CS 485 Final Report
Claire Takenaga, Gavriel Warren, Joel Tang

Using NLP Models on Tweets to Predict Politician Party Affiliation

Abstract

With the common presence of social media in our society, politicians are able to communicate to their constituents with little mediation from traditional media outlets. This study investigates the core thematic differences between the Republican and Democratic parties by analyzing their language on Twitter. We leverage natural language processing (NLP) techniques, specifically using the PoliBERTweet model, to predict the political party affiliation of American politicians based on their tweets. Our approach involves fine-tuning an annotated dataset of politician tweets and their respective party affiliation through PoliBERTweet and employing a logistic regression model to create a binary classification task. Our research seeks to answer how accurately an NLP system can determine party affiliation, the significance of various linguistic features, and the contextual nuances that may elude the model.

Introduction

Twitter and other social media platforms gave politicians unprecedented access to their constituents. Before this access via social media, politicians primarily relied on traditional media outlets, like newspapers, television, and radio, to reach their potential voters. A natural consequence of this is that the nature of speech is a more formal and structured approach to communication. With direct access through social media, like twitter, the tone of speech has shifted significantly. Thereby allowing politicians to bypass the traditional media filter and interact directly with the public in real-time.

By bypassing the structural filter from traditional media, politicians can share their thoughts, political positions, and reactions to events in real time. Another consequence is that the communications are in a more informal and personal manner which allows a politician to form a more personal connection. The politicians respond to constituents' concerns and participate in trendy public discourse without the delay or filter that comes with traditional media coverage. While this creates a nightmare for the politicians public relations team, the authenticity can help politicians appear more relatable and accessible to the average citizen.

Not only is social media used for deepening the connection between politician and voter, it has also become a vital tool for political campaigns. Politicians are now able to attract supporters, organize events, and fundraise more effectively. They can more easily play a facade to create an image of who they think voters want to see in office. They can also target specific demographics with tailored messages, monitor public opinion, and adjust their strategies in response to the feedback from their campaign.

We aim to leverage the type of connection to identify and analyze the core thematic differences between the Republican and Democratic parties. Our plan to do this is by breaking down the language used by each party on the most used social media platform for politicians, Twitter.

Hopefully by doing so we will uncover priorities that distinguish them. We will then track and compare how these themes and priorities evolve over time, reflecting shifts in party priorities, ideology, leadership, and response strategies to events.

To accomplish this we will use a natural language processing model to predict the American political party given a tweet by an American politician. Given the prevalence of the major political parties in politician tweets, we will be focusing on only analyzing texts from either Democratic or Republican identified politicians. In doing this political analysis using natural language processing systems, we seek to answer the following research questions:

Research Questions:

- How accurately can an NLP system predict political party affiliation through linguistic analysis on tweets?
- What are the most important types of words (i.e. parts of speech, specific words or phrases) in determining the political party of the politician's tweet?
- What nuances or contextual scenarios in human understanding of the Tweet does the NLP model not capture?

Related Work

The article "PoliBERTweet: A Pre-trained Language Model for Analyzing Political Content on Twitter" [1] discusses the development of the language model "PoliBERTweet", which was trained by the Twhin BERT base model, an existing pre-trained language model founded by Google for specifically processing tweets. The Twhin BERT base model is trained on a large corpus of tweets and therefore captures Twitter-specific characteristics such as tokenization based on hashtags, emojis, mentions, and the overall informality of language used in tweets. The PoliBERTweet model is trained on 83 million US 2020 election-related English tweets, and evaluates the model on a "stance prediction task", wherein the model is trained to identify the author's position towards a target for a given tweet (i.e. whether the author feels positively, negatively, or neutral towards target "Trump").

Another research paper titled "Predicting U.S. Political Party Affiliation on Twitter" [2] aims to solve a very similar task to ours. They discuss developing a binary classification model that identifies a U.S. political figure's political affiliation based on a single tweet. The paper uses neural network algorithms and specifically shows that the GRU model (Gated Recurrent Unit) correctly classifies political parties of politicians with 91.6% accuracy. While our project will likely use a traditional machine learning algorithm such as Naive Bayes or Logistic Regression models.

