Predicting U.S. Political Party Affiliation on Twitter

Catherine Lee, Jacob Shiff, Sridatta Thatipamala
Department of Electrical Engineering, Stanford University
Graduate School of Business, Stanford University
Engineering, Google
{catfranl, jshiff, sthatip}@stanford.edu

Abstract

With a multitude of divisive issues in today's political climate, politicians have migrated to Twitter to communicate publicly with their constituents and other politicians. In this paper we develop various neural network algorithms to predict users' U.S. political party affiliation based on singular tweets. We choose to focus on politicians' tweets, for which we have ground-truth labels of political party affiliation. We demonstrate that a GRU model can classify the party affiliation of tweets with 91.6% accuracy (and 8.4% error rate, close to the estimated Bayes error of 5%). Our model can be used to facilitate cross-party dialogue amongst Twitter users in an effort to minimize the national political divide.
(Github repository: https://github.com/sridatta/cs230).

1 Introduction

Twitter has become a popular platform for many politicians. Through tweets, retweets, @mentions, and #hashtags, politicians are able to respond quickly and directly to their constituents. While Twitter enables this dynamic discourse on political issues, it also lends itself to intellectual isolation, with limited dialogue across parties.

In this paper, we develop a binary classification model to identify a US political figure's political affiliation (not explicitly captured by Twitter) based on a single tweet. The input to the model is a sequence of words or tokens. We then use a recurrent neural network (RNN) to output a binary class label which predicts the tweet author's political party. With this model, we aim to detect political affiliation from a text of 140 characters or fewer. This classification can ultimately be used to facilitate cross-party dialogue, expose users to content from the opposing party, and develop other applications to moderate political discourse in the United States.

2 Related work

Our project is based on the assumption that Twitter discourse is highly partisan. To show the assumption has a credible basis, we present a study done by Conover et al. (2011), which uses a "combination of network clustering algorithms and manually-annotated data" to "demonstrate that the network of political retweets exhibits a highly segregated partisan structure, with extremely limited connectivity between left- and right-leaning users" [10].

Many researchers have used neural networks to study political language and identify political bias in various texts. A basic baseline for evaluating political bias uses traditional surface lexical modeling, or bag-of-words representation, which is what Gerrish and Blei (2011) used in predicting Congressional voting patterns based on their political leaning and written bills [6]. A group at the University of Maryland focused challenging the bag-of-words representation for political ideology detection with an RNN model [3]. By using RNNs, they were able to able to capture syntactic structure and semantic meaning. This provides an advantage in comparison to the bag-of-words model, as phrases may take on entirely new meanings from individual words alone. They demonstrated that compositionality from the RNN increased performance and allowed the model to outperform the bag-of-words model. From this, we decided to use an RNN as our baseline model. From there, we experimented with LSTM and GRU models.

Makazhanov and Rafiei (2013) focused on predicting political preferences of Twitter users [7]. They based their predictions on a user's language-model alignment with that of a political party's and evaluated their predictions based on the Alberta 2012 general election. With an F-measure score, their model outperforms the aforementioned sentiment and text classification approaches. A major difference between their model and ours is their construction of a user's profile based on their interactions with different political parties and evaluation of positive/negative terms during interactions.

In our own model, as inputs, we used the data collected by computer scientist at Purdue University Kristen Johnsen for her project on "Leveraging Behavioral and Social Information for Weakly Supervised Collective Classification of Political Discourse on Twitter" [1]. Johnsen was more interested in using neural networks to frame Twitter issues, as brought forth by politicians. However, the data gathered was useful as each tweet was already labeled with party affiliation.

3 Dataset and Features

We used the Congressional Tweets Database [1], a database of Twitter tweets of the 114th U.S. Congress members from 2014 through 2017, to develop our model. The dataset includes tweets on topics like gun control, terrorism, abortion, and immigration, and excludes campaign tweets and location updates. The database includes 86,472 tweets. The classes are roughly evenly balanced, with 45,023 (52%) tweets from Republican politicians and 41,619 (48%) tweets from Democratic politicians.

