FINE-TUNING BERT FOR SENTIMENT ANALYSIS

MICHAEL LIU, GENFU LIU

INTRODUCTION:

In this project, we will be exploring BERT (Bidirectional Encoder Representations from Transformers), a state-of-the-art NLP model introduced by the researchers at Google AI Language back in 2018, and also using it to enhance the performance of sentiment analysis on the IMDb movie reviews dataset. We fine-tune the pre-trained BERT model using IMDB dataset, aiming to experiment as well as improve the strategies and approaches used in their paper - **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**. Our results demonstrated the effectiveness of BERT in capturing complex language patterns and achieving superior performance compared to traditional sentiment analysis methods and other deep learning-based models.

APPROACH SECTION

In this section, we describe our approach to fine-tuning the BERT model for sentiment analysis on the IMDb dataset, including the detailed fine-tuning process we took to create our model. We've also created a Google Colab Notebook where we created the model.

https://colab.research.google.com/drive/1QJb_KHLsofFC4Cvn8xI3FbtiiQjftkkp?usp=sharing

DATASET INTRODUCTION

We will be using the IMDB Dataset from Kaggle. This is a popular dataset that can be used for sentiment analysis tasks, it consist of 50,000 different movie reviews, half of them are categorized as **positive** while the other half are **negative**.

```
Example training data:
Review: One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked.
Sentiment: 1
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. The...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there's a family where a little boy ...	0
4	Petter Mattei's "Love in the Time of Money" is...	1

DATA LOADING/PREPROCESSING

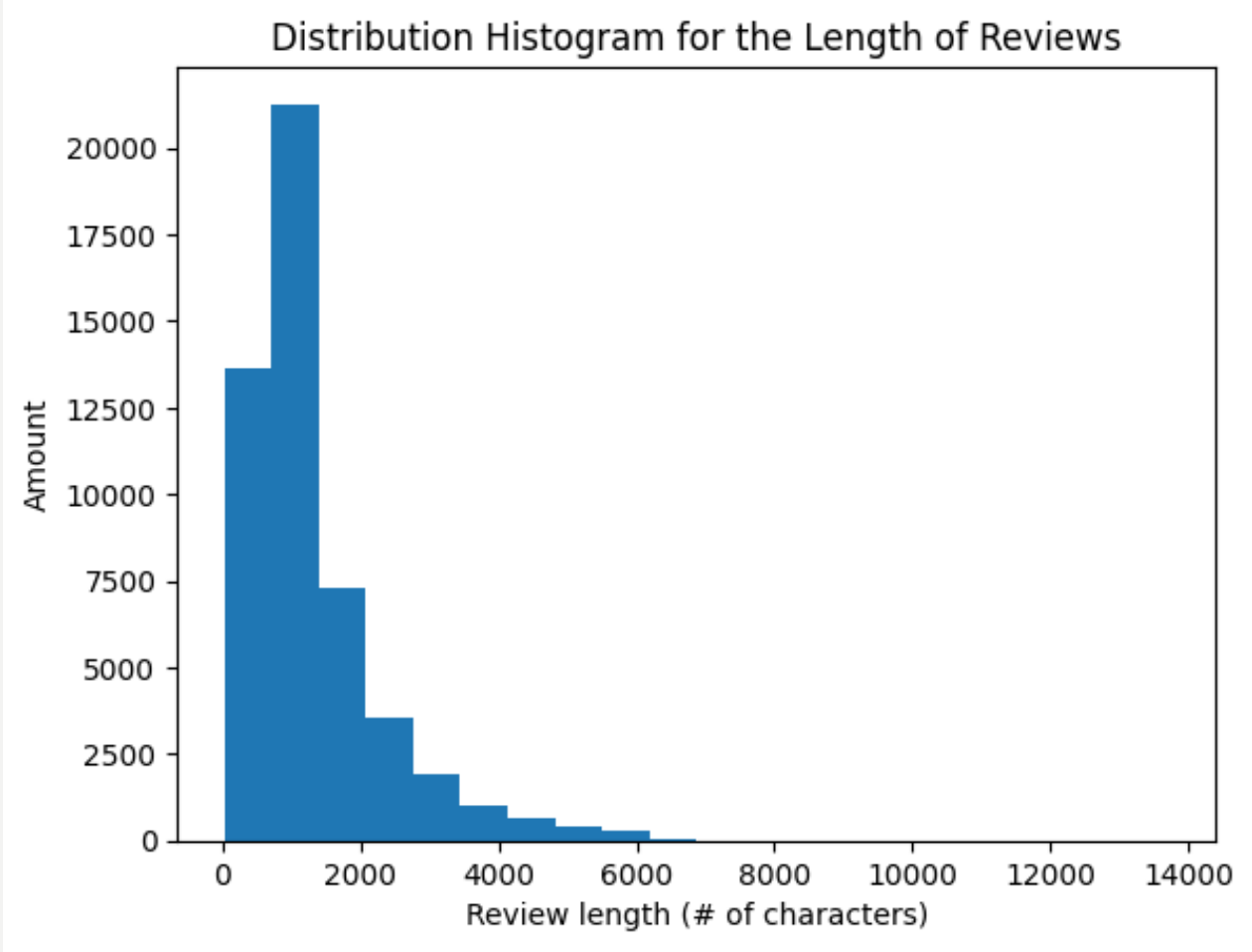
The first step is loading the IMDB dataset in and split the training set into training, validation and testing subsets as usual. After that, the dataset must also be preprocessed to ensure compatibility with the BERT model.

