W266 Final Project

Text-Based Political Sentiment Classification

Neha Kumar Sayan Das

UC Berkeley School of Information

{neha.kumar, dasxx179}@berkeley.edu

Abstract – With the upcoming election, the need to effectively summarize candidates' stances on hot-button issues emerges as even the most well-informed voters struggle to decide who best represents their ideologies. A text-based classifier is constructed that analyzes candidate statements and returns stances on a set of key political issues. This study compares the effectiveness of a baseline LSTM model with various levels of fine tuned BERT models to determine political stances from a corpus. Ultimately, the results suggest that a BERT model that is fine-tuned with further in-domain pre-training has a promising approach and warrants further exploration.

1 Introduction

The 2020 Democractic presidential primary race is historically diverse: consisting of numerous candidates from various backgrounds. However, diversity comes with disparate viewpoints, and for an average voter, it is often difficult to discern between these differing attitudes on political topics most relevant to them.

A classifier that automatically summarizes the positions of these candidates would help viewers by succinctly expressing the candidates' platform. Even the most well-intentioned voters who strive to stay informed struggle with the inundation of information from each candidate in a crowded race as of early 2020. Currently, research in this field is limited to sentiment analysis of tweets from President Trump. However, with the proposed model, voters would no longer have to parse through lengthy articles, jumbled debates, and a myriad of advertisements to determine who to cast their valuable vote for. Indeed, the classifier will do the grunt work instead, so the voter can sleep at ease knowing that their vote went towards a candidate who will fight for their beliefs.

Of course, to effectively summarize the innumerable amount of information concerning these candidates, it is important to explore and evaluate the performances of a variety of neural network models and architectures to construct solid baseline models. The authors propose fine-tuning a pre-trained BERT model from the huggingface library to take advantage of bidirectional transformer architectures, considered to be the current state-of-the-art [5].

2 Prior Research and Related Work

Pretrained models allow for organizations with extensive computational resources to train complex neural network architectures on expansive datasets. These pre-trained datasets can then be fine-tuned on domain-specific data to perform specific tasks, such as document and sentence classification, with notably higher accuracy. Prior to Google's BERT, the pretrained OpenAI GPT was the state-of-the art technology [1]. However, its major shortcoming was that it finetuned its parameters based on only the preceding words in a sentence. In other words, the self-attention layers of the Transformer are unidirectional. BERT overcomes this flaw by employing a bidirectional approach, using context on both sides of a masked word to predict it. The group Sun et al. validate the effectiveness of BERT in a series of experiments on the IMDb dataset and conclude that pretraining BERT models on a fairly small corpus make it more effective at executing topic classification tasks [2]. Han et al additionally validate the improvement in BERT accuracy with unsupervised in-domain pretraining, particularly when the lexicon of the test data is likely to contain words that are outside the vocabulary of the normal language [7].

Political data has been a topic of interest analyzed by other NLP researchers. For example, Johnson et al [4] was one of the first in the field to computationally analyze politicians ' stances on key is-

sues based on their tweets, championing a supervised, Probabilistic Soft Logic model. Their model analyzes the frequency of predetermined keywords associated with each issue supplemented with the politician's party preference and temporal data on when the tweet was posted. Since this study focuses on politicians that are all in the same party in the same race, this project strives to create a classifier based purely on text data without any additional information on a candidate's political stance or any numerical or temporal features.

Adhikari et al [6] use the BERT-large model for document classification and found that the incremental benefit between BERT-large was only slightly higher than a bidirectional LSTM, noting that the bidirectional LSTM is much more computationally efficient to train than BERT-large ' s 340M parameters. However, other studies have shown a more marked improvement between the bi-LSTM architecture and BERT classification, such as d ' Sa et al ' s study on the classification of toxic speech [3]. To the best of the authors' knowledge, the BERT architecture has not yet been applied to a political corpus to determine candidate polarity on key issues.

3 Background and Approach

3.1 Bert Architecture

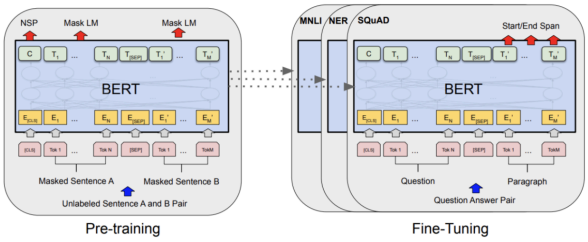


Figure 1: BERT Architecture from [1]

BERT uses the Transformer, an attention mechanism that is able to learn contextual relations between different words in a text. A basic Transformer consists of two major components: an encoder to read in the text input and a decoder to produce a prediction for the task. BERT is unique because it is bidirectional in nature, so the Transformer encoder is able to read the sequence of text all at once, allowing the model to learn the context of a word based on its entire surroundings. This fact in particular is why BERT has obtained state-of-the-art results on various NLP tasks. In this paper, a pre-trained BERT model is fine-tuned and run on a classification output layer. The BERT model is fine-tuned by using HuggingFace's Transformers library, a library that provides a pytorch interface for working with BERT. Using a script included in huggingface, BERT is fine-tuned on the different levels of training data using a masked language modeling loss.

Ultimately, it was interesting to see the application of BERT in a political setting, for not only did BERT perform much better than the baseline model with regards to F1 score, but as shown in the extremity score visualizations in Figure 8, BERT was decently successful in classifying the different political categories from the test data. Moreover, this project explores various fine-tuning strategies to optimize the different layers of BERT that capture different levels of textual information by experimenting with hyperparameters and training: the learning rate is varied to account for issues such as the prominent overfitting problem, and the model is further pre-trained with in-domain training data [2].

