Liberals v.s. Conservatives – a Text Classification Application

Xinping Xue

Abstract

One application of text classification on political science is to identify the political leaning from the written text. In this project, several models are trained and compared to classify the political ideology of the written text as liberal or conservative. It is found that BERT performs best among all models. We attempt to address the overfitting issue and hypothesize the potential cause of it by examining the examples misclassified by the model. Several patterns of misclassification is found and summarized and potential directions for future work to improve the model performance are pointed out.

1 Introduction

Political ideology in the United States is usually described by a left-to-right spectrum, with the left-leaning ideology is classified as liberalism and right-leaning ideology classified as conservatism. In short, each of these ideologies mainly consists of two parts, i.e., views on social issues and views on economic issues.

On social issues, conservative ideology typically adheres to Christianity and supports moralizing people according to religion principle. They hold dear to traditional values and religious laws, often advocating vouchers for Christian schools and school prayer.[1][2] These traditions are reflected in the opposition to abortion, homosexuality, feminism, explicit sex education, etc. Liberalism, on the other hand, supports that everyone has the freedom of private life without being discriminated by the things like tradition or religious prejudice. Liberals often supports the rights for LGBT groups, the freedom of abortion, woman's right and equal voting rights for all adult citizens.[3]

On economic issues, the attitude towards freedom apparently seems reversed. Conservatives often support to have a small government and free market. They often oppose government intervention to the economy, advocating tax cuts, government spending reduction, less regulation, privatization for efficiency, etc.[4] Free economy is generally in favor of the people who are born in a higher social class, those who possessing more resource, or those who are willing to work hard and with great talent. On the other hand, liberals, for the aim of equality, support the redistribution of wealth through tax system. They support capitalism with government intervention, advocating issues like increasing minimum wage, building a mandatory healthcare system.[5]

The main goal of the project is that given some comments suggesting the author's view, we want to classify her view as leaning towards liberalism or conservatism.

There are several potential applications for the task. An apparent application is that by predicting one's views from her comments, we are able to target the political advertisement and news media recommendations for this person. e.g. Liberals typically prefer watching MSNBC while conservatives prefer watching FOX news. One less direct application is to help design campaigning strategies. Because in classification tasks we often output an estimated probability instead of a hard label, we might be able to identify each person's relative location on the left-right political spectrum according to the probability score obtained from her comments. This might help presidency candidates to identify swing voters, whose political ideology is often in the middle, so that candidates can better design their campaigning strategy to meet the needs of these voters in the light that the votes of swing voters often dictate the election result.[6]

2 Data Preparation and Description

The data is downloaded from Kaggle in a table form in *this site*[7].

The source of the data is Reddit. There are in total around 13000 examples. Each example in the dataset corresponds to one post in Reddit, which consists of several entries, including the title of the post, body text of the post if it exists, the subreddit (a.k.a. community) where the post is posted, and the political lean of the subreddit (liberal or conservative), etc.

Note that here we use the political lean of the subreddit as our label, which serves as a proxy of political lean of the post. This proxy is valid because in Reddit, there are mechanisms to prevent posts from the opposite view from being posted on wrong community. e.g. These posts typically will be deleted by subreddit supervisors to prevent fierce verbal attacks between people with opposite views and such authors are often prohibited from posting and the members of these left or right subreddits often conforms to the political lean of these subreddits. The ratio of number of examples labeled as conservatives to liberals is about 0.35:0.65, which is slightly imbalanced.

To obtain text features, the title and body text of each post is concatenated together, seperated by a newline escape sequence. The labels are converted into 0 or 1 with 0 represents conservative and 1 as liberal. We further split the data into train-validation-test sets with proportions as 0.8, 0.1 and 0.1, respectively. Because BERT has a required form of tokenizer while models like LSTM and DAN don't, we will preprocess them in different ways, which will be discussed in the next section.

3 Models

The model we used includes a DAN model, a bidirectional LSTM model, a bidirectional GRU model and a BERT pretrained and finetuned model. The metrics used includes accuracy, precision, recall and f1-score. The loss function is standard and the optimizer used is AdamW with default arguments.