```
imdb_dataset = pd.read_csv('CS462-Final-Project/IMDB Dataset.csv')

# Changing the labels to numerical form, 1 is positive, 0 is negative
imdb_dataset['sentiment'] = imdb_dataset['sentiment'].replace('positive', 1).replace('negative', 0)

# Spliting the data into training (80%), validation (10%), and testing (10%) sets.
train_data, others = train_test_split(imdb_dataset, test_size=0.2)
validation_data, test_data = train_test_split(others, test_size=0.5)
```

DATA ANALYSIS

Next, we analyze the data to help us choose the different hyperparameters for the model

As we can see from the histogram, most of the movie review data seems to have around 1000 - 2000 characters. This is somewhat good for us since the BERT models we will be using can only handle sequence lengths up to 512 tokens. This means that we will have to truncate most of our dataset to 512 tokens to satisfy the model's inputs, but also means that we will be wasting probably half of our data. Still, this is much better than having reviews that are too short in length, which then we would need to pad them with extra tokens, which might lead to inefficiencies in model training and may negatively impact model performance



HYPERPARAMETERS

Here are all the hyperparameters that we will be using to train the model:

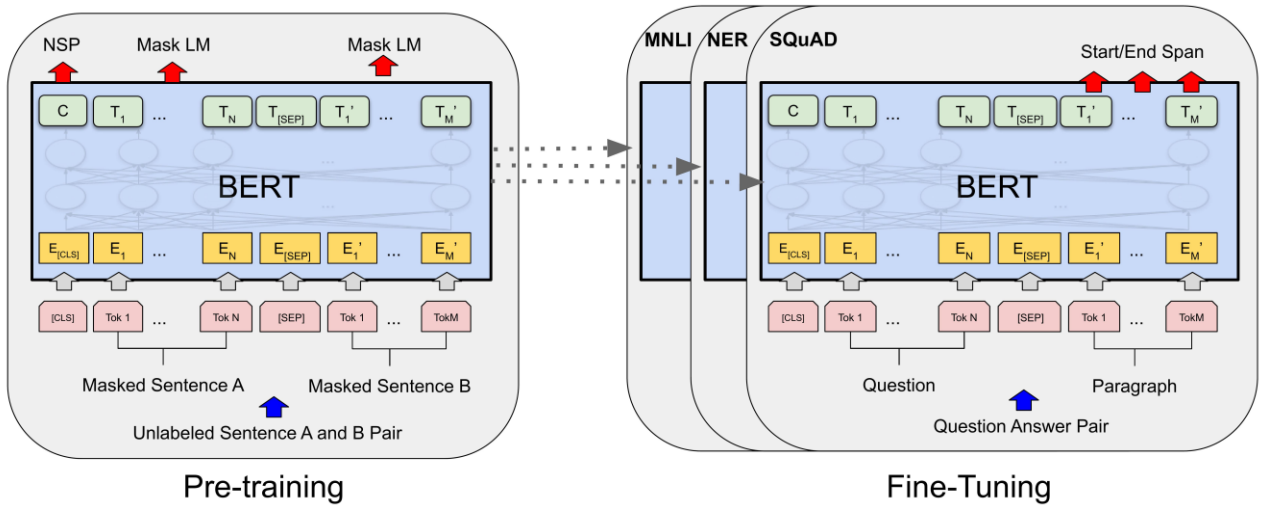
- `max_len`: This is the max token length that our inputs can have as mentioned above
- `batch_size`: We choose a batch size of `32` which is the same as the model trained with the GLUE benchmark in the paper. Since the GLUE benchmark also has a sentiment analysis dataset on movie reviews similar to our imdb dataset. We decide to use some of the same hyperparameter as them
- `epochs`: We will also be fine-tuning the model for 3 epochs similar to the model in the paper for experimentation. Later on we will increase this to produce a better final model.
- `learning_rate`: The paper recommends some best learning rates for this task which are among `5e-5`, `4e-5`, `3e-5`, and `2e-5`, we will try all of them to optimize for the best one

