Classifying the partisan learning of Twitter text using NLP methods

Zhen Guo and Zheyu Zhang

Abstract

Classifying Twitter text into partisan stans is an important step in understanding political polarization on social media. Nevertheless, in order to save human labor, it is best to use Natural Language Processing (NLP) methods to "learn" the labels of twitter text. This project aims to evaluate the performance of logistic regression, support vector machine, feedforward neural network, and recursive neural network on classifying the partisan leaning of Twitter text into Democrat or Republican categories. We pretrained the texts into word embeddings as features to fit the models. The above-baseline accuracy of the models indicates that linguistic features can be used to predict partisan leaning of the texts.

Introduction

Partisan polarization has gained increasing attention in the U.S., and Twitter, in particular, has become a place to express extreme political ideology, which has triggered much research on political tweets. The partisan leaning analysis is helpful in various contexts, such as political analysis, social media monitoring, or marketing research. However, it is sometimes hard to identify the political stands of massive tweet texts. By using machine learning models such as logistic regression, support vector machine, and neural network, we can develop an efficient routine for classifying text into the political stands.

The use of machine learning in analyzing human generated texts is called natural language processing (NLP). This paper aims to evaluate the performance of different NLP models for classifying the partisan leaning of Twitter texts. Specifically, we build four models, including logistic regression, support vector machine, feedforward neural network, and recursive neural networks, and train them with tweets of politicians from both Republican and Democrat party of the United States. We pretrain tweets into word embeddings using Sentence-BERT, then fit the four classification models that we build.

Our findings provide insights into the relationship between linguistic features and political partisanship on Twitter and demonstrate the potential of NLP methods for analyzing social media data. This research has implications for understanding the dynamics of political discourse on social media and may be useful for political campaigns, social media platforms, and policy makers.

Related work

Numerous studies have examined the relationship between social media and political behavior, particularly in the context of Twitter [1]–[4]. Previous research has employed a variety of

approaches, including content analysis, sentiment analysis, and network analysis, to understand the dynamics of political discourse on the platform. In order to understand political behavior on social media through a macro-level scope, researchers seek ways to automatically conduct tasks that would be labor-intensive otherwise, such as identifying partisan leaning of the social media content. Several studies have explored the use of NLP for sentiment analysis and classification of political content on Twitter [5], [6]. Various models have been built for determining the partisan leaning of tweets using manually coded data [7], [8]. However, because of the lack of no standard methods in quantifying measurement of partisanship, there is no parallel evaluation of the performance of different models. In this paper, we build on this existing literature by evaluating four well-known NLP models for their accuracy in classifying partisan leaning of tweets, providing insights for future studies on choices of models.

Task statement

We plan to use supervised machine learning approach to build models that can predict bianary partisan leaning of given tweet texts. However, hand-labeling data are subject to human bias and usually have small sample size. Therefore, we aim to train the model with tweets of known politicians. We assume that the partisan leaning of the tweets is align with the partisanship of the politicians. In this way, we can acquire large datasets with precise labels.

Our next step is to pretrain the tweets into word embeddings, which then are fitted into the model. We do this based on the theory that the language patterns of politicians in different parties are diverse [9], [10]. For example, when discussing issues of climate change, democrat congress members are more likely to use the word “clean energy,” whereas republicans prefer the word “energy tax”[9, p. 487]. Word embedding is powerful in capturing the meaning by representing context of texts, which is suitable in this case. To better grasp the context of words, i.e., word sequence, in the model, we also implement recursive neural network which takes the order of features into account.

We plan to use 70% of the dataset as training set and 30% as testing set for evaluation. The flow of the task is illustrated in figure 1.

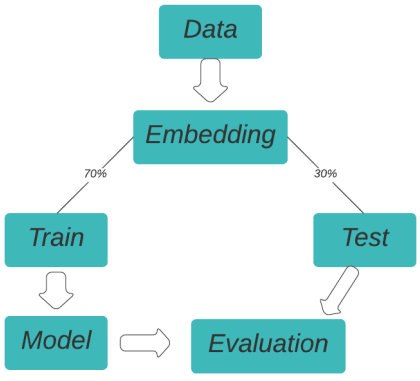

Figure 1 Flow of the task.

Data

In this project, we will use binary classification to mark each tweet as Democrat or Republican. Our dataset is from Kaggle. This dataset contains tweets of US politicians who are given their party identity, either as a democrat or republican. Each party has 24 representatives, and each politician's 200 tweets from May 2018 are collected in this data set. There are 84502 unique tweets in the dataset. The number of instances in Democrat party is 42067, whereas the number of republican tweets is 44392. Thus, the dataset is even in terms of class distribution. The distribution of instance of each class is displayed in figure 2.

