Natural Language Processing Homework-2

Mohit Yadav
yadav171@umn.edu

1 Introduction

The goal of this homework was to fine-tune a pre-trained large language model from Hugging Face and analyze the results. This report outlines my methodology for doing so and includes an analysis of the results obtained from the fine-tuned model.

2 Description of Task and Model

For this assignment, I chose the task of sentiment analysis using the Stanford Sentiment Treebank (SST-2) dataset (Socher et al., 2013), which contains movie reviews labeled with sentiment labels. The labels are either Positive or Negative. The dataset includes 67,349 labeled training examples, 872 labeled validation examples, and 1,821 unlabeled test examples. For this task, I selected the pretrained BERT model (Devlin et al., 2018) and fine-tuned it on the SST-2 dataset.

3 Hardware Used

I used a Dell Alienware system equipped with an NVIDIA GeForce RTX 3090 Ti GPU with 24GB of memory to train and test my model.

4 Model Training

I trained the model for 10 epochs on the hardware previously described. To monitor the training process, I used Weights & Biases. Throughout the training, I tracked various parameters to ensure proper performance. The plot of the time-weighted average training loss per batch, shown in Figure 1, indicates that the model was training effectively, as the loss consistently decreased with each step.

Figure 2 presents the training and validation loss per epoch. While the training loss steadily decreased as the training progressed, the validation loss increased, signaling clear overfitting of the model. This trend is also evident in the accuracy plots in Figure 3, where the training accuracy approaches 100%, while the validation accuracy slightly decreases.

Given the simplicity of the task, the pre-trained model was able to learn the classification task within a single epoch, and any further training led to overfitting.

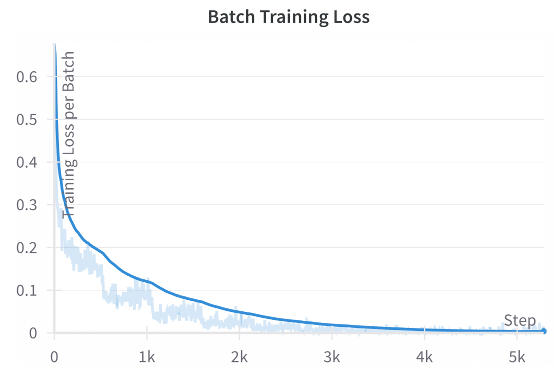

Figure 1: Training Loss per Step.

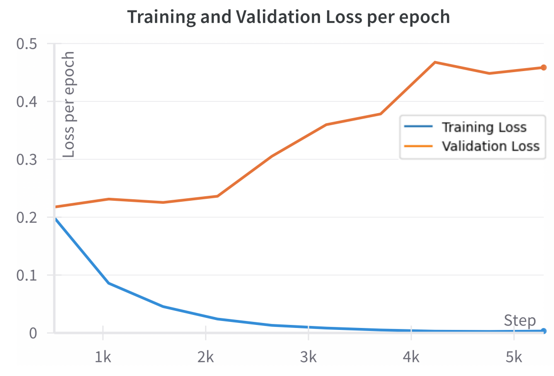

Figure 2: Training and Validation Loss per epoch

5 Evaluation Metric

I used accuracy as the evaluation metric for my model training.

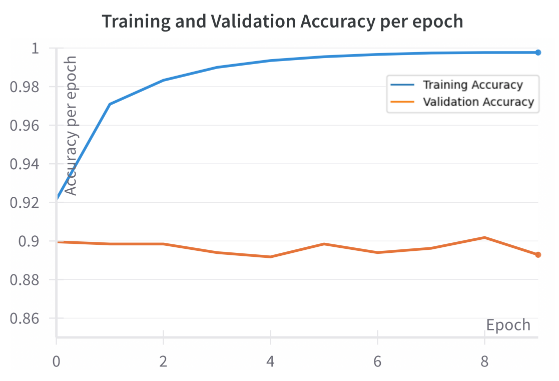

Figure 3: Training and Validation Accuracy per epoch

6 Test Set Performance and Comparison

Since the dataset does not contain labeled test data, I was unable to perform a quantitative analysis on the test set. However, I conducted a similar analysis on the validation set. I achieved a maximum accuracy of 90.18% of the validation data. This result is slightly lower than the BERT base model performance of 91.2% on the same data. The small difference may be attributed to the lack of a precise data cleaning step. If this achieved accuracy translates to the test set, I would rank 62nd on the leaderboard for the Sentiment Analysis on SST-2 Binary Classification task on the Papers with Code website.

7 Training and Inference time

Training the model for 10 epochs on the training data with a batch size of 128, using the hardware described in 3, took 15.86 minutes. In contrast, making an inference on a single data point takes approximately 13 milliseconds. Both of these times were calculated in the code and are printed in the supplementary code file.

8 Hyperparameters

The hyperparameters utilized during training are detailed in Table 1.

Parameter	Value
Epochs	10
Batch Size	128
Learning Rate	2e-5
Optimizer	Adam
Scheduler	StepLR with $\gamma = 0.9$

Table 1: Hyperparameters

9 Error Analysis

The analysis of incorrect predictions can be found in the supplementary file *CSCI5541-F24-HW2-Mohit-Yadav.csv* attached to this submission. I conducted the error analysis based on the errors proposed by (Eremyan). The file includes incorrect predictions from the models, along with their error types and potential solutions.

10 Error Visualization

To visualize the errors, I extracted the *hidden_states* from the model outputs and performed Principal Component Analysis (PCA) to reduce the dimensionality from 768 to 2. I then created a scatter plot of the results using *matplotlib*. The output is displayed in Figure 4.

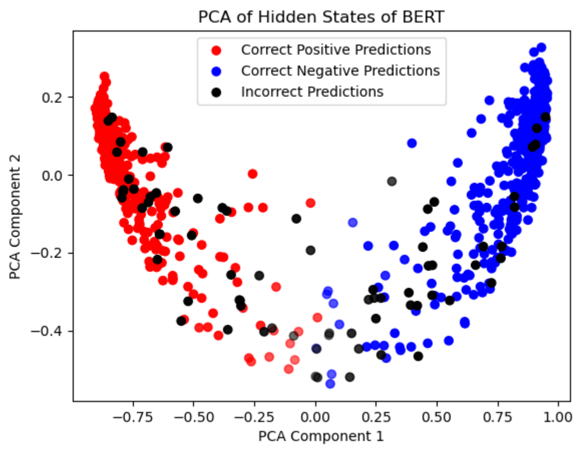

Figure 4: 2D projection of the model outputs after PCA

11 Most Challenging Part

In this assignment, I struggled with creating the custom training class in the code. The difficulty arose because we inherited from the Trainer module, which utilized a different model creation pipeline than what I was accustomed to. I found it challenging to determine which specific parts I needed to code and which aspects were abstracted by the inherited class.

12 Other Resources Used

I used ChatGPT to debug the code and improve the English in this report. I used code in this video (Biases, 2020) to utilize Weights & Biases.

References

Weights Biases. 2020. integrate weights biases with pytorch.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Rudolf Eremyan. Toptal developers. Sep 24, 2024.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.