Preprocessing. We enriched each entry in the Database, which included the Tweet ID and party affiliation (Democrat or Republican), with the full tweet text using a Tweet Downloader [2]. We then ran the dataset through the following preprocessing steps: (1) tokenizing text by splitting by space; (2) padding sentences to a maximum of length of 32 words; (3) cleaning punctuation from ends of words; and replacing all links, hashtags, and "@" mentions with placeholders .

The final step was necessary because the vocabulary of the tweets was too large and sparse, with many hashtags and links being used only a handful of times. After replacing these one-off tokens with placeholders, the total vocabulary size after preprocessing was 18,713.

We split the 86,472 entries into training, development, and test datasets, as seen in Table 1.

Data set	Number of examples	Percentage of dataset
Training	60,649	70%
Development	12,996	15%
Test	12,997	15%

Table 1: Data split for our model

4 Methods

Input. The input to our model was a tweet, which we processed into a sequence of tokens. Neural networks operate on vectors so we had to convert each token into a vector. We accomplished this by using word embeddings. Word embeddings "represent (embed) words in a continuous vector space where semantically similar words are mapped to nearby points" [8].

We initially used a pre-trained GLoVe embedding [9], which learn word embeddings by predicting the number of times a pair of words occur together in a "context window". Specifically we used glove.6B.300d, which maps a vocabulary of 6 billion words (from Wikipedia) into a 300-dimensional vector space. 82% of our tweet vocabulary had a matching vector in the GLoVe matrix. The remaining 18% of words were mapped to randomly generated vectors.

Model. Since our inputs are sequences of words, it was natural to use a recurrent neural network as the basis for our model. An RNN is a neural network that operates recursively, in a loop. At the core of the RNN is a “cell” which maintain a state vector h. At each time step, it accepts an input $x^{<t>}$ and previous state $h^{<t-1>}$ to produce a new state $h^{<t>}$ and new output $y^{<t>}$.

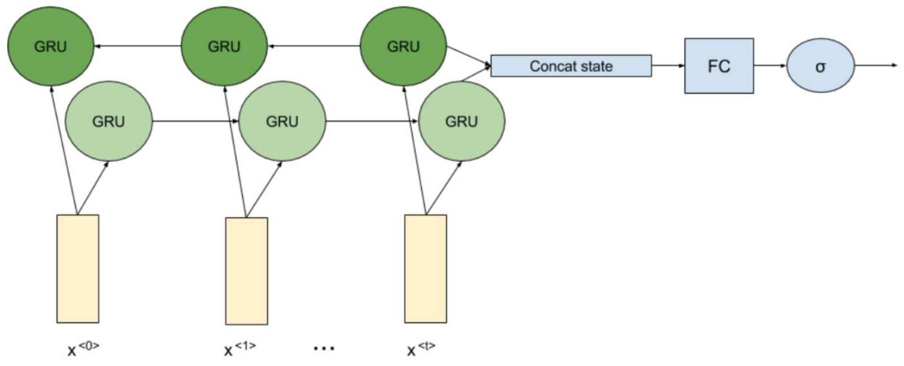

Figure 1: Final Model architecture (Bidirectional GRU model with fully-connected layer + sigmoid)

We first developed our baseline model to establish a benchmark for accuracy. For our baseline, we used a single-unit RNN with a state size of 128. This baseline model achieved a test set accuracy of 77.2%.

To reduce the avoidable bias, we experimented with several more sophisticated models, including RNNs with LSTM cells and multi-layer RNNs. We ultimately selected a bidirectional GRU architecture. This architecture uses two GRU cells. A GRU cell is a type of RNN cell which addresses the “vanishing gradient” issues of a vanilla RNN. The “vanishing gradient” problem causes recent inputs to have much greater influence on the output than inputs seen in previous timesteps. GRU addresses this by updating the cell’s hidden state to more carefully preserve existing state which still taking into account new inputs.

The first step (“reset”) selectively forgets parts of the hidden state. The second step (“update”) sets new information into the hidden state.

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$
$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$
$$h'_t = \tanh(W^x_t + r_t \odot Uh_{t-1})$$
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t$$

The model is “bidirectional” because one GRU unit processes the sequence of tokens from in the forward direction (from t=1 to t=k) while the other GRU unit processes the sequence in reverse (from t=k to t=1). This architecture allows each input token to influence the tokens that come after it (in the forward layer) as well as those that come before it (in the backward layer).

