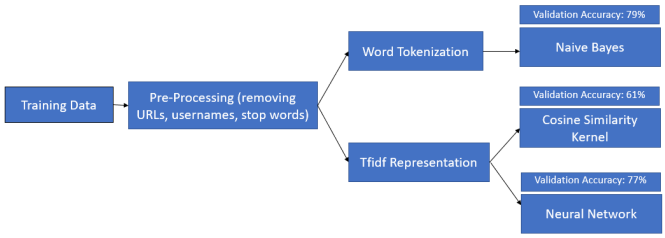
Final Project Report for CIS 419/519: Introduction to Machine Learning Classification in Political Twitter

Author 1: Khizrah Naveed KHIZRAH@SAS.UPENN.EDU
Author 2: Sidarth Subramanian SIDARTH@SAS.UPENN.EDU
Author 3: Francesca Marini FMARINI@SAS.UPENN.EDU

Abstract

In this project, we sought to develop a political alignment Tweet classifier using gold standard sentiment analysis techniques which classified political Tweets as either Democrat or Republican (based on the American two-party political system). With such a classifier, we can determine the political alignment of a particular Twitter account which can be useful when determining political support for certain parties before elections, creating recommendation systems used for recommending articles with certain political leanings and many other such applications. To see how our model performed on new real data, we extracted new Tweets from famous Democratic and Republican politicians and ran our models on them, post training and post validation. The results are included below. To train our model, we used a variety of different techniques (Elghazaly, et al.; Vishal, et al.) which are in illustrated in our pipeline Figure 1 below.

Figure 1. Pipeline of Project



University of Pennsylvania, CIS 419/519 Course Project.

1. Final Report Submission

1.1. Datasets & Pre-Processing

We initially acquired two datasets online in order to train our classifiers. The primary dataset we used for this project was a Kaggle dataset called Democrat Vs. Republican Tweets, in which all political Tweets that had been labeled as either Democrat or Republican[1]. This dataset contained approximately 170,000 instances which came in csv format in which labels were a single feature, as were the text of the Tweets themselves.

We primarily decided to work with the Democrat Vs. Republican Tweets dataset and discarded the second dataset (Political Twitter Corpus) as it was not appropriately labeled. The Democrat vs. Republican dataset was only a single train set of data, so we manually processed and randomly split the data into a train and validation set, such that around one third of the data comprised the validation set. While pre-processing, we removed all hashtags and handles, as well as urls from the text of the Tweets. Using the nltk (Natural Language ToolKit) library[2], we tokenized the Tweet text, and then we removed all English stopwords (Algorithm 1). We did this for both the train set and the development set.

Algorithm 1 Preprocess(x)

(Tweet, label) = x
Tweet = Remove URLs, usernames, characters (Tweet)
Tweet = wordTokenize(Tweet)
Tweet = RemoveStopWords(Tweet)
return (Tweet, label)

For our second and third models, we did the same preprocessing (Algorithm 1) up until the tokenization step but instead of tokenizing, we created a tfidf representation of the Tweets which were used later for Cosine Similarity test and Neural Network training.

[1] https://www.kaggle.com/kapastor/democratvsrepublicantweets
[2] https://www.nltk.org

1.2. Models

We decided to use a Naive Bayes Classifier[3] to perform the binary classification task of labeling political Tweets as either Democrat or Republican (Algorithms 2; 3, 4). We chose this type of classifier because it is a generally efficient model for training and predicting (Nakov, et al.; Vanzo, et al.), which has been known to scale well. Naive Bayes models are effective in predicting probabilistically and focusing only on the most relevant features for making predictions. In particular, such models are frequently utilized for text classification tasks, such as the sentiment analysis tasks which we are performing in this project. We trained nltk's Naive Bayes Classifier model on our train set and predicted on our validation set.

