Modelling IMDb reviews using BERT

Matt Ludwig

1 Abstract

In this assignment I implemented the Bidirectional Encoder Representations from Transformers (BERT) using PyTorch to predict the sentiment of movies reviews from the IMDb movie review dataset. When comparing the BERT model to several traditional baseline machine learning models, I found that BERT achieved better accuracy and that it took significantly longer to train. On the IMDb testing, BERT achieved an accuracy of 93.8320%. I also used the Random Forest model to obtain the top twenty important features or words used in predicting the sentiment of reviews and produced various heatmaps to examine the attention mechanism in one of the transformer blocks in BERT for correctly predicted and incorrectly predicted reviews. In addition, I also implemented several other machine learning models such as GPT2 and XLNet and compared their accuracies and AUROC scores with those obtained from BERT.

2 Introduction

In this assignment I acquired and preprocessed the IMDb movie review dataset, implemented various machine learning models to serve as baseline models for BERT as well as implemented several other transformer based models to compare BERT with. The IMDb movie review dataset is a commonly used data set for natural language processing and has become popular for binary sentiment classification due to the fact that it contains more samples previous benchmark datasets. For instance, the IMDb dataset was been widely used in machine learning research and has been used to test a variety of machine learning techniques. One of the uses of the dataset has been to test variations of BERT. For instance, in "XLNet: Generalized Autoregressive Pretraining for Language Understanding" the authors introduce a new model called XLNet which is an autoregressive model. Unlike BERT it also uses a word embedding matrix as well as LSTM and is able to overcome the pretraining discrepancy that BERT suffers from. For more information on XLNet we refer the reader to [YDY+19]. Another paper that uses the IMDb dataset and is a variation of BERT is "DistilBERT, a distilled version of BERT, smaller, faster, cheaper and lighter". DistilBERT reduces the size of BERT by 40% while retaining 97% of its language understanding capabilities. For more information on DistilBERT see [SDCW19]. Another paper that uses the IMDb dataset and that implements a model that improves upon the BERT architecture is "Univeral Language Model Fine-tuning for Text Classification". In this paper, the authors noted that transfer learning, which is when a model that is trained on one task is re-purposed on a second task, has greatly impacted computer vision, but has been more difficult in the world of natural language processing. Therefore, they introduced the Universal Language Model Fine-tuning (ULMFiT), which can be applied to any NLP task and which has resulted in more advances in transfer learning in the domain of natural language processing. For more information we refer the reader to [HR18].

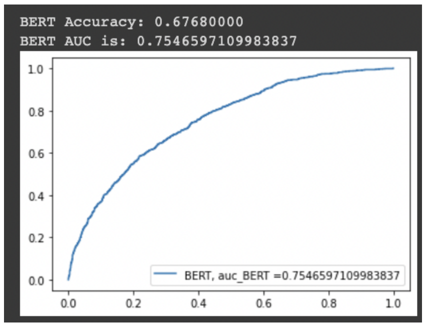
3 Datasets

The dataset itself consists of $50,000$ movie reviews which are split between $25,000$ training and $25,000$ testing samples. The dataset is provided in both a raw text format (which I used in implementing BERT and other transformer based models) and in an already processed bag of words format (which we used in implementing the baseline machine learning models). After downloading the IMDb Reviews data, I preprocessed the data by converting the data into tabular format from the bag of words format. I also filtered out words that appeared in less than 1% of documents and more than 50% of documents. I then used absolute z-scores as associations with continuous rating scores $(1-10)$ by using the Simple Linear Regression Hypothesis in order to select the most important words in predicting the sentiment of each IMDb review. I used this preprocessed version of the data when running baseline models that I used to compare BERT with. For BERT, I used the BertTokenizer from bert-base-uncased to first tokenize each of the sentences into individual words. I then prepared token embeddings and token Ids as well as padded each of the sequences of token ids so that all of the reviews would be off the same length. When trying to obtain the ROC curve for BERT, I encountered memory issues with the CUDA gpu that the model was running on and ultimately found that running the model on a subset of the $50,000$ training and testing reviews as well as only tokenizing and using a small number of tokens for each review reduced the size of the tensors that the CUDA gpu was processing which resolved the issue of the CUDA gpu running out of memory. While I used basic gradient clipping in training the BERT model, further work could perhaps use more advanced methods of gradient clipping to resolve this issue of the CUDA gpu running out of memory.