In DAN model, the preprocessing steps include tokenization of the text features using the word tokenizer from NLTK, lowering the cases, removing all punctuations, lemmatization of all words, removing stop words, and padding all text features into the same length according to the length of longest features. We delegate these tasks into a specific Dataset and use dataloader to batch the data during training and validation. The preprocessing of LSTM and GRU models are similar except that first we don't remove stop words so that the original order of words in sentences can be maintained for these sequence models, and second we don't padding them into the same length and we don't batch features into minibatches while training as RNNs are generally less parallelizable.

As for the model architectures, DAN consists of an embedding layer, an average pooling layer over non-padding word embeddings, and a two-layer feedforward block which includes a hidden layer and ends with a sigmoid function. In the feedforward block, one batch normalization layer and one dropout layer are included for the purpose of accelerating training and regularization, respectively. Hyperparameters selected according to the validation performance include the number of embedding dimensions, number of hidden layer dimensions, and the dropout rate. Other hyperparameters like number of epochs or minibatch size are not selected according to the validation set performance.

For LSTM and GRU, their architectures are similar to each other. Each includes an embedding layer, a bidirectional LSTM or GRU with one or multiple layers, and a two-layer feedforward block. Note that the hidden states output by the final cell in both direction for all LSTM(or GRU) layers are concatenated together as the input of the feedforward block. Hyperparameters include number of embedding dimensions, number of hidden state dimensions in the LSTM(or GRU), number of LSTM(or GRU) layers, and the dropout rate in the feedforward block. Note that other hyperparameters are not selected by validation performance but are chosen directly according to other factors. e.g. learning rate is decided according to the loss curve. Number of epochs is decided according to how fast the training loss converges, etc.

BERT, on the other hand, uses its own BertTokenizer according to Hugging Face documentation. The pretrained parameters are from "bert-base-uncased". In the preprocessing step, the maximum length of each feature is set to be 128. Text longer than this will be truncated and the text shorter than this will be padded to this length. The main reason we choose 128 is that if we set the length longer, it will run out of RAM during training given the upper limit for RAM in Google Colab. Training instances are also batched into minibatches during training for parallelization purpose.

The model architecture consists of a BERT pretrained model, whose pooler output is subsequently taken as input for a two-layer feedforward block. At first, all hyperparameters within BERT architecture are set according to the Bert default hyperparameters. The hyperparameters selected on the validation set include dropout rate in the feedforward network and number of epochs.

Later on, to address overfitting problem, weight decay rate and dropout rate inside the BERT configuration, which includes attention dropout rate and hidden dropout rate, and the dropout rate in the feedforward block are selected according to the validation performance. Overfitting issue will be discussed in more detail in the next section.

4 Result and Quantitative Analysis

The training and validation of each model can be summarized into several steps:

First, we select several configurations of hyperparameters for each model, we train it using such configurations and then according to the validation f1-scores, we select the best configuration of hyperparameters for each model. All the configurations chosen and the validation results for each model are presented in Appendix.

Second, for each model with its selected hyperparameters, we test their performance on the testing set instead of on the validation set. The result on the test set is presented in the following table:

	precision	recall	f1_score
DAN	0.8045	0.7747	0.7893
GRU	0.7668	0.7604	0.7636
LSTM	0.7842	0.8272	0.8051
BERT	0.7963	0.8808	**0.8364**

Table 1: Performance of models on test set

From the resulting table we can clearly see that BERT performs much better compared to all the other models. Actually in the validation set, even the BERT configuration with the worst performance among all BERT configurations achieves a larger f1-score than any other model with its best configuration, indicating that BERT might be superior than all other models in terms of performance. However, during the training, BERT is perceivably slower and more costly to train for each epoch than all other models. Although the configurations of hyperparameters sampled for GRU and LSTM are identical, LSTM has a better performance overall on both validation set and test set, which is consistent with our knowledge that the number of parameters to train in LSTM is larger than that in GRU given the same set of hyperparameters.

