Political Polarization in the Time of Twitter

Pranali Rekhawar
prekhawar@ucsd.edu

Abstract

Social Media has become extremely influential in engaging voters. With the 2008 and 2016 US elections, it became evident that social media is a tool that politicians use extensively. In this paper, we analyse the twitter feed of the 115th Congress and try to distinguish between a tweet by a Republican senator and a Democrat Senator. We utilize various embedding and classification techniques to fulfil this task and achieve over 70% accuracy across all models. We observe that the two parties have a clear distinction in their messaging and this reflects in the tweets by the senators.

1 Introduction

Barack Obama was the first presidential candidate in US history to use Internet to reach out voters at a grass-root level. With the launch of MyBarackObama.com, he was able to reach to a wider audience and involve them in his campaign. This saw him tremendous success.

In the 2016 elections, Donald Trump popularized the use of social media, especially Twitter and Facebook, to disseminate campaign information, engage with voters and most importantly as a voice of attack. This saw him an unexpected success that no polling, even a day prior to election, was able to capture. Given the success of Mr. Donald Trump, we can infer that social media is here to stay as a vital tool in political strategies.

We thus analyse the twitter feed of senators of the 115th Congress to analyse how are they reaching out to their voters. We analyse tweets from the heavyhitter senators of each party and try to develop a model to predict if the tweet was by a Democratic senator or a Republican senator. We try to understand what do Democrats talk about? and What do Republicans talk about? Can we use a tweet to determine if the user is a Republican or a Democrat?

2 Exploratory Data Analysis

The Tweets data from the Harvard Dataverse (Harvad dataverse, 2018) contains tweets by US Senators of the 115th Congress. The 115th Congress' members and their respective Twitter accounts have been compared with https://www.socialseer.com/resources/us-senator-twitter-accounts/ (accessed September 27, 2018). Two account names have been substituted: SenatorSanders: SenSanders, SenBennetCO: SenatorBennet. The tweets have been scraped in .json until February 11, 2019. The tweets' plain text has been rid of http links and concatenated per user until September 26, 2018 and saved in .txt format. and split into separate folders 'Republican', 'Democrat', and 'Independent'. The picture links from the tweets have not been removed. The tweets have been scraped with the Python library: https://github.com/taspinar/twitterscraper.

The data contains tweets from 47 Democrat Senators and 51 Republican Senators. The information available for the tweets by senators are:

1. html: The original html tag for the tweet
2. fullname: The full name for account
3. user: The user id of the account
4. id: The id of the tweet
5. text: The processed tweet
6. likes: The likes on the tweet
7. replies: The number of replies on the tweet
8. retweets: The retweets for the tweet
9. timestamp: The timestamp the tweet was posted

We add an additional flag, "party" to signify the Party to which the senators belong to.

In total, there are 79,409 tweets in the data, with 40,747 tweets tagged as tweets from Republican Senators and 38,662 tweets tagged as tweets from Democrat senators. We focus only on the text column for further analysis.

Following steps have been used to process the text column:

- Only the http links have been removed from the tweets. Thus, we first separate out the hashtags from the tweet using simple regex and save it in a separate column.
- We separate multi word hashtags into its components using the Segmenter from the ekphrasis package
- We clean the text of mentions, picture links, hashtags, etc. using the twitter-preprocessor package. Thus now we are only left with the relevant segments of the tweet
- We remove stopwords from the tweets to only keep relevant words. We use a combined stopword set from NLTK and WordCloud libraries. We further update our stopwords with words: today, tomorrow, yesterday, also, like, https, and, icymi, of which today, also and like has heavy usage by senators from both the parties but do not significant value to the tweets
- We remove any digits from the tweet and convert it to a lowercase for easier processing
- We further lemmanize the text to reach each word's root form and remove any punctuations from the data
- We remove tweets that have text length less than or equal to 5

Using the above steps we now have a processed version of the tweet and the hashtags in its base form. This processes tweets can be used for further analysis. We further need to understand if the hashtags used by the senators are relevant for our purpose and if it should be included or not in the original tweet. To do so, we run the hashtags through a similar pipeline of pre-processing and we generate a wordcloud based on the frequencies of the words in the segmented hashtag column to understand its importance.

