2016 United States Presidential Election Tweet Classification

A Preprint

Amine Bellahsen
Department of Applied Mathematics
Polytechnique Montréal
Montréal, CA
amine.bellahsen@polymtl.ca

Hamza Sghir
Department of Industrial Engineering
Polytechnique Montréal
Montréal, CA
hamza.sghir@polymtl.ca

Abderrahim Khalifa
Department of Computer Science
Polytechnique Montréal
Montréal, CA
abderrahim.khalifa@polymtl.ca

November 12, 2019

Abstract

With the constant growth of social networks in recent years and their ever-easier access, many People are turning to these platforms when it comes to discussing various topics and events such as politics. That's why analyzing trends on these platforms is becoming more of a better alternative than the classic polls that have long existed. The idea of this project is to rely on the social network Twitter and a set of tweets, which we know in relation to the 2016 US election, to determine if they are in touch with the Democratic or Republican candidate. Then from a sufficiently representative set of these tweets produces the day of the election, to be able to predict the results by determining which candidate is the most "trend", that is to say the most talked about and also the expressed sentiments towards the two parties.

1 Introduction

Nowadays, on the Internet, many sources generate huge amounts of daily information. In addition, as many emerging controversial subjects like politics and human bias and even the Russian interference in the 2016 United States presidential election have created a lot of debates last years, it is important to understand how it works and try to apply machine learning tools on this subject. For that matter, we will use tweets related to the 2016 United States presidential election in order to extract sentimental features along with classifying, depending on the state, if it is more concerning the democratic party or the republican one.

To do this our approach will be divided to 3 main parts, which will be presented through this report:

- Dataset Retrieval: Constructing the Dataset from an Existing Dataset and the Twitter API
- Text processing and training of the models. Hyperparameter tuning is done by cross validation, and we train 4 classification different algorithms: Logistic regression, Support vector machine, Naîve Bayes, Random forest.
- Apply the best model on the tweets written during the election day. Then a study of the results is performed.

2 Data set

In order to perform this study, we use the data set "2016 US Presidential Election Tweet Ids", from the "Harvard Dataverse" research data directory. This dataset brings together nearly 280 million of tweets related to the 2016 US elections and was collected between July 13, 2016 and November 10, 2016. These tweet ids are broken up into 12 collections composed into tweets of democratic parties and republican parties, along with tweets during the three presidential debates and finally during the election day.

We took from the whole collections 7 of them as follows:democratic-candidate-timelines.txt, democratic-convention-filter.txt, democratic-party-timelines.txt, republican-candidate-timelines.txt, republican-convention-filter.txt, republican-party-timelines.txt, election-day.txt.

Since we already know to which party the first six collections belong, the problem of building statistical models becomes a supervised learning problem, a classification task where the aim is to use tweets as input to predict whether it comes from the democratic or republican part. For that matter, we establish labeling by giving the value 0 to the tweets concerning the Republican party, and the value 1 to the tweets concerning Democratic party.

The next step will be to use the last data set (election-day), in order to make statistical inference about the tweets that were made during the election day.

To make the data acquisition, we need to use our twitter developer status and the ids collected from the collections in order to retrieve customized in formations about each tweets. The features we choose for the first 6 collections are tweet's id & tweet's text. For the last collection, we choose tweet's id & tweet's text & user's location & user's screen name

3 Preprocessing

The stage of preprocessing is a crucial step for the development of a fairly powerful model. Here we propose some treatments that we will perform on tweets to reduce the vocabulary and remove the tokens that are not used in our problematic. The pipeline presented in Figure 1 was employed to clean the text for each tweet. The url on twitter are all reduced to a link t.co which do not generally give any idea about the content of the tweet. We replace all the url in the tweet by the token. All punctuation marks were stripped, as it has been suggested by our team that they couldn't be useful in our classification. Next, all the emoji has been classified into two categories: positive and negative. This will give an idea of the type of emotion expressed in the tweet. Subsequently, All Hashtages and TAGs were stripped as they not relevant for our classification. A number as well does not allow in a model BoW to know what the nature of the tweet is unless it is put in a context specifying the unit for example. We replace all the numbers in the tweet by the token. Lastly, all tweet were lemmatized. Stemming reduces the word to its stem, using predefined rules which vary depending upon the stemming algorithm used. Stemming may also result in removal of derivational morpheme.

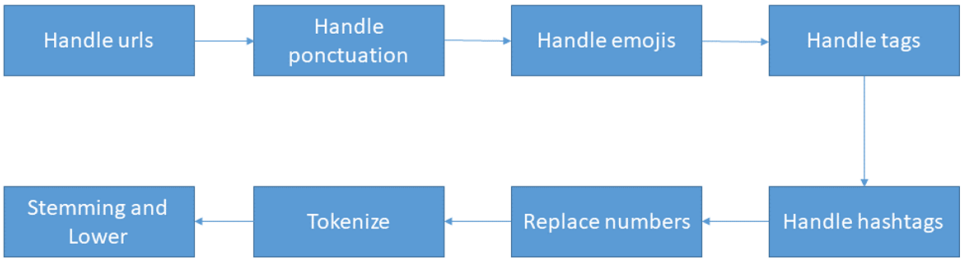

Figure 1: Preprocess Pipline

4 Methodology

In any text mining task, and before building statistical models, it is crucial to extract good & useful features of the provided data. The first part of this section is about extracting those features; the second part is about the built statistical models. The final part is about inferring using the chosen best model and performing sentiment analysis about the written tweets during the 2016 election day.

