EEE586: Statistical Foundations of Natural Language Processing Assignment 2

Atakan Topcu, *Bilkent University*

***Abstract*—Syntax and grammar are the cornerstones of language. It is how humans operate and maintain their self in the world. Thus, teaching a deep learning model to understand syntax and grammar is essential to reach a level of self-consciousness. For this purpose, I have implemented a couple of transformer architectures with trainable regression heads. This way, I could practice my knowledge of transformer models and Hugging Face API. In this assignment, I started by first utilizing the API of Hugging Face, where there was already a function for generating a BERT model with a regression head used for syntax classification. Further, I pushed my trained model to the Hugging Face repository. Apart from using the model from API, I have also come up with a new way to classify syntax without only using the "[CLS]" token of the BERT model. For this part, I have used native PyTorch instead of Hugging Face API [1]. Since the last model didn't follow the Hugging Face convention, I wasn't able to push to model to the hub. Nonetheless, The aim of this assignment was to utilize deep learning tools for computational linguistics[1].**

***Index Terms*—Natural Language Processing, Transformers, Hugging Face**

I. Introduction

In this study, I used the CoLA (Corpus of Linguistic Acceptability) dataset from the GLUE (General Language Understanding Evaluation) to classify sequences using the BERT (Bidirectional Encoder Representations from Transformers) language model [2], [3]. I used Optuna, a hyperparameter optimization framework, to undertake hyperparameter tuning in order to get the best results. In addition, I created a novel method for CoLA classification by pooling all of the BERT model's tokens rather than just the [CLS] token. After the training and fine-tuning procedure, I uploaded the model to Hugging Face, a well-known platform for sharing and deploying natural language processing models. This study introduces a new technique to improve CoLA classification results and shows the utility of BERT for sequence classification problems.

The main models used will be explained in the Architectures section. To begin with, we will provide an overview of the BERT model, including information about the equations and parameters. Then, we will discuss the techniques used to classify the CoLa dataset. Next, we will provide information on the quantitative properties of the corpora, such as their size, content, and other relevant details. Finally, we will summarize the implementation details, including the programming infrastructure used, the libraries utilized, and the implementation decisions made.

The Results section of this study is presented in four subsections, where the findings are arranged systematically. Visual aids, such as loss curves and MCC results, will be utilized to display the findings and will be explained in detail.

The Discussions & Conclusions section will provide an interpretation of the experimental findings. A comparison will be made between the expected and actual experimental results. The methodology of the study will also be reviewed. Lastly, a brief and concise summary of the project's main idea will be presented.

Overall, we have four main tasks to investigate:

- Utilize Hugging Face API to conveniently train a pre-trained model and push it to the hub.
- Examine the relationship between the performance of the model and the hyperparameter values.
- Examine the usability of other tokens of BERT for the CoLa classification task.
- Learn to use a pre-trained transformer model and fine-tune it.

[1]My Hugging Face Hub: https://huggingface.co/Attakuan/bert-base-uncased-finetuned-cola

TABLE I
EXAMPLE COLA DATASET

Sentence	Label
Our friends won't buy this analysis, let alone the next one we propose.	1 (acceptable)
One more pseudo-generalization and I'm giving up.	1 (acceptable)
"One more pseudo generalization or I'm giving up."	1 (acceptable)
"One more pseudo generalization or I'm giving up."	1 (acceptable)
"Day by day the facts are getting murkier."	1 (acceptable)
"We yelled ourselves."	0 (unacceptable)

II. ARCHITECTURES

Here, the architectures used for the assignment are further discussed. The dataset used in this study is the CoLa dataset from the GLUE benchmark. The dataset includes sentences, labels, and ids, as shown in Table I.

Dataset

The General Language Understanding Evaluation (GLUE) benchmark uses the CoLA (Corpus of Linguistic Acceptability) dataset to assess how well a language model can discern whether an English sentence is grammatically acceptable. This task is difficult since it comprises over 10,000 grammatically correct or incorrect statements and contains semantically ambiguous sentences [2]. The Matthews correlation coefficient (MCC), which has a value between -1 and 1, indicates a negative and positive correlation, respectively, and is used to assess the effectiveness of models using the CoLa dataset for training. For the assignment, I have used this dataset using the Transformers library for Python, which pulls the dataset from Hugging Face's database. The dataset is split into three parts: train, validation, and test. However, the labels for the test part are not given. Thus, I have only used training and validation datasets for this assignment to have quantitative results.

Transformer Architecture

In this assignment, I have used a pre-trained BERT model for classification. The language model architecture, BERT (Bidirectional Encoder Representations from Transformers), was created by Google [4]. It captures bidirectional context using a transformer-based neural network architecture that processes inward and outward input text. By predicting masked terms in the input sentence, BERT uses an unsupervised approach to pre-training. During fine-tuning, BERT is trained on a particular downstream job by adding a task-specific layer to the pre-trained model and adjusting the model's parameters. For the classification tasks, it was found that using only the [CLS] token is sufficient and optimal. The general architecture of BERT is given in Fig. 1.