Based on the wordclouds generated from the hashtags (Figures 1 and 2), we see a distinct difference between the hashtags used by Democrats and Republican senators. The Republican party are talking more about their policies, like Tax Reform, Tax cuts and Job Act whereas we see Democratic senators in two modes, attack and social policies. We see them talk about trump shutdown and gop tax scam. Issues like Net Neutrality also has a high frequency. We can assume this was because the Republican party, at that time, controlled both the

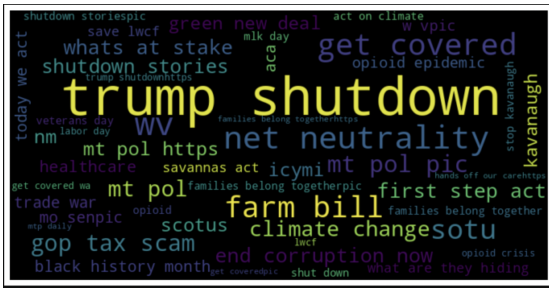

Figure 1: Wordcloud of hashtags used by Democratic Senators

Figure 2: Wordcloud of hashtags used by Republican Senators

Presidency and Chamber of Congress and therefore were able to implement their agenda relatively unrestrained. Thus we decide to combine the processed tweet data and the processed hashtag data back together.

We also tried using NERs to tag persons and organisations in the dataset as a single entity instead of a bigram (Ex. The National Chemical Laboratory becomes the_national_chemical_laboratory). However, due to the nature of the tweets, extremely small number of tweets were relevant and we do not believe it added any additional value to our text pre-processing pipeline.

3 Predictive Task

The important task with this dataset is to predict whether a given tweet was written by a Democrat or not, or by a Republican or not. At it's core, this is a simple classification task. We generate predictions for each class separately and thus treat it as a multi-class classification problem.

Evaluation Metrics:

Accuracy: Accuracy is the proportion of true results predicted among the total number of cases examined. Given that our dataset is a balanced dataset, accuracy is the quintessential metric of comparison to test model performance on new data.

AUC ROC Curve: AUC ROC curve measures how well the probabilites from the positive class are separated fromt the negative class. We plot the True Positive Rate against the False Positive Rate at various thresholds to generate the ROC curve. AUC is scale invariant as it measure how well predictions are ranked rather than their absolute values. In simple terms, the AUC indicates the probability that the classifier will rank a randomly chosen positive observation higher than a randomly chosen negative one.

4 Text Classification

We compare the old school approach of Bag of Words with a simple machine learning model with the popular Word Embeddings with a deep learning neural networks and the state of the art Language Models used with transfer learning from attention based models. (Pietro)

4.1 Bag-of-Words

The Bag of Words model is simple. It builds a vocabulary from a corpus of documents and counts how many times the word appears in each document. To put in another way, each word in the document becomes a feature and a document is represented by a vector with the same length of the vocabulary. However, this approach causes a significant dimensionality problem. The larger the number of documents that you have, the larger will be the vocabulary and larger will be the feature matrix. Thus, preprocessing steps like word cleaning, stop word removal, stemming/lemmatization are necessary when we use a bag of words model.

Term frequency however is not necessarily the best representation of the text. In fact, words with the highest frequency often have very little predictive power over the target variable. To address this problem we use the advanced variant of the Bag of Words, that, instead of simple counting uses the **term frequency - inverse document frequency**. Essentially, the value of the word increases with count but is inversely proportional to the frequency of words in the document.

We first set up the Tf-Idf vectorizer with a 20,000 words limit capturing unigrams and bigrams. We fit the vectorizer on the corpus created from the processed tweets. We further transform the fitted vecotrizer to form the input for our classification model.

The chosen model for the classification task is

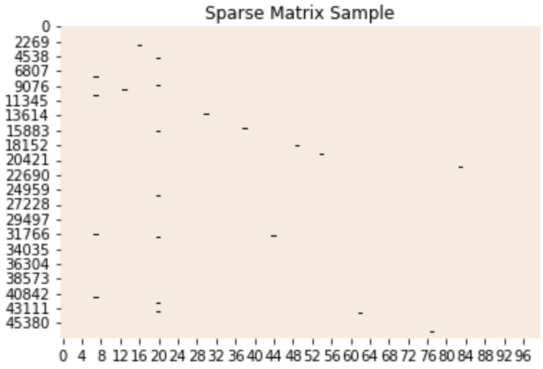

Figure 3: Random sample from feature matrix (non zero values in black)

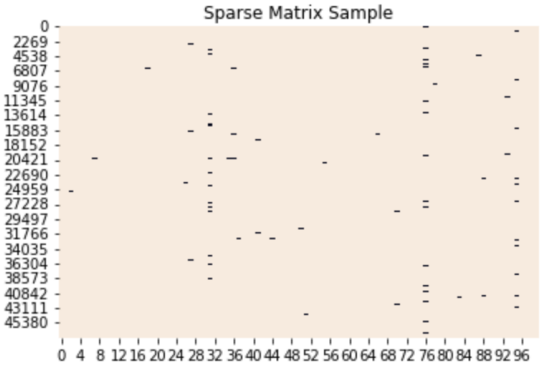

Figure 4: Random sample from reduced feature matrix (non zero values in black)

the Naive Bayes Algorithm, a probailistic classifier that uses probability to make predictions based on prior knowledge of conditions that might be related. This algorithm is most suitable for such a large database as it considers each feature independently, calculates probability of each category and then predicts the category with the highest probability. This model gives an accuracy of 0.8 and an AUC of 0.89.

