EEE 486 STATISTICAL FOUNDATIONS OF NATURAL LANGUAGE PROCESSING ASSIGNMENT 2 REPORT

MELİH BERK YILMAZ

21803702

Table of Contents

ABSTRACT

In this assignment, the use of the BERT (Bidirectional Encoder Representations from Transformers) model is explored for text classification tasks on the Corpus of Linguistic Acceptability (CoLA) dataset. The aim of this assignment is to explore the hyperparameters of the BERT Language model and their effect on the training procedure. The effects of learning rate, number of epochs, max length of input sequence, and dropout rates on model performance are investigated, with evaluation metrics based on the Matthew Correlation Coefficient (MCC). Additionally, we explore alternative ways of text classification using BERT models and evaluate their performance using the same set of hyperparameters.

INTRODUCTION

The main objective of this assignment is to provide a practical experience in text classification using the BERT language model. In particular, we focus on the Corpus of Linguistic Acceptability (CoLA) dataset to examine the model's performance. Although pre-trained BERT models can enhance the accuracy of text classification, it is essential to fine-tune the model with different classifier units to accommodate the variability in the number of classes in distinct tasks. Hence, in the first part of the assignment, we retrieve the BERT feature extractor from the Huggingface library and train only the classifier head to significantly reduce training time. Furthermore, we optimize the model's hyperparameters, including the learning rate, number of epochs, max length of input sequence, and dropout rates, to enhance the accuracy and MCC scores. In the second part of the assignment, we investigate an alternative document representation approach, specifically the mean of the hidden state vector, in contrast to the traditional method of using the [CLS] token. Similarly, we optimize the model's hyperparameters to identify the optimal settings for the mean pooling approach.

ARCHITECTURES

In this part of the assignment, you are required to come up with an alternative method to handle the document representation other than the [“CLS”] method. The [CLS] token in BERT is commonly used to represent the entire input sequence, allowing the model to perform text classification tasks by predicting the probability distribution over a set of target labels based on the first vector in the hidden state used for document representation. This can be visualized as:

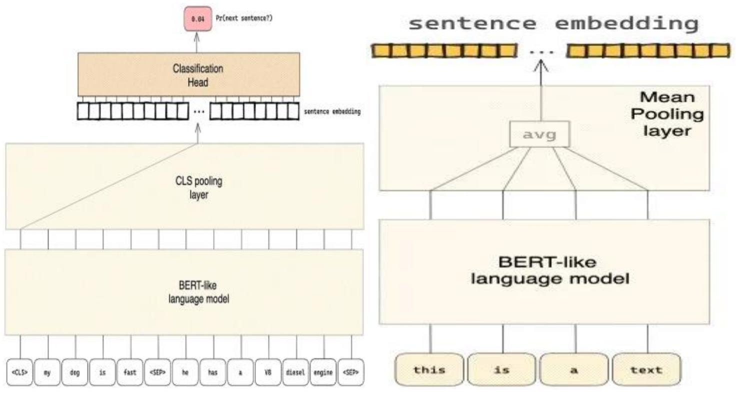


Figure 1 (Left): CLS Document Representation Method [1]

Figure 2 (Right): Mean Pooling Based Document Representation [1]

There are other ways to handle the BERT architecture, which can be divided into two categories: learning-based and non-learnable methods. An example of a learning-based method is using attention weights, while examples of non-learnable methods include taking the element-wise mean or maximum of the vectors in the dimension of the stack of vectors. [1, 2] The latter can be visualized as follows:

$$[CLS]_{mean} = {}^{1}\!/_{n} * \sum_{i=0}^{n} h_i \quad (1)$$

$$[CLS]_{max} = max(h_i) \quad (2)$$

Mean Pooling and Max Pooling based document representation approaches can be formalized as in equation (1) and (2) respectively. I used Mean Pooling based document representation for non-learnable approaches.