3.2 Evaluation Strategy

The goal of the model is to take in a political candidate's statements, and output whether they are for or against 7 predetermined political issues on a scale of 0 to 1. This brings up the question of an effective evaluations strategy. For instance, if one model says candidate X has a score of 0.7 on the issue of free college tuition, and another model says that same candidate scores 0.8 for college tuition, which one is more accurate? There is no sound numeric ground-truth that can be used in this case.

To resolve this, the model is trained on sentences of labeled articles that are clearly for or against a given issue. There are 7 total issues, and 2 sides for each issue, meaning that this is now a classification problem where k = 14. 10% of the sentences are reserved as the development dataset. Since these sentences are from articles that have a "known" label (ex: this sentence is from a pro-life article), the micro F1 score was used to objectively determine the relative performance of different models compared to one another.

The best-performing model will be run against the statements from various Democratic candidates. Each candidate's statements will be split on the sentence level, and each sentence will be assigned one of the fourteen labels. The results from all the sentences

will be aggregated per candidate to assess their overall stance on each of the 7 issues, with the aggregation methodology discussed in Results. Note that this approach is using a single-label multiclass approach. The aggregation simulates a multi-label effect as candidates can score highly for multiple classes when all sentences from their statements are combined.

Drawbacks of the model should be discussed. Firstly, individual sentences in the training data may not necessarily have any relevance to the overall topic when taken out of context. (For example "She agreed with him.") These types of sentences pollute the training data. Additionally, there can be sentences in the testing data that are not related to any of the predetermined topics, revealing the need to have the ability to not assign any label if a sentence is off-topic. Despite these drawbacks, the ability to have an objective evaluation metric on the development data drove the decision by the researchers to use the approach described above.

3.3 Datasets and Data Preparation

Three levels of training data were utilized to fine tune BERT and run the Sequence Classifier Transformer from the Huggingface library [5]: a 48M word political corpus composed of politically concerned news articles (referred henceforth as D1 data), a set of news articles that were scraped from websites concerning the set of key political topics (referenced henceforth as D2 data), and lastly a labeled, compact set of articles on each side of each political topic (referenced as D3 data). These datasets were constructed with the purpose of refining the model's understanding of language in a political context before executing the final sequence classification task.

Testing data for each political candidate was obtained by scraping text from their interviews with the New York Times and their platforms from their campaign websites. The trained model was also tested on Trump tweets, a publicly available dataset, as a point of comparison against the Democratic candidates.

To prepare the datasets for training, each sentence was split into its own record. Sentences that were less than 10 characters were removed from the dataset to ensure short phrases – such as "Yes." "No." "Why?" "Sure" – were not factored into the model. Furthermore, to fine-tune BERT for the target task of classification, it is important to consider that the maximum sequence length of BERT is 512. For this study, the text from these datasets were preprocessed with a maximum sequence length of 128 words, for the average sentence length in the datasets were less than 128 words, therefore it minimized the amount of time and padding needed rather than setting the maximum sentence length to the full 512. Any sentences that were under 128 words were padded to ensure each embedding vector for BERT is the same length. The preprocessing steps can be found within the project git repo in the appendix.

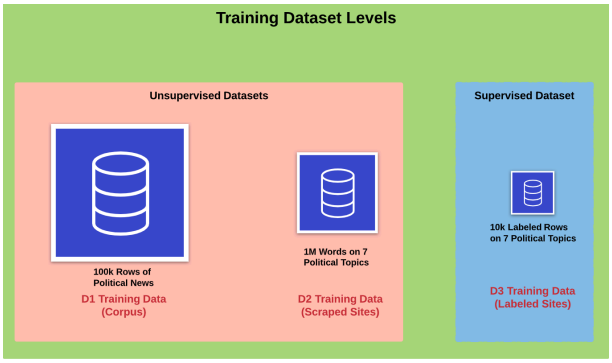


Figure 2: Schematic of Different Training Datasets

3.4 Model Construction

The model is a multi-class, single-label classifier that assigns a single label on each individual sentence within a block of text. The assigned labels for each class are aggregated to summarize stance on a set of given issues, as will be discussed in the Results section of this paper. At its core, the model is a Sequence Classifier Transformer using the pretrained bert-base-uncased model as a starting point. This model was further fine-tuned using the run_language_modeling.py script in the examples directory of the huggingface github repo. In terms of tooling, the authors opted for the pytorch implementation initially on Sagemaker and then completed the study on Google Colab as the example scripts were more readily usable on Colab as opposed to the Sagemaker environment.

A variety of model architectures were created to determine which design led to the most optimal prediction. As mentioned in the Datasets section, there are three different levels of data that exist in the creation of the model. Therefore, models were constructed based on permutations of these layers of data. For instance, a model that was created based

solely on D3 data was compared to a model constructed based on D3 data pre-trained on D2 and D1 data. It is important to note that the D3 data must be included in each of these model permutations, for it is the classification layer that is used to calculate the final results.

Adapting BERT to the task of sentiment classification is tricky, for there are several factors to consider in order to optimize the performance. Firstly, the various layers that exist in BERT encompass different textual information, so it is important to understand which layer performs best for the task. For example, the lower layers of a BERT model capture more broad information, whereas the deeper layers are more specific [2]. Furthermore, it is important to be aware of the issue of overfitting BERT. To solve this, it is important to alter hyperparameters such as the learning rate until a favorable model is achieved.