```
[6]  max_len = 512 # Choosing the maximium length possible for this BERT model as metioned above
     batch_size = 32 # GLUE
     epochs = 3
     learning_rate = 1e-05 # 5e-5, 4e-5, 3e-5, and 2e-5
```

LOADING AND CONFIGURING THE PRE-TRAINED BERT MODEL

Now we are ready to load the model!

We use the pre-trained BERT base model as the starting point for fine-tuning. The BERT base model consists of 12 layers, 12 self-attention heads, and 768 hidden units, with a total of 110 million parameters. We add a classification layer on top of the BERT base model to perform binary sentiment classification, where the output logits are mapped to the corresponding sentiment labels, as well as a sigmoid function to map the output to a probability distribution over the two classes.



TRAINING THE MODEL

We first define the optimizer and the loss function and train the model as usual.

We chose the **Adam Optimizer** and **CrossEntropyLoss** as the Loss Function for our model.

Next, we follow the usual steps for training a neural network, for each epoch, we use our DataLoader to iterate over the batches and do the forward pass, loss computation and backward pass for each batch. To see how our model is doing, we use the simple function accuracy_score from the sklearn liberary to compute the accuracy of our results and print it out. At the end of each epoch, we use the validation set to monitor the performance of our model based on new, unseen data.

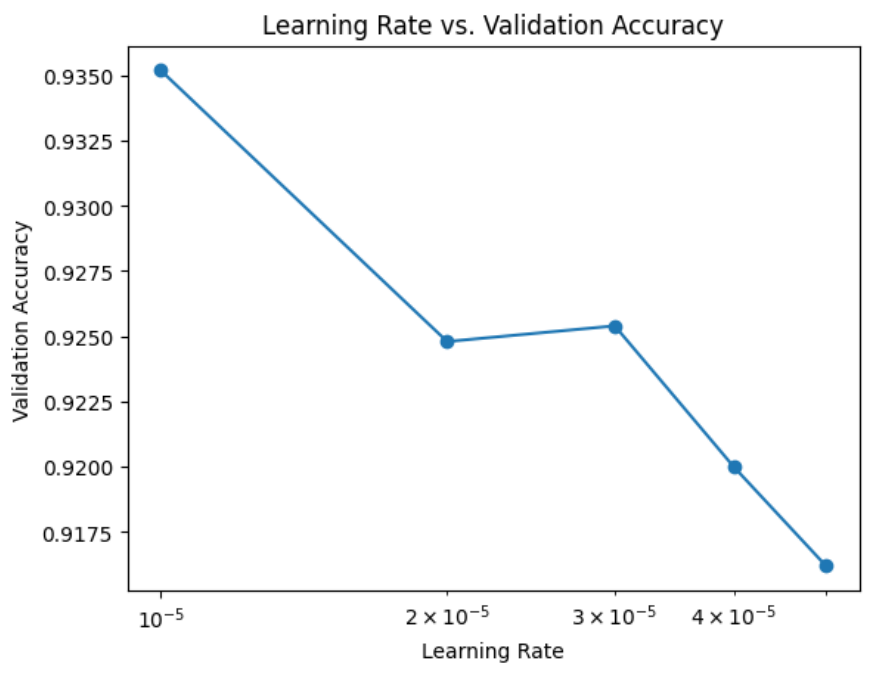
EVALUATING THE MODEL AND HYPERPARAMETER OPTIMIZATION

After we finish training the model, we then use the test dataset to evaluate the model's performance based on the accuracy of the results on complete new and unseen data.

And after setting up all the training and evaluation steps, we can finally start to tune our hyperparameter, we will only be testing different learning rates.

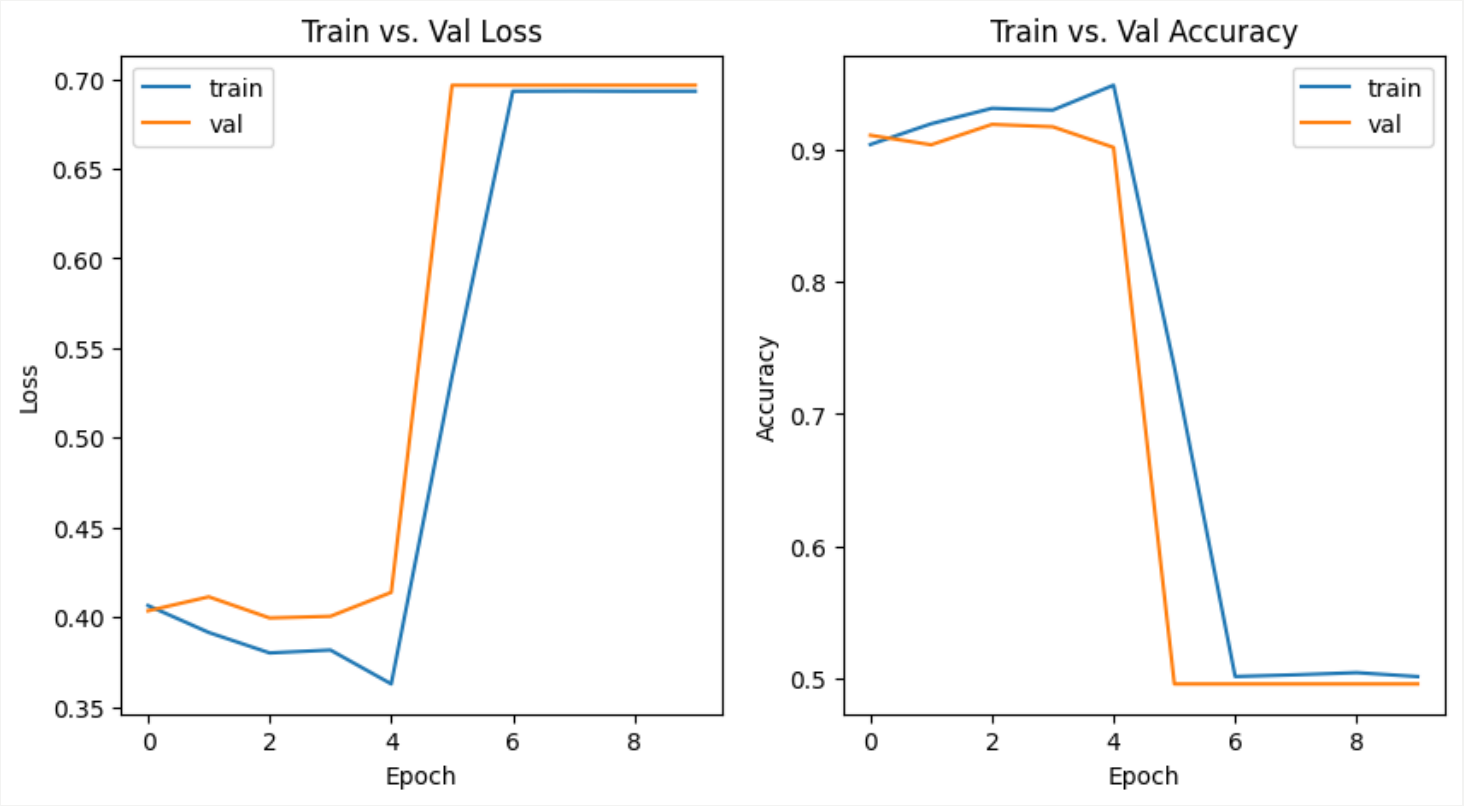
```
learning_rates = [5e-5, 4e-5, 3e-5, 2e-5, 1e-5]
```

Clearly, smaller the learning rate, the better.



HYPERPARAMETER OPTIMIZATION

During the hyperparameter tuning process, we also found that our models starts overfitting and the accuracy drops significantly at around 5 epochs, so at the end we decided to train our model with only 4 epochs, while that quite a low number, with the amount of complexity the original BERT model has, we are still able to get quite a high accuracy.



TRAINING THE FINAL MODEL

After going through this process, we now have the best hyperparameters, a good model architecture, and all the training and evaluation steps step up. We will now train a final model for 4 epochs and save it. Although we could train for more to increase the accuracy even more, I don't think we would have enough resources nor time.

USING THE MODEL ON REAL DATA

Now that we've created our Movie Review BERT Model with a fairly high accuracy, we can use it on some real movie reviews that I've found on a review website.