The main problem during the training is overfitting. From the training plot obtained during the training of a BERT model, including both accuracy curves and loss curves, we can clearly see that the performance on validation set stops increasing after several epochs. Although the training accuracy keep rising toward almost 100%, validation accuracy is stuck around 80%. Similar phenomena also happened in the training of all other models after several epochs.

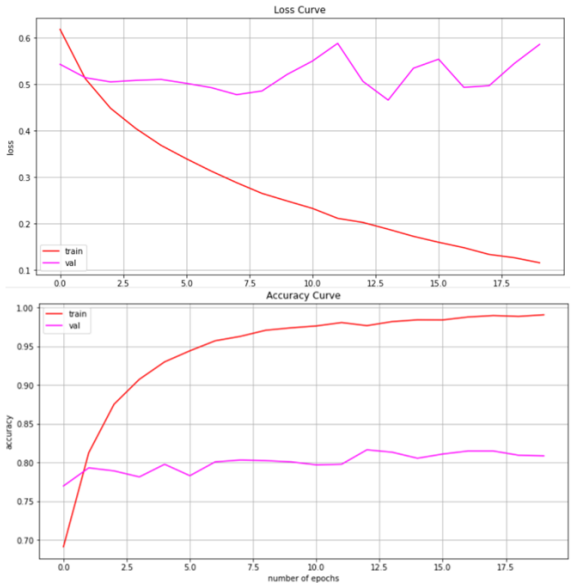

Figure 1: Training Curves for one BERT model

Because BERT works the best among all the models, it is the main focus of our subsequent analysis. To handle the overfitting issue on BERT, I used two regularization techniques. First, tune up dropout rate both inside the BERT architecture and the feedforward layers. Second, tune up weight decays in the AdamW optimizer. In practice, I choose several combinations of these two hyperparamters to train the models, the result on validation set is in the Table 2.

We choose the best-performed combinations on the validation set and test its performance on the testing set. Although on the validation set the selected new model achieved a higher f1-score, on the test set, as is shown in Table 3, it only achieves a slightly lower f1-score compared with the previously selected configuration.

This indicates that the better performance on the

num_epochs	dropout_rate	weight_decay	accuracy	precision	recall	f1_score
5	0.1	0.01	0.7977	0.7920	0.9370	0.8584
5	0.4	0.02	0.8000	0.7980	0.9298	0.8589
5	0.2	0.03	0.8062	0.7954	0.9477	**0.8649**
5	0.4	0.05	0.7813	0.8333	0.8323	0.8328
5	0.2	0.1	0.7852	0.7940	0.9073	0.8468
5	0.4	0.1	0.7844	0.8588	0.8026	0.8297

Table 2: Validation result for BERT to address overfitting

	precision	recall	f1_score
BERT	0.7963	0.8808	**0.8364**
BERT-new	0.8210	0.8260	0.8235

Table 3: Comparison between BERT models on test set

validation set might be due to selection bias. These techniques including weight decay and dropout might be incapable of alleviating overfitting issue for this application. We will further discuss the overfitting issue in the section of qualitative analysis.

Next, the confusion matrix of the new BERT model on the test set is presented, from which we can see that the number of misclassification examples of the two types are relatively balanced, which is promising given that the training dataset is slightly inbalanced toward liberals. The model generally performs well.

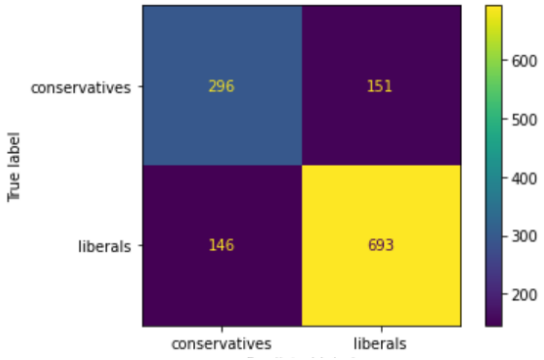

Figure 2: Confusion matrix for new Bert model

5 Qualitative Analysis and Future Work

By sampling from examples misclassified by the final BERT model on the test set, we might be able to get a taste of what kind of mistakes the model tends to make and potentially help us figure out why some types of mistakes are frequently made.