As such, apart from the learning rate, other hyperparameters were varied to achieve a desirable model: including batch size, number of epochs, learning rate, Adam epsilon coefficient, and whether D1 and D2 pretraining was performed to begin with.

3.5 Bert Architecture

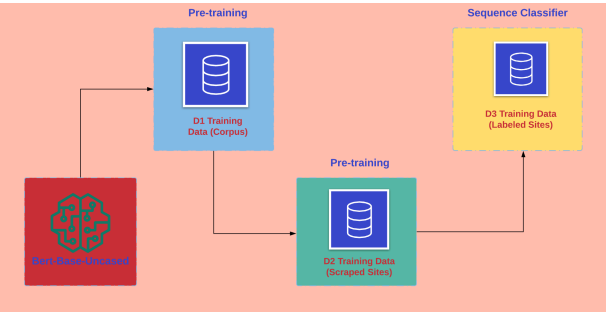


Figure 3: Schematic of Model Architecture

4 Results

4.1 Preliminary Results and Takeaways

A baseline model was generated using the LSTM architecture, and it generated an F1 score of 0.12 on the development dataset. To contrast, the bert-base-uncased model is pre-trained on the first two levels of training data with unsupervised masked language modeling and next sentence prediction tasks to enhance its understanding of words in a political context. Afterwards, BERT is trained for the target task of sentiment classification by using supervised learning on the third, labeled dataset. Training on this supervised set ultimately allows BERT to act as a classifier that provides predictions for the text classification. This process is inspired by the research paper from Sun [2].

In the early iterations of the study, the authors came across a handful of challenges. Note that the earliest iterations ran the study on 3 topics instead of the full 7 to get an understanding of the data pipeline and workflow.

One key challenge faced was the tooling to use. Initially, Sagemaker was the tool of choice. However, neither Huggingface nor sagemaker had documentation on the AWS-specific parameters that needed to be defined. This led to many training jobs crashing even with a small number of epochs. Ultimately the study was conducted on Google Colab pro, which did not give the option to train on multiple GPUs and parallelize jobs across GPUs like Sagemaker, but was much quicker to set up. For Google Colab pro to be a feasible tool, D1 pretraining had to be split into 10 smaller subsections, each taking 2-3 hours, as Google Colab can terminate processes mid-way.

Another key challenge was ensuring quality of the scraped D2 data. The scraper pulled the html for each article via BeautifulSoup; however, this picked up many formatting strings that contaminated the corpus. To ensure high data quality, these html strings were manually removed as there was no clear pattern that could remove these programmatically.

Lastly, there were some signs of overfitting in dev versus training data. Training data reached F1 scores well over 0.9 while dev data F1 scores barely crossed 0.7. This was resolved by increasing the learning rate, so the final parameters have less precision and are less likely to overfit. Also, special care was taken to ensure that the curve of the dev F1 models did not ever drop when plotted against epochs.

4.2 Hyperparameter Tuning

The F1 score for the development data was optimized by experimenting with the hyperparameters mentioned above. The table of results that summarize the F1 scores of the baseline and various BERT models is shown in Figure 4. The model that proved to be most effective at classifying the text is a pre-trained BERT model further pre-trained on both layers of training data ultimately fine-tuned on a classification layer, resulting in an F1 score of 0.722 on the development dataset. The full table of different

hyperparameter values and their results are in the Appendix.

Pretraining?	Model	F1 Score
No	LSTM	0.12
No	BERT	0.70
Yes. D1 + D2	BERT	0.72

Figure 4: F1 Scores

The figures below show the hyperparameter optimization exercises to determine the optimal configuration to achieve the highest possible F1 score. Figure 6 shows each hyperparameter being adjusted while holding other hyperparameters constant. For cleanliness, only the F1 score on the dev data is plotted. Figure 5 compares the F1 score on the training dataset compared to the development dataset. This reveals that after 2 epochs, the incremental benefits of each epoch begin to diminish on the development dataset; however, the training dataset continues to climb, so this result is indicative of overfitting on the training dataset.

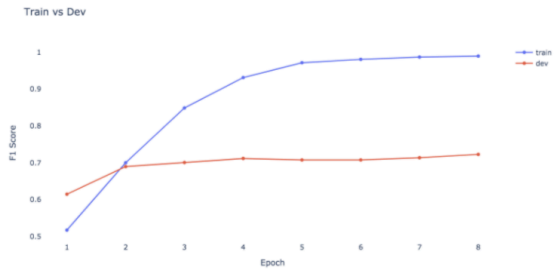


Figure 5: F1 of Train vs. Dev

Ultimately, the ideal hyperparameters were (1) pretraining on both D1 and D2 data, (2) a learning rate of 1e-4 on D1, D2, and D3 training, (3) an Adam epsilon score of 1e-9, and (4) a batch size of 8. These various hyperparameter adjustments are shown in Figure 6.

Once the best-performing model was determined by optimizing the F1 score on the development dataset, a metric was developed to characterize the polarity of a corpus on a given political issue. This metric, developed by the authors and shown in Figure 7, is referenced henceforth as the "extremity score:"

In the formula, Pro and Anti represent the pro and anti statements respectively. The output of this formula is a value between 0 and 1, where 0 repre-

Figure 6: From top to bottom, left to right, we are demonstrating the effects of D1 learning rate, D2 learning rate, D3 Adam Epsilon, D3 Learning Rate, Pretraining Effect, Batch Size. Enlarged versions of the graphs and detailed results from specific hyperparameters are in the Appendix.

$$\frac{1}{2} * \left(\frac{sum(Pro) - sum(Anti)}{sum(Pro) + sum(Anti)} + 1 \right)$$

Figure 7: Extremity Score Formula

sents an extremely anti statement and 1 represents a completely pro statement. From a coding implementation standpoint, a try-except clause is added to output 0.5 if there are no statements provided that are for or against a given issue.