Data

We found our dataset on the website, Kaggle, which is a large data science platform which contains datasets covering a variety of different topics. The dataset we chose to use is titled "Democrat vs. Republican Tweets" [3], which contains the 200 most recent tweets from 433 U.S. politicians as of May 2018, for a total of 86,461 tweets. Only tweets belonging to politicians affiliated with either the Democratic or Republican party are listed, and there is a fairly even split

between the number of Democratic vs Republican tweets in the dataset (49% Democrat, 51% Republican). The dataset is given as a CSV file, and contains 3 columns labeled as: Party, Handle, Tweet. The 'Party' column refers to the political party affiliation of the politician that authored the tweet. The 'Handle' column refers to the Twitter handle of this politician. The 'Tweet' column refers to the actual tweet that was posted by this politician. For our research purposes, we did not analyze the 'Handle' values seeing as we are only focusing on political party affiliation and the words that make up the tweets themselves, rather than the specific author of the tweet.

The dataset is somewhat outdated as it only contains tweets up until 2018. Seeing as political opinion and topics of discussion have likely changed quite a bit in the last 6 years, the model may not perform an accurate linguistic analysis of current political tweets if trained on this data. However, our purpose in analyzing these political tweets is focused on seeing how well an NLP model can learn and predict political party affiliation based on the given data, so we will analyze results knowing the specific political climate with which those tweets were written. We are also interested in seeing if there are specific words or parts of speech that are important in determining the political party affiliation of a political tweet. We must also keep in mind that the results may not be indicative of the entirety of political tweets, seeing as this dataset is 6 years outdated and focused only on tweets from politicians and therefore may miss some nuances in how the general public perceives the political atmosphere in the United States. We hope that analysis on our NLP model will help shed light on what some of these nuances might be and the differences between how politics are discussed by politicians themselves versus how they are discussed by the people who support these figures. Therefore, we might gain insight on how political views are viewed and conveyed to the public via popular social media platforms–especially those which use more colloquial and informal language–such as Twitter.

Method

At first, we explored the pre-trained language model Twhin BERT base, which has been evaluated on various NLP tasks such as POS tagging, NER (named entity recognition), and text classification. We utilized the Twhin BERT base tokenizer to preprocess the Tweet data, as it is specifically trained on Twitter language and characters. The Twhin BERT base tokenizer employs various tokenization methods to preprocess the data, including breaking down words into "subwords" (morphemes), identifying and handling special characters such as hashtags, mentions, URLs, and emojis (#, @, replacing URLs and mentions with special tokens), removing retweets, and other standard normalization processes (lowercasing letters and removing accents). The Twhin BERT base tokenizer also uses a vocabulary that incorporates words, tokens, and subwords/morphemes frequently used in Twitter posts.

After initially exploring Twhin BERT base, we found PoliBERTweet, a more specialized pre-trained language model for our classification task. PoliBERTweet has the same properties of Twhin BERT base, being able to be used as a tokenizer to preprocess tweet data in addition to its specialization for politically related tasks. Being trained on 83 million tweets related to the 2020 political election, we decided to use this model first for our classification task.

We imagined that using the PoliBERTweet to preprocess and tokenize our dataset would be beneficial, as Twitter content often contains language and characters not commonly used in other types of more formal texts, such as articles or speeches. In addition, the political side of the PoliBERTweet could pick up politically related words better. For model development, we employed multiple methods and compared them. We used Python packages sklearn, pandas, and numpy for logistic regression with different vectorization methods, along with a Naive Bayes implementation.

