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

Date: 8 October 2023

On the last report, we have discussed about different fine-tuning techniques like Full fine tuning, Feature Extraction, Layer-Wise Fine Tuning, Adaptive Fine Tuning and LoRA. This reports main focus is to play with the P-Tuning. But before knowing about P-tuning we need have a good understanding about PEFT. This fine tuning techniques is one of the most popular fine tuning technique backed by hugging face PEFT library.

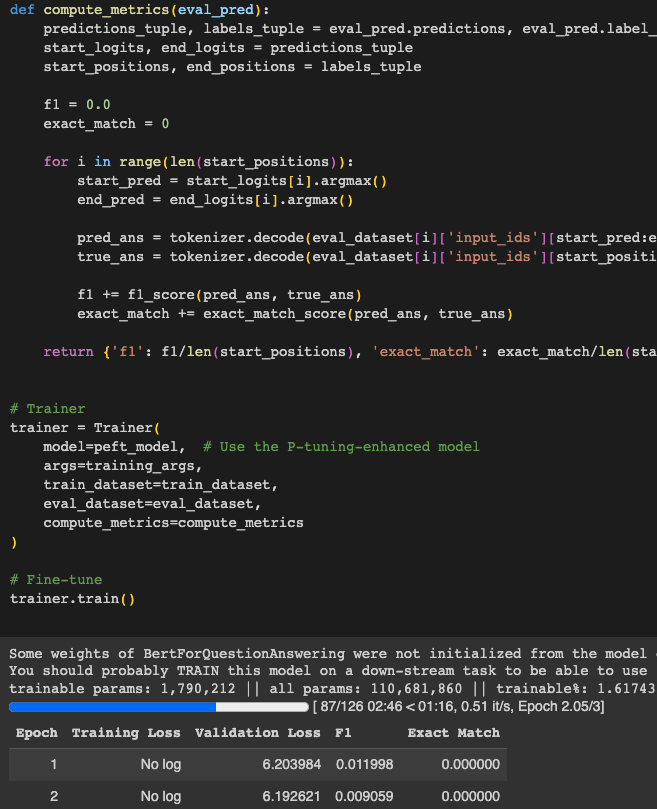
Parameter-efficient Fine-tuning (PEFT) is a strategic approach in fine-tuning that refines pre-trained models by adjusting only a subset of their parameters, rather than the entire model. This method provides several advantages, especially for large-scale models. First, it substantially reduces computational costs and speeds up the fine-tuning process. Second, by tweaking fewer parameters, PEFT mitigates the risk of overfitting, especially in scenarios with limited training data. Additionally, PEFT can help overcome catastrophic forgetting, a phenomenon where the model loses its initial pre-trained knowledge during exhaustive retraining. Among PEFT there are many types of peft-techniques available, among them LoRA, qLoRA, Prefix tuning and P-Tuning are the best resulting ones. In last report we have put our mind into LoRA, so this week we will try to talk a bit about the other two.

Prefix tuning is an innovative technique designed to improve the versatility of large language models. Instead of training the entire model on a new task, prefix tuning introduces a series of task-specific vectors or prefixes to the beginning of the input. This approach is ingenious because these prefix parameters are the only elements that undergo optimization. They integrate with the model's hidden states across every layer, and the input tokens can attend to them just like they would with other tokens. The advantage of prefix tuning is its efficiency— it utilizes approximately 1,000 times fewer parameters than a full finetuning process. This means that a single expansive language model can be adapted for a multitude of tasks without the need for extensive retraining.

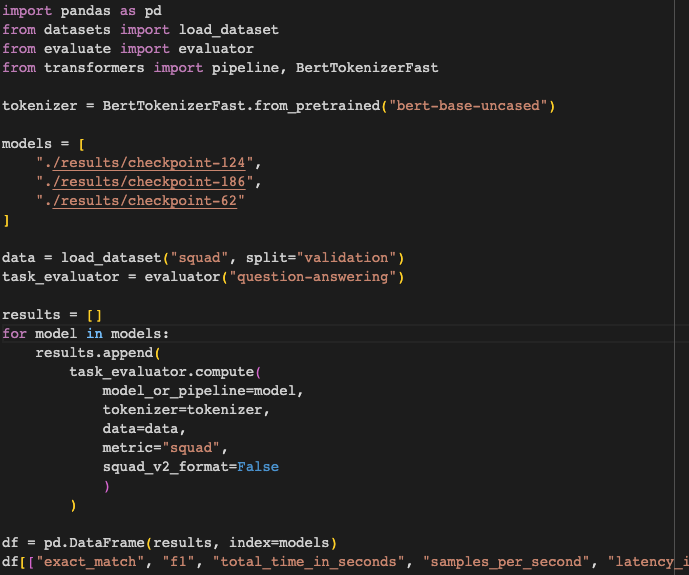
P-tuning, short for "Prompt-tuning", offers a novel approach to fine-tuning large language models. Unlike conventional methods that require retraining vast numbers of parameters, P-tuning focuses on optimizing specific tokens, called prompts, to guide the model towards desired behaviors for downstream tasks. This method reduces the resource-intensive nature of training large models, making it more efficient. Traditional prompt methods often rely on manual crafting, which can be labor-intensive and may require vast validation sets to discern the most effective prompts. P-tuning, however, automates this process, searching for the best prompts within a continuous space. In essence, it merges the power of large language models with the flexibility of prompts, delivering specialized models that perform efficiently on specific tasks without the overhead of traditional fine-tuning.