4 Results

4.1 Model Hyperparameters and Architectures

Using the scikitlearn libraries, I implemented logistic regression, support vector machine, random forest and XGBoost models and found their binary classification accuracy on the IMDb dataset. For the logistic regression model, I did not implement any regularization and found that the model gave an accuracy of 84.32%. For the support vector machine model, I used the default radial basis function kernel and found that it gave an accuracy of 83.992%. For the hyperparameters for the random forest model, I used 800 estimators, each with a maximum tree depth of 50 and found that it gave an accuracy of 79.865%. For the XGBoost model, I used the default hyperparameters which consisted of a gradient boosting tree for the booster, an alpha value of zero, a max depth of 6 and a lambda value of 1 for regularization and found that it gave an accuracy of 80.616%. For the naive Bayes model, I used scikitlearn's MultinomialNB and used the default settings which meant that the model would not assume a uniform prior on the two class probabilities and would have an alpha value of one for the additive Laplace smoothing parameter and found that it gave an accuracy of 83.912%. For the BERT model, I used a pretrained model which was version of the Base-BERT model called 'bert-based-uncased' which had 12 hidden layers, 768 hidden nodes, 12 attention heads and a total of 110 million parameters. In implementing the model I followed the tutorial that was listed on the assignment sheet and that can be found here [AA]. In training the model, I used the Adam optimizer with a learning rate of 3×10^{-6}, a batch size of 2 and with 10 epochs. Due the memory issues with the CUDA gpu, I decided to train the model on a subset of 500 of the 50,000 reviews. In tokenizing the reviews, I reduced the dimension of the tensors by limiting each review to only the first 25 tokens. With these hyperparameters, I was able to obtain an accuracy of 67.68% and an AUROC score of .7546 which is shown in the plot below. I was able to run the model on all 50,000 and get an accuracy of 93.8320% but due to the CUDA gpu memory, I was unable to obtain an ROC plot and an AUROC score for this model.



```
BERT Accuracy: 0.67680000
BERT AUC is: 0.7546597109983837
```

Figure 1: ROC Curve, Accuracy and AUROC score for BERT on a subset of the IMDb dataset

4.1.1 Baseline vs. BERT Model

To compare the binary classification accuracy of all of the models, I generated a single plot with ROC curves of BERT, LR, SVM, RF and XGBoost on the IMDb test data and a bar plot comparing their AUROC scores. These plots are shown below:

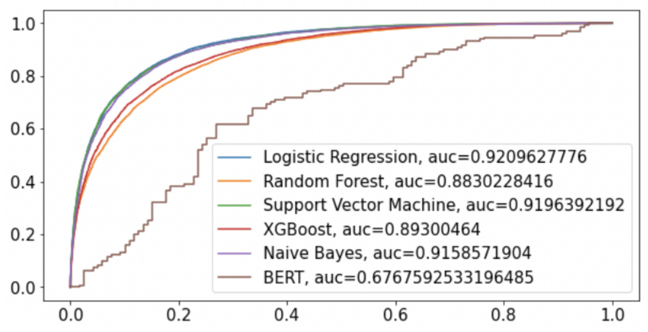


Figure 2: A single plot containing four ROC curves of BERT, LR, SVM, RF, XGBoost and Naive Bayes on the IMDb test data

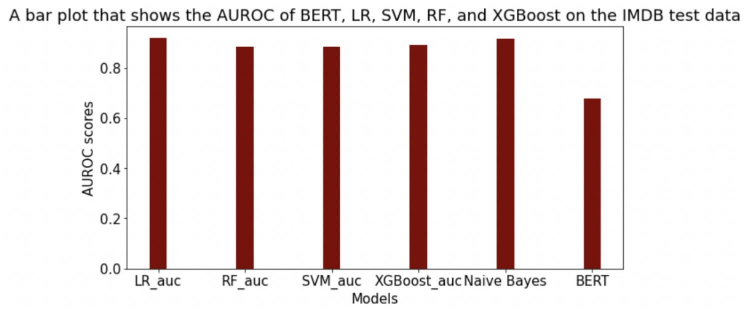


Figure 3: A bar plot that shows the AUROC of BERT, LR, SVM, RF, XGBoost and Naive Bayes on the IMDb test data.

