Congressional Tweets Prediction Report

Jiarui Chen and Xuanyu Shen

***Abstract*— The project mainly predicts the political party association of tweets. A Ridge Regression Model was used to make the prediction. In addition, a temporal analysis on the favorite count, a frequency count on the hashtags, and a sentiment analysis were conducted on the dataset for descriptive analysis.**

I. INTRODUCTION

This report is about predicting the party affiliation of the tweet message sent by congressional politicians. Our data set contains all tweets shared from the politicians. The data covers a period between the years 2008 - 2021 and contains 857,803 tweets. Our goal is to implement a model that can mostly predicted the party affiliation. We expected our target accuracy to 90%, and currently we made it 88.705%. For descriptive analysis, we conducted Temporal Analysis to understand the trend of favorites and retweet counts for the two parties in the chronological order by calculating the average favorite and retweet counts for the two parties each year. Then we visualized hashtag ratio to analyze two parties' key concentrations and discussion trends. A sentiment analysis was also conducted to study the attitude of tweets. In temporal analysis, we found that between 2013 and 2017, Democratic had average 200 more favorite counts than Republican steadily, but fall down immediately after Republican leader Donald Trump took charge of the White House (In 2019, the Democrats had 81 more favorites per post, but then Republican had 217 more favorite counts than Democrat in the following 2020). In sentiment analysis, we found that most of the tweets had positive sentiments (47.9% and 49.2% for Republicans and Democrats respectively), and Democrats tended to have clearer attitude in their tweets. These interesting explorations provided us interesting background before our prediction.

II. DATA PREPROCESSING

Before descriptive analysis and building the classification model, preprocessing was conducted on the dataset. We first observed that, for each tweet in the dataset, the text was surrounded by b" ", so we used the Python strip() method to handle this formatting issue. Next, texts were tokenized and POS-tagged. Tokenization helped to treat each word separately, and POS-tagging labeled the part of speech of the token. With the identified part of speech, lemmatization was performed on the dataset so that words were converted back to their dictionary forms. This step helped to categorize the words with the same meaning but different forms together to reduce the noise for training the model. Then, numbers, punctuations, emojis, websites, and word with fewer than 2 characters were removed. In addition, we also observed some tweets start with "rt", which stands for retweet. Since this information does not contribute to the context of the tweet itself, "rt" were also removed from the text. Finally, tweet text in the original data set were replaced by the pre-processed clean text.

III. DESCRIPTIVE ANALYSIS

The descriptive analysis in this part is only based on the congressional_tweet_training_data.csv provided. The provided testing set is not included in the analysis because the actual political party associated with the tweet is unknown.

A. *Temporal Analysis on Favorite Counts*

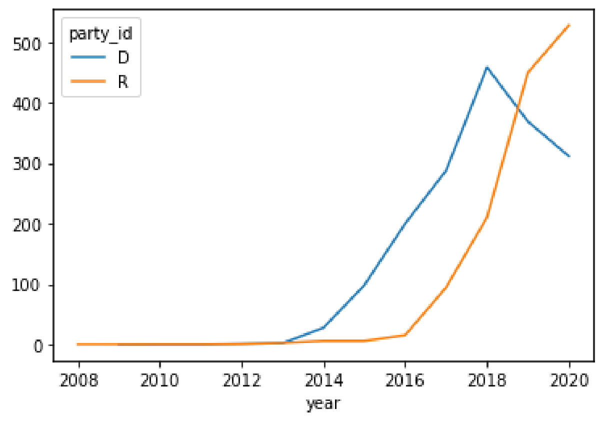

Fig. 1. Change in Favorites Counts of the Two Parties

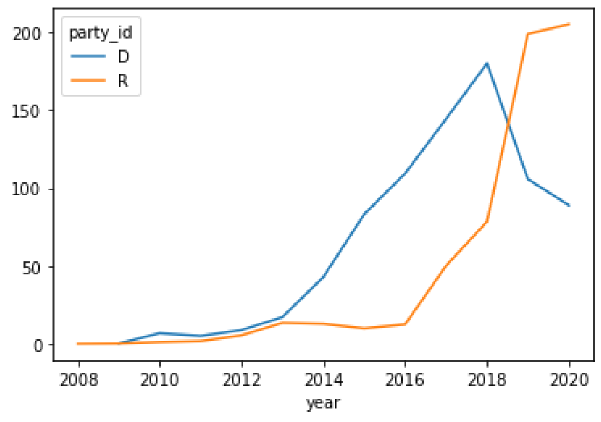

Fig. 2. Change in Retweet Counts of the Two Parties

We first conducted Temporal Analysis on "favorite_counts" and "retweet_counts". We measured the average amounts of favourite_counts and retweet_counts for each party every year, and tried to find the tendency for the two parties. We fulfilled this goal by constructing a new data frame which contained years, average favorites, and average retweet for the party. We used line charts to visualize the tendency (Figure 1 and Figure 2), and we found that when democrats took control of the White House (2009-2017), they had higher average favorites and retweet per tweet. However, when republican leader Donald Trump took charge of the White House, Democrats experienced a turning point and the trend fall down sharply from 2017 to 2018. The change in favorite and retweet counts highly correlate with each other.