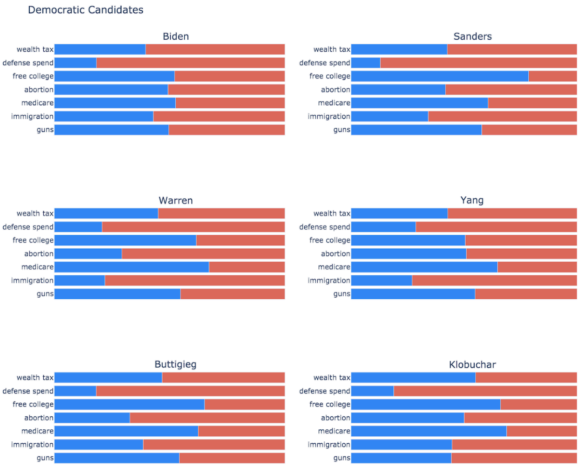


Figure 8: Candidates Extremity Score Visualization

5 Discussion of Extremity Scores

From the extremity scores, we can see some key trends that the model was able to capture, as well as some unexpected results. Firstly, we do see that the model was able to detect Sanders's strong position on free college; however it did not capture his position on medicare-for-all. For almost all candidates, the model did not capture that they were against guns

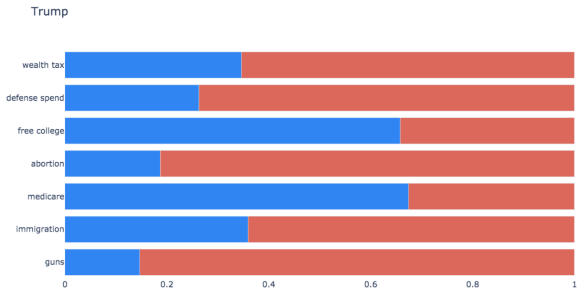


Figure 9: Trump Extremity Score Visualization

or leaning away from guns. Looking holistically at the extremity scores of the top two candidates, Biden and Sanders, we can also see that Biden is depicted as a more “moderate” candidate with extremity scores that hover around 0.5 compared to Sanders.

When looking at Donald Trump’s tweets, the extremity scores deviate further from expectations. Firstly, it suggests that Trump is against gun usage and would be a strong champion of gun control measures Secondly, it suggests that he is a strong advocate of free college and medicare for all. This could be a direct result of the tweets actually being about neither of the topics, and the model still strives to categorize each sentence to exactly one class, leading to the heavy polarization.

6 Comparison of Results

Regarding the D3 model pre-trained on both the D1 and D2 layers, it was unnatural to see that there was no conceivable difference when experimenting with the batch size, for a batch size of 16 did not perform better or worse than a batch size of 8. This detail was particularly mystifying because other research groups attain success with larger batch sizes that are greater [2]. It is possible that 16 compared to 8 is not significant enough of an increase in batch size to see a notable difference with the given dataset size, so larger batch sizes should be explored. Additionally, it appears that the ideal learning rate for the classification layer is 1E-4, and the ideal Adam Epsilon value is 1E-9 or 1E-8, as both epsilon values led to similar results.

Overall, the maximum F1 score that was attained is approximately 0.72, a marked improvement from the LSTM baseline which scored only 0.12. Overall, the additional training of the D2 layer made a small improvement in the model’s F1 score when compared to just fine-tuning the model with the D1 layer. It is possible that the model may eventually peak at this F1 value due to overfitting. Regardless, the model appeared to benefit from additional training data in D2, despite the marginal improvement.

However, it appears that similar to the results of other research on strategies to fine-tuning BERT, further in-domain pre-training does augment the results [2]. The F1 score of the bare bert-base-uncased model with an D3 model classification layer reached a max of .7. While the performance increase is marginal (.72 compared to .7), it is possible that experimenting with different types of test data and possible augmentation of the training data may result in a more profound difference, for other researchers utilizing similar strategies agree that in order for in-domain pre-training to be successful, it is necessary to have a large corpus of unlabeled data [2].

In summation, as mentioned, larger batch sizes should be explored to see how the model reacts, for other research groups noting success with batch sizes of 24 and even 32 [2]. Furthermore, larger corpus sizes should be used in future work for training, and it would be interesting to explore different types of test data as well.

7 Conclusion

Automation is a pervasive theme of the 21st century, as society aims to accomplish tasks as efficiently as possible. As information becomes more abundant and NLP algorithms become increasingly capable to detect nuances in context, text-based classification models become a viable and practical way for voters to make an informed assessment of political figures. While this paper only examined presidential candidates, a similar approach can be leveraged for local elections that typically get less of a spotlight and suffer from poorer voter turnout due to apathy or a lack of information on candidates. There is obvious room for improvement such as partnering with large sources of data such as Google to pretrain BERT models specifically for this use case. Thus, the findings here are to be taken as a directional improvement to summarizing political stances rather than a finished solution.

8 Further Research

Two clear courses for further study would be recommended by the authors. Firstly, more extensive computational resources should be leveraged to pretrain the model on an even larger dataset of unlabeled in-domain data.