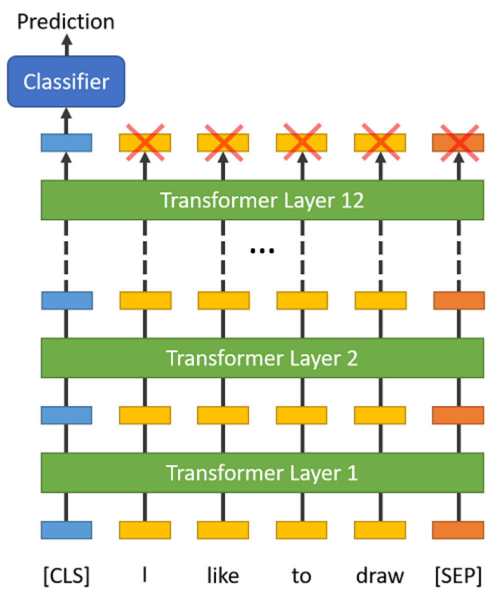

Fig. 1. Classical BERT architecture with [CLS] token used as document representation.

Though I have fine-tuned a [CLS] token with a linear regression head in the first part, I have also used the other tokens in the second part (see Fig. 2). For the regression problem, I have used a 1-layer linear classifier with a sigmoid activation function and binary cross entropy loss function in the second part. For that, I have pooled and taken the mean of the pre-trained BERT's hidden states and then passed it into the linear classifier. Thus, unlike the conventional way, instead of just using the [CLS] token for document representation, I have used all the available tokens and pooled them for classification.

For the hyperparameter setup of the first part:

- Dropout = 0.2.
- Learning Rate between 1e-6 and 1e-3.
- Epochs between 3 and 10.
- Batch size range of [4, 8, 16, 32, 64].

For the hyperparameter setup of the second part:

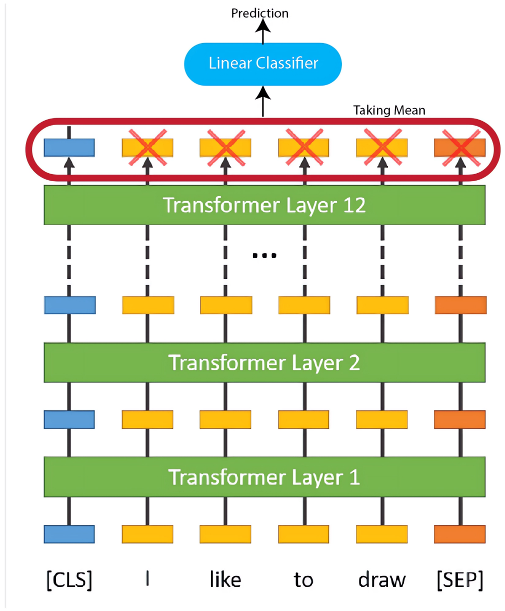

Fig. 2. Proposed model that pools all the tokens and passes it into the linear classifier.

- Learning Rate in between 1e-6 and 1e-3.
- Epochs in the range of [1,5,10].
- Batch size in the range of [16, 32, 64].
- Maximum length in the range of [16, 32, 64].

For the hyperparameter search, I used the Optuna library, and there were 2 trials in the first part and 3 trials in the second part. For the visualization of results, Wandb has been utilized in Google Colab.

III. Results

In this section, we conduct training and testing for classical BERT classification from Hugging Face API and the proposed method.

Part 1: Hugging Face BERT Fine-Tuning

In the first part, we were asked to use a predefined model BertForSequenceClassification that comes with an untrained classifier head. The main plots for train loss, validation loss, MCC curve, and learning rate plot are given in Fig. 3, Fig. 4, and Fig. 5.

We can see from Fig. 4 that MCC is generally over 0.57 after some time which can be considered a high value. This performance has been achieved after finding the best parameters using the Optuna library. The following hyperparameters were used during training:

- learning rate: 1.1521858230688484e-05.

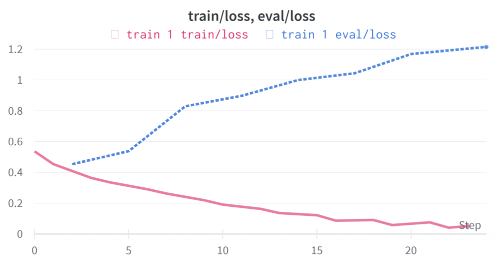

Fig. 3. Train and Eval Loss for BertForSequenceClassification model.