However, we still wanted to test how well PoliBERTweet compares with Twhin BERT base to see the significance of PoliBERTweet being tested to specifically find the political party affiliation of a given tweet. In addition to testing with PoliBERTweet and Twhin BERT base, we used a normal, control, BERT model called BERT base uncased. This BERT model is a popular BERT model developed by Google to understand the context surrounding words. It is uncased, meaning it processes all text as if it were lowercase. This model has been trained on an extremely large corpus of text from Wikipedia and BookCorpus. This model serves as a strong basic model that could be applied to many different types of NLP tasks. Comparing BERT base uncased to Twhin BERT base and PoliBERTweet would demonstrate the relative effectiveness of these specialized models.

We first created a logistic regression model that uses word counts as its features. Utilizing a binary classifier, our model predicts either "Democrat" or "Republican" to indicate the political party affiliation of the politician who authored the given tweet. Then, to compare logistic regression to other models, we also used naive bayes and linear regression as classification algorithms. In comparing these models with each other, we used the BERT base uncased and PoliBERTweet models to extract word embeddings as features, and then use them in the naive bayes, logistic regression, and linear regression models. Similarly, we had these models predict either “Democrat” or “Republican” as the political party affiliation from the politician tweet.

Finally, we wanted to test these classification models, logistic regression, naive bayes, and linear regression, with and without PoliBERTweet and BERT base uncased. We wanted to see the impact of the BERT models in how they fine-tune and tokenize the text.

Results

1. **Baseline Algorithm–Zero Rule Algorithm**:

The baseline algorithm we decided to use is the Zero Rule Algorithm, where each label for tweets in the test set will be predicted as the label/class that occurs most frequently in the training data. Therefore, since our classes are “Democrat” or “Republican”, this algorithm will predict all tweets in the test set to have a label of either “Democrat” or “Republican”–whichever class label occurs more often in tweets in the train set. We then compared our model’s accuracy in predicting labels after learning from the actual labels in the training data to this baseline algorithm’s results. The results of the baseline algorithm are as follows: 42068 Democrat Tweets and 44392 Republican Tweets. Therefore, the baseline algorithm should always choose the class “Republican” and would predict tweet labels with a 51.3% accuracy.

2. **Preliminary Experiment–Logistic Regression with Bag-of-Words Features**:

For the preliminary experiment, we used a logistic regression model using word counts as features, which utilized our code from Homework 2. We used a 80%, 20% training and test split of our dataset. After training the Logistic Regression model using word counts as features, we got the following results:

Overall Report	Precision	Recall	F1-Score	Support
Democrat	0.79	0.77	0.78	21097
Republican	0.79	0.81	0.80	22133
Accuracy			0.79	43230
Macro Avg	0.79	0.79	0.79	43230
Weighted Avg	0.79	0.79	0.79	43230

3. **Pre-trained BERT with Various NLP Models**:

The first method we tried was using two different pre-trained BERT models to extract word embeddings as features, and then use different models, namely Naive Bayes, Logistic Regression, and Linear Regression, to predict the labels of the tweets in the test set. The BERT models we used were the bert-base-uncased and PoliBERTweet (using the "kornosk/polibertweet-mlm model"). Since Linear Regression predicts a continuous value as an outcome, we set the decision threshold to 0.5 to determine whether the predicted outcome was "Democrat" or "Republican". Furthermore, we split our dataset using a 80%, 10%, 10% split for the training set/development set/test set, and took a subset of 7500 of the 86,460 tweets in the dataset to train and test our models. Therefore, there were 6000 tweets in the training set, 750 in the development set, and 750 in the test set.

We looked at the overall accuracy, precision, recall, and f1-score for each of the Naive Bayes, Logistic Regression, and Linear Regression models, using both the bert-base-uncased and PoliBERTweet variation. The results are as follows:

a. **Bert-base-uncased**:

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.60	D: 0.61 R: 0.59	D: 0.55 R: 0.64	D: 0.58 R: 0.61
Linear Regression	0.59	D: 0.60 R: 0.58	D: 0.54 R: 0.64	D: 0.57 R: 0.61
Naive Bayes	0.49	D: 0.49	D: 0.40	D: 0.44

		R: 0.49	R: 0.58	R: 0.53

b. **PoliBERTweet**:

