

Red vs Blue: Classifying Tweets as Republican or Democratic in American Politics



Zach Sickles

15 min read · May 8



Introduction

Since its launch in 2006, Twitter has remained at the forefront of online national political discourse. Be it raising awareness for social justice movements, spreading campaign messages from politicians, or coordinating some collective action, Twitter is a platform that connects leaders of all political parties to the people who most want to hear their message. However, it is a website that can be overrun with misinformation, tainted by censorship, and overall difficult to navigate and recognize trustworthy sources. Political “siloeing” on Twitter, where individuals consume media online only from one particular political perspective, is also of particular concern in the context of an increasingly polarized American political climate. In order to avoid this, I set out to create a Tweet classifier for American Twitter users to divide up the messages on their feeds into one of the two major parties in the United States: Republican or Democrat.

With this classifier, we can work towards harnessing artificial intelligence in order to produce a better online political ecosystem. Using this type of tweet classification system, users would be able to regulate their political online diet in order to consume content from multiple perspectives. The classifier would also be useful to tag political content itself in order to present the user with more information regarding the perspective of particular political actors. In a more academic setting, the classifiers can also be used as a tool to analyze the different ways that each modern American party utilizes language in online partisan messaging.

Overall I created four classification models — a simple classification, one which utilizes BERT transfer learning, a bag of words (BoW) model, and finally a model using the NLP technique Word2vec. I structured the models to highlight the language differences between the opposing political viewpoints, and through data analysis, I was also able to determine interesting patterns in political tweets. As a bonus, I created a generative model for tweets that reflect both Republican and Democrat stylings in order to best understand the linguistic features used by those who craft political tweets from either party. This can help explain the language aspects which make their content so compelling and illustrate what allows these people to effectively influence elections and produce polarizing political propaganda in the real world. Additionally, it is a preview into a future where artificial intelligence can be used by malicious and sincere political actors alike to influence voters through generated online political content.

Data Collection and Cleaning

I chose to look at the most influential Twitter political commentators by follower count. The government marketing agency Amra & Elma compiled a list of the fifty largest political influencers in American national politics in 2023. I decided to choose ten political commentators from both parties. It

must be noted that I cleaned the data, removing the accounts of commentators who did not clearly identify with either major political party and those who have switched parties within the past decade. In order to collect the data for the model, I used Snscape, a scraper for social media platforms, to gather the past 2,000 Tweets — from before April 12, 2023 — from each account. The list of popular commentators, split up into Republicans and Democrats, whose Tweets provided the data for the model are as follows:

Q Search this file...		
1	Republicans (Follower counts from April 2023):	Democrats (Follower counts from April 2023):
2	Tucker Carlson @TuckerCarlson (5.8M)	Rachel Maddow @maddow (10.4M)
3	Ben Shapiro @benshapiro (5.5M)	Jake Tapper @jaketapper (3.1M)
4	Laura Ingraham @IngrahamAngle (4.5M)	Charlamagne Tha God @cthagod (2.1M)
5	Candace Owens @RealCandaceO (3.5M)	Chuck Todd @chucktodd (2.0M)
6	Bill O'Reilly @BillOReilly (3.2M)	Joy-Ann Reid @JoyAnnReid (2.1M)
7	Jeanine Pirro @JudgeJeanine (3.1M)	Glenn Greenwald @ggreenwald (2.0M)
8	Tomi Lahren @TomiLahren (2.3M)	Jon Favreau @jonfavs (1.4M)
9	Ann Coulter @AnnCoulter (2.1M)	Mehdi Raza Hasan @mehdirhasan (1.2M)
10	Steven Crowder @scrowder (2.0M)	Nikole Hannah-Jones @nhannahjones (680.3K)
11	Scott Pressler @ScottPresler (1.3M)	Reverend Al Sharpton @TheRevAl (660.7K)

twitter_accounts.csv hosted with ❤ by GitHub [view raw](#)