To improve model performance even further, we allowed backpropagation to update values of the embedding matrix itself (a technique which we refer to in this paper as “trainable embeddings”). This technique allows the embedding matrix to adapt to the text distribution of political tweets, rather than remain static on the dataset it was initially trained on.

Output. After feeding the input sequence through the bidirectional model, both GRU cells have a final hidden state. We can treat this state as a vector representation of the entire tweet. To produce a binary classification output, we concatenate each GRU unit’s 128-dimensional state vector into a single 256-dimensional vector. Then we use a fully-connected layer to project this vector into one dimension, and pass it through a sigmoid activation to get a (0, 1) output.

Loss, Optimization, Hyperparameters. We used a cross entropy loss function because it is historically a good loss function to optimize accuracy of a binary classification problem.

$$L(\hat{y}, y) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}))$$

We used an Adam optimizer [11] to improve convergence of loss function during training. Adam is an adaptive learning rate algorithm. It adjusts per-parameter learning rates based on the moving average of gradient updates (first moment) and squared gradient updates (second moment).

Regularization. To reduce the model's variance (as measured by the difference between training and test accuracy), we applied two regularization techniques: L2 regularization and dropout. We applied L2 regularization to the fully-connected output layer and to the updates to the embedding matrix. L2 regularization prevents overfitting by adding the L2 norm of the model parameters to the loss function. This incentivizes the model parameters to be close to 0. We applied dropout to the GRU hidden state. Dropout randomly shuts off or masks parts of the model's activation during training time. This has a regularizing effect because the model must learn to represent the same information using multiple different parameters.

We tuned the model's hyperparameters by performing a random search of the hyperparameter space. We searched for L2 lambda on a logarithmic scale from 10e-4 to 10e-1, and dropout keep_prob on a linear scale from 0.4 to 0.8. The following are the hyperparameters used in our final implementation.

Learning rate	Batch size	Hidden units	Dropout rate (keep_prob)	L2 regularization (lambda)
1e-3	64	128	0.67	0.0438

Table 2: Hyperparameters used in final LSTM implementation

5 Experiments, Results, and Discussion

After training each model for 10 epochs, we were able to achieve a 91.6% test set accuracy using the bidirectional GRU model.

Model	Training accuracy	Testing accuracy
0. RNN	77.6%	77.2%
1. GRU	91.0%	86.4%
2. GRU (with trainable embeddings)	99.4%	87.9%
3. bi-directional GRU	95.9%	88.7%
4. bi-directional GRU (with trainable embeddings)	98.3%	91.6%

Table 3: Training and testing accuracy based on model

With a test set of $n = 12,997$ examples, Table 4 shows the results of our model in a confusion matrix:

	Predicted: Democrat (0)	Predicted: Republican (1)	
Actual: Democrat (0)	True Democrat = 5632 (43.3%)	False Republican = 585 (4.5%)	6217 (47.8%)
Actual Republican (1)	False Democrat = 512 (3.9%)	True Republican = 6267 (48.2%)	6779 (52.2%)
	6144 (47.2%)	6852 (57.7%)	

Table 4: Confusion matrix of our results

Precision: 91.7%	Recall: 90.6%	Accuracy: 91.6%

Table 5: Final metrics

To analyze misclassified tweets, we performed an error analysis on 50 misclassified tweets (see Table 6 for a list of examples). Of those 50, we found that 10 tweets (20%) were entirely non-partisan and did not provide any signal about political affiliation. We were able to manually classify 22 (44%) of them correctly and got 18 (36%) of them wrong. Since our classifier had a 8.4% error rate and we (average of expert humans) could only reduce this error by 44%, we believe the Bayes error rate for this problem to be close to 4.7%.