4.1 Feature Extraction

Logistic regressions, SVM and other very common classification models require entries that are all the same size, which is not necessarily the case for data types such as texts, which may have a variable number of words. For handling that we're going to use two important methods:

- **CountVectorizer** : It is a representation of comments by vectors whose size is equal to the size of the vocabulary, and which is constructed by counting the number of occurrences of each word. Thus, each token is here associated with a dimension.
- **TF-IDF** : The use of the frequency of gross word appearance, as is the case with CountVectorizer, can be problematic. Indeed, few tokens will have a very high frequency in a comment, and because of this, the weight of these words will be much larger than the others, which will tend to bias all the weights. Moreover, the words that appear in most documents do not help to discriminate them. TF-IDF is a method to overcome this problem. It weights the vector using an inverse document frequency (IDF) and a frequency of terms (TF) [1].

4.2 Model implementation

Several models were implemented from Scikit-learn as a machine learning library for the Python programming language. It turns out that since the tweets were retrieved using the same number of requests, the generated data sets were balanced; that means that the accuracy metric can be a good choice. In fact, the performance metric we are going to use is:

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative}$$

4.2.1 Naive Bayes Classifier

A Naive Bayes Classifier determines the probability that an example belongs to some class, calculating the probability that an event will occur given that some input event has occurred.

We used a Multinomial distribution for Naive Bayes from sci-kit learn[source] with a smoothing parameter α. The smoothing parameter redistributes the weight of features, by transferring some of the weight to the features which have zero weight. $\alpha = 0$ is no smoothing, $\alpha = 1$ is Laplace smoothing and for any other α, it is called Lidstone smoothing [2].

4.2.2 Logistic Regression Classifier

Logistic Regression outputs predictions about test data points on a binary scale, zero or one. If the value of something is 0.5 or above, it is classified as belonging to class 1, while below 0.5 if is classified as belonging to 0.

Each of the features also has a label of only 0 or 1. Logistic regression is a linear classifier and therefore used when there is some sort of linear relationship between the data [3].

4.2.3 Support vector machine Classifier

Support vector machine is a representation of the training data as points in space separated into categories by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall [4]. There are three main parameters which we played with when constructing a SVM classifier: Type of kernel, Gamma value, C value.

4.2.4 Random Forest Classifier

Random forest is an ensemble model that grows multiple tree and classify objects based on the "votes" of all the trees. An object is assigned to a class that has most votes from all the trees [5].

The hyperparameters are either used to increase the predictive power of the model or to make the model faster. we used different values for different hyperparameters such as : number of estimators, maximum number of features, bootstrap and criterion.

4.3 Hyperparameter tunning

To select the best model, we first chose optimum parameters for Logistic Regression, Random Forest, SVM and Naive Bayes. This was done using 5-Fold Cross Validation and choosing the best value based upon the Accuracy as the right classification metric for balanced data. Hyperparameter tuning was executed using Scikit-learn's GridSearchCV [6], which exhaustively considers all parameter combinations to find the hyperparameter space that achieves the best cross validation score.

On the other hand, in order to make the vectorizer, transformer and classifier easier to work with, scikit-learn provides a Pipeline class that behaves like a compound classifier. The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters.

4.4 Sentiment analysis

Another layer was added to the methodology for sentiment analysis. We build a model using airlines tweets in order to score the feeling of twitter users about election tweets. The same pre-processing from the previous project was performed for the 2016 election day database.

- The first step was to use the best selected classification model in order to predict whether a tweet is concerning the democratic party or the republican party.
- We then extracted the state from which the tweet was sent based on its location.
- For each of the concerned parties, we used the sentiment analysis model to evaluate the sentiment regarding the tweet.
- Finally, we average the sentiment score for each party, grouped by state.

5 Results

After splitting the database into a training set and a testing set, we performed a 5-fold cross validation on the training set along with hyper-parameter search, then evaluated the 4 models and have chosen the one which gives the best validation score. The SVM was selected with a validation accuracy $= 90.3\%$. Now testing the selected model on the test set gave us an accuracy of 96.07% which validates our choice.

This model was used to flag each tweet by one of the two political parties.

Next, we used the sentiment analysis pipeline in order to determine the expressed feelings in the tweet toward the corresponding party. For each tweet we extracted the state from which it was sent based on the user's location. For each US State, we averaged the sentiment score and calculated the portion of tweets belonging to each party.