Secondly, the authors realized that sentences that were not directly related to any topic were getting labeled as for or against the seven chosen issues. To address this phenomenon, a different approach would need to be employed. As a first step, each sentence should first be flagged on whether or not it is related to a certain topic before being assigned a side for the topic. This means there would be multiple binary classification models, one for each topic, to determine whether the model is related to the sentence subject. Then, all the sentences related to a topic would be passed through another binary classifier to determine whether the sentence is for or against the topic.

Literature References

[1] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018). https://www.aclweb.org/anthology/N19-1423.pdf

[2] Sun C., Qiu X., Xu Y., Huang X. (2019) How to Fine-Tune BERT for Text Classification?. In: Sun M., Huang X., Ji H., Liu Z., Liu Y. (eds) Chinese Computational Linguistics. CCL 2019. Lecture Notes in Computer Science, vol 11856. Springer, Cham https://arxiv.org/pdf/1905.05583.pdf

[3] Ashwin Geet d'Sa, Irina Illina, Dominique Fohr. BERT and fastText Embeddings for Automatic Detection of Toxic Speech. SIIE 2020 - Information Systems and Economic Intelligence, Feb 2020, Tunis, Tunisia. Ffhal-02448197 https://hal.inria.fr/hal-02448197/document

[4] Johnson, K., & Goldwasser, D. (2016). Identifying stance by analyzing political discourse on twitter. In Proceedings of the First Workshop on NLP and Computational Social Science. https://www.aclweb.org/anthology/W16-5609.pdf

[5] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Brew, J. (2019). Transformers: State-of-the-art Natural Language Processing. arXiv preprint arXiv:1910.03771. https://arxiv.org/pdf/1910.03771.pdf

[6] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. (2019) DocBERT: BERT for Document Classification. David R. Cheriton School of Computer Science, University of Waterloo. https://arxiv.org/pdf/1904.08398.pdf

[7] Xiaochuang Han, Jacob Eisenstein. (2019). Unsupervised Domain Adaptation of Contextualized Embeddings for Sequence Labeling. Georgia Institute of Technology. https://arxiv.org/pdf/1904.02817.pdf

Appendix

Project Repo

https://github.com/dasxx179/W266-Final-Project

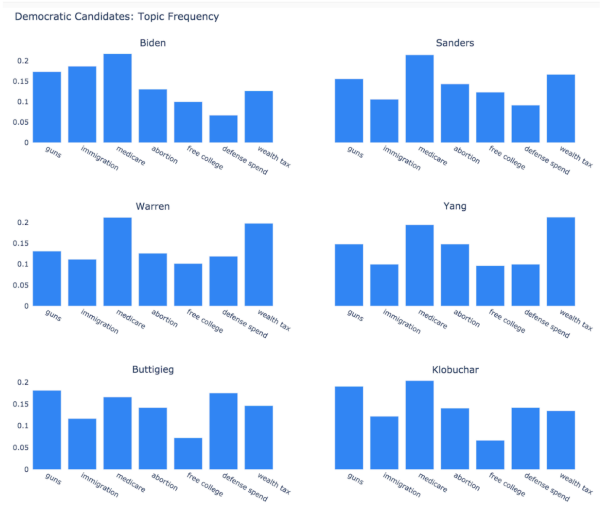
Helpful References

https://github.com/awslabs/amazon-sagemaker-examples/tree/master/sagemaker-python-sdk/pytorch_mnist

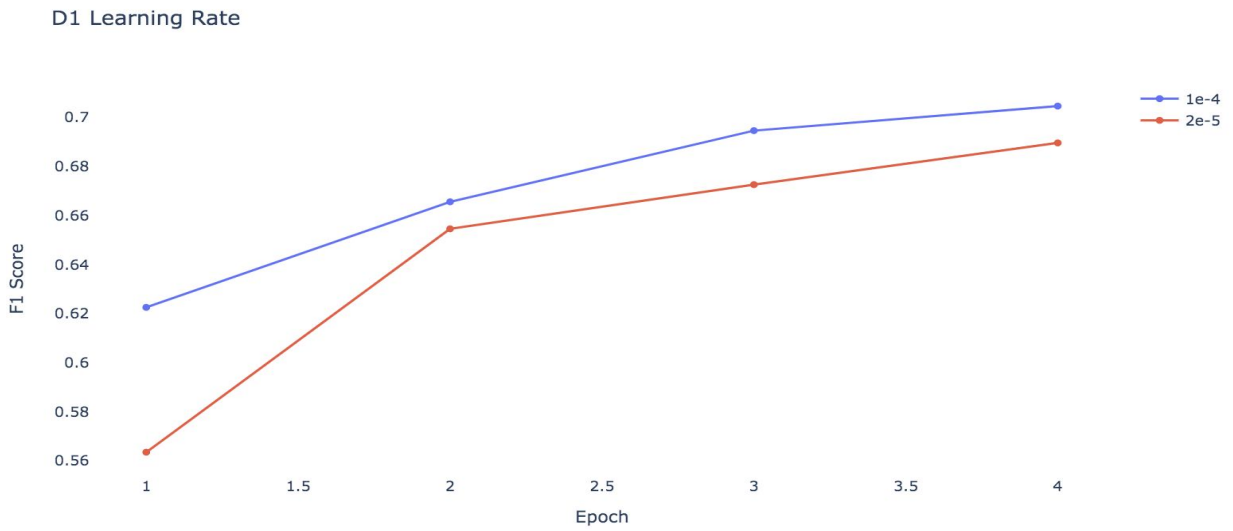
https://mccormickml.com/2019/07/22/BERT-fine-tuning/