# this week’s practical research overview

On week 8’s meeting, when I showed my work to my tutor, he find out an obvious error in my pipeline, and the error is for almost for all the epochs all other models my models f1 score and the exact match is same. Here in the function, what exact match function does is that, it compares the predicted value with ground truth and if these exactly matches then we get a exact match, but in f1 score we take the harmonic mean of precision and recall. In a general sense f1 should always be larger than exact match. But for me it was actually generating close values, not exactly identical but not close to same which defines there’s something wrong with it. I have tried to find out the where’s not functioning properly, experimented with different types approach, but I kept getting error. By giving my f1 and exact match function I asked chatGPT what’s wrong with it. I asked the same question again and again, but every time the summary it’s answer is, these 2 functions are working properly, if your exact match and f1 score is same, there might be something else which is why you are getting the result.

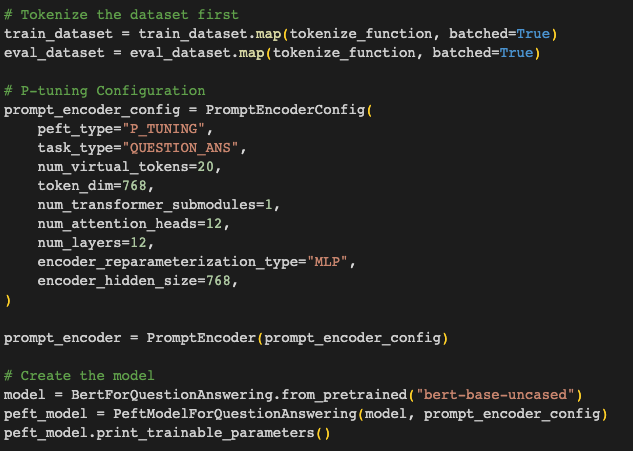


After trying the solve the error for a long time, I gave up and took another approach. There’s already a pre-build library in hugging face to evaluate a model. I tried to integrate that with my code but there was one component, model or pipeline it kept getting me error. So I have taken a different approach. So now, to evaluate my model, I finish the training for my model save it in the local directory then from the local directory I load that into my evaluate model or pipeline component then evaluate my trained model.



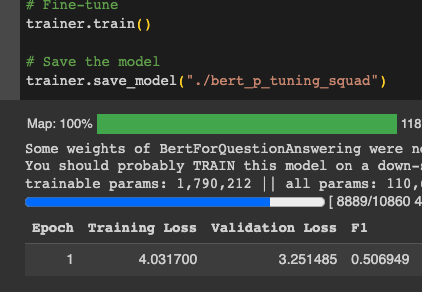
P-Tuning BERT with SQuAD:

On week 9 I have done PEFT LoRA to train the model and I got f1 score of around 49%. As this week my main goal was to try out P-tuning so I have played a bit with it and kept the dataset same so that we can compare the result.



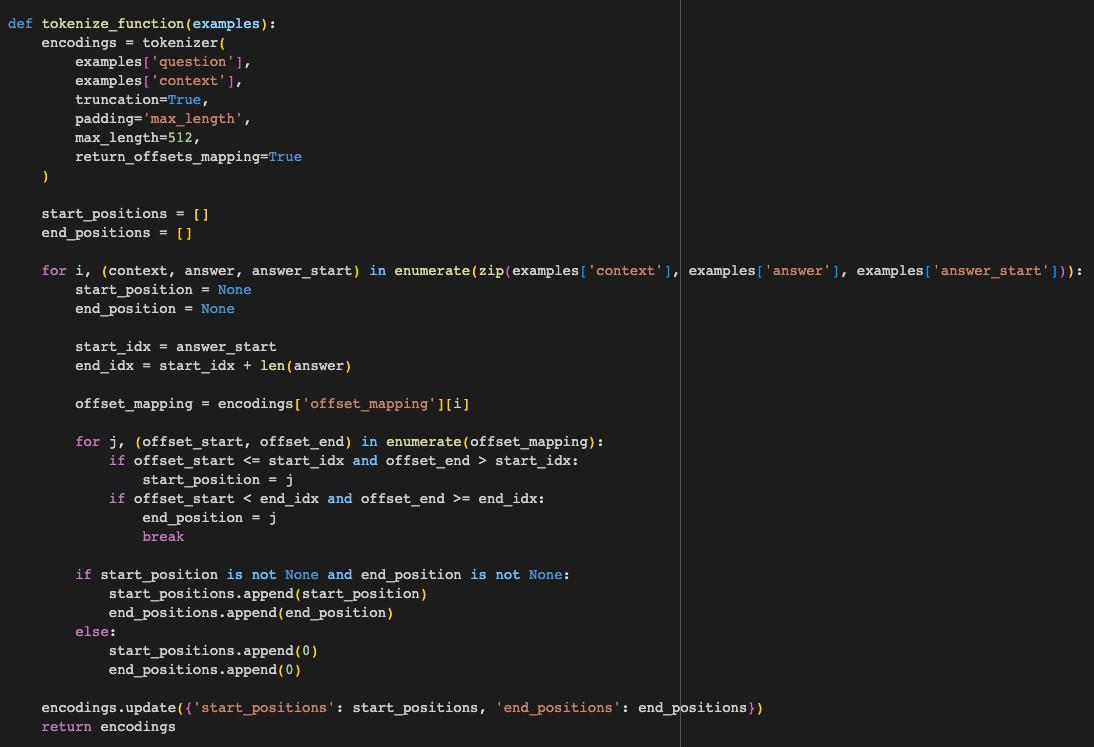
I was struggling to run the P-tuning configured model, and was getting error. So for BERT, when we tokenize the dataset the max length can be 512, BERT doesn’t support more than that. When we were configuring the P-tuning configuration, there’s one value called virtual tokens. So for p tuning virtual token + tokenize dataset’s max length should be 512 or less. The result for the peft lora and peft p tuning is like neck to neck.