The following two plots show the results obtained using our methodology.

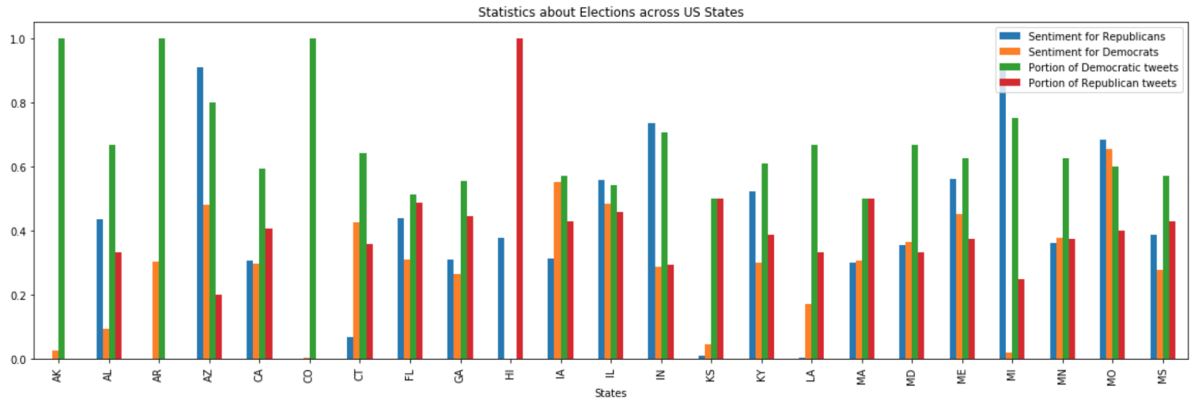

Figure 2: Statistics about 2016 elections across US states

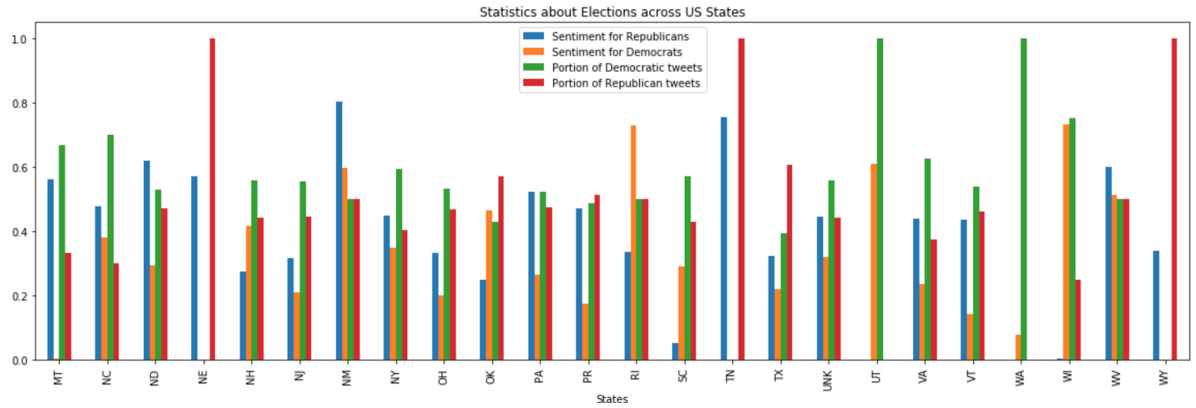

Figure 3: Statistics about 2016 elections across US states

We notice that in some states, our classification model flagged all the tweets for one of the two political parties. At first sight, we may conclude that the political party will be majority in that state. However, the sentiment analysis process brings us more insights for the analysis. As a matter of fact, all Alaska's (AK) tweets were predicted to belong to the democratic party, but the average sentiment score for these tweets was significantly low (negative). Another example is Wyoming (WY) state, which shows that all its tweets belong to the republican party, and their average sentiment score was significantly low. It is then difficult to make direct assumptions about whether a state will be democratic or republican. However, for the majority of the states, we got tweets for both parties. Combining the sentiment score of both parties within a state along with the proportions of tweets for each party, we can come up better conclusions: Rhode Island (RI) can show that there is nearly the same proportion of tweets for both parties; by weighting these proportions with the correspondent sentiment score, we can leverage the democratic party.

The location data extracted from twitter was not always reliable and doesn't point to any valid location. Therefore, all tweets with invalid locations were assigned to the same cluster state: UNK.

6 Discussion

Broadly speaking, one might infer that a political party will win if it's most talked about. Adding other estimates can help us find more insights about those parties. Sentiment analysis can be a good start, but that might have some shortcomings with bad conclusions. Further work might concern extracting keywords that may have a big influence on the two parties.

Furthermore, one of things we might also approach regarding to this methodology is if the tweets are significantly representative to real world surveys. Does a randomly sampled population of tweets really gives unbiased estimates for the future of presidential elections ?

References

[1] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. Piscataway, NJ, 2003.

[2] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[3] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848, 2002.

[4] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[5] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.