B. *Hashtag Frequency*

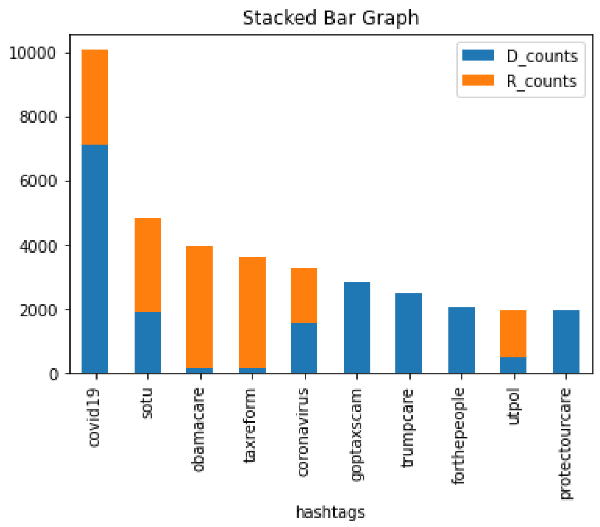

Fig. 3. Stacked Bar Chart for Hashtag Frequency by Party

After Temporal Analysis, we conducted analysis of the specific feature, which is "hashtags". "Hashtags" feature provides crucial information about the two parties' most frequent tags. To startup, we first transform the hashtags value into lower case. Then we implemented pandas function to group "hashtags" by the two parties and count the frequencies of each "hashtags" value. We chose the top 10 frequency value to analyze hashtags label and build a new data frame to contain these values. A stacked bar chart was created to visualize the proportion of participation of these two parties in the topics (Figure 3). What we found is that Democrats mentioned most about covid19, and had a huge tag ratio among "goptaxscan", "trumpcare","forthepeople", and "protectourcare". Republican's tags were concentrated more on "sotu", "obamacare", "taxreform", and "urpol". Republicans are not active tweeters of "goptaxscan", "trumpcare", "forthepeople", and "protectourcare".

C. *Sentiment Analysis*

For the final part of our descriptive analysis, we conducted sentiment analysis on the tweets and calculated the polarity sentiment score using the textblob package. We defined positive tweets as the ones with sentiment score higher or equal to 0.05, negative posts as the ones with sentiment score lower or equal to -0.05, and neural post be the rest. We found that (Figure 4), for both parties, most of their tweets convey positive sentiments. For Republicans, 47.9% of their posts were positive, and the proportion was 49.2% for the Democrats. We also found that there were not many negative posts for both parties. In the dataset, 11.1% and 13.3% posts were negative respectively for Republicans and Democrats. Our final finding is that Democrats tended to express clearer sentiment in their tweets. For both positive tweets and negative tweets, Democrats have a higher proportion than Republicans, which means that they have a lower proportion in terms of neutral posts.

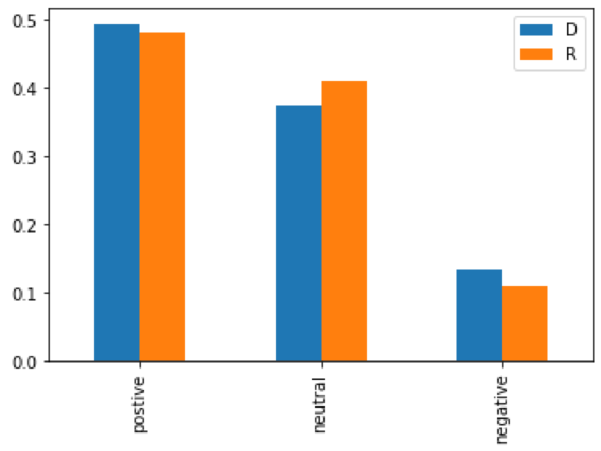

Fig. 4. Proportion of Sentiment by Party

IV. CLASSIFICATION METHODS

A. *Setting up*

After data preprocessing and descriptive analysis, models were built to predict the party associated with the tweet. The X we used for our model was the "text" attribute, and the Y we used was the dummied "party id" attribute (i.e they were converted to 0 and 1). To convert text input to numerical input for model training, TFIDF vectorizer was used, and the two parameters we specified were ngram range and max features. ngram range tells the model whether to create bi-grams or tri-grams, and max features tells the model to only use terms with high frequencies for building the feature spaces.