Kaggle URL: https://www.kaggle.com/datasets/kapastor/democratvsrepublicantweets

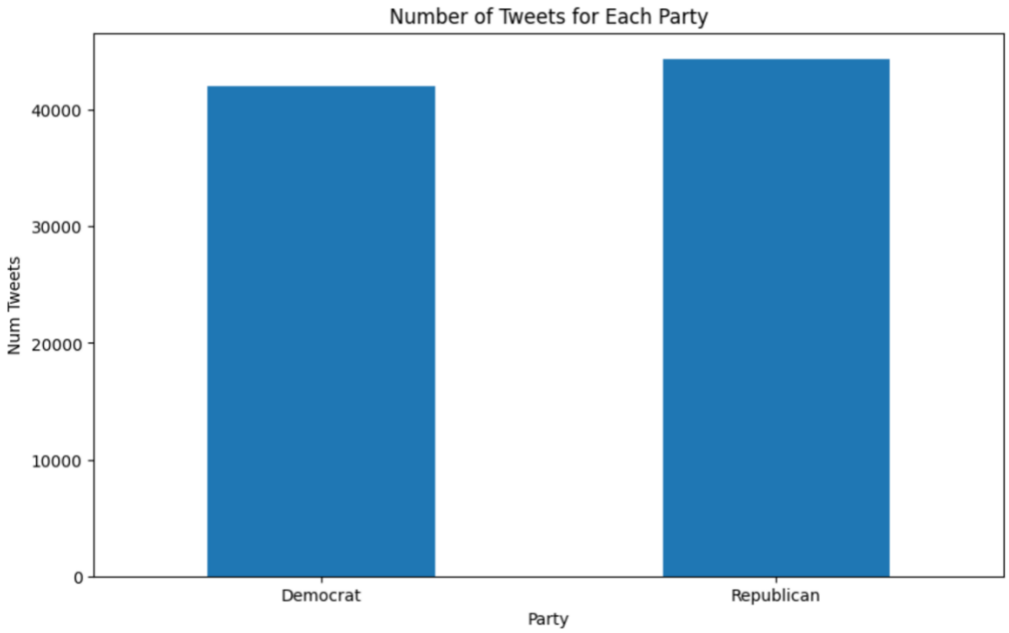

Figure 2 Class instances of dataset.

Methods

This project aims to use the tweets of known politicians to train a model of classifying tweets. To achieve this, we pretrain the tweet texts into word embeddings. Then, we use the embeddings to train four models: logistic regression, support vector machine, feedforward neural network, and recursive neural network.

Table 1 NLP models

Model	Function or Layers	Hyper parameters
Logistic regression	sigmoid	max_iter = 1000, penalty = l2
SVM		kernel = linear
FNN	3 layers, 2 hidden layers ReLU, one dropout layer with sigmoid for output	input_size = 768 hidden_size = 128, 256, 512 num_classes = 2 num_epochs = 5, 10, 20 p_dropout = 0.8
RNN	1 hidden layer, LSTM	embedding_dim = 768

		hidden_size = 128, 256, 512 num_epochs = 5, 10, 20

Pretrain word embedding

For encoding sentences, we utilize the Sentence-Transformer library [11], which provides a convenient high-level interface to work with sentence embeddings. We specifically use the 'paraphrase-distilroberta-base-v1' model [12], a powerful pretrained model known for generating semantically meaningful sentence embeddings.

Logistic regression

Logistic regression is widely used in classification tasks. The model takes the input vector and applies the sigmoid function. We use the *sklearn* package for the logistic regression model and its training. As for hyperparameters, we use the default penalty, which is l2 and set the max_iter to 1000.

Support vector machine

Support vector machine is another popular machine learning model for classification problems. The mechanism of this algorithm is to maximize the distance between the two groups. Again, we use the *sklearn* package for the model and training. As for hyper parameters, we use the linear kernel.

Neural network

Feedforward Neural Network

We build a feedforward neural network classifier from PyTorch [13] NN module, which consists of two fully connected layers with ReLU activation functions and a dropout layer for regularization to avoid overfitting. We select sigmoid activation function for output layer since it is a binary classification problem. We choose Adam optimizers with a learning rate of 0.001 and cross-entropy loss function for criterion. We start with 128 hidden nodes and 5 epochs for training and gradually tune hyperparameter for optimizing our neural network. 10-fold cross validation are applied to each grid search and best average validation accuracy model are saved as model.pt file followed by test dataset evaluation.

