|  |  |  |
| --- | --- | --- |
|  |  | Research notes  NAZMUS SAMMO-103512692 |

Date: 24 September 2023

In the last report I have discussed all the data pre-processing steps which comes before in which way we want to fine tune our model. After we are done with the data pre-processing. We need to take into consideration which way we want to fine tune our model. Basically, this depends more on what we want to achieve by fine tuning this model. For my case I want to fine tune the BERT model for question answering. For question answering there are several techniques of fine tuning which we can adapt in the below paragraph I will shed a light upon a few popular techniques which can be done to fine tune our pre trained model BERT.

Fine Tuning Techniques:

If we have a rich dataset specific to our question-answering (QA) task, our best bet is to go with **Full Fine-tuning**. We would adjust every layer in the model to tailor it to our specific needs. This is especially useful if our task differs substantially from the original training task, as it allows us to maximize the model's performance for our problem.

On the flip side, if we're working with a limited dataset, we should consider **Feature Extraction**, also known as "Frozen Features." In this approach, we would keep most of the pre-trained model untouched and only modify the final layers. This will help us avoid overfitting while still gaining from the foundational knowledge already present in the model.

When we're sitting on a moderately sized dataset, **Layer-wise Fine-tuning** could be the perfect compromise. We'd start by freezing the lower layers of the model, then gradually unfreeze and fine-tune the upper ones. This allows us to balance the benefits of the pre-trained model with customization for our QA task.

If we're looking to get really detailed with our fine-tuning, **Adaptive Fine-tuning** is the way to go. By using specialized techniques like slanted triangular learning rates, we can adjust the learning rate for different layers to our needs. This can be helpful if we believe that some layers require more fine-tuning than others to achieve optimal performance for our QA system.

If we want our model to be a jack-of-all-trades and excel in multiple areas, we'd be looking at **Continual Learning**. Here, we'd train the model sequentially on different tasks so it learns to adapt without forgetting prior tasks. This is particularly beneficial if we have multiple, sequential QA tasks to tackle.

Alternatively, if our tasks are interrelated, we might opt for **Multitask Fine-tuning**. In this strategy, we'd train the model on multiple tasks simultaneously, allowing it to develop a more rounded understanding that can be beneficial across tasks.

If our QA system needs to be specialized in a certain area, like legal or medical queries, **Domain-adaptive Fine-tuning** could be our best option. We'd first adapt the model to the broader domain and then fine-tune it for the specific QA tasks we're interested in. This makes the model both a domain expert and specialized for our tasks.

Lastly, when we want a quick and minimally invasive adaptation, **Prefix-tuning** can come to the rescue. In this approach, we add some task-specific parameters at the beginning, enabling the model to better handle our specific QA task while maintaining most of its original behaviours.

Now I want to talk a bit about the datasets I have chosen:

SQuAD v2:

So, we've decided to use the SQuAD v2 dataset to fine-tune our BERT model for question-answering, and there are some good reasons for that. First off, this dataset is huge—it's got over 100,000 questions that can be answered and another 50,000 that can't. This is awesome because it teaches our model to find the right answers but also helps it figure out when there's no answer to give. The questions come from Wikipedia, which means our model will get good at answering all sorts of questions on different topics. The dataset is already divided into a big training set and a smaller validation set, so we can train and test our model easily. Plus, the dataset is trusted by researchers, which gives our project some extra credibility. And don't worry about size; the whole thing is about 175 MB, so it's not going to hog all our computer space. All in all, using SQuAD v2 lines up perfectly with our goal to make a question-answering model that's both smart and practical.

MedMCQA:

This dataset is stacked with real-world stuff—over 194,000 multiple-choice questions from actual medical entrance exams like AIIMS & NEET PG. And it's not just the sheer number; the questions span 21 medical subjects and cover 2,400 healthcare topics. That's like a full-blown medical course packed into a dataset! What's cooler is that each question tests more than 10 types of reasoning skills. That means our model will learn to think deeply about a wide range of medical topics. This dataset even includes detailed explanations for each answer, which is like having a medical textbook built right in. And since the data comes from a mix of mock tests, past AIIMS exams, and NEET exams, we've got a balanced training and testing ground.