The X_train (input to the model) is of the shape 74235 x 20000 and is pretty sparse. In order to drop some columns and reduce the matrix dimensionality, we perform Chi-Square test to determine whether the feature and the target are independent and keep only features with a p-value above 0.9. Using this methodology we were able to reduce the number of features from 20,000 to 4,343. Refer to Figures 3 and 4.

Re-running the Naive Bayes Classifier on the reduced feature set gives us an accuracy of 0.79 and an AUC of 0.88.

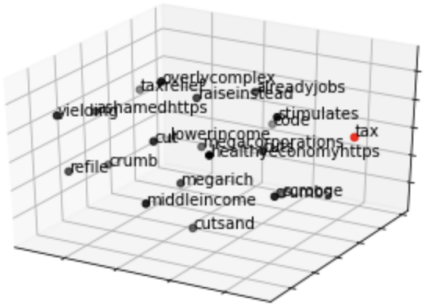

Figure 5: Word "tax" and its context - as trained on W2V Skip Gram

4.2 Word Embedding

Word Embedding is the collective name for feature learning techniques where words from the vocabulary are mapped to vectors of real numbers. These vectors are calculated from the probability distribution for each word appearing before or after another. To put it another way, words of the same context usually appear together in the corpus, so they will be close in the vector space as well.

We are using the Word2Vec by Google (Mikolov and Dean, a). We opted out of using a pre-trained model and are instead training the Word2Vec model on our corpus. We are using skip-gram, which uses the approach of predicting the context from the given word.

Before creating the model, we first transform our data into a list of n-grams. We try to capture unigrams, bigrams and trigrams. For the WordVec, we are using target size of word vectors as 300, for the window, i.e., the maximum distance between the current and predicted word, I'm using half the average length of the text, 8. For training the algorithm, I'm using skip-gram. Refer to Figure 5.

We use the vectors as weights for the neural network. We first transform the corpus into a padded sequence of word ids to get a feature matrix. We then create a embedding matrix so that the vector of the word with id N is located at the Nth row. We finally build a network with an embedding layer that weighs every word in the sequences with the corresponding vector.

We create the network with following structure:

1. Embedding layer: The Embedding layer takes the sequences as input and the word vectors as weights
2. Attention layer: We add this layer as it captures the weights of each instance and allows us to build an explainer (Dzmitry Bahdanau and Bengio)
3. Bidirectional LSTMs: We add two layers of Bidirectional LSTMs to model the order of words in a sequence in both direction
4. Dense Layer: We add two final dense layers that will predict the probability of each category

We check the performance of the model on a validation subset before testing it on the actual test set. The maximum accuracy reached during validation during some epochs is 0.77. We complete the evaluation of our Word Embedding model by predicting on the test set and comparing for the same metrics. On the test set we get an accuracy of 0.71 and an AUC of 0.81.

4.3 Language Models

Language Models, or Dynamic Word Embeddings, overcome the biggest limitation of the classic Word Embedding approach: polysemy disambiguation, a word with different meanings is identified by just one vector. This was initially made popular by ELMO (Peters and Zettlemoyer) which doesnt apply a fixed embedding but using a bidirectional LSTM, looks at the entire sentence and then assigns an embedding to each word.

In the 2017 paper, Attention is all you need (Vaswani and Polosukhin), it was demonstrated that sequence models like LSTM can be completely replaced by Attention Mechanisms.

Google's BERT (Devlin and Toutanova) continues the trend of naming models after cartoon characters, and combines ELMO context embedding and several Transformers, plus it's bidirectional. The vector BERT assigns to a word is a function of the entire sentence, therefore, words can have different vectors based on the context.

We are using Distil-BERT (Huggingface.co), a lighter version of BERT, pre-trained on 66 million parameters instead of 110 parameters, to fine tune. We need to pre-process our data in a format that is expected by the Distil-BERT. To this, we add a [CLS] token to signify the beginning of the text, [SEP] to signify the ending of the sentence and [PAD] to bring all the inputs to same length. We finally put all together in a tensor to get the feature matrix that will have the shape 2 (ids, masks) x Number of documents in the corpus. We apply the same pre-processing to the test dataset.

To apply transfer learning to a pre-trained BERT

model, we essentially summarize the output of BERT into one vector with Average Pooling and then add two final Dense layers to predict the probability of each category.

We check the performance of the model on a validation subset before testing it on the actual test set. The validation accuracy for the model is 0.7029. We complete the evaluation of our Language Model by predicting on the test set and comparing for the same metrics. On the test set we get an accuracy of 0.71 and an AUC of 0.8.