Recurrent Neural Network

We also utilize a recurrent neural network (RNN) with long short-term memory (LSTM) [14] architecture to perform text classification. The LSTM model is designed to handle sequential data and is suitable for processing text data. We used one LSTM layer, which takes the embedded text input and passes it through the LSTM cell. The LSTM layer is responsible for maintaining and updating the hidden state and cell state based on the input sequence. We tested 128, 256 and 512 hidden nodes in LSTM layer. Another linear layer fully connected to the LSTM

layer, which maps the hidden states to the two output classes for binary classification. Same cross validation methods are applied to RNN.

Results

After training all the models, we record the testing accuracy and compare them. Among all the models, RNN with 5 number of epochs regardless of number of hidden layers has the highest accuracy, which is 73.5%. Logistic regression with l2 penalty and sigmoid function has the lowest accuracy, which is 68.6%. SVM and FNN has 68.74% and 72.50% accuracy respectively. This is illustrated in figure 3.

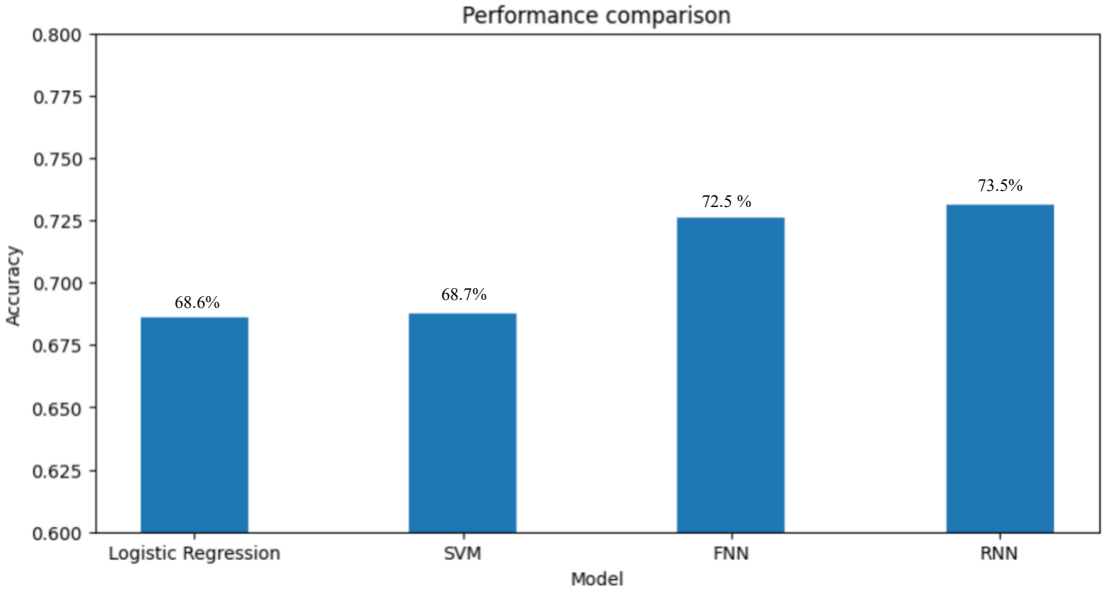

Figure 3 Performance of the four models on testing data.

Figure 4 illustrates the performance of FNN varied by the number of epochs and the size of the hidden layer. The accuracy of model changes slightly when change the number of epochs. More epochs do not necessarily improve the performance of model. The best performance is with 10 epochs with 72.84%, comparing to 72.41% and 72.82% with 5 and 10 epochs, holding the size of hidden layers the same. We believe that more epochs might result in overfitting, because the training accuracy is 90% when use 20 epochs, as comparison to 87% when use 10 epochs. When changing the number of hidden layers from 512 to 128, while holding the number of epochs the same, the accuracy is also similar, with 72.84% and 72.58% respectively. Overall, the performance of FNN is about 4% more accurate than logistic regression, even though only two hidden layers with ReLU activation functions are added before the sigmoid output layer.