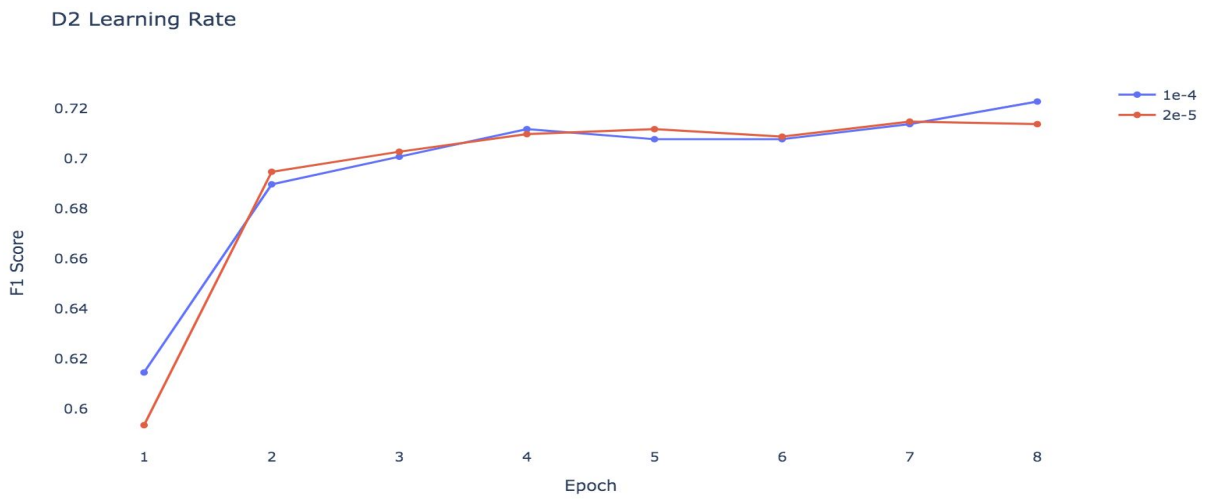
https://github.com/huggingface

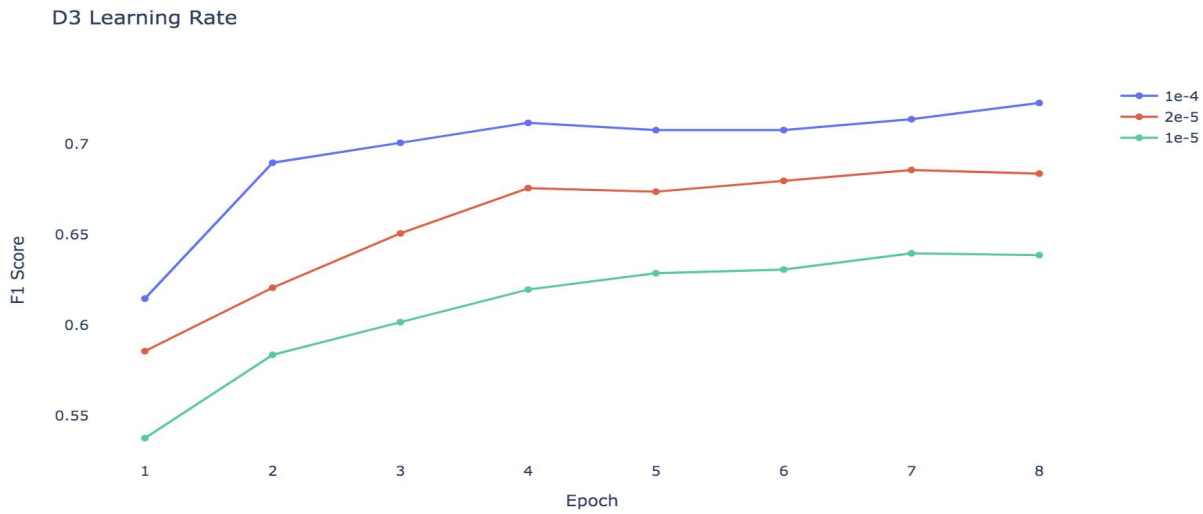
Extra Figures and F1 Score Data



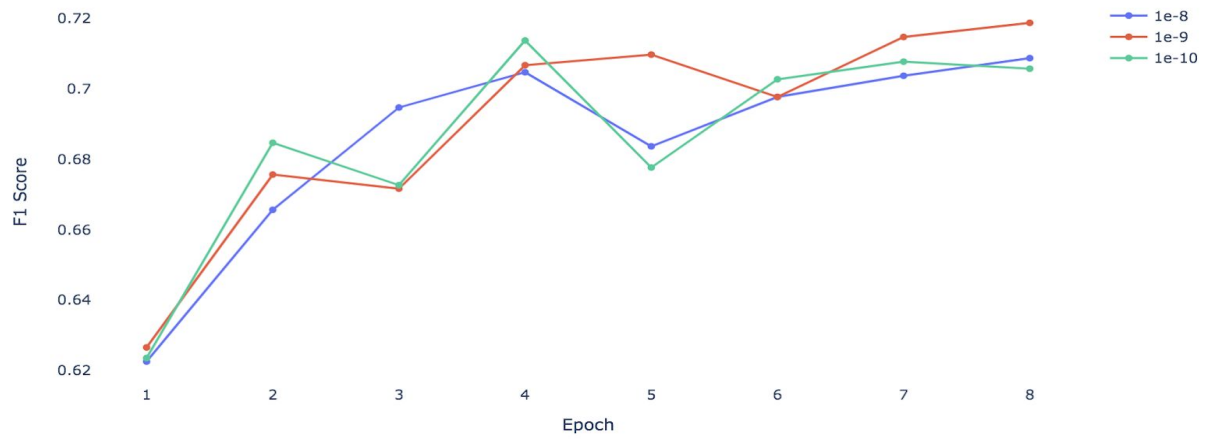
The figure above takes the sum of the pro and anti labels for each topic to produce a frequency distribution of each topic for each political candidate. In conjunction with the extremity scores, a view like this will allow an understanding of which issues are a more significant portion of the candidate's platform.