The prominent problem is the model's capability of dealing with long text. When the text is very long, the part longer than 128 words will be truncated by BERT to avoid memory overflow. This causes an issue. When the text is very long, sometimes the key information will be hidden in the middle of the paragraph, which will be lost after the truncation. In the following misclassfied example, the key information to confirm that the author leans towards liberalism is highlighted in blue, which shows that the author believes socialism is well-maintainable with good people in charge, which is a typical justification of socialism made by the left. However, the first half of the text only serves as a introduction to the topic, from which we can confirm nothing. This problem can be solved by increasing the maximum length allowed, which will require more RAM.

> "Does anyone else feel like the discussion of which system is best gets frequently conflated with technology and people? Im not making an argument more of an obersation that i would like some input on. Let me explain. Any system will fail horribly if the people are inadequate to maintain it, for whatever reason. ...(skip hundreds of words)... Phrases like "capitalists are evil." (So are non-capitalists). Or "socialism will distribute resources better." Depends on the people in charge. ... A government is more a vehicle to accomplish what is desired rather than the actual thing causing it."

Besides this long example, Table 4 containing some misclassified short sentences on the test set. Among these examples, even human cannot identify the label of the second, third, fifth and the last examples. The second example features the **lack of background information**, in which the exact reference of "Rioters" is unclear. If it refers to the

rioters in Jan 2020 provoked by Donald Trump, then the sentence is probably liberal. If it refers to the rioters in "Black Life Matter" movement, then it is conservative. The third example contains **ambiguity between sarcasm and extreme views**. It might be written by someone with extreme liberal views, believing every worker should have his/her own press. It can also be written by some conservatives being sarcastic against those liberals who demanding the press to be excessively free. We can not decide the political leaning without extra information like the tones. In the fifth example, the text description alone in the post is unrelated to politics. The political content is in the picture associated with *this post*. The last example is just **neutral discussion**, from which we can't detect its stance either.

These examples, although not many, might corrupt the model if they are in the training set. The pattern learned from these examples whose labels are not solely determined by the text feature might be contradictory against other unseen instances. Overfitting towards these examples might causes some ungeneralizable patterns are captured in the model parameters, which might lead to poor performance on the unseen validation and test set. This might partly explain why we can't solve the overfitting issues merely by tuning up dropout rate and weight decay. In the light of this, one possible solution to alleviate the overfitting problem is to manually identifying or discarding examples whose labels are unidentifiable from the text feature.

Notice that there are some punctuations very helpful as to correctly identify the misclassified examples. In the sixth example in the table, ** is used to wrap some word, which denotes **bold** text, indicating an accentuation of wrapped text. One opposite case is when the word is wrapped in "", which might suggest an ironic tone. One possible direction for the future work might be developing a mechanism to utilize these punctuations to improve the model performance.

6 Conclusion

In this project, a text classification application in political science is introduced to identify the stance of written text as either liberal or conservative. We use DAN, bidirectional LSTM, bidirectional GRU and BERT to do the task and found that BERT can achieve the best performance with a relatively balanced types of errors. To address the most prominent overfitting issue, hyperparameters including dropout rate and weight decay are tuned but the overfit-alleviating effect is not exhibited on the test set. One potential cause of overfitting is hypothesized as the noiseness of the dataset, in which the labels of some examples cannot be inferred from the written text for different reasons. Several misclassification scenarios are found and analyzed. To further improve the model performance, future work might include using a larger RAM, training on a less noisy dataset, and developing some mechanism to utilize punctuations instead of simply discarding them.

References

[1] Clyde Wilcox (2018). Onward Christian Soldiers?: The Religious Right in American Politics. Taylor & Francis. p. 96. ISBN 9780429974533.