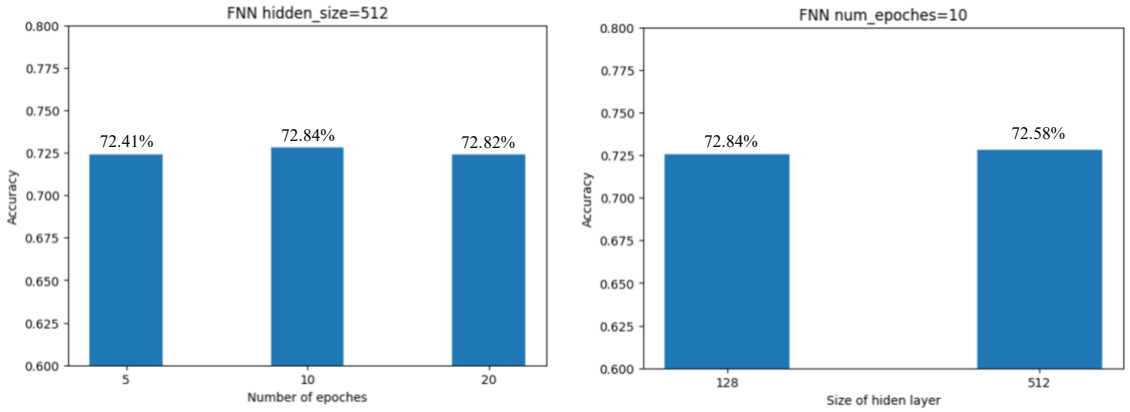

Figure 4 FNN performance varied by number of epochs and size of hidden layer.

Figure 5 demonstrates the impact of varying the number of epochs and the size of the hidden layer on the performance of the RNN model. The accuracy of the model only shows slight changes when the number of epochs is altered. Not surprisingly, increasing the number of epochs does not necessarily lead to an improvement in the performance of the model. The optimal performance is achieved when the model is trained with 5 epochs and a hidden layer size of 128, resulting in an accuracy of 73.23%. In comparison, when the model is trained with 10 epochs and the same hidden layer size, the accuracy slightly decreases to 72.48%. Our findings suggest that increasing the number of epochs may cause overfitting, as evidenced by the training accuracy of 84% with 10 epochs, compared to 79% with 5 epochs. See figure 6 for reference of training accuracy versus validation accuracy.

Changing the number of hidden layers from 128 to 512 while maintaining the number of epochs at 5 results in the same accuracy of 73.23%. Overall, the RNN model outperforms the FNN model by approximately 1% in terms of accuracy.

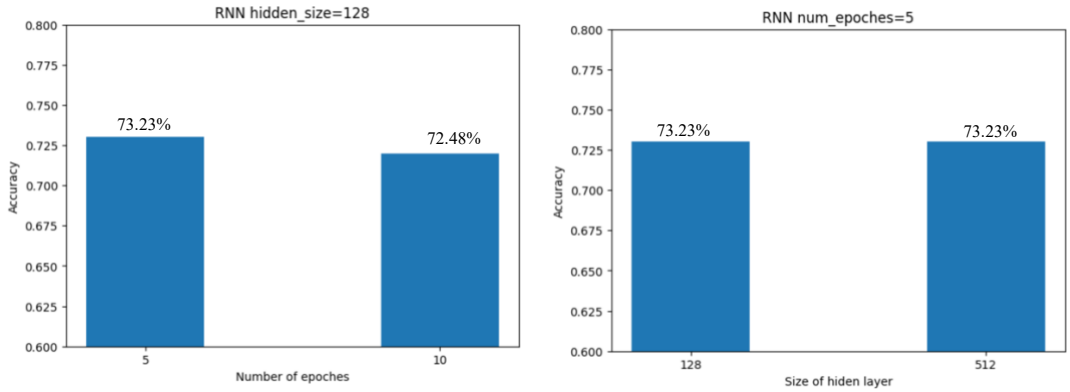

Figure 5 Performance of RNN varied by number of epoches and size of hidden layers.

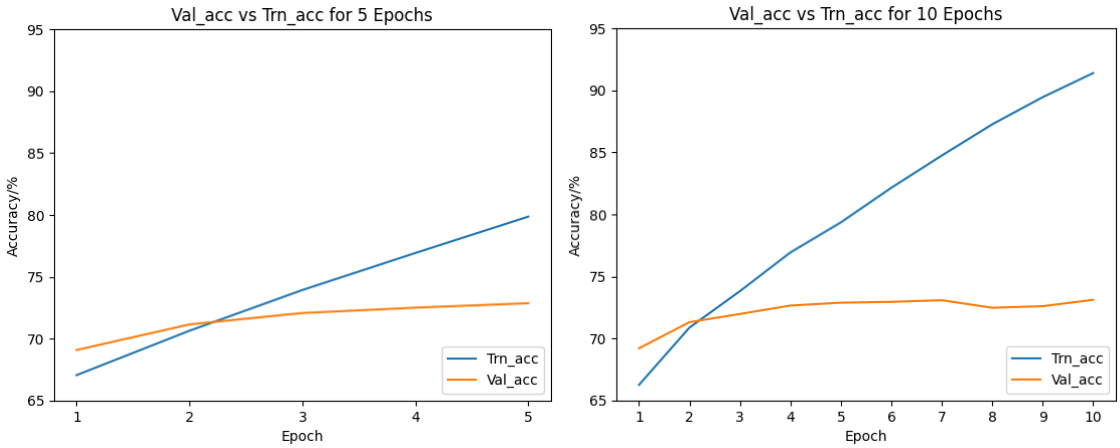

Figure 6 Training Accuracy vs Validation Accuracy of RNN varied by number of epoches.