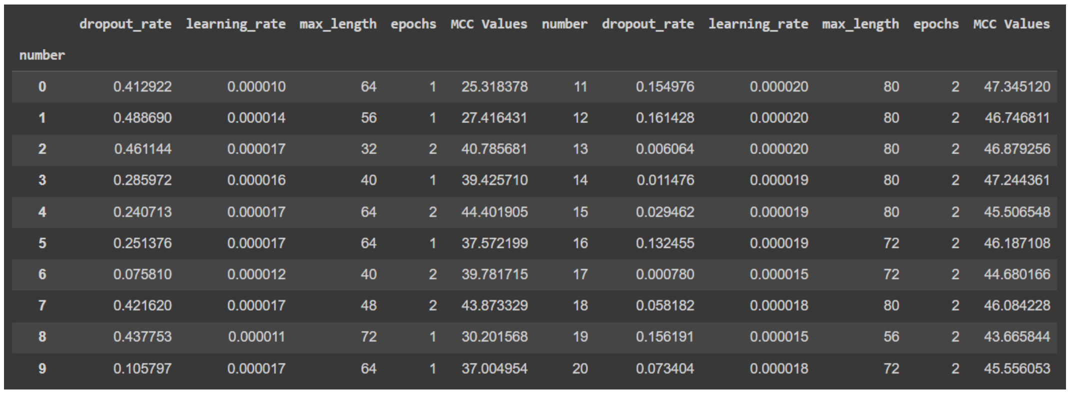
To introduce more complexity to the model architecture for improved results, we can replace the classifier layer with a learnable layer that operates on the output vectors of the feature extractor part of the BERT model. This approach involves converting the output vectors into the sum of weighted vectors, allowing the model to prioritize the features that have a greater impact on the classification task. In Figure 2 of the original paper, the vectors are directly summed and divided by the number of vectors. However, in this approach, each vector is multiplied by a weight factor, enabling us to assign more weight to the vectors that are more important for the task at hand. By multiplying each vector with a small constant near 0, we can effectively ignore the features that have a negligible impact on the classification task, leading to improved model performance. This method ensures that if a vector has more impact on the class label, the model will prioritize the feature extraction coming from this vector, and vice versa. Therefore, the model will allocate more resources to the features that are more important, while down weighting or discarding the less important ones. This method can be formalized as:

$$[CLS]_{out} = \sum_{i=0}^{n} a_i * h_i \quad (3)$$

RESULTS

Part 1:

Firstly, the CoLA dataset is downloaded and a pretrained and frozen BERT model is created with a classifier header specialized for the CoLA dataset as explained in steps a-c. Then , the model is tuned with adjusting learning rate, maximum length of input sequence, number of epochs and learning rate.



number	dropout_rate	learning_rate	max_length	epochs	MCC Values	number	dropout_rate	learning_rate	max_length	epochs	MCC Values
0	0.412922	0.000010	64	1	25.318378	11	0.154976	0.000020	80	2	47.345120
1	0.488690	0.000014	56	1	27.416431	12	0.161428	0.000020	80	2	46.746811
2	0.461144	0.000017	32	2	40.785681	13	0.006064	0.000020	80	2	46.879256
3	0.285972	0.000016	40	1	39.425710	14	0.011476	0.000019	80	2	47.244361
4	0.240713	0.000017	64	2	44.401905	15	0.029462	0.000019	80	2	45.506548
5	0.251376	0.000017	64	1	37.572199	16	0.132455	0.000019	72	2	46.187108
6	0.075810	0.000012	40	2	39.781715	17	0.000780	0.000015	72	2	44.680166
7	0.421620	0.000017	48	2	43.873329	18	0.058182	0.000018	80	2	46.084228
8	0.437753	0.000011	72	1	30.201568	19	0.156191	0.000015	56	2	43.665844
9	0.105797	0.000017	64	1	37.004954	20	0.073404	0.000018	72	2	45.556053

Table 1: Parameters Combinations vs MCC Scores Table

Table 1 shows the parameter combination for each trial and its corresponding validation MCC score at the last column. It is hard to say something about the parameter importance by only looking at this table.