[2] Glenn H. Utter; James L. True (2004). Conservative Christians and Political Participation: A Reference Handbook. ABC-CLIO. pp. 51–53. ISBN 9781851095131.

[3] Starr, Paul (2012). "Center-Left Liberalism". In Coates, David; Smith, Kathy (eds.). The Oxford Companion to American Politics. Oxford University Press.

[4] Giridharadas, Anand (April 14, 2021). "Welcome to the New Progressive Era". The Atlantic. Retrieved March 24, 2022.

[5] Hudelson, Richard (1999). Modern Political Philosophy. M.E. Sharpe. ISBN 978-0765600219.

[6] Eulau, Heinz; Fiorina, Morris P. (1981). "Retrospective Voting in American National Elections". Political Science Quarterly. 96 (4): 671. doi:10.2307/2149903. ISSN 0032-3195. JSTOR 2149903

[7] The dataset source: https://www.kaggle.com/datasets/neelgajare/liberals-vs-conservatives-on-reddit-13000-posts

Text	Prediction	Ground Truth
Sen. Ted Cruz Gives Update on 2024 Presidential Run - Says He Would 'Absolutely' Consider Running As Long As Donald Trump Does Not	Liberal	Conservative
Rioters accused of erasing content from social media, phones	Conservative	Liberal
A Worker-Owned Press is the Only Free Press	Liberal	Conservative
Why Socialism Failed	Liberal	Conservative
My MIL got me an apron for Christmas!	Conservative	Liberal
Capitalism explained via a comic—yes, it *is* exploitation	Conservative	Liberal
'Market Societies vs. Societies with Markets'	Liberal	Conservative

Table 4: Some misclassified examples by BERT

Appendices

Hyperparameter selections for Each Model:

embedding_dim	hidden_dim	dropout_rate	accuracy	precision	recall	f1_score
100	10	0.2	0.7494	0.8161	0.7967	**0.8063**
200	30	0.2	0.7191	0.8077	0.7491	0.7773
500	50	0.3	0.7409	0.7920	0.8193	0.8054
1000	100	0.5	0.7284	0.7867	0.8026	0.7946
200	30	0.5	0.7268	0.8086	0.7634	0.7853
1000	100	0.2	0.7253	0.7954	0.7812	0.7882

Table 5: Validation result for DAN, bolded value corresponds to the selected hyperparameters

embedding_dim	hidden_dim	dropout_rate	num_lstm_layers	accuracy	precision	recall	f1_score
100	10	0.2	1	0.7230	0.7947	0.7776	0.7861
200	30	0.2	2	0.7292	0.7700	0.8359	0.8016
500	50	0.3	1	0.6981	0.7938	0.7277	0.7593
1000	100	0.5	2	0.7160	0.7192	0.9287	**0.8106**
200	30	0.5	3	0.7113	0.7859	0.7681	0.7769
1000	100	0.2	2	0.6934	0.7770	0.7455	0.7609

Table 6: Validation result for LSTM, bolded value corresponds to the selected hyperparameters

embedding_dim	hidden_dim	dropout_rate	num_gru_layers	accuracy	precision	recall	f1_score
100	10	0.2	1	0.7198	0.7730	0.8098	0.7909
200	30	0.2	2	0.7120	0.7787	0.7824	0.7805
1000	50	0.5	1	0.6973	0.7534	0.7990	0.7755
200	30	0.5	2	0.7253	0.7387	0.8977	**0.8105**
1000	100	0.2	1	0.7105	0.7371	0.8668	0.7967

Table 7: Validation result for GRU, bolded value corresponds to the selected hyperparameters

num_epochs	feedforward_dropout_rate	accuracy	precision	recall	f1_score
5	0.2	0.7914	0.7845	0.9394	**0.8550**
5	0.5	0.7930	0.7937	0.9239	0.8538
10	0.4	0.7953	0.8074	0.9025	0.8523
10	0.2	0.7953	0.8705	0.8074	0.8378
20	0.6	0.7946	0.8260	0.8692	0.8470

Table 8: Validation result for BERT, bolded value corresponds to the selected hyperparameters