In order to determine which features that the random forest model deemed important, I also generated a horizontal bar plot which the top 20 important features on the y-axis and Gini importance of the mean decrease in impurity on the x-axis. The mean decrease in impurity is calculated for each feature by looking at the average of how the feature decreases the impurity of each split in which it is used in the decision tree. The final mean decrease in impurity for the feature is then calculated as the average of this score for each decision tree in the random forest. A plot of the top 20 important features in determining the sentiment of an IMDb movie review is shown in the figure below.

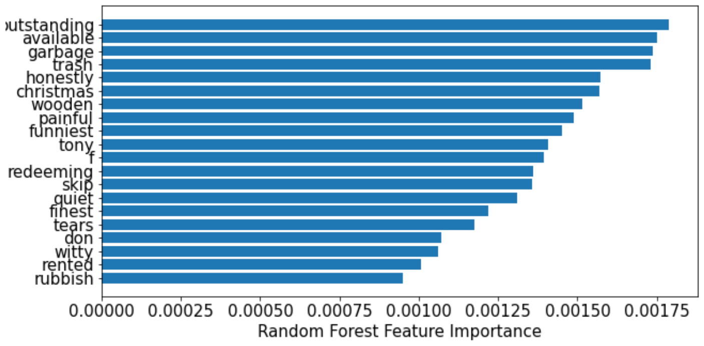


Figure 4: A horizontal bar plot showing the top 20 important features from RF on the IMDb data with the feature importance scores as the x-axis and the feature names (i.e., words) as the y-axis.

I also looked at which features the BERT model deemed important in predicting the sentiment of an IMDb movie review by examining several attention matrices between the words and the class tokens. While I did not have enough time to look at heatmaps for positive and negative reviews that BERT predicted correctly and incorrectly, I was able to generate a heatmap which shows which words in the start of a correctly predicted positive review that BERT pays attention to. While it does not show the attention between the $[CLS]$ token and the top words that the model pays attention to, it does show the attention between the $[CLS]$ token and the first words of the review. It is interesting to note how the word ”incredible” has a high value for the attention head in layer 6. This can be seen in the figure below.

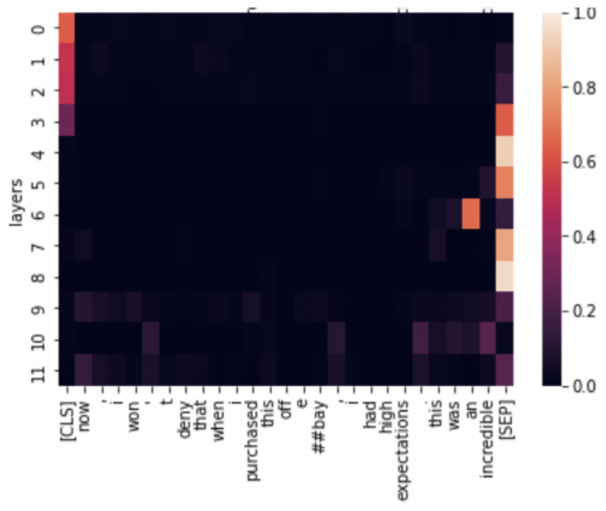


Figure 5: A heatmap showing one of the attention matrices for the heads in each layer and the $[CLS]$ token and the first words of the review. [CLS] token.

4.2 GPT2 and XLNet Models

Looking at a kaggle leaderboard for the different machine learning models used for sentiment analysis on the IMDb dataset, I decided to implement a few other models and compare their accuracy to the BERT model that I implemented.