Fine Tuning Technique PEFT LoRA:

PEFT stands for Parameter-Efficient Fine-Tuning. It's a technique designed to fine-tune large-scale language models like BERT in a way that requires fewer parameters to be updated during the training process, making it more efficient in terms of computational resources.

**Resource Efficiency**: PEFT allows you to fine-tune the model by only updating a subset of the parameters, thus reducing the computational load. This is particularly beneficial if you have limited computational resources.

**Faster Training**: Because fewer parameters are being updated, the training process is typically faster. This can be crucial when you need to iterate and experiment multiple times.

**Reduced Overfitting Risk**: By updating fewer parameters, the model is less likely to overfit to the specific task or dataset. This is beneficial when working with smaller datasets, which is often the case in specialized fields like medical or legal question answering.

**Preserve Generalization**: Since not all parameters are fine-tuned, the model retains more of its pre-trained generalization capabilities. This is helpful for question-answering tasks that might require a broad understanding of language and knowledge.

**Task-Specific Adaptation**: Despite being parameter-efficient, PEFT still allows for targeted tuning. You can specify which layers or parameters are more crucial for your task and focus on fine-tuning those.

Problems with Full Fine Tuning:

Full fine-tuning of all layers in a model like BERT comes with its own set of challenges. One major concern is the risk of overfitting, especially when dealing with small datasets. The model could become too tailored to the training data, losing its ability to generalize to new information. Additionally, the computational cost can be quite high. Fine-tuning every layer in a large, complex model requires significant computational resources, often necessitating specialized hardware like high-end GPUs or TPUs. So, the problem I was facing with full fine tuning was, when I was working with medmcqa dataset when I used the full dataset for fine tuning with colab the training time was 16 hours I was still doing that anyway but after around 7 hours of training I couldn’t use it as colabs time expired. So, I bought the colab pro to use faster gpu, with v100 gpu the training time was reduced, and I was getting f1 score of 0.4 then on the last step the storage of colab pro was full so I couldn’t proceed with the training anymore. So, this time to overcome these overfitting and resource problem I am going with the PEFT LORA techniques.

Fine-Tuning BERT With SQuAD V2:

Previous fortnight I have experimented BERT with squad model with the technique of full fine tuning. But as I didn’t know that I have keep the results, but I couldn’t. But now I am taking different approach to fine tune the model. So, this time I am going to use PEFT lora instead of fully fine tune the model.

The Tokenize function:



PEFT LORA configuration:

A screen shot of a computer

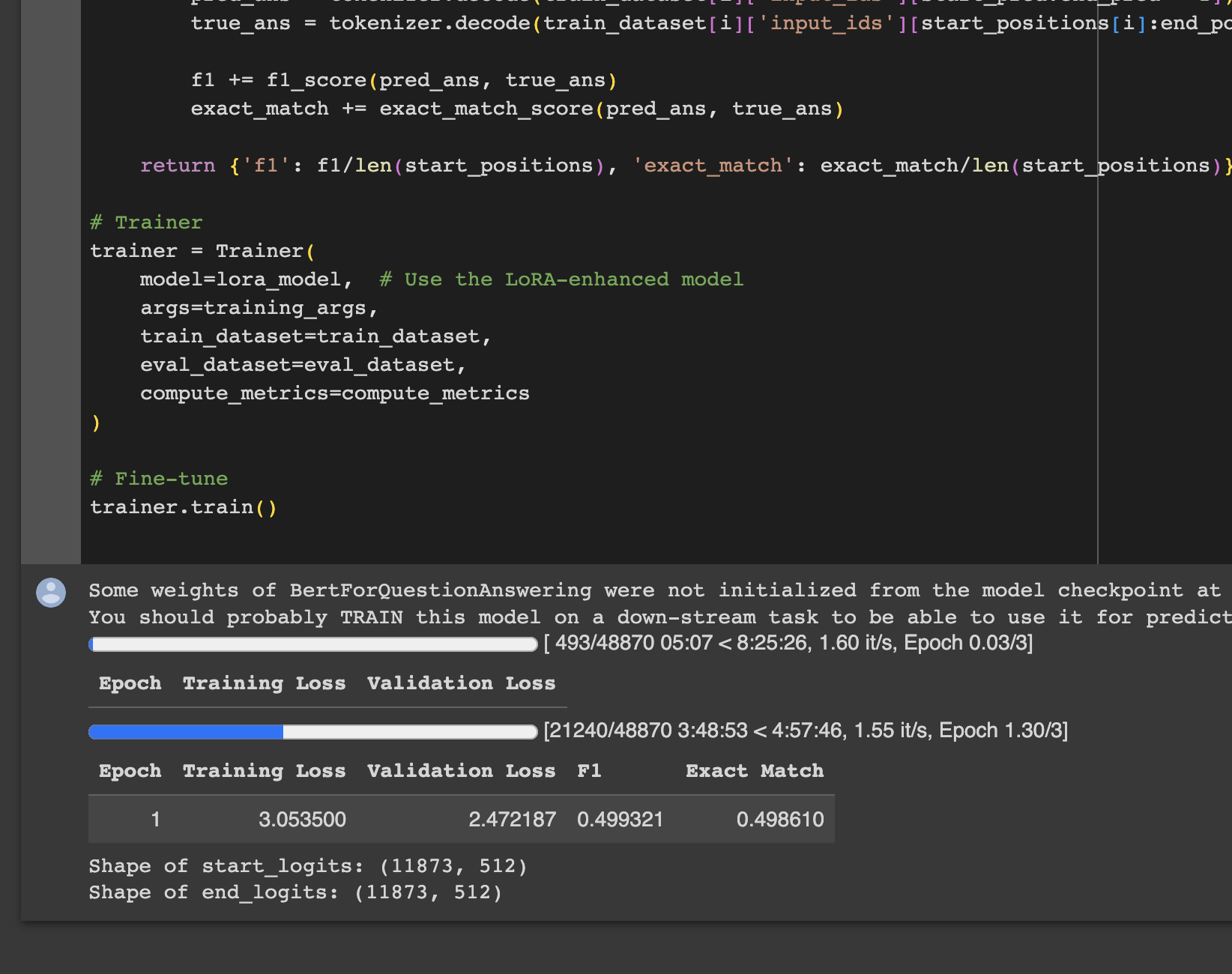
Description automatically generated

Other Configs:

A screen shot of a computer screen

Description automatically generated

Result of the training:



So for this training when I was using peft lora techniques for fine tuning bert the result after running 1 epoch was was good enough. Because the f1 and exact match scores looks solid. As well as if look at training loss which is greater than exact match, so the model is not overfitting as well. Which is a good a sign compared to the base model evaluated with squad dataset.

Fine Tuning BERT with MedMCQA:

Tokenize function and Lora config: (using the same lora config as I used in the squad)

A screen shot of a computer program

Description automatically generated

Other Functions:

A screen shot of a computer

Description automatically generated

Result of the Training:



Reference:

*Tuners*. Available at: https://huggingface.co/docs/peft/main/en/package\_reference/tuners

*Prompt tuning for causal language modeling*. Available at: https://huggingface.co/docs/peft/main/en/task\_guides/clm-prompt-tuning

*Semantic segmentation using Lora* *Semantic segmentation using LoRA*. Available at: https://huggingface.co/docs/peft/main/en/task\_guides/semantic\_segmentation\_lora

*P-tuning for Sequence Classification* *P-tuning for sequence classification*. Available at: https://huggingface.co/docs/peft/main/en/task\_guides/ptuning-seq-classification

*Lora for token classification* *LoRA for token classification*. Available at: https://huggingface.co/docs/peft/main/en/task\_guides/token-classification-lora

Tinkerd.net. (n.d.). 'Machine Learning - LoRA', Available at: <https://tinkerd.net/blog/machine-learning/lora/>

Stanford University. (n.d.). 'Final Report 169568390', Available at: <https://web.stanford.edu/class/cs224n/final-reports/final-report-169568390.pdf>

Jaotheboss. 'PEFT with BERT', Medium, Available at: <https://jaotheboss.medium.com/peft-with-bert-8763d8b8a4ca>

Available at: <https://www.youtube.com/watch?v=gHsm1F2muiM>

Keras.io. (n.d.). 'Parameter-Efficient Fine-Tuning of GPT-2 with LoRA', Available at: <https://keras.io/examples/nlp/parameter_efficient_finetuning_of_gpt2_with_lora/>

Reddit. 'Doubt Regarding LoRA', Available at: <https://www.reddit.com/r/LanguageTechnology/comments/13dhsab/doubt_regarding_lora/>