Predict A Voter's Political Orientation Using Their Tweets

Pranav S **Meghana K.** **Sravya K.** **Srikar B.** **Anusha P**
Georgia State University
{pnadigapusuresh1,mkodali1,lkambhampati1,sbalmuri1,apalisetty1 }@gsu.edu

Abstract

Election polls, like any other polls, is an opinion survey designed to indicate the opinion of the population. Social media platforms like Twitter contain a multitude of data that can be used to analyze the trend. These platforms contain large amounts irrelevant data, so the data we use for our polls needs to be selected carefully. In this project, we used NLP and statistical methods to predict if a tweet is political or non-political, and if political, the political party orientation of the tweet. Our experiments show that while the the general pre-processing techniques used on the text corpus are useful, additional pre-processing done on the tweets increased the accuracy of the predictions.

1 Introduction

Election polls, like any other survey, is an opinion survey designed to indicate the opinion of the population of the election [1]. Traditionally these polls are conducted through telecommunication or online survey. The problems these traditional polling methods became apparent during the U.S. 2016 general election where the polls of most national media has shown that Hillary would win the primaries [2]. One disadvantage that the traditional polls pose is that a voter is aware that the responses are recorded and could be biased [1]. Another reason is that the traditional polls have included higher number of retired population ignoring the millennial. In order to select appropriate sample of the voting population we can use the social media data in addition to the traditional polls.

Social media platforms like Twitter,Facebook, and Reddit has opened a new portal of communication and interaction. All the users on these platforms have freedom to express their opinions on anything adhering to the rules of the platforms. As a by-product of this freedom there is a vast amount of data being generated every minute. The abundance of data and its unmediated access has attracted the interest of the markets.

In this project, we have used the publicly available tweets and experimented different text processing methods to transform the corpus into data that can be consumed by different machine learning methods. We have tried 5 different methods and compared their accuracy with different text pre-processing which has shown consistent results for all the methods used.

2 Data

The Raw data is extracted using the public twitter API. We collected 52,000 tweets from various accounts with distribution show in Figure 1. For the democrats the data is collected from the official democrats account, from Joe Biden's account, from Bernie Sanders account and few other officials who represent the Democrats party. For the republicans the data is collected from the official republicans account, from Donald Trump's account and also from other officials who represent the Republican Party. For the Non – Political tweets the data is selected from various twitter accounts on music, food, movies etc.

Submitted to 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). Do not distribute.