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.56	D: 0.57 R: 0.56	D: 0.51 R: 0.62	D: 0.54 R: 0.58
Linear Regression	0.57	D: 0.57 R: 0.56	D: 0.51 R: 0.62	D: 0.54 R: 0.59
Naive Bayes	0.48	D: 0.48 R: 0.49	D: 0.42 R: 0.55	D: 0.45 R: 0.52

Based on these results, we can see that the regular bert-base-uncased BERT model performed better for our task in predicting the political party label for the given tweets. Although the accuracies are quite close, the bert-base-uncased model consistently had higher accuracies than the PoliBERTweet model. This was surprising to us because the PoliBERTweet model was trained on a large corpus of political tweets, and thus we imagined that it would be able to better preprocess and extract features for our dataset of political tweets as well.

Looking at the difference between the individual Naive Bayes, Logistic Regression, and Linear Regression models after extracting the features using the BERT models, we can see that the logistic regression model consistently performed the best for both BERT variations. The linear regression model predicted labels with the second highest accuracy, and the naive bayes model performed the worst at predicting labels. Since linear regression predicts a continuous outcome, whereas logistic regression predicts a discrete outcome, it follows that the logistic regression model performed better for our binary classification task, as logistic regression predicts the probability of instances being categorized into various classes, which would be the two political parties in our case. Logistic regression likely performed better than naive bayes because the naive bayes model assumes independence between features. Since our model took in textual input in the form of political tweets, it may follow that we cannot always assume that there is independence between features, as words in the tweets may be correlated with each other depending on the political topic discussed.

4. **Fine-Tuning BERT**:

One method we pursued was to test and analyze differing pretrained Bert models along with the original model. To achieve this goal, we again used a logistic regression model with word embeddings extracted from various BERT models, and then fine-tuned each model to compare the results of no fine-tuning versus a fine-tuned BERT model. In this trial, we again used a 80%, 10%, 10% split for the training, development, and test sets, with a subset of 6000 tweets from the initial dataset: 4800 tweets in the training set, 600 in the development set, and 600 in the test set.

We used the “kornosk/polibertweet-mlm” PoliBERTweet model as our main BERT model, “twhin-bert-base” for the generic Twitter BERT, and “bert-base-uncased” for the BERT base model.

a. **Logistic Regression with Word Embeddings Extracted from BERT**:

Average 12-layer accuracies for 4800 train 600 test

Model	Validation Accuracy (%)	95% Confidence Interval (%)
PoliBerTweet	59.4	[55.4, 63.3]
Generic Twitter Bert	58.5	[55.6, 63.4]
Generic Bert	63.7	[59.8, 67.5]

For the initial logistic regression model with word embeddings extracted from the three BERT models, we obtained the following results to the left. Averaging the 12 layer accuracies for each model, we can see that none of the three models perform very well. Additionally, they are all relatively similar to the point that the 95% confidence interval indicates that the current amount of testing is inconclusive to decisively indicate that one model is more accurate than the others. Additionally all three models are marginally higher than the baseline algorithm accuracy of 51%. Extrapolating this to our preliminary experiment results of using Bag-of-Words features with a logistic regression implementation, the results are considerably worse. The logistic regression model created through using a bag-of-words features resulted in an 79% correctness. This appears to indicate that the relationship between words in tweets is less impactful than the overall connection and relationship between them. However, this cannot be the case in general as sarcasm and contextual nuance plays an important role when determining the political bias for the author of a tweet. Therefore, we may see a different story play out after tuning our three models.

b. **BERT with Default Training Arguments**:

The three models were tuned with “default” training arguments. The “default” arguments are composed of using 2 training epochs, a batch size of 8 per device during training, a batch size of 64 for evaluation, evaluation and logging occurring after each epoch, log level being set to “error”, and using pytorch's “adamw” implementation. The other “default” arguments used were the pre-existing default training arguments.