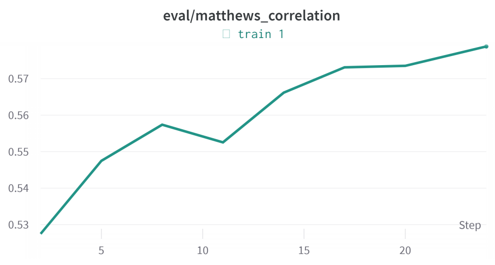

Fig. 4. MCC curve for BertForSequenceClassification model.

- train batch size: 8.
- eval batch size: 16.
- epochs: 8.

The trained model is pushed to the hub in Hugging Face, given at the start of the report.

Part 2: Proposed BERT Fine-Tuning

In this part, results are given in Fig. 6 and Fig. 7. We can see from Fig. 7 that MCC is nearly 0.35 after some steps, significantly lower than the vanilla model. This performance has been achieved after finding the best parameters using the Optuna library. The following hyperparameters were used after the hyperparameter search:

- learning rate: 9.685521437204136e-04.
- batch size: 64.
- max length: 32.
- epochs: 10.

For the main framework, I have utilized the codes given in the tutorial with some modifications. Since the proposed model wasn't required to be pushed to the hub, I didn't conform to the conventions of Hugging Face API.

IV. Discussions & Conclusions

All in all, the main idea of this investigation was to explore and understand how to fine-tune

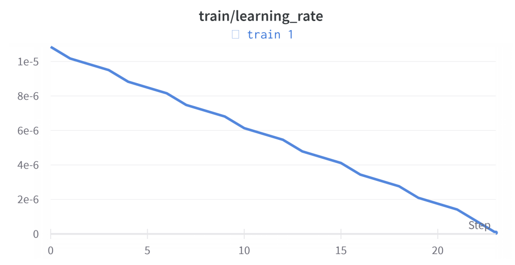

Fig. 5. Learning rate curve for BertForSequenceClassification model.

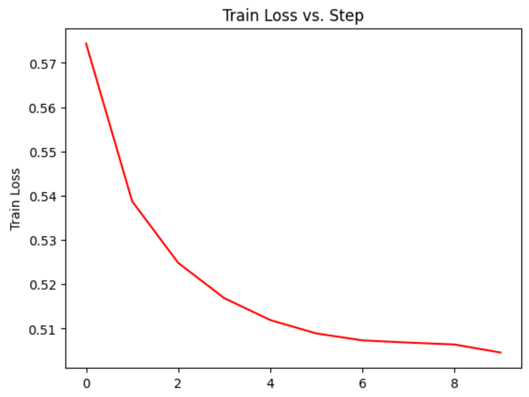

Fig. 6. Train loss for the proposed model.

a transformer model and make adjustments to its architecture. In the first part, I used the Hugging Face API to fine-tune a Bert model for syntax classification and push it to the hub. For this task, I tried a variety of hyperparameter values during hyperparameter search and used the best values for training the model. Since I have used a pre-defined model from Hugging Face in the first part, I have only used the [CLS] token as a document representation in the first part (see Fig. 1). From Fig. 4, we can see that MCC is relatively high, meaning the [CLS] token was indeed able to represent the document optimally. Nevertheless, BERT can be considered an 'old' model due to the fast-changing nature of deep learning research. Thus, newer models might be able to achieve even higher MCC scores.

In the second part, I tried to challenge the idea that using only the [CLS] token is sufficient and pooled all the tokens of the BERT by taking their mean (see Fig. 2). As we can see from Fig. 7, this was a bad idea. Not only the MCC score dropped, but also the training loss of the model also increased (see Fig. 6). Furthermore, the robustness of the model has also decreased, meaning fluctuation in MCC scores during hyperparameter search was significantly apparent. Furthermore, instead of a linear classifier, I have also trained a 2-layer multilayer perceptron (MLP) to see if there would be any improvements. However, since there weren't significant changes to the results, I didn't share the results.

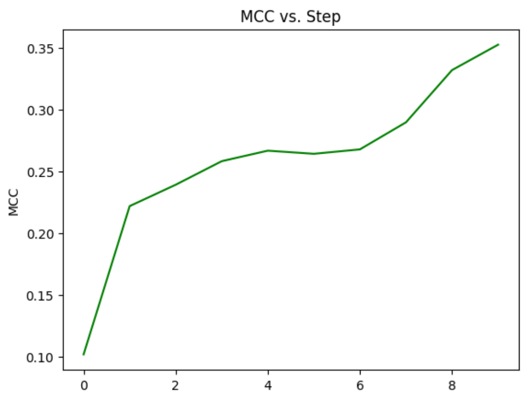

Fig. 7. MCC curve for the proposed model.

In the end, I learned how to utilize Hugging Face API along with fine-tuning a popular transformer model. Furthermore, by coming up with a new way to represent a document, I have learned how to modify the model and understand why [CLS] token is usually preferred over other tokens in the hidden state.

References

[1] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[2] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," 2019.

[3] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing," 2019.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.