D1 Learning Rate
F1 Score
Epoch
1e-4
2e-5
0.7
0.68
0.66
0.64
0.62
0.6
0.58
0.56
1
1.5
2
2.5
3
3.5
4

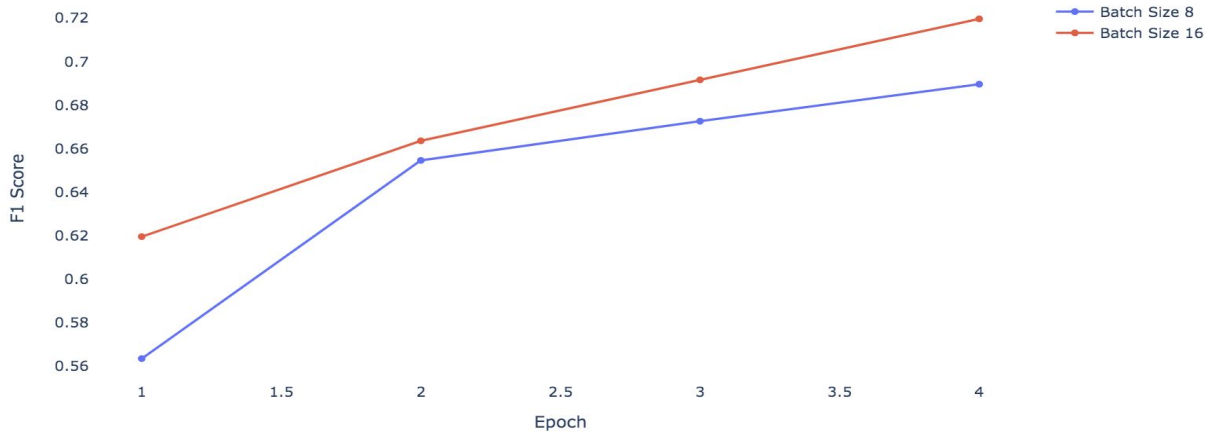

D2 Learning Rate
F1 Score
Epoch
1e-4
2e-5
0.72
0.7
0.68
0.66
0.64
0.62
0.6
1
2
3
4
5
6
7
8