Accuracies for test "default" setting 4800/600

Model	Validation Accuracy (%)	95% Confidence Interval (%)
PoliBerTweet	78.0	[74.7, 81.3]
generic twitter bert	67.0	[64.6, 69.4]
generic bert	74.7	[71.2, 78.1]

To the right are the results after running a split of 4800 tweets for training and 600 for development, which in this case is the test set. Surprisingly, the generic twitter bert model has the lowest increase in accuracy compared to the previous results. The generic twitter bert model also is decisively statistically worse than the other two models as the upper limit of its confidence interval, 69.4%, is below the lower limit of the other 95% intervals. Comparing the better two models in terms of accuracy, PoliBERTweet slightly outperforms the baseline bert model. This may be due to the “default” arguments favoring PoliBERTweet, which may therefore require additional fine tuning.

We also saw varying levels of accuracy for each of the 12 layers for a varying volume of tweets against the PoliBERTweet model, which can be seen in the table below. Unsurprisingly we observe an increase in accuracy from the 4000 train 500 test split to the 4,800 train 600 test split.

Surprisingly, when the triple the amount of tweets being processed, there does not appear to be an increase in accuracy, with the highest accuracy being layer 5 of the 4,800/600 split. We therefore kept this split for the next step of fine tuning.

Validation Accuracy and 95% Confidence intervels for varying volumes on PoliBerTweet

Layer	4,000/500 Acc (%)	95% CI (%)	4,800/600 Acc (%)	95% CI (%)	12,000/1,500 Acc (%)	95% CI (%)
1	49.2	[44.8, 53.6]	56.3	[52.4, 60.3]	58.9	[56.4, 61.4]
2	52.0	[47.6, 56.4]	58.3	[54.4, 62.3]	60.4	[57.9, 62.9]
3	52.4	[48.0, 56.8]	61.0	[57.1, 64.9]	61.3	[58.8, 63.7]
4	55.4	[51.0, 59.8]	62.2	[58.3, 66.0]	60.4	[57.9, 62.9]
5	53.2	[48.8, 57.6]	63.3	[59.5, 67.2]	59.5	[57.0, 62.0]
6	53.2	[48.8, 57.6]	61.5	[57.6, 65.4]	59.3	[56.8, 61.8]
7	55.4	[51.0, 59.8]	61.0	[57.1, 64.9]	60.2	[57.7, 62.7]
8	53.0	[48.6, 57.4]	62.2	[58.3, 66.0]	58.7	[56.2, 61.2]
9	52.8	[48.4, 57.2]	60.2	[56.2, 64.1]	59.1	[56.6, 61.6]
10	53.2	[48.8, 57.6]	59.5	[55.6, 63.4]	61.1	[58.6, 63.5]
11	53.2	[48.8, 57.6]	58.3	[54.4, 62.3]	60.8	[58.3, 63.3]
12	51.8	[47.4, 56.2]	56.7	[52.7, 60.6]	60.9	[58.5, 63.4]

notation is train/test

c. **Fine-Tuning BERT**:

To fine tune the model, we used the most accurate value for the parameter being tested in subsequent parameter testing. We tested five parameters in the following order. Learning rate, number of training epochs, batch size, warmup size, and weight decay. The results in changing the learning rate did not appear to be of any statistical significance between learning rates. Nevertheless, a learning rate of 4e-5 was kept for the subsequent tunings. Unfortunately, fine tuning the rest of the arguments–number of training epochs, batch size, warmup size, and weight decay–did not yield more accurate results. Although the differences between the values of the arguments were statistically insignificant, the downward trend appears to indicate that there is a significant difference between the values of the arguments. In the future we may increase the volume of tweets processed as this may have inhibited the ability for the arguments to take effect. Additionally, we may experiment with increasing the number of training epochs and batch size together. Increasing the number of training epochs drastically increases the run time while increasing the batch size has the opposite effect.