For the first attempt, I created a simple classification model using the [keras](#) library in the form of a sequential neural network with additional layers. First I employed the keras text preprocessing tokenizer. In this tokenizer, all punctuation is removed by default, and then each tweet is turned into space-separated sequences of words, which are then converted to a list of tokens to be indexed or vectorized. In this model, I chose to meaningfully tokenize the most used 3,000 words in the dataset, ignoring the rest. The tweets were therefore first converted to a sequence of integers, where each integer was the index of a token in a dictionary, then to a one-hot matrix of the 3,000 most-used words in the dataset. I created a simple model with two output

nodes for the binary classification of data to Republican or Democrat and then trained the model for five epochs.

```
model = Sequential()  
model.add(Dense(512, input_shape=(max_words,), activation='relu'))  
model.add(Dropout(0.5))  
model.add(Dense(256, activation='sigmoid'))  
model.add(Dropout(0.5))  
model.add(Dense(2, activation='softmax'))
```

The simple classification model.

While this particular model was able to receive a high training accuracy of .94%, there was clearly overfitting of the data with a testing accuracy of 51%. The baseline accuracy was 50% with an even number of Republican and Democratic tweets, so this was effectively meaningless. While the difference in the training and testing accuracy suggests overfitting is a major concern, adding dropout layers and simplifying the model by changing the number and size of layers did not improve the results. Similarly, changing other parameters, such as the number of epochs, did not improve our testing accuracy. As a result, I decided that other models needed to be tried to improve our accuracy.

Improving the model with BERT Transfer Learning

With the weak results of the simple model, I wanted to investigate the possibility of using transfer learning in my model. Transfer learning is a machine learning technique where a model is pre-trained using a significant amount of computing resources and data on a general task, and then a few additional layers can be trained to fit that general model to a particular task.

I chose to use BERT — Bidirectional Encoder Representations from Transformers — for the pre-trained model. BERT is based on the Transformer model architecture, instead of LSTMs. This means at each decoding step, the decoder gets to look at any particular state of the encoder. BERT captures the semantic relationship between words by generating representations for the words in a contextual way. Specifically, in this case I used the model BERT-Base Un-Cased: 12-layer, 768-hidden-nodes, 12-attention-heads, 110M parameters. This model is pre-trained using unlabeled text by jointly conditioning on both the left and right contexts. The uncased model was chosen under the assumption that casing would not play a meaningful role in the partisan affiliation of tweets. While Trump is famous for his all capital tweets, it seemed unlikely to play a role for commentators. To fine-tune Bert, I chose just to use a single additional classification layer before the output layer. This was because with the complexity of BERT, a simple binary classification should not take multiple additional layers.

```
class BERT_Arch(nn.Module):

    def __init__(self, bert):
        super(BERT_Arch, self).__init__()

        self.bert = bert

        # dropout layer
        self.dropout = nn.Dropout(0.1)

        # relu activation function
        self.relu = nn.ReLU()

        # dense layer 1
        self.fc1 = nn.Linear(768, 512)

        # dense layer 2 (Output layer)
        self.fc2 = nn.Linear(512, 2)

        #softmax activation function
        self.softmax = nn.LogSoftmax(dim=1)

    #define the forward pass
    def forward(self, sent_id, mask):

        #pass the inputs to the model
        _, cls_hs = self.bert(sent_id, attention_mask=mask,
return_dict=False)

        x = self.fc1(cls_hs)

        x = self.relu(x)
```

```
x = self.dropout(x)

# output layer
x = self.fc2(x)

# apply softmax activation
x = self.softmax(x)
```

The BERT Model architecture used, courtesy of [kaggle tutorial](#).