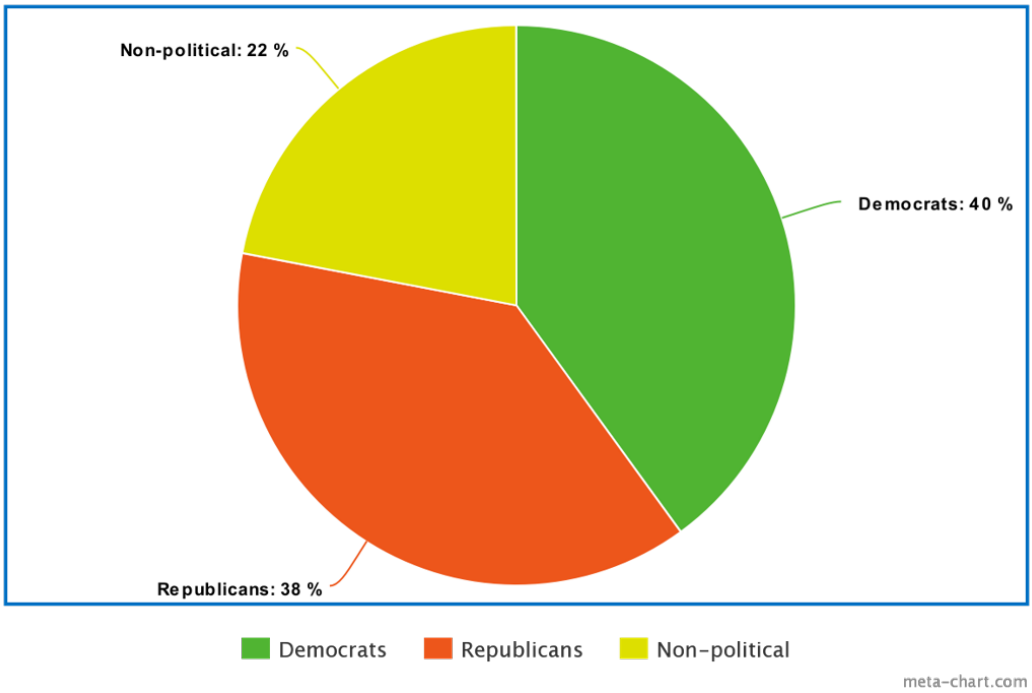


Figure 1: Distribution of data

2.1 Ground Truth

The ground truth for training and evaluation is formed by the accounts the tweets were extracted from. All the data from Republican party representative's tweets grouped as class Republicans, the data from the Democrat party representatives grouped as class label Democrats. Finally, the remaining data is grouped to represent the class label Non-Political

2.2 Data Pre-Processing

Basic text pre-processing, Bag Of Words, was applied: stop words removal, stemming, and tokenization. Then, we removed all the emojis and the hyperlinks. We repeated the experiments with and without the hashtags and the mentions removing them together and individually.

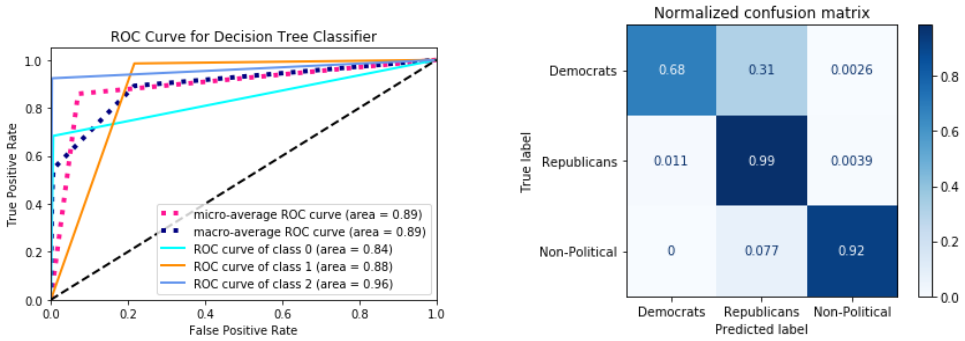
3 Experiments

We ran experiments using Multinomial Naïve Bayes, Random Forest Classifier, Decision Tree Classifier and KNN Classifier.

3.1 Decision Tree

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features [3]. In a decision tree, each branch shows the feature values specified at that node. And every test result in branches, representing varied test outcomes. The basic decision tree algorithm constructs decision trees in a recursive divide-and-conquer manner.

We used the Scikit learn's implementation to fit the model and have observed that the model is overfitting one class using the Gini as the measure for quality of split. We allowed the model to fit until the leaves are pure and set the remaining parameters to default values. The results we observed are summarized in Figure 2.



(a) ROC curve (b) Confusion Matrix

Figure 2: Decision Tree Metrics

The label 0,1,2 represents Democrats, Republicans, and Non-Political respectively. As a single decision tree is overfitting we experimented with Random Forest Classifier.

3.2 Random Forest Classifier

Random Forest classification algorithm consists of many decision trees. It uses feature randomness when building each individual tree[4]. Random forest is a type of learning where we join different types or the identical algorithm multiple times to create a more efficient prediction model. This algorithm combines multiple decision trees, resulting in a forest of trees, called "Random Forest".

We have used number of estimators as 100 and achieved an accuracy of 0.915.

The following is the ROC curve for Random Forest Classifier.

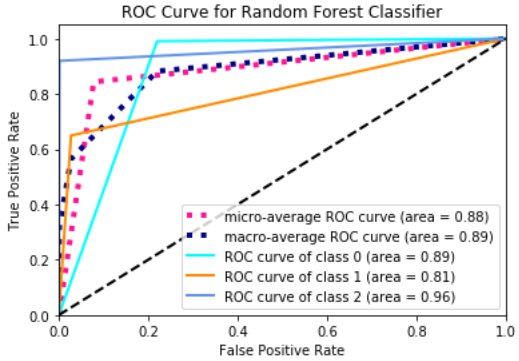


Figure 3: ROC for Random Forest Classifier

There was a great improvement on the accuracy still there was overfitting.

3.3 KNN Classifier

This algorithm is used to classify by finding the K nearest matches in training data and then using the label of closest matches to predict. Generally, we use Euclidean distance to find the closest match [4].