Varying Training Epochs for PoliBerTweet

Model	Num Train Epochs	Validation Accuracy (%)	95% Confidence Interval (%)
PoliBerTweet	2	81.5	[78.4, 84.6]
PoliBerTweet	4	79.8	[76.6, 83.0]
PoliBerTweet	6	78.3	[75.0, 81.6]
PoliBerTweet	8	77.2	[73.8, 80.5]

Varying Learning Rate for PoliBerTweet

Model	Learning Rate	Validation Accuracy (%)	95% Confidence Interval (%)
PoliBerTweet	2e-5	78.2	[74.9, 81.5]
PoliBerTweet	3e-5	79.7	[76.4, 82.9]
PoliBerTweet	4e-5	81.5	[78.4, 84.6]
PoliBerTweet	5e-5	78.0	[74.7, 81.3]

Varying Weight Decay for PoliBerTweet

Model	Weight Decay	Validation Accuracy (%)	95% Confidence Interval (%)
PoliBerTweet	0.0	81.5	[78.4, 84.6]
PoliBerTweet	0.05	81.3	[78.2, 84.5]
PoliBerTweet	0.1	81.2	[78.0, 84.3]
PoliBerTweet	0.2	80.8	[77.7, 84.0]

Varying per_device_train_batch_size for PoliBerTweet

Model	Batch Size	Validation Accuracy (%)	95% Confidence Interval (%)
PoliBerTweet	8	81.5	[78.4, 84.6]
PoliBerTweet	16	76.2	[72.8, 79.6]
PoliBerTweet	24	75.7	[72.2, 79.1]
PoliBerTweet	32	75.0	[71.5, 78.5]

Varying Warmup Steps for PoliBerTweet

Model	Warmup Steps	Validation Accuracy (%)	95% Confidence Interval (%)
PoliBerTweet	0	81.5	[78.4, 84.6]
PoliBerTweet	100	80.3	[77.2, 83.4]
PoliBerTweet	200	80.7	[77.5, 83.8]
PoliBerTweet	300	80.0	[76.8, 83.2]

We then tested the best fine tuned model against the testing set of 600 tweets. The best training argument values were: learning rate=4e-5, number of training epochs=2, batch size=8, warmup steps=0, weight decay=0. The resulting accuracy came out to be 69.0% with a 95% confidence interval of 65.3% to 72.7%. It appears that we significantly overfit as this is a statistically significant difference from the prior accuracy of 81.5% during the fine tuning against the development set.

Misclassified Tweets

We also wanted to look at the tweets that were wrongly classified in order to analyze our research question in order to see where the model is not able to capture certain aspects of our human understanding of natural language. In analyzing the misclassifications of these tweets there is clear evidence that several factors are contributing to the misclassification of tweets. For example, a tweet from NovaMBB thanking @RepTomMacArthur was misclassified. Despite the positive nature of the message towards a Republican representative, the lack of context in the specific token RepTomMacAuthur led to the classification of the tweet being democratic. Furthermore, a tweet commending Brad for his dedication to serving New Jersey residents could have been misclassified because the model associated positive sentiments with a particular party, despite the politically neutral nature of the message. Lastly, a tweet advocating for a comprehensive scientific study on gun violence could have been misclassified due to language patterns, as the model might have mistakenly interpreted the call for evidence-based decision-making as aligning with Republican values.

In summary, these misclassifications likely arise from a combination of factors, including ambiguity in tweet content, contextual understanding, named entities, local focus, language patterns, and biases present in the training data. These examples underscore the complexity of political classification tasks and the challenges inherent in accurately discerning the political affiliations of social media content.