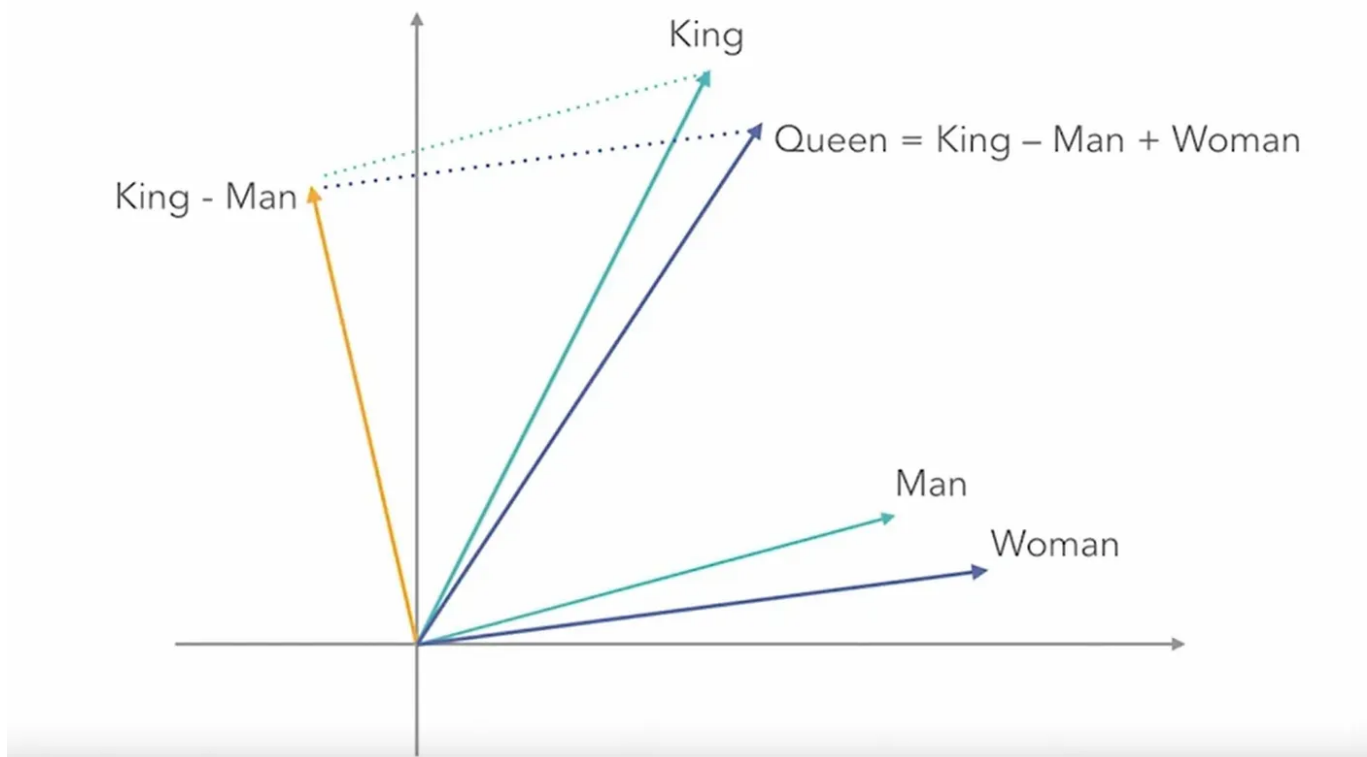
To create the model, I used the [transformers](#) and the [sklearn](#) libraries. For tokenization of the data I used the “BertTokenizerFast” from HuggingFace, which my research recommended and had built-in masking and tokenization features that fit the pre-trained model. It was also chosen to increase to 20 Training Epochs in order to maximize the accuracy of the model on the training data. After training on the data, the model was able to predict the correct political party at a more accurate weighted rate of 71 percent.

	precision	recall	f1-score	support
0	0.73	0.65	0.69	2450
1	0.69	0.76	0.73	2525
accuracy			0.71	4975
macro avg	0.71	0.71	0.71	4975
weighted avg	0.71	0.71	0.71	4975