Algorithm 2 BuildVocab(x)

```
allWords = []
for (words, label) in x do
    allWords.extend(words)
end for
list = freqDist(allWords)
return list.keys()
```

Algorithm 3 extractFeatures(x)

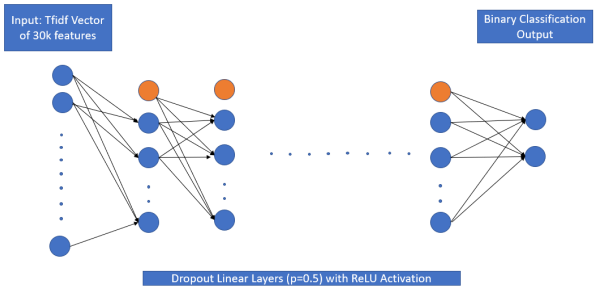
```
TweetWords = set(Tweet)
features = dict()
for (words) in wordFeatures do
    feature["contains" + word] = (word in Tweet)
end for
```

Algorithm 4 Naive Bayes

```
Input: Dataset of size m
Remove all columns except Tweet and label
PreprocessedTweets = []
for i = 0 to m do
    PreprocessedTweets.append(Preprocess(i))
end for
wordFeatures = BuildVocab(PreprocessedTweets)
trainingFeature = nltk.classify.applyFeatures(
extractFeatures, PreprocessedTweets)
nltk.NaiveBayesClassifier.train(training_features)
```

Following Naive Bayes, we also wanted to see how well a Cosine Similarity kernel (Algorithm 5) on a tfidf representation of Tweets would classify. To do this we used sklearn's in built pairwise.cosine_similarity function. Once we had created our tfidf representation we also decided to use that representation to train a Neural Network to see if a learned model could perform better (Ghiassi, et al.) than Cosine Similarity and Naive Bayes. We initially tried to integrate an LSTM layer in our Neural Network as well however due to some technical errors we had to abandon that approach and continue with a simple linear layer model (Algorithm 6). Figure 2 below gives the basic structure of our Neural Network.

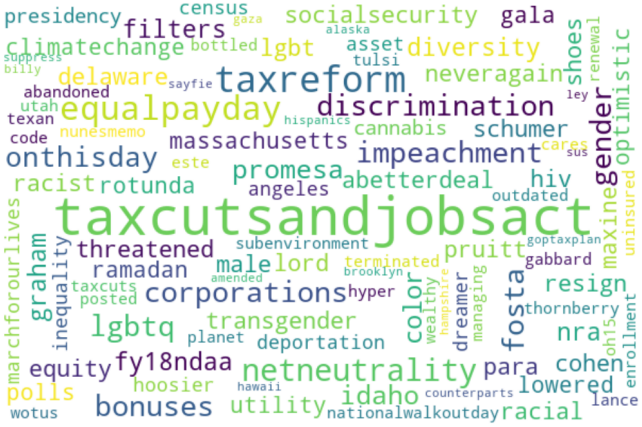
Figure 2. Neural Network Model Architecture



1.3. Results

We made predictions on our validation set of Democrat and Republican Tweets, using all three models developed and have included results below.

Figure 3. Most Important Features for Classification



For the Naive Bayes model, we were able to use nltk's "most informative features" method to extract from our Naive Bayes classifier, the words which occur in Tweets that are most useful in making a binary classification of either Democrat or Republican. Such useful words include "taxreform", "netneutrality", "lgbtq", "corporations", "socialsecurity", "climatechange", "gender", and "impeachment", among others. We were able to represent this data nicely using a word-cloud (Figure 3). In addition, we were able to extract the words which were most frequent in those

[3] https://towardsdatascience.com/creating-the-twitter-sentiment-analysis-\program-in-python-with-naive-bayes-\classification-672e5589a7ed

Tweets which were tagged as Democrat and Republican and representative word-clouds of those (Figures 4 and 5). Ultimately, this model was able to achieve a validation accuracy of 79.75%.

Regarding our second model, Cosine Similarity kernels are known to perform very well in classifying documents with different writing styles (Castellucci, et al.; Karanasou, et al.); unfortunately, both Democrats and Republicans have a tendency to speak in the same tone / manner in political Tweets. Because of this, we were not able to get substantially great results with this approach. We we were able to achieve an accuracy of about 61% which was lower than both our Naive Bayes and Neural Network models.

Finally, for our Neural Network model trained on tfidf representations, we were able to achieve a validation accuracy of 77%. We had hopes of getting a higher accuracy for the NN model however we faced several limitations. The first one was how much RAM Google Colab could allocate to for our use at no cost. Since we had over 1,000,000 different features including words, numbers, names etc creating sparse tfidf vectors for each Tweet took a lot of ram and caused our run time to crash several times. Towards the end, we ended up focusing on just the 30,000 most frequently occurring words to create the tfidf representation which was then passed in to the first layer to the Neural Network.

2. Figures and Algorithms

2.1. Figures

Our first two figures above depict, respectively, a visualization of our project process and a representation of our Neural Network model.