Examples of Highest Confidence Misclassified Tweets:

Text: RT @NovaMBB: @RepTomMacArthur Thank you for your support! We couldn’t do it without you, #NovaNation. Now, #LetsMarchNova. https://t.co/wi7…
True Label: 1, Predicted Label: 0, Confidence: 0.995

Text: Brad has been an incredible member of my team and works tirelessly on behalf of the people of New Jersey. I'm so pr… https://t.co/xzXofExh1x
True Label: 0, Predicted Label: 1, Confidence: 0.994

Text: I'm urging Congress to instruct relevant federal agencies to conduct a comprehensive scientific study on gun violen… https://t.co/3dk57Xgn2U
True Label: 1, Predicted Label: 0, Confidence: 0.994

Discussion and Future Work

Our experiments with different BERT models and fine-tuning efforts yielded several insights into their performance in predicting political bias from tweets. We tested PoliBERTweet, TwhinBERT base, and BERT base uncased, finding none significantly outperformed the others due to accuracies only slightly higher than the baseline algorithm, which had a 51% accuracy. This was unexpected since we assumed models like BERT were able to surpass simpler approaches such as Bag-of-Words with Naive Bayes, which achieved 80% accuracy in prior experiments without the use of BERT.

The BoW approach's superior performance suggests that the relationship between individual words in tweets may be less impactful than their overall connection. While BERT models are designed to capture contextual nuances, they may require substantial fine-tuning to be effective for this task.

Fine-tuning experiments showed PoliBERTweet slightly outperformed the baseline BERT model, while TwhinBERT base significantly underperformed. Increasing the training dataset size from 4000/500 train test split to 4800/600 train test split led to only a marginal accuracy increase, indicating a plateauing effect. Despite tuning hyperparameters like learning rate, training epochs, batch size, warmup size, and weight decay, we did not achieve significant improvements in accuracy. The final testing of our best fine-tuned model resulted in 69.0% accuracy, lower than the 81.5% during fine-tuning, indicating overfitting.

In the future, we could extend our research to more and different politically related datasets. For example, in our research, we came across a different Kaggle set that had political tweets from 2021. However, we decided not to use it in this paper due to the fact that each tweet author was not labeled with their respective political party affiliation and manually annotating the large dataset would have been too time consuming. In the future, we could manually annotate this and more datasets to train and test our data. In addition to giving more data for our classification task to train and test on, we could see the association between the 2018 and 2021 results. We could find the voting and political trends of 2018 and 2021 and compare it to the results from the two datasets and find if there is any significant detail. Of course, this applies to any data that could fit into our model. More recent and diverse data would make this model much stronger.

In addition to finding more recent and diverse data, we could find metadata and include it in the annotations. This will allow the model to have more data to train on besides just text and potentially become more accurate. Metadata such as the date and time of the tweet, the politician's party affiliation, geographic location, and engagement metrics (likes, retweets, comments) can provide valuable context that helps the model's ability to discern patterns and make more informed predictions.

Currently, a big limitation to our model is that it only classifies the political party affiliation as “Democrat” or “Republican”. Of course, there are multiple party affiliations as well as specific

issues that politicians could focus on more than others such as healthcare, immigration and economy. With this in mind, we could alter our binary classification model into a model that displays multiple statistics on a politician. We could output a specific political party affiliation outside of Democrat and Republican as well as output a gauge of how much or how little a politician puts emphasis on specific political issues such as healthcare.

Moreover, expanding the classification approach to include various advanced machine learning techniques could enhance the model's performance. In addition to fine-tuning with BERT and using the classification algorithms that we used, we could explore other binary classification methods, such as using neural networks. In addition to finding more methods to create the task, we can use known LLMs to compare to our model such as ChatGPT or Gemini.

References:

[1] Kawintiranon, Kornraphop, and Lisa Singh. "PoliBERTweet: A Pre-trained Language Model for Analyzing Political Content on Twitter". *Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022)*, 20-25 June 2022, pp. 7360–7367, www.lrec-conf.org/proceedings/lrec2022/pdf/2022.lrec-1.801.pdf. Accessed 15 May 2024.

[2] Lee, Catherine, et al. "Predicting U.S. Political Party Affiliation on Twitter". *Stanford University*, 2018, cs230.stanford.edu/projects_spring_2018/reports/8291017.pdf. Accessed 15 May 2024.

[3] Pastor, Kyle. "Democrat vs. Republican Tweets." *Kaggle*, 27 May 2018, www.kaggle.com/datasets/kapastor/democratvsrepublicantweets/data. Accessed 15 May 2024.