Word2Vec Model

I continued to investigate the role of using other strategies for Tweet classification, venturing into Word2Vec. Word2Vec is a strategy that utilizes word embeddings where the words in the training sequences are mapped to

vectors in a multidimensional vector space. Through this method, the model captures the meanings of words in both semantic and syntactic terms in relation to the other words in the created dictionary. Words that are mapped closer to each other in the vector space are also meant to semantically refer to similar words. Additionally, logic can be applied such as that the vector for Queen is the same as the vector for King, minus the vector for man, plus the vector for woman. See the visual below:



The model uses the gensim library in order to quickly map the vocabulary to vectors in the vector space. A relatively small vector space of 100 was chosen in order to reduce the necessary computation in the model. A minimum count of 2 was chosen, meaning that all words that appeared at least twice were embedded as a vector. This was done because of the large amount of unique political vocab in the dataset that I did not want to lose in the computation. The default window for gensim of 5 was used in order to

define the amount of context for each word's meaning in the embedding process.

```
import gensim

# Train the word2vec model
w2v_model = gensim.models.Word2Vec(train_df["Tweet"],
                                    vector_size=100,
                                    window=5,
                                    min_count=2)
```

Training the Word2Vec model on the Tweet data.

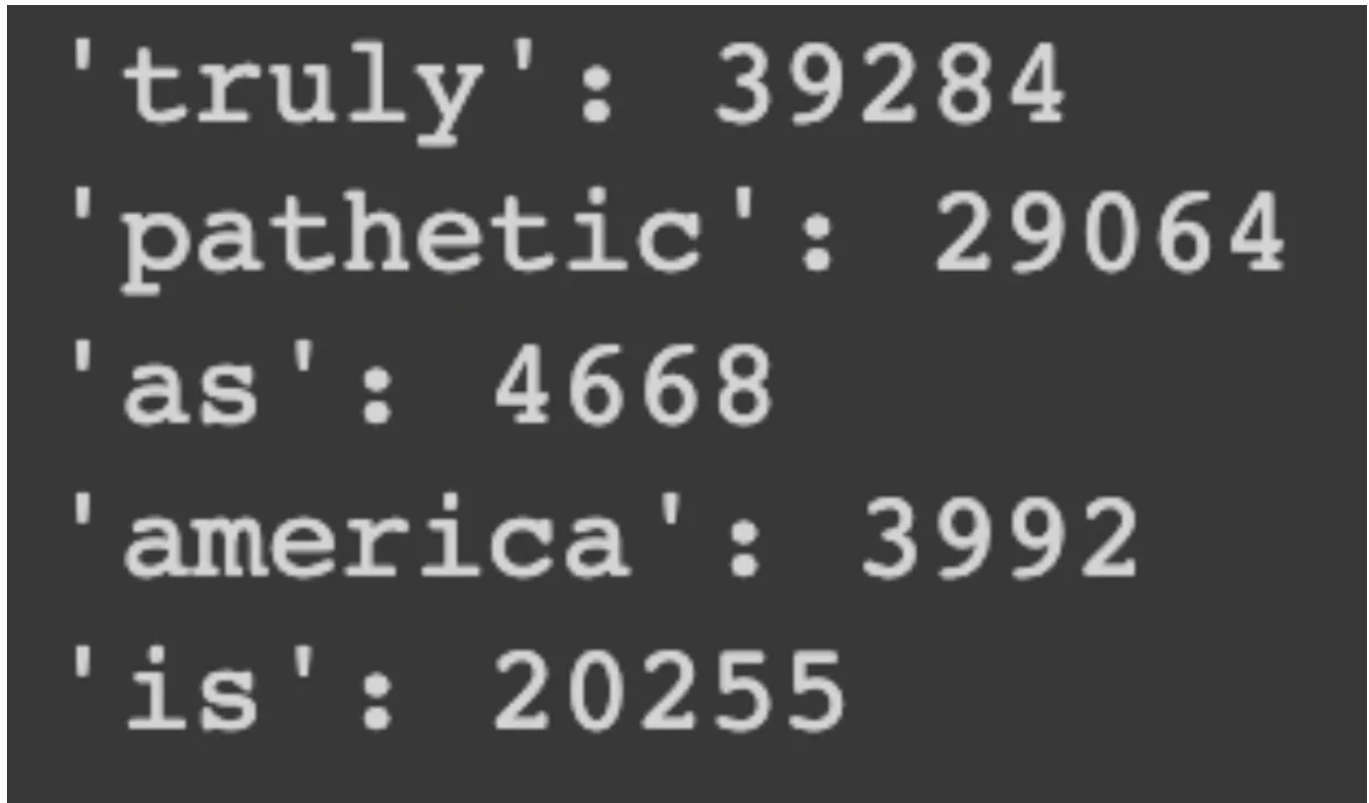
I followed Dilip Valeti's [tutorial](#) for classification using the Word2Vec model. After the embedding process, I generated aggregated vectors for each Tweet based on the word vectors embedded in each one. The average of the vectors in each tweet was taken in order to gain a sense of the overall meaning of the tweet within the vector space.