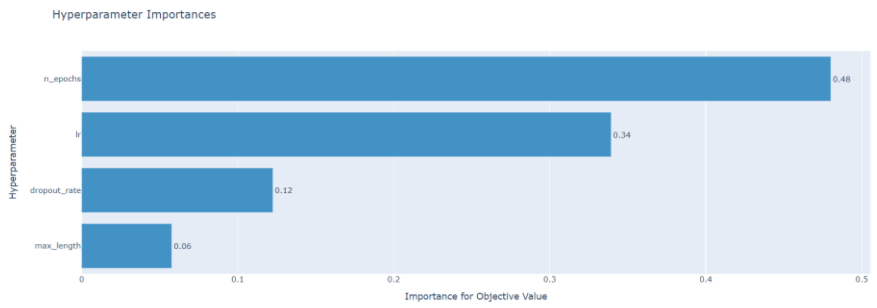


Figure 3: The Importance of The Hyperparameters

Figure 3 is obtained by the function of Optuna library, which automatically calculates the importance of each parameter to maximize the MCC score. From figure 3, it is clear that the number of epochs has the greatest impact on the increase of MCC score. For lower number of epochs values, this is logical since the model needs further training for the specific task even though it is pretrained. The learning rate is the second most important, it is obvious as well since if it is too high, it causes so much saturation and affects the learning period negatively. The dropout rate and maximum input sequence length has not much effect on the results.

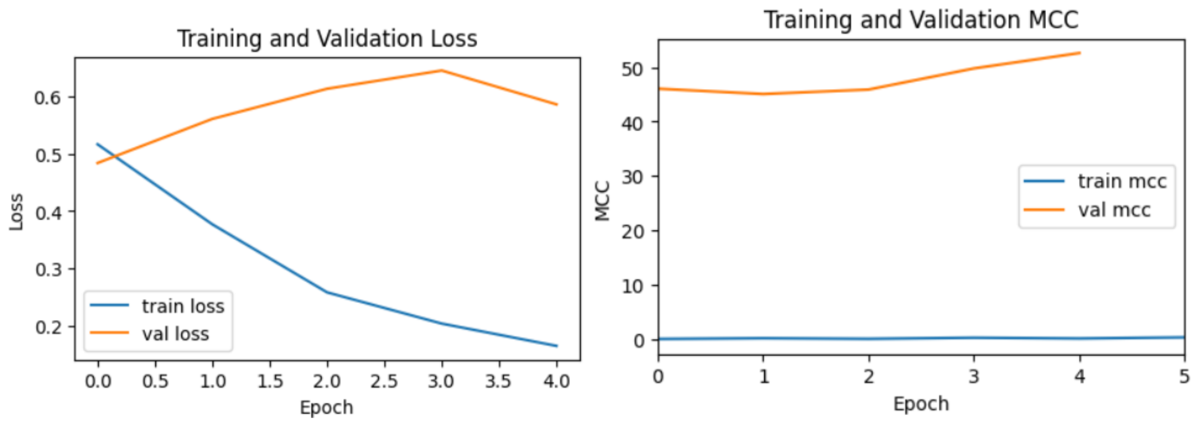


Figure 4: Training vs Validation Loss and MCC Score Plots

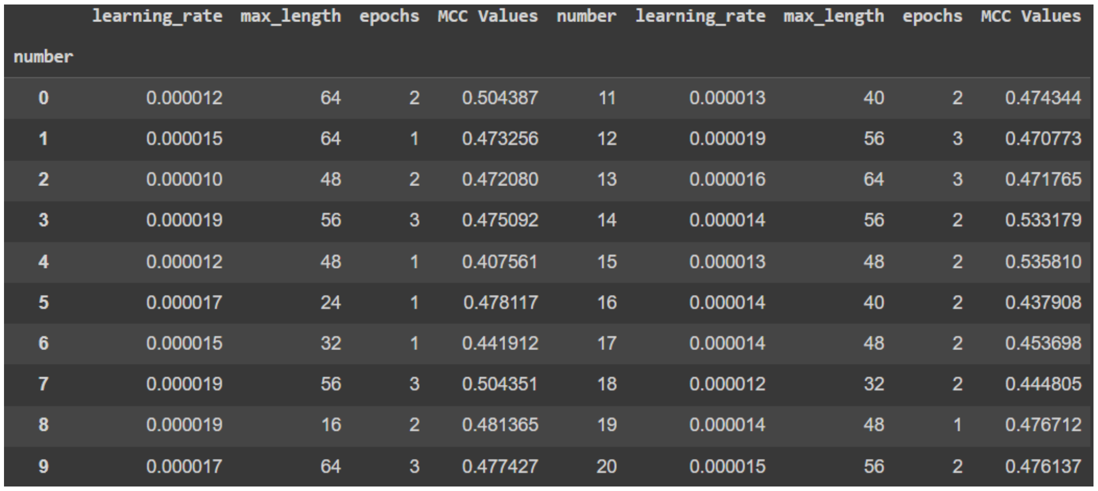
As seen in the above figure, validation loss doesn't decrease that much but the MCC score increases for each epoch. After the optimization step, the initial model is trained with the best parameter combinations again and loaded to Huggingface's hub.