Conclusion and Future

In conclusion, this project has successfully demonstrated the effectiveness of various supervised machine learning models for predicting the binary partisan leaning of given tweet texts.

For encoding sentences, we utilized the Sentence-Transformer library, which provides a convenient high-level interface to work with sentence embeddings. We specifically employed the 'paraphrase-distilroberta-base-v1' model, a powerful pretrained model known for generating semantically meaningful sentence embeddings. This approach has facilitated the creation of high-quality input data for our machine learning models.

Through careful experimentation and comparison, we have determined that the LSTM RNN model outperforms other models with over 73% accuracy, showcasing its superiority over Feedforward Neural Networks, SVM, and logistic regression models. Thus, our results highlight the potential of RNNs as a robust and effective approach for predicting partisan leaning in tweet texts, opening up new possibilities for various applications in the realm of political analysis and beyond.

References

[1] J. A. Tucker *et al.*, “Social Media, Political Polarization, and Political Disinformation: A Review of the Scientific Literature.” Rochester, NY, Mar. 19, 2018. doi: 10.2139/ssrn.3144139.

[2] O. D. Apuke and B. Omar, “Fake news and COVID-19: modelling the predictors of fake news sharing among social media users,” *Telemat. Inform.*, p. 101475, Jul. 2020, doi: 10.1016/j.tele.2020.101475.

[3] D. Bamman, B. O’Connor, and N. Smith, “Censorship and deletion practices in Chinese social media,” *First Monday*, Mar. 2012, doi: 10.5210/fm.v17i3.3943.

[4] S. Hong and S. H. Kim, “Political polarization on twitter: Implications for the use of social media in digital governments,” *Gov. Inf. Q.*, vol. 33, no. 4, pp. 777–782, Oct. 2016, doi: 10.1016/j.giq.2016.04.007.

[5] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau, “Sentiment Analysis of Twitter Data,” in *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, Portland, Oregon: Association for Computational Linguistics, Jun. 2011, pp. 30–38. Accessed: Apr. 27, 2023. [Online]. Available: https://aclanthology.org/W11-0705

[6] E. Martínez-Cámara, M. T. Martín-Valdivia, L. A. Ureña-López, and A. R. Montejo-Ráez, “Sentiment analysis in Twitter,” *Nat. Lang. Eng.*, vol. 20, no. 1, pp. 1–28, Jan. 2014, doi: 10.1017/S1351324912000332.

[7] F. M. F. Wong, C. W. Tan, S. Sen, and M. Chiang, “Quantifying Political Leaning from Tweets, Retweets, and Retweeters,” *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 2158–2172, Aug. 2016, doi: 10.1109/TKDE.2016.2553667.

[8] M. Iqbal, “Determining Political Inclination in Tweets Using Transfer Learning”.

[9] Z. Albert, “Partisan Policymaking in the Extended Party Network: The Case of Cap-and-Trade Regulations,” *Polit. Res. Q.*, vol. 73, no. 2, pp. 476–491, Jun. 2020, doi: 10.1177/1065912919838326.

[10] S. C. Guntuku, J. Purtle, Z. F. Meisel, R. M. Merchant, and A. Agarwal, “Partisan Differences in Twitter Language Among US Legislators During the COVID-19 Pandemic: Cross-sectional Study,” *J. Med. Internet Res.*, vol. 23, no. 6, p. e27300, Jun. 2021, doi: 10.2196/27300.

[11] Reimers, Nils, and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks." *arXiv preprint arXiv:1908.10084* (2019).

[12] Henderson, Matthew, et al. "A repository of conversational datasets." *arXiv preprint arXiv:1904.06472* (2019).

[13] Paszke, Adam, et al. "Pytorch: An imperative style, high-performance deep learning library." *Advances in neural information processing systems* 32 (2019).

[14] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

Supplementary Data:
https://drive.google.com/file/d/1qogx047dUHZXd6yapjvxqNdGuBlUUZBZ/view?usp=sharing