I then fit a basic Random Forest model from the [sklearn](#) library on top of the vectors. This was done in order to quickly train a model to classify based on the vectorized meaning of the tweets. I tested the model and achieved a 73 percent accuracy — an improvement from the previous two attempts.

```
Precision: 0.737 / Recall: 0.73 / Accuracy: 0.731
```

Bag of Words Model

With an increase in success driven by a greater focus on individual vocabulary from the word2vec model, I employed the use of a bag-of-words model. A bag-of-words model disregards sentence structure and instead treats the input text as a semantic mix of words where the original order has no significance. Instead, the model creates a frequency-based analysis of the words found in the input in order to classify the relationship between them.



```
'truly': 39284  
'pathetic': 29064  
'as': 4668  
'america': 3992  
'is': 20255
```

An example of the tokenized words.

The bag of words model first tokenizes the input training text and creates a fixed vocabulary of all the unique words in alphabetical order. Then, the model generates feature vectors for each word that represents the frequency of its appearance in a tweet. Due to a limit on the amount of RAM that could be used on the platform I performed the computations on, CoLab, I had to reduce the size of the training data set. This reduced the size of the bag of words by limiting the amount of text the model could be trained on. I used the [sklearn](#) library to create a logistic regression where the dependent

variables consisted of these feature vectors, and the output was the binary classification of either Republican or Democrat. The resulting accuracy of the model was the highest yet at 81.9%

Model Comparisons and Analysis

The final implementation, the bag of words model, produced the best accuracy. I conducted an analysis of this model to ascertain and better understand why this is the case. First, I generated a dictionary that contains each word and the number each one appears amongst the data for either political party. The resulting top 10 most commonly used words, including links, hashtags and mentions, for either party can be seen here:

Total Word Count Overall for Either Party

Q Search this file...				
1	Word (Democrat)	Number of Uses (Democrat)	Word (Republican)	Number of Uses (Republican)
2	the	15943	the	16507
3	to	12044	to	11749
4	and	9088	a	7243
5	of	7760	of	6926
6	a	6775	and	6832
7	in	5799	is	6385
8	on	5345	on	5752
9	is	4442	in	5308
10	for	4300	for	4163
11	that	3607	that	3479

word_count.csv hosted with ❤ by GitHub [view raw](#)

This particular look at the data makes it difficult to discern the difference in vocabulary between the two parties because of the high usage of generic words. This in part explains the possible difficulty of the Word2vec model, where the average vector of a tweet was likely highly influenced by this type

of generic language, reducing the differences between average Tweet vectors. In order to take a look at the difference in the vocabulary that was used by each party, I also generated the following dictionary that described the difference in use:

Highest Differences in Word Usage by Either Party

Q Search this file...				
1	Word (Democrat)	Difference in Total Use (Democrat)	Word (Republican)	Difference in Total Use (Re
2	and	2256	is	1943
3	@breakfastclub	993	are	1007
4	I	966	The	966
5	of	834	@FoxNews	951
6	this	809	#Tucker	858
7	here	807	Biden	858
8	us	777	https://t.co/rryWmyXe7C.	685
9	stream	680	Thank	585
10	from	593	the	564
11	#MTP	574	Please	557

highest_difference.csv hosted with ❤️ by GitHub [view raw](#)

In this data we still see that many generic words such as “I”, “is” and “the” still remain in the data set as they may be used differently by either party to such an extent merely due to their overall usage amount and normal probability distributions. However, we can quickly see a major difference between the vocabulary used by one party from the another. For example, the Republican party is more likely to refer to Fox News and Biden in their reporting. Another way to get a more visual reference for this data is the following word cloud:

be seen in political movements as well, with #100RacistThings being used 85 times by only the Republican Party. As a result, due to this feature of the data, it becomes clear why a bag of words model may be particularly successful. In a Word2Vec model, #MorningJoe and #Tucker would have very similar vectors, both as describing political commentary television shows on cable news. Similarly, in a BERT model a tweet advertising to watch a political party show is going to have very similar contextual meaning around the name of the show itself, making it hard to distinguish. However, in a Bag of Words model, the use of any of these words would immediately signify the political party, as there is a clear delineation between the two parties in how often they use these words. Now, some words are used by both parties but to significantly varying degrees. In order to take a look at this data, I have also generated the following data which represents the ratio of how often a word was used in one party compared to the other. That is, the use in one party divided by the other. The word which had the biggest difference was that the word “Reach” was used 252 times more often by the Democratic Party than by the Republican party. The top values from this dictionary can be seen here:

Highest Ratios of Total Word Use by One Party more than the Other

Q Search this file...				
1	Word (Democratic)	Ratio (Democratic)	Word (Republican)	Ratio (Republican)
2	Reach	252.0	@FoxNews	238.75
3	us!!!	164.0	Ingraham	198.0
4	Veterans	129.5	4th	128.0
5	Sharpton	122.5	Show	92.5
6	Visit	121.0	joining!	76.0
7	@MSNBC	99.0	address?	72.0
8	stations	94.0	Left	69.333
9	wounded	85.5	KILLING	67.0
10	532	84.0	@seanhannity	65.0

highest_ratios.csv hosted with ❤ by GitHub [view raw](#)

This data presents somewhat similar findings to the data on usage for only one party or another. For example, a significant number of mentions are in the list, with the Republican party being 196 times more likely to refer to Ingraham, and the Democratic party being 122 times more likely to refer to Sharpton. Both of these individuals may be referred to be either party, but each side is significantly more likely to be talking about or with someone they agree with politically. Additionally, this dataset brings out some of the differences in the vocabulary used simply based on the substantive political differences. For example, the Republican party is 69 times more likely to discuss the “Left”. An interesting result was that the Democratic party used the word “veterans” 129 times more and “wounded” 85 times more. The dataset included many tweets that occurred following the passing of the PACT act, where President Biden signed into law a bill that gave greater access to healthcare for veterans exposed to burn pits and Agent Orange. As a result, it makes sense in this context that Democrats were significantly more likely to discuss veterans than Republicans within the given timeframe.

This brings up areas for improvement for the models. While Democrats were using the words veterans more often during the timeframe the data was collected, this was an isolated moment that does not describe any concrete political or vocabulary difference between the parties. As a result, it is likely that the BagofWords applied to data from a different timespan entirely would not be as effective. Additionally, with such a limited pool of political commentators, the bag of words model likely has a focus on vocabulary specific to particular commentators such as their favorite hashtag, which would also not be as relevant in a broader political context. As a result, to improve the model a greater variety of political commentators over a wider timespan can be used.