Part 2:

In this part of the assignment, it is required to come up with a different method for handling CLS token document representation. The architecture details are explained in the Architectures part. Thus, this section will provide only the results.

Mean Pooling Based Document Representation Model:

Since this method doesn't require any external parameters, I didn't change anything in the optimization code, only the optimization of dropout rate is removed.



number	learning_rate	max_length	epochs	MCC Values	number	learning_rate	max_length	epochs	MCC Values
0	0.000012	64	2	0.504387	11	0.000013	40	2	0.474344
1	0.000015	64	1	0.473256	12	0.000019	56	3	0.470773
2	0.000010	48	2	0.472080	13	0.000016	64	3	0.471765
3	0.000019	56	3	0.475092	14	0.000014	56	2	0.533179
4	0.000012	48	1	0.407561	15	0.000013	48	2	0.535810
5	0.000017	24	1	0.478117	16	0.000014	40	2	0.437908
6	0.000015	32	1	0.441912	17	0.000014	48	2	0.453698
7	0.000019	56	3	0.504351	18	0.000012	32	2	0.444805
8	0.000019	16	2	0.481365	19	0.000014	48	1	0.476712
9	0.000017	64	3	0.477427	20	0.000015	56	2	0.476137

Table 2: Parameters Combinations vs MCC Scores Table

In this model, the importance of the learning rate is 0.62, the importance of the maximum sequence length is 0.33 and the importance of the number of epochs is 0.05. (Plots occupy too much area). We can see that the learning rate has the greatest impact on the model performance. This may stem from the fact that the model is very fragile to small changes in the learning rate. In this scenario, the number of epochs has not much effect since the models are pretrained.

Then the model is trained with the best parameter combinations again to obtain the validation results.

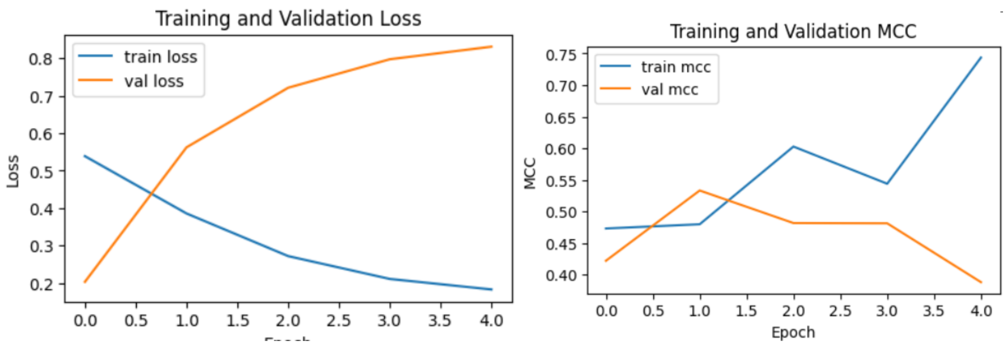


Figure 5: Training vs Validation Loss and MCC Score Plots

As seen in the above figure, validation loss doesn't decrease, indeed, it increases in each epoch. The validation MCC score is increased for just one epoch, then starts to decrease. This is the indicator of overfitting. Therefore, after finding the best parameter combination, several epochs are enough to see the full performance of the parameter setting. As seen in figure 5, the model reaches a maximum of 55 MCC score whereas the CLS document representation method reaches a maximum of 51 MCC score. Briefly, this method yields better results compared to classical CLS methods.