Original tweet	Prediction	Actual	Analysis
joining @TeamCavuto at 4:15 on @FoxNews to discuss the burwell nomination hearings #TSonTV	Democrat	Republican	@FoxNews is indicative of Republican but we replaced all "@" mentions with placeholders.
saddened by the violence in colorado my thoughts and prayers go out to all the victims and their families of this senseless tragedy	Democrat	Republican	Non-partisan tweet which does not indicate any political affiliation
i call on speaker boehner to convene the house to debate and vote on syria by wednesday of next week	Republican	Democrat	Mention of "Speaker Boehner" which might not occur frequently enough to learn a strong signal
this SCOTUS ruling creates even more uncertainty for americans trying to comply w the ACA.	Republican	Democrat	Reference to a context ("SCOTUS ruling") which lies outside the tweet itself.

Table 6: Examples of misclassified tweets and error analysis

The algorithm was impressively accurate in classifying tweets, identifying subtle nuances in language between Republicans and Democrats. Table 7 lists a few examples of tweets that were correctly classified:

Tweet	Prediction
Obamacare continues to threaten seniors enrolled in popular MedicareAdvantage program	Republican
no child should end her life cold amp alone struggling for her last breath inside an abortion clinic	Republican
by a show of RTs, who thinks Congress should be focusing jobs instead of repealing Obamacare for the 37th time	Democrat
house gop once again blocks vote to prevent individuals on terrorist watch list from buying guns	Democrat

Table 7: Examples of correctly classified tweets

We hypothesize that the model, with a training error of 1.7% (above our estimated Bayes error), is overfitting. We employed regularization techniques like dropout and L2 regularization, but additional regularization techniques like early stopping could potentially further minimize overfitting.

6 Conclusion/Future Work

A limitation of our approach is that we were unable to test our model on non-politician users, because Twitter does not collect political party affiliation data. It is possible that the language distribution of politician's tweets is different than that of civilian users and that our algorithm would not effectively generalize to a civilian population. Future work could include surveying Twitter users to collect political party affiliation. With this information, we could assess the accuracy of our algorithm on a civilian population and further tune the algorithm for civilian tweets.

Another limitation we would like to address is our model's inability to use tweet history in its prediction. Our model would likely improve if we inputted several tweets (simulating a tweet history), rather than a single tweet. A simple approach would be to use our existing model to classify individual tweets and feed those classifications into another model to make an aggregate prediction.

7 Contributions

All members contributed equally to this paper. Sri took the lead in deploying the first iteration of the model. All members assisted with subsequent iterations and paper and poster write-ups.

References

[1] K. Johnson, D. Jin, D. Goldwasser. *Leveraging Behavioral and Social Information for Weakly Supervised Collective Classification of Political Discourse on Twitter*. ACL, 2017. Github repository, https://github.com/kmjohnson/twitter-framing.

[2] E. Summers, N. Ruest. *twarc*. University of Maryland, 2017. GitHub repository, https://github.com/DocNow/twarc.

[3] M. Iyyer, P. Enns, et al. *Political Ideology Detection Using Recurrent Neural Network*. 2014. https://people.cs.umass.edu/m̃iyyer/pubs/2014_RNN_framing.pdf

[4] C. Olah. *Understanding LSTM Networks*. 2015. Github repository, http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[5] S.M. Gerrish, D. M. Blei. *Predicting Legislative Roll Calls from Text*. International Conference on Machine Learning, 2011. http://www.cs.columbia.edu/b̃lei/papers/GerrishBlei2011.pdf

[6] A. Makazhanov, D. Rafieie. *Predicting Political Preference of Twitter Users*. IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, 2013. https://ieeexplore-ieee-org.stanford.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=6785723

[7] Tensorflow. *Vector Representations of Words | TensorFlow*. TensorFlow, 2018. https://www.tensorflow.org/tutorials/word2vec.

[8] J. Pennington, R. Socher, C. D. Manning. *GloVe: Global Vectors for Word Representation*. 2014. https://nlp.stanford.edu/projects/glove/.

[9] M. D. Conover, J. Ratkiewicz, M. Francisco, B. Gonc¸alves, A. Flammini, F. Menczer. *Political Polarization on Twitter*. Fifth International AAAI Conference on Weblogs and Social Media, 2011. https://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/viewFile/2847/3275

[10] D. P. Kingma, J. Ba, *Adam: A Method for Stochastic Optimization*. ICLR Conference, 2015. https://arxiv.org/abs/1412.6980.

[11] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.