4.2.1 GPT2

Following an online tutorial [GM], I implemented GPT2 for text classification using Hugging Face Transformers. While BERT uses a stack of encoder transformers, GPT2 uses a stack of decoder transformers and while BERT uses the first token embedding to make a prediction, GPT2 uses the last token embedding to make a prediction. I found that GPT2 gave an accuracy of 83% after 4 epochs. I also implemented the model with a batch size of 30 a max text sequence length of 60. I also used the AdamW or Adam with weight decay fix from the Huggingface library with a learning rate of $2 * 10^{-5}$

4.2.2 XLNet

I also implemented a XLNet model which had an even accuracy than BERT on the leaderboard for the IMDb dataset following this online tutorial [ES]. XLNet has a similar architecture to BERT but has a different approach to pre-training. Whereas BERT masks 15% percent of words in word prediction pretraining task, XLNet uses a different scheme and uses a subset of a word's neighbors in its prediction so that the every word is not indirectly seen through the words used to predict it. I used the XLNet model with a max sequence length of 64, one epoch, a learning rate of $1 * 10^{-5}$, a weight decay of .01 and a training batch size of 128. With these hyperparameters, I found that XLNet gave an accuracy of 92.2%. A bar plot comparing the accuracies of all of the models is shown below:

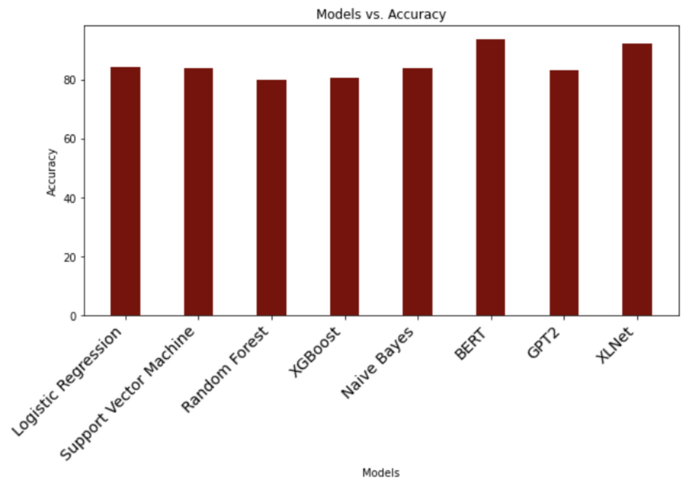


Figure 6: Bar plot of Accuracies of all of the Models

5 Discussion and Conclusion

Overall, I found that the BERT model achieved the highest accuracy of 93.8320% on the IMDb movie review dataset when compared to other baseline machine learning models and to the GPT2 and XLNet models. I also found that training the BERT model took significantly longer than any of the baseline machine learning models or the GPT2 and XLNet models. When implementing the BERT model, I also encountered several issues involving the available memory in the CUDA gpu and ultimatley had to train the model on a subset of the original dataset and with a smaller sequence of tokens for each review in order for the GPU to hold the tensors that were required to generate the ROC and AUROC scores for the BERT model. Encountering this issue made me realize the value in having different BERT models such as DistilBERT or bert-small which contain a subset of the original BERT model parameters and can be trained faster and require less memory for the GPU. Further investigations, could look at more advanced methods of gradient clipping or other ways to get around the limited memory of the CUDA gpu so that it would be possible to generate ROC and AUROC scores for the BERT model that is trained on the entire dataset. Another possible direction for further investigation would to find ways increase the accuracy of the baseline machine learning models. One possibility would be to project the IMDb data onto an embedding space using Word2Vec or pretrained GloVe embeddings and to run the baseline models using these embedded tokens and see if the resulting accuracies is closer to the accuracies achieved by the BERT, GPT2 and XLNet models.

6 Statement of Contributions

I completed this assignment alone. I contacted my groups telling them this and they agreed to let me work alone and that we would receive separate grades for this assignment.

References

[AA] https://www.kaggle.com/code/atulanandjha/bert-testing-on-imdb-dataset-extensive-tutorial/notebook Atul Anand.

[ES] https://colab.research.google.com/github/eugenesiow/practical-ml/blob/master/notebooks/Sentiment$_{A}nalysis_{M}ovie_{R}e$
$iHQBc_{R}jAtI9Eurgene\ Siow.$

M] https://colab.research.google.com/github/gmihaila/ml$_{t}hings/blob/master/notebooks/pytorch/gpt2_{f}inetune_{c}lassification.ipynbscr$
$3NmlcqdctXlBGeorge\ Mihaila.$

R18] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification, 2018.

CW19] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019.

DY^{+}19] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2019.