Political Ideology Detection in 2020 US Election Tweets

Annika Dahlmann, Ryan Song, Divya Ramamoorthy

University of Michigan

April 1, 2022

1 Introduction

Media sources play an influential role in many individuals political decisions. Specifically, Twitter seems to be a source in which people get a lot of their information from nearing election time. Objectivity is important in journalism, especially when reporting on political agendas that can influence everyday life for individuals. In this project, we aim to classify the ideological bias of a person's tweets regarding the United States 2020 Presidential election. Ideological bias for our purpose can fall into one of the three categories: conservative bias, liberal bias, or neutral. Ideology-detection has been an important and relevant task within the natural language processing space as people rely on media to seek fast and easy information to make political decisions. The rise of Twitter as a media source has introduced an increased possibility of such biases as not as much time or resource goes into curating Tweets and individuals are restricted in the number of characters to defend their point in a singular Tweet.

Previous work has highlighted certain biases that exist in common news sources. For example, Fox News is known to foster more conservative bias. However, the goal of our project is to explore this same ideology bias classification with Twitter as our source. An example of how such bias can exist is that liberals use the phrase "estate tax" and conservatives use "death tax", two phrases to describe the same political statement where no neutral term exists. This will be the type of distinguishing we hope to extract from our Twitter data. Specifically, when training, the input to our model will be a singular Tweet and the possible classes (conservative, liberal, neutral), and the output will be the predicted class bias. When testing, we will only input unseen Tweets into our model and then output the predicted label.

2 Related Work

There has been a considerable amount of previous research done on sentiment analysis, specifically regarding political alignment classification in different forms of media.

Prior research on the subject includes work done in 2014 by researchers at the University of Maryland, who applied a Recursive Neural Network model to identify political positions suggested by given sentences.[4] In their work, political annotations on both the phrase and sentence level were collected and used as training data. The use of Recursive Neural Networks captured syntactic and semantic composition together and did not rely on hand-made dictionaries or rule sets. The dataset contained a mix of ideological statements that were enriched, using the work of past research studies, to have a higher likelihood of containing bias and were balanced to contain an equal number of sentences corresponding to the two political parties in study. Given this labeled data, an element in the vector space representing a sen-

tence with liberal bias was distinct from the vector of a conservative-associated sentence. The cross entropy loss was minimized using optimal model parameters, and the main distinction between this research and prior studies was the eradication of bag-of-words modeling and hand-designed lexicon.

Our research explores a question similar to the work done by the Maryland researchers, but we plan on using Twitter data rather than crowd-sourced sentences. We also are using a Logistic Regression model instead of a Recursive Neural Network model to classify the political ideology evinced by Tweets. This angle differs from the research previously explained and our approach should provide a different angle at the research question.

3 Changes since Project Proposal

3.1 Method

Originally in our project proposal, we mentioned only using logistic regression as our model. Amongst further analysis, we have decided to use logistic regression as our baseline and try to make enhancements from there. This will include using alternative models such as, Long Short-Term Memory (LSTM) and Naive Bayes to attempt to get the highest scoring model.[3] Additionally, we plan to use a variety of hyperparameter tuning techniques in addition to cross validation, which is the only tuning method we mentioned in our project proposal. In addition to cross validation, we will use random search and grid search to obtain the highest performing hyperparameters.

3.2 Evaluation

In our proposal, we originally mentioned that we would be using accuracy as our only metric for evaluation our model. However, past research has showed that it is important to report multiple evaluation metrics in order to get a full picture of how the model is performing. For this reason, we have chosen to include f1-score. This metric combines precision and recall to form a metric that performs well with binary classification which is our case since we are trying to classify a tweet into one of two political alignment categories, Democratic or Republican.

4 Data Pre-processing

4.1 Data Set

In order to learn our model we intend to use a dataset consisting of over 2.5 million 2020 US Election tweets, divided into tweets sent from politicians as well as non-politicians.[1] We can easily train our classifier over this dataset because each tweet is annotated with the political alignment of the tweet itself, either Democratic or Republican. Tweets lend themselves well to training classifiers because they are short and quick to analyze–this means that we can more easily process a large number of tweets than say, news articles, and thereby train our classifier on a larger number of data points.

4.2 Data Cleaning

In order to have our data prepared to input into our model, we have a couple of data cleaning steps. First, we removed any hyperlinks that might have been included in the original tweets. Next, we tokenized the lowercase version of each tweet using Natural Language Toolkit's (NLTK) TweetTokenizer, which is similar to the NLTK library word tokenizer that we've used throughout the course of this course. The only difference is that TweetTokenizer keeps hashtags intact.