5 Literature Survey

The paper Distributed Representation of Words and Phrases and their Compositionality (Mikolov and Dean, b) introduces extensions to the skip-gram model that improve both the quality of the vectors and the training speed. The main two methods introduced were, the subsampling of the frequent words, through which they obtained significant speedup and more regular word representations and an alternative to hierarchical softmax called negative sampling. Using the HS-Huffman, they were able to achieve an accuracy of 47% with subsampling compared to the accuracy of 19% with no subsampling.

Language model pretraining has led to significant performance gains but careful comparison between different approaches is challenging. Thus, this paper, RoBERTa: A Robustly Optimized BERT Pretraining Approach (Yinhan L.), presents a replication study of BERT pretraining (Devlin and Toutanova) that carefully measures the impact of many key hyperparam- eters and training data size. They found that BERT was significantly undertrained, and can match or exceed the performance of every model published after it. These results highlight the importance of previously overlooked design choices.

Social media has become extremely influential when it comes to policy making in modern societies especially in the western world. In the recent (February 2022) paper, Politics and Virality in the Time of Twitter: A Large-Scale Cross-Party Sentiment Analysis in Greece, Spain and United Kingdom (Dimosthenis A.), they attempt to analyse tweets from politicians from different countries and explore if their tweets follow the same trend. Utilising state-of-the-art pre-trained language models they performed sentiment analysis on multilingual tweets collected from members of parliament of Greece, Spain and United Kingdom, including devolved administrations. The analysis indicates that politicians' negatively charged tweets spread more widely, especially in more recent times, and highlights interesting trends in the intersection of sentiment and popularity.

Model	Accuracy	AUC	F1- Dem	F1- Rep
TF-IDF	0.79	0.88	0.79	0.79
WordVec w/ NN	0.71	0.81	0.65	0.75
BERT	0.71	0.8	0.65	0.75

Table 1: Result - Model Comparison

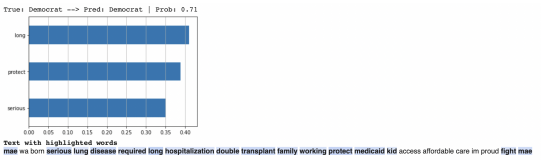

Figure 6: Explainability - Word Vec

6 Results and Conclusion

The table summarizes the results for all the models explained in the Text Classification section. We see that the simplest model, a TF-IDF encoding with a Naive Bayes classifier performs the best and is also more easily able to separate between the Democrat and Republican tweets.

From the explanability figures, Figures 6 and 7, we observe that while Word Vec does capture more words due to how it was trained, the words with highest probability are long, protect, serious. Of which only protect seems to be relevant to the context. However, the TF-IDF model captures words seemingly more accuratly, assigning high proability to words like medicaid and affordable. Despite assigning transplant to Republican label.

While the models have not been fine tuned to the desired level, it is quite revealing that even among the basic version of the models, the Naive Bayes model with a TF-IDF encoding outperforms others. I believe this is partially because of the exhaustive text pre-processing pipeline which might've removed important contextual information.

We find that even a basic model is able to dis-

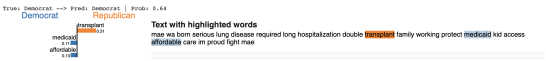

Figure 7: Explainability - TF-IDF

tinguish between the Democrat and Republican tweets with over 70% accuracy and further scope for improvement. This suggests that the polarization between the political parties in US is wide enough for a clear distinction.

References

Chang M. Lee K. Devlin, J. and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding.

Jose C. C. Dimosthenis A., Alun P. Politics and virality in the time of twitter: A large-scale cross-party sentiment analysis in greece, spain and united kingdom.

Kyunghyun Cho Dzmitry Bahdanau and Yoshua Bengio. Neural machine translation by jointly learning to align and translate.

Harvad dataverse. 2018. *Tweets - US Senators of 115th Congress until Sept 2018.*

Huggingface.co. Distilbert.

Chen K. Corrado G. Mikolov, T. and J. Dean. a. Efficient estimation of word representations in vector space.

Sutskever I. Chen K. Corrado G. Mikolov, T. and J. Dean. b. Distributed representations of words and phrases and their compositionality.

Neumann M. Iyyer M. Gardner M. Clark C. Lee K. Peters, M. and L. Zettlemoyer. Deep contextualized word representations.

Mauro Di Pietro. Text classification with nlp: Tf-idf vs word2vec vs bert.

Shazeer N. Parmar N. Uszkoreit J. Jones L. Gomez A. Kaiser L. Vaswani, A. and I. Polosukhin. Attention is all you need.

Naman G. Jingfei D. Mandar J. Danqi C. Omer L. Mike L. Luke Z. Veselin S. Yinhan L., Myle o. Roberta: A robustly optimized bert pretraining approach.