For KNN Classification, we have used sklearn library to import the classifier. Hyperparameter tuning is performed on KNN classifer to find the best match for n_neighbors. A hyperparameter is a parameter where we set values before fitting the model. We have tried both Grid Search CV and Random Search CV as part of hyperparameter tuning to achieve better results.

Table 1: Metrics of different K values of KNN classifier using GridSearchCV

Metric	K=3	K=5	K=7	K=13	K=19	K=21	K=23
Accuracy	0.89	0.87	0.93	0.872	0.892	0.894	0.902
Precison	0.88	0.89	0.89	0.90	0.897	0.884	0.879
Recall	0.93	0.93	0.91	0.935	0.922	0.920	0.919
CV Score	0.914	0.91	0.895	0.890	0.886	0.88	0.91

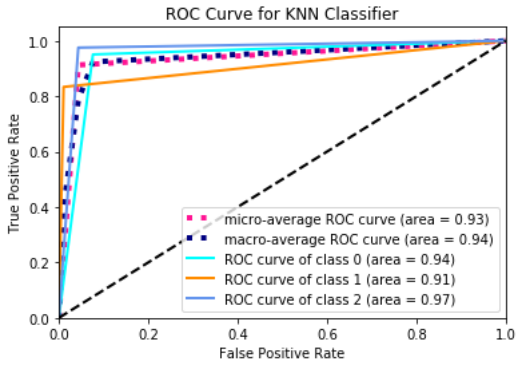


Figure 4: ROC curve for K=7

- Through sklearn's GridSearchCV, we have selected a range of k= 1 to 30 for number of nearest neighbors and achieved the best score at k = 7. The mean CV score has been increased from 0.89 to 0.93
- In Random Search CV, Best parameter was k =14 and the mean cross validation score has been increased from 0.89 to 0.94

The disadvantage for these methods is that inference of a new datum requires pre-processing and out of vocabulary word is dropped which might be useful for prediction.

3.4 Word2Vec

Word2Vec is a prediction based Embedding which means this method provide probabilities to the words which will work perfectly for tasks like word analogies and word similarities. Word2Vec can rely on either Continuous Bag-of-Words (CBOW) or Skip-Gram. CBOW predicts the probability of a word given a context, on the other hand, Skip-Gram predicts the context given a word. A context can be a single word or a group of words. CBOW finds the word with maximum probability in the given context. So, there will be an issue with infrequent words. Whereas, Skip-Gram treats both frequent and infrequent words same way, thus learns to interpret even rare words better. In this project, names Joe and Biden don't appear together very often in our training tweets, the cosine similarity is not that good when we use CBOW. Similarly, with words Real Donald and Trump. Skip-Gram outperformed CBOW in finding cosine similarity between words Joe and Biden. Below figure shows cosine similarity values predicted by CBOW and Skip-Gram.

```
w2v_model.wv.similarity('joe','biden')
```
```
0.12217695
```
```
w2v_model.wv.similarity('realdonald','trump')
```
```
0.13505428
```

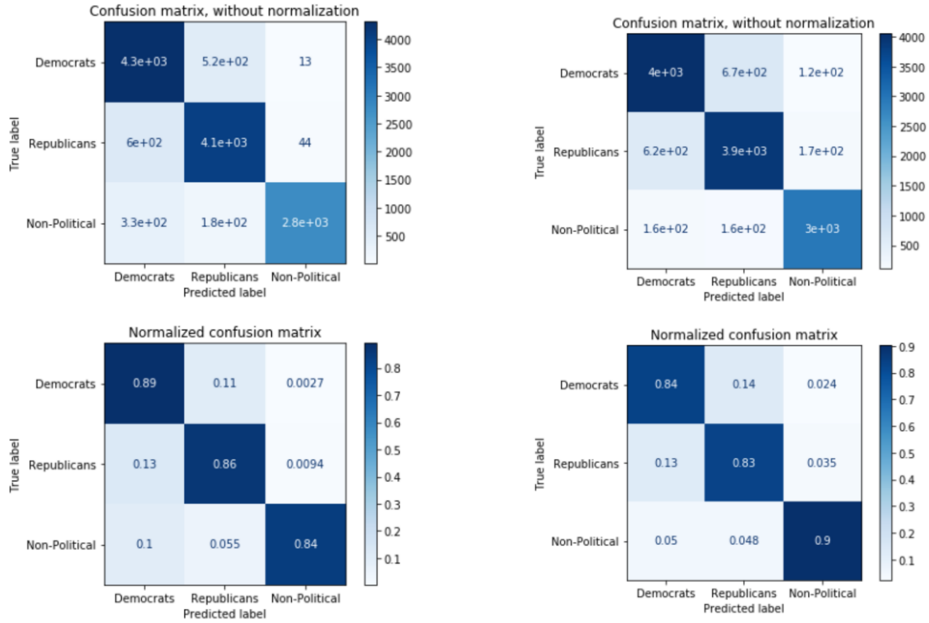
(a) CBOW cosine similarity

```
w2v_model.wv.similarity('joe','biden')
```
```
0.8777736
```
```
w2v_model.wv.similarity('donald','trump')
```
```
0.78913105
```