The third step of our data cleaning pipeline was removing any stop words like "and", "is", "a", etc. Next, we converted every word to its stem using Porter stemmer which is also in the NLTK package. Stemming helps improve indexing efficiency and accuracy by only storing one form of a given word. Finally, we masked rare words (words occurring less than 2 times) as "UNK" (unknown) tokens in our frequency counts so that when encountering a word in testing that was not seen in training, we have a count to include.

4.3 Train, Validation, and Test Splits

For our classification we will split our data into three groups: training, validation, and testing. We will not be using all 2.5 million tweets for efficiency. Training and validation will be used to train our model and validate the best hyperparameters. We will use 10,000 tweets in total for these two splits, 8,000 tweets for training and 2,000 tweets for validation. And separately, we will use 2,000 other tweets for testing.

5 Methodology

Our group intends to use a Logistic Regression model in order to classify tweets by party affiliation. At a high-level we will implement the following procedure for training:

1. Pre-process data
2. Train LR model
3. Hyperparameter selection with Cross Validation

The first step of our training pipeline involves the preprocessing of our raw tweet data which is detailed in the Data Cleaning subsection. We propose retrieving data from both the collection of tweets written by politicians as well as from the collection of tweets written by non-politicians in order to cover the largest variety of tweet authors possible from both sides of the political spectrum.

5.1 Feature Engineering

To build our features, we will build a frequency dictionary, similar to HW1. We will take the tweet text and political affiliation as input and then count occurrences of each word in a dictionary object. The key of the dictionary will be a tuple of the word and political affiliation (1 representing Republican and 0 for Democrat) and the value corresponds to the count for that given word and affiliation.

5.2 Training

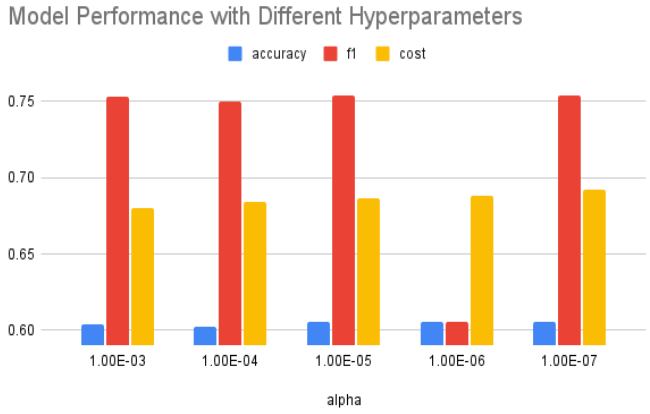
Following this we will pass our data into our logistic regression model, which will utilize a sigmoid function as well as a gradient descent function for the cost function.[2] After extracting the features of each tweet we will train our model, and obtain a set of optimal weights. These optimal weights will serve as the final parameters for our trained model.

5.3 Evaluation

In order to properly analyze the quality of our model, we will have a portion of the data set strictly used for testing. This set will not have been used for any part of training the model to ensure our model is not overfit to the training data and will perform well on unseen data. We will use accuracy as the actual metric to evaluate the performance of our test data, which will represent the number of times our predictions correctly matched the true labels assigned.

6 Experiments

So far, we have implemented a very basic logistic regression model in order to obtain baseline results that we can use as a reference. Our baseline model achieved an accuracy of 0.602 and an f1-score of 0.749. In our final model we hope to better tune the parameters and include other features which will improve upon this accuracy.



7 Future Work

Our initial attempts at classifying the political affiliation given a tweet had mediocre results meaning there's still room for improvement.

7.1 Model

First, we will try out different models to compare and contrast which performed best. Two options we are considering are LSTMs and Naive Bayes.

7.2 Feature Engineering

In addition to creating a frequency dictionary just with tokenized tweets, we are also going to explore how the model performs if we train using n-grams, using a value of n that performs the best.

7.3 Training

Finally, we will explore adding checkpoints into our training step. These checkpoints may be used directly, or used as the starting point for a new run, picking up where it left off. For our purpose, the checkpoint are the weights of the model. These weights can be used to make predictions as is, or used as the basis for ongoing training.

References

[1] L. Hu. *Political Partisanship Tweets*. 2022.

[2] A. Mashalkar. *Sentiment Analysis Using Logistic Regression and Naive Bayes*. Supervised ML Overview. 2020.

[3] *Natural Language Processing with Python*. 2009.

[4] *Political Ideology Detection Using Recursive Neural Networks*. University of Maryland Computer Science, Linguistics, iSchool, UMIACS. 2014.

[4] [2] [3] [1]