Weighted Mean Pooling Based Document Representation Model:

Since this method doesn’t require any external parameters, I didn’t change anything in the optimization code, only the optimization of dropout rate is removed. Maybe, the weight initialization could be parameterized, but i didn’t think it is an essential parameter worth tuning.

Firstly, the model is optimized over the learning rate, maximum sequence length and number of epochs. I will not share the table again due to page limit (it can be seen in the ipynb). In this case, the importance of the number of epochs, learning rate and maximum sequence length are 0.92, 0.07 and 0.02 respectively. This is because I introduced a new learnable layer, which increases the number of learnable parameters and the overall model requires more epochs to generalize to validation and test data. Therefore, this is an expected outcome. If I were to increase the range of epoch numbers for the hyperparameter tuning part, the results would be different.

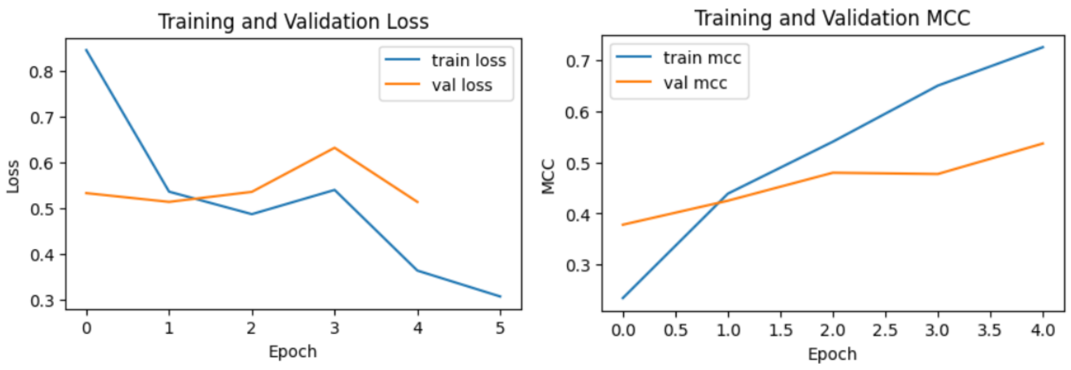


Figure 6: Training vs Validation Loss and MCC Score Plots

In this case, the model requires a higher number of epochs as it is seen in the above figure. Even though the validation loss fluctuates, the validation MCC score increases in each epoch, which is a sign that there is still room for improvement. The validation MCC score is higher compared to the CLS method and lower compared to the Mean Pooling method. However, as explained before, this method could reach a much higher validation MCC score with a higher number of epochs.

DISCUSSION AND CONCLUSION

In this assignment, the BERT language model is used for text classification tasks with the CoLA dataset. The aim of the assignment is to gain hands-on experience with one of the essential and fundamental Natural Language Processing algorithms. In the first part, the BERT model is optimized over the learning rate, dropout rate, maximum sequence length and number of epoch parameters with the usage of CLS document representation technique and the model performance is observed. This part was challenging for me since some of the parameters are not included in the training arguments. In the second part, I proposed 2 ways of handling document representation, one is learning based weighted pooling and the other one is non learnable based mean pooling document representation. In this part, the bert model must be separated into two as feature extractor and classifier for output manipulation. It was hard with the code provided to us. Therefore, I created my own class to implement this separation so that I easily handled the hidden state manipulation.

REFERENCES

[1] H. Zhang, Y. Gong, Y. Shen, W. Li, J. Lv, N. Duan, and W. Chen, "Poolingformer: Long document modeling with pooling attention," *arXiv.org*, 24-Oct-2022. [Online]. Available: https://arxiv.org/abs/2105.04371. [Accessed: 07-May-2023].

[2] M. Leys, "The art of pooling embeddings 🎨," *Medium*, 20-Jun-2022. [Online]. Available: https://blog.ml6.eu/the-art-of-pooling-embeddings-c56575114cf8. [Accessed: 07-May-2023].