased. We could not determine a relation between the afinn score of each tweet with the number of re-tweets received by a tweet. We found a definitive relation between the poll rating and the tweet replies, the more the replies a user got the better poll rating is observed, implying a direct relationship between them.

For Topic Modeling the algorithm which has consistently given good scores on different train/test splits and random subsets is the *Support Vector Machine (SVM)*, it gave us an accuracy of 70-75 % initially. So, a significant amount of time was spent to tune the parameters and pre-process the input text data to obtain a higher accuracy score. After the preprocessing had been done we obtained an accuracy score of over 85% for the SVM which we believe was it's best. The SVM wtih SGD gave us the best accuracy scores in the range of 85-90%. The below plot shows the error rates with and without performing cross-validation with the error rate on the Y-axis:

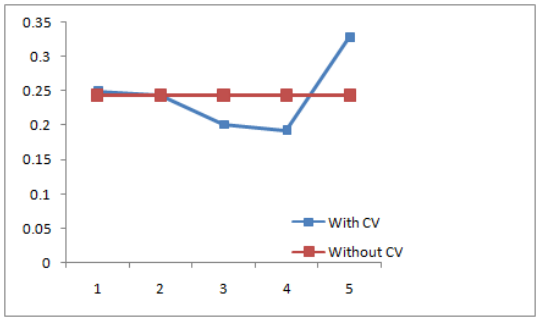

Figure 2: Process Flow

The plots in the **Jupyter Notebook** give the accuracy rates of different predictive models. We hope this study will serve as a basepoint for further research in the area of social media and its effects on elections. There are always newer hypothesis to be verified or refuted as the data in social media is growing exponentially.

ACKNOWLEDGMENTS

The authors would like to thank Professor Vincent Malic for his valuable suggestions and helpful comments. The authors would also like to thank the Python open source developer community for supporting and sharing the packages used in this project.

CONTRIBUTIONS

- Divya Rajendran: Contributed to the central idea behind our study, selection of token words, model selections, main concepts, data to be mined, time frames needed for data extraction, the programs to be used with suggestions on the data to be mined. Wrote the code for processing and making predictions using the poll data. Mined 15000 tweets from both the candidates prior 2016 election.
- Pramod Duvvuri: Modified and executed the existing code to suit our needs, extracted the data. He found the issue with obtaining data for a month at once, suggested and extracted 10000 tweets per one run for every three days before and after a major date (like nomination date). Wrote all code for the performing LDA, fitting different classifiers.
- Samanvitha Pradhan: Manually labeled 700 tweets and contributed towards the final report. Wrote code for sentiment analysis and generated plots for visualizing end results.

REFERENCES

[1] https://reachma.org/6-different-types-abuse/
[2] Natural Language Processing: https://en.wikipedia.org/wiki/Natural_language_processing
[3] Natural Language Toolkit: http://www.nltk.org/
[4] Count Vectorizer: http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
[5] Preprocessing using NLP: https://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html
[6] Scikit Learn Documentation: http://scikit-learn.org/stable/documentation.html
[7] Pipeline: http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
[8] Naive Bayes: https://en.wikipedia.org/wiki/Naive_Bayes_classifier
[9] Multinomial Naive Bayes: http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
[10] NumPy Documentation: http://www.numpy.org/
[11] Pandas Documentation: https://pandas.pydata.org/
[12] Logistic Regression: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[13] Support Vector Machines (SVM): http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[14] SVM with Stochastic Gradient Descent (SGD): http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
[15] Random Forest Classifier: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[16] Decision Tree with Bagging: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html
[17] Voting Classifier: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html
[18] Author Identification using Deep Learning: https://web.stanford.edu/class/cs224n/reports/2760185.pdf

[19] Plotly Documentation
https://plot.ly/python/
[20] The Book
Hands-On Machine Learning with Scikit-Learn and TensorFlow

[21] GetOldTweets-python
https://github.com/Jefferson-Henrique/GetOldTweets-python
[22] 2016 Election Polls Data Set
https://www.kaggle.com/fivethirtyeight/2016-election-polls/data