Figure 3 above shows the word-cloud for the most important features used when classifying the Tweets. As we can see “taxcutsandjobsact” and “taxreform”, “discrimination” and “gender” are some of the most important features used in determining the sentiment of the Tweets. Figures 4 and 5 below show word-clouds for the most important frequently-occurring words in Democratic and Republican Tweets respectively.

Figure 6 details information from our Neural Network, showing how the training and testing accuracy changed during training over time. As we can see below, the model learned relatively quickly and reached around 75% accuracy after the first few epochs but then unfortunately plateaued.

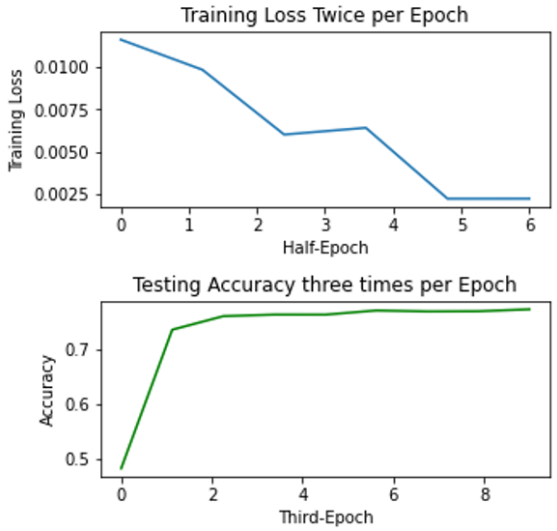
2.2. Algorithms

For the Naive Bayes model, we used Python’s inbuilt nltk library to train on and classify political Tweets. The bulk

Figure 4. Most Frequently Occurring Democratic Words

Figure 5. Most Frequently Occurring Republican Words

Figure 6. Loss and Accuracy



of the work however came from pre-processing the data (Algorithm 1) into a format that could be utilized by the nltk.NaiveBayesClassifier.

For the Cosine kernel and NN models, we had the same initial pre-processing (Algorithm 1) up until word tokenization. From there we utilized different functions to create tfidf representations, primarily sklearn's CountVectorizer and TfidfTransformer classes. For Cosine kernel model we used sklearn.metrics.pairwise.cosine_similarity for predictions (Algorithm 5). Algorithm 6, meanwhile, relates the process of pre-processing and training our Neural Network model.

Algorithm 5 Cosine Similarity Kernel

Input: Dataset of size m
Remove all columns except Tweet and label
PreprocessedTweets = []
for $i = 0$ **to** m **do**
  PreprocessedTweets.append(**Preprocess(i)** sans word Tokenize)
end for
Dem = All Dem Tweets appended together
Rep = All Rep Tweets appended together
Corprus = [Dem, Rep]
X = **CountVectorizer.transform(corpus)**
tfidf = **TfidfTransformer.fit.transform**(X)

Algorithm 6 Neural Network

Input: Dataset of size m
Same as before until prepocess loops (sans word Tockenzie)
Corpus = [PreprocessedTweets]
X = **CountVectorizer.transform**(max.feature = 30k)
tfidf = **TfidfTransformer.fit.transform**(X)
model = **NLPNetwork()**
NLPNetwork.train(tfidf, labels)

3. Trump, McConnell, and Co: A New Generation of Tweeters

Since the entire motivation behind this paper was to see how well a classifier could predict political alignments, our team decided to manually create a small dataset of over 180 recent Tweets from some core Democratic and Republican politicians including President Trump, Senator McConnell, Secretary Pompeo and Senator Graham from the Republican Party and Senator Warren, Governor Cuomo, President Obama and Speaker Pelosi from the Democratic Party. We wanted to see how well each of our models would work when classifying these politicians and the results are shown in Table 1.