B. *Model Selection*

We tried three different models before reaching our final model selection: Multinomial Naive Bayes, Logistic Regression, and Multi-Layer Perceptron Classifier (Neural Network). A train-test-split with ratio 7:3 was applied on the training set, which allowed us to have a rough idea about the model performance before submitting our prediction using the provided test set. For each of the models, we all started with the default parameter settings. If the result seemed to be promising, we then proceeded to the model tuning process.

We found that Multinomial Naive Bayes and Logistic Regression model yielded an accuracy around 86% and 87% in general. For Multinomial Naive Bayes, we tried to use the boosted version of it by adding the AdaBoostClassifier(). A boosted version of a model create a collection of the chosen model, and each model in the collection focus on the part where the previous model classified wrong. The boosting method is generally considered as a way to improve model performance. However, in this case, the boosted versions of the two models both yielded an accuracy score around 56%, which made us move on to other models. The Neural Network with a hidden layer of size 50 did yield a slightly higher accuracy of 88%, but it was computationally expensive to train the model even with the early stopping condition. Therefore, it also didn't become our final choice.

Our final choice was Ridge Classifier based on ridge regression. The ridge classifier converts the Y variable to {-1, 1} and transforms the classification question to a regression question [1]. Since the model is based on the ridge regression, L2 regularization is applied to avoid overfitting. The model was quick to train and yielded high accuracy before tuning (around 87%).

C. Model Tuning

To achieve a higher accuracy rate, we proceeded to adjust the max feature and ngram range parameters previously mentioned. We found that limiting the ngram range to (1, 1) resulted in higher accuracy compared to the settings in which bi-gram and tri-gram are allowed. In terms of the max feature parameter, we found that a value that is too low (in this context, 5,000 and 10,000 for example) yielded poor results since the feature space was too small to capture the useful data for making correct prediction. It does not mean that a large value for max features is necessarily good. When increasing max feature to around 130,000, we found that as we further increase the value, the training accuracy increased, but there was no significant evidence that the testing accuracy was improved, which was a sign of overfitting. To better illustrate this point, we made a line chart (Figure 5) showing the impact of max_features parameter on training accuracy and test accuracy.

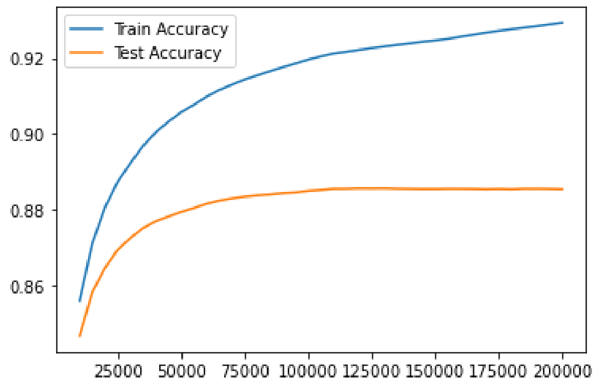

Fig. 5. Max_Features' Impact on Training and Testing Accuracy

The plot showed that the overfitting happened earlier 130,000. Actually, when "max_features" was around 110,000, test accuracy have no significant changes. Therefore, our final choice for the parameter is (1, 1) for the ngram_range and 114,000 for the max_features. For the Ridge Regression Model itself, we kept the default setting.

V. Classification Results

Using the 7:3 ratio for train test split, our test accuracy was 0.885448. To better understand what our model predicted well and what it predicted badly, we used a confusion matrix to take a further look (Table 1).

TABLE I
Confusion Matrix of the Model Prediction

	Actual 0	Actual 1
Predicted 0	70079	9843
Predicted 1	10529	87460

The confusion matrix showed that the model predicted 13% of the actual Republicans' tweet wrong, and it predicted 10.1% of the actual Democrats' tweet wrong. This suggested that the model may be better at identifying tweets from the Democrat Party. For the prediction submission on Kaggle, we changed the train-test split ratio to 99:1 to gain more training data, and the final accuracy on Kaggle was 88.705%, which indeed showed some improvement.

For team evaluation and criticizing, we think we did a good job in the competition. We made a great effort through trying out a great variety of models and model tuning methods, and both team members contributed to the work equally. However, this does not mean that our work was perfect. There is still room for improvement. When training the model, we have always been trying to use the sentiment analysis results from the descriptive analysis in the model to provide an additional dimension of data for enhancing the classification accuracy. However, we were not able to improve the model using this information, along with the favorite count and retweet count. If given more time, we would try to come up with a way to utilize these pieces of data for model input.

References

[1] "Ridge Classifier" sklearn Documentation. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear.RidgeClassifier.html. [Accessed April 2nd, 2022]