(b) Skip-Gram Cosine Similarity

Figure 5: Cosine Similarity of CBOW and Skipgram

Then, we used the KNN with 7 nearest neighbors and Multinomial Logistic Regression for classification. The results are shown in the Figure 6



(a) Skip-Gram with KNN

(b) Skip-Gram with Multinomial Logistic Regression

Figure 6: Comparision of Skip Gram with KNN and Multinomial Logistic Regression

3.5 Fast Text

FastText is one of the efficient libraries used for text classification and word embedding [5]. This is based on skipgram model, where each word is represented as a bag of character n-grams. A vector representation (embedding) is associated to each n-gram for a word. Thus, we can represent a word by the sum of the vector representations of its n-grams. By leveraging n-grams from individual words based on their characters, there is a higher chance for rare words to get a good representation since their character based n-grams should occur across other words of the corpus.

After testing the model, we achieved a score of 0.89.

3.5.1 Data Preprocessing

To improve the score, we have performed Preprocessing on the data by removing all the punctuations, stop words, Retweets, Hashtags, @names which then reduced our score to 0.85. After few hits and trails we observed that Hashtags and @names perform important role in classification of the tweets. Therefore, we had added back Hashtags, @names in the preprocessing step which increased our score to 0.92

3.5.2 Hyperparameter Tuning

FastText represents a word by bag of character of N-grams. Therefore, we tuned the parameter value for the model by varying the n-values from 2 to 6 also changing learning rate and number of epochs. By this we achieved a score of 0.936 with WordNGrams=2, lr=0.7 and number of epochs =50. Comparison of the measures with their techniques is shown below:

Metric	Precision	Recall	F1 Score
Before Preprocessing	0.891	0.891	0.891
After Preprocessing	0.850	0.850	0.850
Including #tags-@names	0.929	0.929	0.929
Hyper Tuning	0.936	0.936	0.936

Table 2: Metrics with different pre-processing

4 Analysis

Form the experiments conducted we found that fast text performs better than other classifiers not just for training but also for inference which makes it highly suitable for real time predictions where data that is being generated is very high. Also, we found that inclusion of the hashtags and the mentions have increased the accuracy of the models.

5 Future Work

Here we tried to predict a tweets political orientation. This acts as a first stepping stone in the big picture of predicting a general election where we want to predict several tweets per second which requires more robust methods. Future work may investigate how to incorporate the regularization and the confidence interval of a prediction to predict the tweet.

6 Related Work

We have adopted an agile working methodology to implement this project. The details can be found app.zenhub.com and we used git for our version control Github.com. An implementation of our project can be found here The Pollsters

References

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Andrew Mercer & Claudia Deane & Kyley Mcgeeney *Why 2016 election polls missed their mark*, https://www.pewresearch.org/fact-tank/2016/11/09/why-2016-election-polls-missed-their-mark/

[3] Breiman, L. &, Friedman &, Olshen R. & Stone C. Classification and Regression Trees. Wadsworth, Belmont, CA, 1984.

[4] Bahrawi, Bahrawi. (2019). Sentiment Analysis Using Random Forest Algorithm-Online Social Media Based. Journal of Information Technology and Its Utilization. 2. 29. 10.30818/jitu.2.2.2695.

[5] Armand J. & Edouard G. & Piotr B. & Tomas M. Bag of Tricks for Efficient Text Classification, 2016