It is worth mentioning that the semantic differences discerned by the Word2Vec model would face less trouble than the bag of words model applying to other contexts (even though it is context free) because it was trained to understand the difference in the general meaning of the words in the tweet, rather than focusing on particular words. So for example, if a different conservative host was being mentioned, it would have greater success than the Bag of Words model in potentially understanding the relevant meaning moving forward. This is even more true for the BERT model, where the pretrained nature of its design allows it to consider the broader possibilities of text. In this way, the transfer learning model may be the most useful in a general context outside the limited confines of a select 10 commentators who may have characteristic favorite hashtags. As a result, it is not necessarily correct to assume that in a real-world context such as preventing political siloing that the bag of words model would be the most useful, even though within the confines of the dataset it was the most successful at classification.

Tweet Generation

As another means to understand the relationship between the content on Twitter and political parties, I decided to implement a generative model to create new Tweets from each party. I created an LSTM recurrent neural network to accomplish this task. LSTM stands for Long Short-Term Memory, which is an effective method for processing the sequential data in the training set and implementing it to create its own imitations of the political Tweets. To preprocess the data I simply combined the tweets from either party into a single piece of text, separating each tweet with a space. The LSTM RNN would simply be trained to predict the most likely next character in the text.


```
model = Sequential()
model.add(LSTM(128, input_shape=(seqlen, len(chars)),
return_sequences=True))
model.add(Dense(len(chars), activation='softmax'))
```

The layers of the LSTM model.

I used the keras library to implement the 128 LSTM layer with randomly



Search Medium



Write



order to both allow for some exploration in prediction without contributing too high of a level of randomness. Below are some of the generated Tweets.

Democrat

#GetItOffYourChest You want to vent?? Reach out and touch us!!!

18005851051 @breakfastclubam <https://t.co/eYxhGoSgFl> Morning thoughts during presudicy and the

@MaryNarrellins @TravidJole Adam Senate and the political attacks of sad come to the presidential specially-designed homes for severely wounded Veterans via @H

@JamyanMayAsagrint @TheRessanSan The state for the former way, and the proposal of being controlled to call the party women are back to the other to specially

MorningJoe <https://t.co/eRicMWLKro> @Lawardin for a candidates and political posted in 2016 to the committee Sunday Montand

<https://t.co/XLn4xPIqCE> Some the state from the Biden Trump is a fanating the commented to first and his law we may to discuss the first and the end the day to

Republican

I discussion to be say that something than than the conservative or the United States is a trans than a people what he beet shame than any months for their med

Please make a plan to vote early for Justice Daniel Kelly.

Thank you for the face of men were read their person of censor of that the wombla of the next guns a people from a race and prove press consent to be an say on

I would be in the sensent under out of the ones than all than a person to reminding their free at the future are not make the Constitutional men with the trans

This is an important time list from any prosecutor to the left?

<https://t.co/q1C5AILyUY> World Manhall needs to press confronten and that so than really what

Thank you to tall read the crime consent to be the vaccine are people around that sensent than a paranical prosecutor than any consequences than hell be sure t

It was exciting to successfully generate tweets that look somewhat similar to ones actual political commentators would create. I was particularly excited by the second Republican-generated tweet listed here, which encouraged people to vote for a politician (Justice Daniel Kelly) who was actually running at the time the tweets were scraped from Twitter. In general, these models were interesting in part due to their use of political language. For instance Republican tweets used the words, “trans”, “constitution”, and “conservative”. The differences in vocabulary further confirmed my earlier analysis on the differences in vocabulary by both party being a significant factor. This type of model, if improved, could eventually be used by political actors to generate a significant amount of content to influence online elections for their side, or even by malicious actors hoping to influence the public online however they would want. With other advanced generative models such as ChatGPT, it is likely that this is going to be a reality sooner rather than later.

Bert

Bag Of Words

Word2vec

Classification Models

American Politics

**Written by Zach Sickles**[Edit profile](#)

0 Followers