Lab-06: AWS Machine Learning University Module 2 Lab 05 Fine Tuning Bert

Martin Demel

Department of Science, Technology, Engineering & Math, Houston Community College

ITAI 2376 Deep Learning in Artificial Intelligence

Patricia McManus

February 26th, 2025.

Introduction

In this laboratory assignment, my main objective was understanding how large language models, like BERT, can be adapted to new tasks with limited data. This lab introduced me to the transformer's library, as well as key concepts such as tokenization, attention masks, and partial freezing of network layers to expedite training.

Body

From the start, I knew that BERT-like models are computationally heavy, so I was prepared to run into potential memory or runtime challenges if I tried to process too large a dataset. Indeed, the lab explicitly advised using only 2,000 data points out of the full 56,000 to keep training time manageable. This limitation became a practical lesson in balancing dataset size and training resources.

I discovered that fine-tuning a language model is remarkably powerful. By freezing the majority of the BERT layers and only training the final classification layer, I could leverage the pre-trained language representations while reducing the computation needed to update over 66 million parameters. I was also reminded of how critical tokenization is for transforming raw text into a numerical form that models can process.

Additionally, I gained a deeper appreciation for the validation process. With only a fraction of the entire dataset, I needed to carefully split my data (90% for training, 10% for validation) to accurately track my model's performance and tune hyperparameters such as the learning rate. Watching the validation loss and accuracy across epochs helped me decide if more epochs were truly beneficial.

Interesting challenge was the decision on how many epochs to train. The lab suggested an initial run of 10 epochs but also posed the question: would more epochs improve the model's performance? (See Figure 1 for the 10-epoch training results and Figure 2 for the 20-epoch results.) When I trained for 20 epochs, I observed that the validation loss improved significantly, dropping from about 0.428 at 10 epochs to 0.341 at 20 epochs. This drop was accompanied by a rise in validation accuracy (from approximately 0.795 to around 0.885 in the later epochs), indicating that further training indeed helped the model generalize better.

```
Epoch 1: train loss 0.656, train acc 0.616, val loss 0.633, val acc 0.620, seconds  47.281
Epoch 2: train loss 0.630, train acc 0.626, val loss 0.606, val acc 0.620, seconds  30.023
Epoch 3: train loss 0.606, train acc 0.655, val loss 0.583, val acc 0.625, seconds  29.996
Epoch 4: train loss 0.585, train acc 0.689, val loss 0.554, val acc 0.700, seconds  30.058
Epoch 5: train loss 0.560, train acc 0.723, val loss 0.536, val acc 0.650, seconds  30.064
Epoch 6: train loss 0.538, train acc 0.753, val loss 0.503, val acc 0.755, seconds  30.061
Epoch 7: train loss 0.513, train acc 0.770, val loss 0.483, val acc 0.740, seconds  30.108
Epoch 8: train loss 0.494, train acc 0.785, val loss 0.459, val acc 0.785, seconds  30.060
Epoch 9: train loss 0.476, train acc 0.791, val loss 0.447, val acc 0.775, seconds  30.087
Epoch 10: train loss 0.463, train acc 0.797, val loss 0.428, val acc 0.795, seconds  30.060
```

Figure 1 - 10 Epochs

```
Epoch 1: train loss 0.660, train acc 0.615, val loss 0.641, val acc 0.620, seconds  30.084
Epoch 2: train loss 0.631, train acc 0.633, val loss 0.618, val acc 0.640, seconds  30.026
Epoch 3: train loss 0.609, train acc 0.650, val loss 0.589, val acc 0.630, seconds  29.999
Epoch 4: train loss 0.585, train acc 0.691, val loss 0.566, val acc 0.645, seconds  30.058
Epoch 5: train loss 0.564, train acc 0.717, val loss 0.540, val acc 0.710, seconds  30.054
Epoch 6: train loss 0.540, train acc 0.743, val loss 0.517, val acc 0.730, seconds  30.094
Epoch 7: train loss 0.519, train acc 0.766, val loss 0.492, val acc 0.780, seconds  30.055
Epoch 8: train loss 0.501, train acc 0.784, val loss 0.472, val acc 0.795, seconds  30.079
Epoch 9: train loss 0.484, train acc 0.791, val loss 0.460, val acc 0.845, seconds  30.076
Epoch 10: train loss 0.467, train acc 0.800, val loss 0.436, val acc 0.840, seconds  30.064
Epoch 11: train loss 0.458, train acc 0.799, val loss 0.421, val acc 0.800, seconds  30.086
Epoch 12: train loss 0.429, train acc 0.818, val loss 0.402, val acc 0.845, seconds  30.067
Epoch 13: train loss 0.428, train acc 0.814, val loss 0.399, val acc 0.815, seconds  30.101
Epoch 14: train loss 0.415, train acc 0.815, val loss 0.389, val acc 0.875, seconds  30.065
Epoch 15: train loss 0.410, train acc 0.820, val loss 0.373, val acc 0.870, seconds  30.094
Epoch 16: train loss 0.402, train acc 0.821, val loss 0.362, val acc 0.860, seconds  30.061
Epoch 17: train loss 0.390, train acc 0.836, val loss 0.354, val acc 0.880, seconds  30.088
Epoch 18: train loss 0.393, train acc 0.828, val loss 0.360, val acc 0.880, seconds  30.061
Epoch 19: train loss 0.386, train acc 0.826, val loss 0.342, val acc 0.885, seconds  30.066
Epoch 20: train loss 0.384, train acc 0.825, val loss 0.341, val acc 0.870, seconds  30.087
```

Figure 2 - 20 Epochs

Working with BERT has reinforced for me the importance of transfer learning in NLP. Initially, I expected that using such a large architecture on a small dataset might lead to overfitting. However, by freezing the bulk of the BERT layers and only updating the classification layer, I was able to strike a balance between leveraging powerful representations and preserving model generality.

I was also pleasantly surprised by how, with careful resource management, we can effectively harness a large model on a relatively small dataset. This taught me to be more open to exploring pre-trained transformers in future projects, especially when data or computational resources are limited.

Conclusion

Through this lab, I sharpened my understanding of how pre-trained transformers such as BERT can be adapted for sentiment classification. Training for 20 epochs (Figure 2) definitely led to better validation performance than 10 epochs (Figure 1), showcasing the importance of iteration and patience when fine-tuning large models. I also learned to effectively manage my computational resources by freezing layers, adjusting the batch size, and monitoring GPU memory.

Above all, this experience underscored how crucial pre-processing, data splitting, and model monitoring are to success in machine learning projects. As I look ahead to future courses or professional pursuits, I will carry forward both the technical skills I have honed and the mindset of continuous experimentation.

Resources:

GeeksforGeeks. (2024, December 10). *BERT Model NLP*. GeeksforGeeks. https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/

GeeksforGeeks. (2024, January 10). *Self attention in NLP*. GeeksforGeeks. https://www.geeksforgeeks.org/self-attention-in-nlp/

GeeksforGeeks. (2024, December 28). *Difference between encoder and decoder*. GeeksforGeeks. https://www.geeksforgeeks.org/difference-between-encoder-and-decoder/