Table 1. Classification accuracies for new Tweets

DATA SET	NAIVE	COS-KERNEL	NEURAL NETWORK
OBAMA	**1.00**	0.900	0.950
PELOSI	**1.00**	0.950	0.850
WARREN	**1.00**	0.950	0.800
CUOMO	**0.950**	0.800	0.850
GRAHAM	0.400	0.250	**0.550**
ROMNEY	0.500	0.550	**0.850**
POMPEO	0.523	0.570	**0.809**
MCCONNELL	0.300	0.500	**0.600**
TRUMP	0.286	0.333	**0.428**
DEM	**0.987**	0.900	0.863
GOP (W/ TRUMP)	0.402	0.440	**0.647**
GOP (W/O TRUMP)	0.431	0.467	**0.704**
OVERALL	0.683	0.649	**0.742**

An important part of machine learning is that testing data should come from a similar environment to that of the training data to ensure generalization. Our training data was a series of Tweets compiled up to early 2018 which meant they probably ranged from a few years before 2018. The political landscape changed drastically over short period of time since the election of President Trump, resulting in some words / rhetorical styles becoming obsolete. This is especially relevant as Trump's Tweets are not included in the training data. Many Republicans have now changed their Tweets to mimic President Trump's style of rhetoric, whether that means prioritizing different issues than before (such as trade wars with China and Mexican walls) or just taking more combative tones against their opponents. The Democratic narrative has changed as well, but their style of Tweeting remains largely the same. As a result, all three models performed well on the new Democratic Tweets but misclassified several Tweets from new Republicans. For President Trump, our models had not encountered a Tweeting style like his before, so they more or less could not classify him with any reasonable accuracy.

Our Naive Bayes and Cosine kernel models tended to overpredict Democrats, often getting a 100% accuracy for some Democrats; however, we can see that they failed to generalize successfully to the Republicans with accuracy lower than random guessing. Our Neural Network model performed well on Democrats and managed to get a solid chunk of Republicans correct, particularly those Republicans who Tweet in a more "traditional" style as seen in the training data. None of the models accurately classified Republicans who Tweet in a "Trumpian" style, as the models had not seen such data before. Hence, of all three models, the Neural Network ultimately generalized the best.

4. Extra-Credit:

We also decided to included the extra credit link here in addition to the form:
https://vimeo.com/415699345

5. Citations and References

Castellucci, Giuseppe, et al. "Unitor: Combining syntactic and semantic kernels for twitter sentiment analysis." Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013). 2013.

Elghazaly, Tarek, Amal Mahmoud, and Hesham A. Hefny. "Political sentiment analysis using twitter data." Proceedings of the International Conference on Internet of things and Cloud Computing. 2016.

Ghiassi, Manoochehr, James Skinner, and David Zimbra. "Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial Neural Network." Expert Systems with applications 40.16 (2013): 6266-6282.

Karanasou, Maria, Christos Doulkeridis, and Maria Halkidi. "DsUniPi: An SVM-based approach for sentiment analysis of figurative language on Twitter." Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015). 2015.

Nakov, Preslav, et al. "SemEval-2016 task 4: Sentiment analysis in Twitter." arXiv preprint arXiv:1912.01973 (2019).

Vanzo, Andrea, Danilo Croce, and Roberto Basili. "A context-based model for sentiment analysis in twitter." Proceedings of coling 2014, the 25th international conference on computational linguistics: Technical papers. 2014.

Vishal, A. and Sonawane, S.S. "Sentiment Analysis of Twitter Data: A Survey of Techniques." International Journal of Computer Applications 139.11 (2016): 5–15. Crossref. Web.

The Goal and Models Used

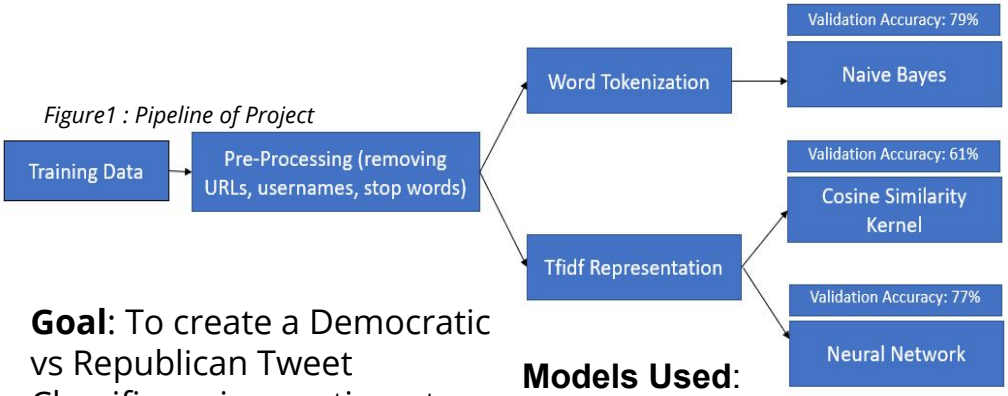


Figure1 : Pipeline of Project

Goal: To create a Democratic vs Republican Tweet Classifier using sentiment analysis