D3 Learning Rate
F1 Score
Epoch
1e-4
2e-5
1e-5
0.7
0.65
0.6
0.55
1
2
3
4
5
6
7
8

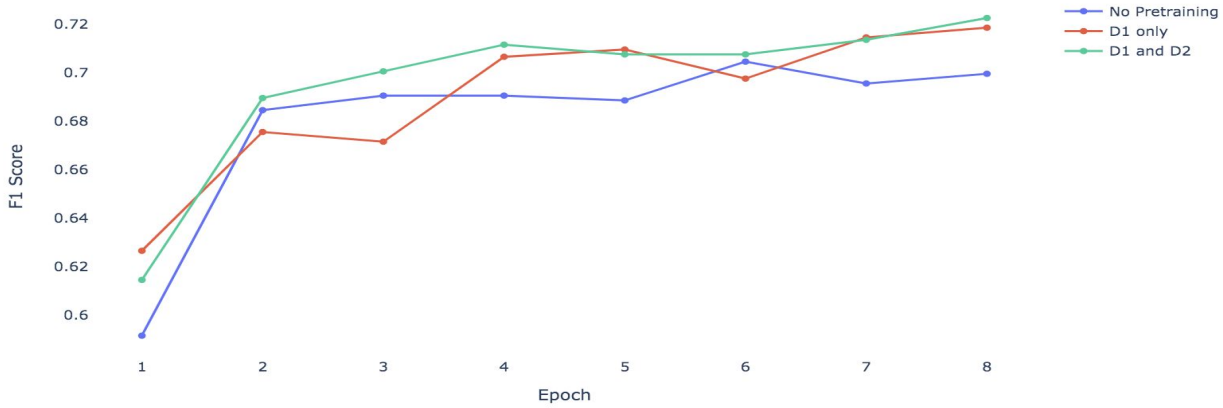
D3 Adam Epsilon



Batch Size



Pretraining Effect



Model	Pretraining	L1 Corpus Size	L1 Learning Rat	L1 Epochs	L1 Per GPU Bat	Num Topics	L2 Learning Rat	L2 Epochs	L2 Per GPU Bat	L3 Adam Epsilo	L3 Learning Rat	L3 Epochs	L3 Per GPU Bat	F1 Train	F1 Dev
LSTM	No														
CNN + Fully Con	No														
BERT	No	100000	1.00E-04			3				1.00E-08	1.00E-04	4	8	[0.5441734417	[0.6009732360
LSTM	L1 + L2														
CNN + Fully Con	L1 + L2														
BERT	L1	100000	1.00E-04	4	8	7	NA	NA	NA	1.00E-09	2.00E-05	4	8	[0.467067200712	[0.570570570570
BERT	L1	100000	1.00E-04	4	8	7	NA	NA	NA	1.00E-09	2.00E-05	8	8	[0.466065865598	[0.579579579579
BERT	L1	100000	1.00E-04	4	8	7	NA	NA	NA	1.00E-09	1.00E-06	8	8	[0.11014686248	[0.196196196196
BERT	L1	100000	1.00E-04	4	8	7	NA	NA	NA	1.00E-09	1.00E-04	8	8	[0.519359145527	[0.626626626626
BERT	L1	100000	1.00E-04	4	8	7	NA	NA	NA	1.00E-08	1.00E-04	8	8	[0.514241210502	[0.622622622622
BERT	L1	100000	1.00E-04	4	8	7	NA	NA	NA	1.00E-10	1.00E-04	8	8	[0.513684913217	[0.623623623623
BERT	none	NA	NA	NA	NA	NA	NA	NA	NA	1.00E-09	2.00E-05	8	8	[0.463618157542	[0.585585585585
BERT	none	NA	NA	NA	NA	NA	NA	NA	NA	1.00E-09	1.00E-05	8	8	[0.402759234534	[0.547547547547
BERT	none	NA	NA	NA	NA	NA	NA	NA	NA	1.00E-09	1.00E-04	8	8	[0.516243880729	[0.591591591591
BERT	L1 + L2	100000	1.00E-04	4	8	7	1.00E-04	4	8	1.00E-09	1.00E-04	8	8	[0.517022696925	[0.614614614614
BERT	L1 + L2	100000	1.00E-04	4	8	7	1.00E-04	4	8	1.00E-09	2.00E-05	8	8	[0.473520249221	[0.585585585585
BERT	L1 + L2	100000	1.00E-04	4	8	7	1.00E-04	4	8	1.00E-09	1.00E-05	8	8	[0.419225634178	[0.537537537537
BERT	L1 + L2	100000	1.00E-04	4	8	7	2.00E-05	4	8	1.00E-09	1.00E-04	8	8	[0.508678237650	[0.593593593593
BERT	L1 + L2	100000	1.00E-04	4	8	7	2.00E-05	4	8	1.00E-09	2.00E-05	8	8	[0.471517578994	[0.588588588588
BERT	L1 + L2	100000	1.00E-04	4	8	7	2.00E-05	4	8	1.00E-09	1.00E-05	8	8	[0.419114374721	[0.537537537537
16 BATCH TRIALS															
BERT	L1	100000	2.00E-05	1	16	7	NA	NA	NA	1.00E-09	2.00E-05	4	8	[0.463061860258	[0.592592592592
BERT	L1	100000	2.00E-05	1	16	7	NA	NA	NA	1.00E-09	1.00E-04	4	8	[0.514018691588	[0.612612612612
BERT	L1	100000	2.00E-05	1	16	7	NA	NA	NA	1.00E-08	1.00E-04	4	8	[0.511125945705	[0.600600600600
BERT	L1	100000	2.00E-05	2	16	7	NA	NA	NA	1.00E-08	1.00E-04	4	8	[0.505117935024	[0.619619619619
BERT	L1	100000	2.00E-05	4	16	7	NA	NA	NA	1.00E-08	1.00E-04	4	8	[0.503115264797	[0.610610610610
BERT	L1	100000	2.00E-05	4	8	7	NA	NA	NA	1.00E-08	1.00E-04	4	8	[0.483867378727	[0.563563563563
BERT	L1	100000	2.00E-05	4	16	7	NA	NA	NA	1.00E-08	1.00E-04	8	8	[0.521680216802	[0.581508515815
BERT	L1	100000	2.00E-05	2	16	7	NA	NA	NA	1.00E-08	1.00E-04	8	8	[0.520596205962	[0.579075425754
BERT	L1 + L2	100000	2.00E-05	1	16	7	2.00E-05	4	16	1.00E-08	1.00E-04	4	8	[0.520325203252	[0.586374695806
BERT	L1 + L2	100000	2.00E-05	2	16	7	2.00E-05	4	16	1.00E-08	1.00E-04	4	8	[0.520054200542	[0.605839416058
BERT	L1 + L2	100000	2.00E-05	4	16	7	2.00E-05	4	16	1.00E-08	1.00E-04	4	8	[0.528184281842	[0.600973236009
BERT	L1 + L2	100000	2.00E-05	1	16	7	1.00E-04	4	8	1.00E-08	1.00E-04	8	8	[0.511014686242	[0.593593593593
BERT	L1 + L2	100000	2.00E-05	2	16	7	1.00E-04	4	8	1.00E-08	1.00E-04	8	8	[0.504116599997	[0.583583583583
BERT	L1 + L2	100000	2.00E-05	4	16	7	1.00E-04	4	8	1.00E-08	1.00E-04	8	8	[0.510013351117	[0.609609609609
BERT	L1 + L2	100000	2.00E-05	1	16	7	1.00E-04	4	16	1.00E-08	1.00E-04	8	8	[0.509457053847	[0.606606606606
BERT	L1 + L2	100000	2.00E-05	2	16	7	1.00E-04	4	16	1.00E-08	1.00E-04	8	8	[0.504339118847	[0.612612612612
BERT	L1 + L2	100000	2.00E-05	4	16	7	1.00E-04	4	16	1.00E-08	1.00E-04	8	8	[0.507231864706	[0.595595595595
BERT	No	100000	1.00E-04			7				1.00E-09	1.00E-04	4	8	[0.516243880725	[0.591591591591