"Really Amazing, just epic and awesome to watch, no other words. A bold showing on the theory and practice of collective oligarchies. A consumerist masterpiece that shows the function of a capitalist world built off of cinema and film industry. In every moment the screen fills up with action, not only from the main character. It should win an Oscar for its screenplay."

Prediction Results: positive review with 99.98% certainty

"Christian Bale is great, Russell Crowe has a good moment, Hemsworth is good but this movie cannot be saved from a terrible script, bad direction and stupid humour that overstays its welcome."

Prediction Results: negative review with 99.98% certainty

CHALLENGES AND IMPROVEMENTS:

In this section, we discuss the challenges encountered during the fine-tuning of the BERT model on the IMDb dataset for sentiment analysis and propose potential improvements to address these issues.

COMPUTATIONAL RESOURCES

One of the main challenge that we came across is not powerful enough computational resources. The BERT model has a large number of parameters, which lead to high computational resource requirements, particularly in terms of memory and processing power. During the process of fine-tuning our model, we found that running our training loop required at least 23 GB of GPU RAM over a really long period of time.

We first tried just using the normal version of Google Colab to run this, but obviously, the weak GPU did not have enough memory. So to solve this, we decided to purchase Google Colab Pro which offered double the GPU memory of the original one and also premium computing units made for high-RAM usage. This allowed us to run the fine-tuning program without it running out of memory and crashing.

HYPERPARAMETER OPTIMIZATION

Another major challenge we faced during this project was the time-consuming nature of the hyperparameter optimization process. As previously mentioned, finding the optimal combination of hyperparameters is crucial for maximizing the model's performance. However, tuning hyperparameters require us to train multiple test models which is extremely time consuming even with powerful GPUs from Google Colab Pro. In particular, training our model with 3 epochs took an average of 40 minutes to an hour because of the vast amount of data in the dataset and the huge parameter count that BERT has.

Since the process of hyperparameter tuning often involves adjusting multiple hyperparameters, such as learning rate, batch size, number of epochs, it is clear that trying to force search the huge space for the best hyperparameter is unrealistic. Instead, we try to take suggestions from the paper for things like batch size, learning rates, and the number of epochs. By doing that, we reduce the search space to an manageable amount that we can manually test.

DATASET MANAGEMENT

Another challenge that we faced was the inefficient organization and management of the dataset. Initially, we did not utilize any of the dataset handling tools provided by PyTorch, so the data management process became messy and cumbersome. This led to difficulties in efficiently loading, processing, and feeding the data into the model during training and testing.