Dataset: Kaggle Democrat Vs. Republican Tweets
- 170,000 instances
- labels as single feature
- tweets as features

Models Used:
1) Naive Bayes
2) Cosine - Similarity Kernel
3) Neural Network

Testing on:
- Validation dataset
- New 2018-2020 (not COVID-19 related), manually collected

Figure 2: Most Important Features for classification in Naive Bayes

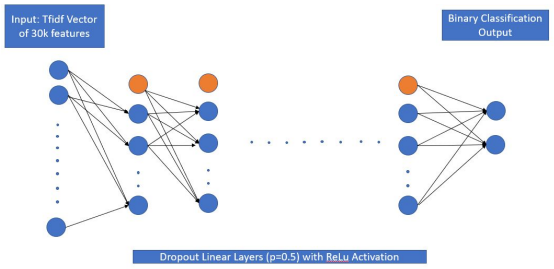


Figure 3: Structure of the neural net

Classification on 2018-20 Data

DEMOCRATS

Overall Accuracy:
Naïve Bayes: 0.987
Cos-Kernel: 0.900
Neural Net: 0.863

Barack Obama
Moderate / Traditional Democrat
Naïve Bayes: 1.00
Cos-Kernel: 0.900
Neural Net: 0.950

Elizabeth Warren
Left-Wing / Non-Traditional Democrat
Naïve Bayes: 1.00
Cos-Kernel: 0.95
Neural Net: 0.800

Andrew Cuomo
Moderate / Traditional Democrat
Naïve Bayes: 0.950
Cos-Kernel: 0.800
Neural Net: 0.850

Nancy Pelosi
Moderate / Traditional Democrat
Naïve Bayes: 1.00
Cos-Kernel: 0.95
Neural Net: 0.85

Liberal Buzzwords:
climate, change, misinformation, LGBT, transgender, Democrats, equality, justice, gender, racism, workers, economic, affordable, American, government, healthcare, jobs, companies, pollution, women, Russia, gun, control, violence, immigrants, sanctions, communities

Models Overall:
Naïve Bayes: 0.683
Cos-Kernel: 0.649
Neural Net: 0.742

Findings:
All models tended to overpredict Democrat, though the neural net generalized best. The rhetoric and style of tweeting of Democrats has not generally changed since the time when the training data was sampled; on the other hand, there is a clear divide on the Republican side, with the neural net performing comparatively to Democrats on some Republicans, while others seemed to be reduced to guesswork. We believe this to be because since Trump's election, the rhetoric and style of tweeting of some Republicans has changed drastically. We notice that the Republicans who had high accuracy tweeted in the more "traditional" Republican style, while those whose accuracies were low took on more "Trumpian" language over time.

REPUBLICANS

Overall Accuracy: (w/ Trump)
Naïve Bayes: 0.402
Cos-Kernel: 0.440
Neural Net: 0.647

Conservative Buzzwords:
war, power, emergency, economy, Republicans, CEOs, corporations, corruption, strengthen, companies, economic, America, socialist, attacks, defend, military, jobs, industry, China, immigration, guns, immorality, Syria, law, allies, private, sector, Israel, ISIS, security, administration, criminals

Mitt Romney
Moderate / Traditional Republican
Naïve Bayes: 0.500
Cos-Kernel: 0.550
Neural Net: 0.85

Mitch McConnell
Right-Wing / Non-Traditional Republican
Naïve Bayes: 0.300
Cos-Kernel: 0.500
Neural Net: 0.600

Mike Pompeo
Right-Wing / Non-Traditional Republican
Naïve Bayes: 0.523
Cos-Kernel: 0.570
Neural Net: 0.809

Lindsey Graham
Moderate / Traditional Republican
Naïve Bayes: 0.400
Cos-Kernel: 0.250
Neural Net: 0.550

Overall Accuracy: (w/o Trump)
Naïve Bayes: 0.431
Cos-Kernel: 0.467
Neural Net: 0.704

The Trump Effect:
We believe that models performed so poorly on Trump's tweets because of how vastly they differed from typical Republican training tweets. Models were reduced to guessing in this case.

Right-Wing / Non-Traditional Republican
Naïve Bayes: 0.286
Cos-Kernel: 0.333
Neural Net: 0.428