Later on, we found out about the Dataset and Dataloader classes provided by PyTorch. hese classes provide a clean and efficient way to manage the dataset, allowing for easier data loading, preprocessing, and batching.

RESULTS

In this section, we present the results obtained from fine-tuning our BERT model on the IMDb dataset for sentiment analysis. We evaluate the performance of our final model and compute the validation as well as test accuracy, and we will also test it on some real data.

During the hyperparameter tuning process, we found that our models starts overfitting and the accuracy drops significantly at around 5 epochs, so at the end we decided to train our model with only 4 epochs, while that quite a low number, with the amount of complexity the original BERT model has, we are still able to get quite a high accuracy.

After training our final model, here are some of our metrics:

- Train Accuracy: 94.05%
- Validation Accuracy: 91.74%
- Test Accuracy: 91.74%

DISCUSSION AND CONCLUSIONS

In conclusion, this study demonstrates the effectiveness of the BERT model for sentiment analysis on the IMDb dataset. By fine-tuning the model and optimizing its hyperparameters, we were able to achieve high performance, surpassing (or being competitive with) existing benchmarks in the literature. Our findings indicate that the BERT model, with its bidirectional pre-training and contextualized representations, is a powerful tool for sentiment analysis tasks and can be successfully adapted to various NLP challenges. The insights gained from this study can guide future research in fine-tuning the BERT model for other NLP tasks or on different datasets. Moreover, the proposed improvements and strategies can be employed to overcome challenges and enhance the performance and efficiency of similar models in the field of NLP. Overall, our study contributes to the growing body of research on the BERT model and its applications, further establishing its potential as a robust solution for a wide range of NLP tasks.

Thank you!!

REFERENCES

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, May 24). *Bert: Pre-training of deep bidirectional Transformers for language understanding*. arXiv.org. Retrieved May 5, 2023, from https://arxiv.org/abs/1810.04805

Bert. BERT. (n.d.). Retrieved May 5, 2023, from https://huggingface.co/docs/transformers/model_doc/bert

N, L. (2019, March 9). *IMDB dataset of 50K movie reviews*. Kaggle. Retrieved May 5, 2023, from https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

A Visual Notebook to Using BERT for the First Time. Colab Notebook. (n.d.). Retrieved May 5, 2023, from https://colab.research.google.com/github/jalammar/jalammar.github.io/blob/master/notebooks/bert/A_Visual_Notebook_to_Using_BERT_for_the_First_Time.ipynb#scrollTo=lZDBMn3wiSX6