Result for P-tuning with SQuAD:



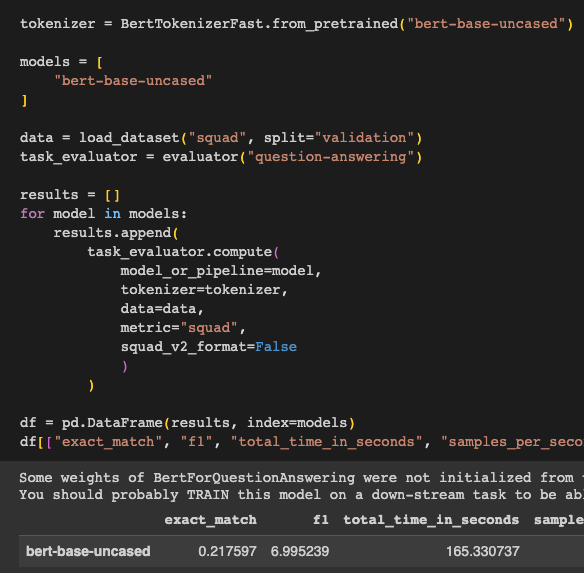
New Dataset CPGQA:

On last week’s meeting tutor suggested us to use the domain specific dataset. We were leaning a lot towards squad which is not our project goal. And he suggested that, we should work on CPGQA, other groups have worked on it and found out pretty good results with this dataset. This dataset is not that recognized dataset so there’s not a lot of information about it, and compared to all the datasets we have worked on, this is the smallest one. It has only 987 train set and 110 test set. Columns: answer: Contains the responses to the questions. answer\_start: An integer value specifying where the answer begins in the context text. question: Contains the queries for which answers are provided in the dataset. context: Provides a detailed backdrop or reference text from which the answer can be extracted. Tokenize Function for the dataset:

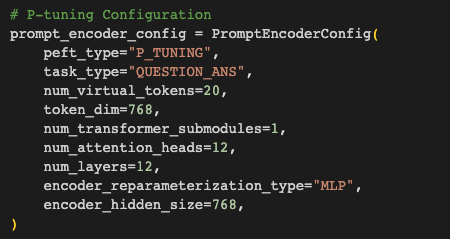


On our last meeting when we were discussing about individual research, one of my group mate presented that, by using the peft method he is getting significant drop on the model’s performance. After the research he found out that, BERT by itself is very tiny model(110M) but when we use peft method it uses only 1 percent of the models parameter(1M only) by training this small parameter, out model doesn’t improve a lot (which will shown in the screenshot). This is why, I have used this dataset CPGQA to fine tune BERT with 2 techniques- Full Fine tuning and PEFT P-Tuning. After fine tuning them with this small dataset I evaluate the model with squad validation split to see how it’s reacting on new data. (Also tutor suggested to do that)

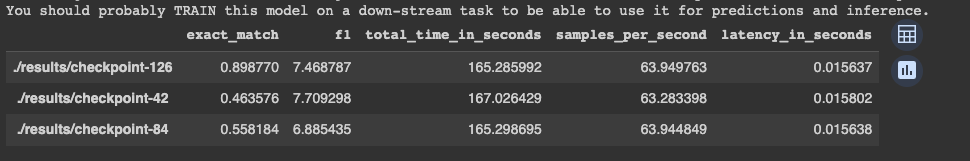
Before any fine tuning, BERT base models f1 score is- 6.9 percent



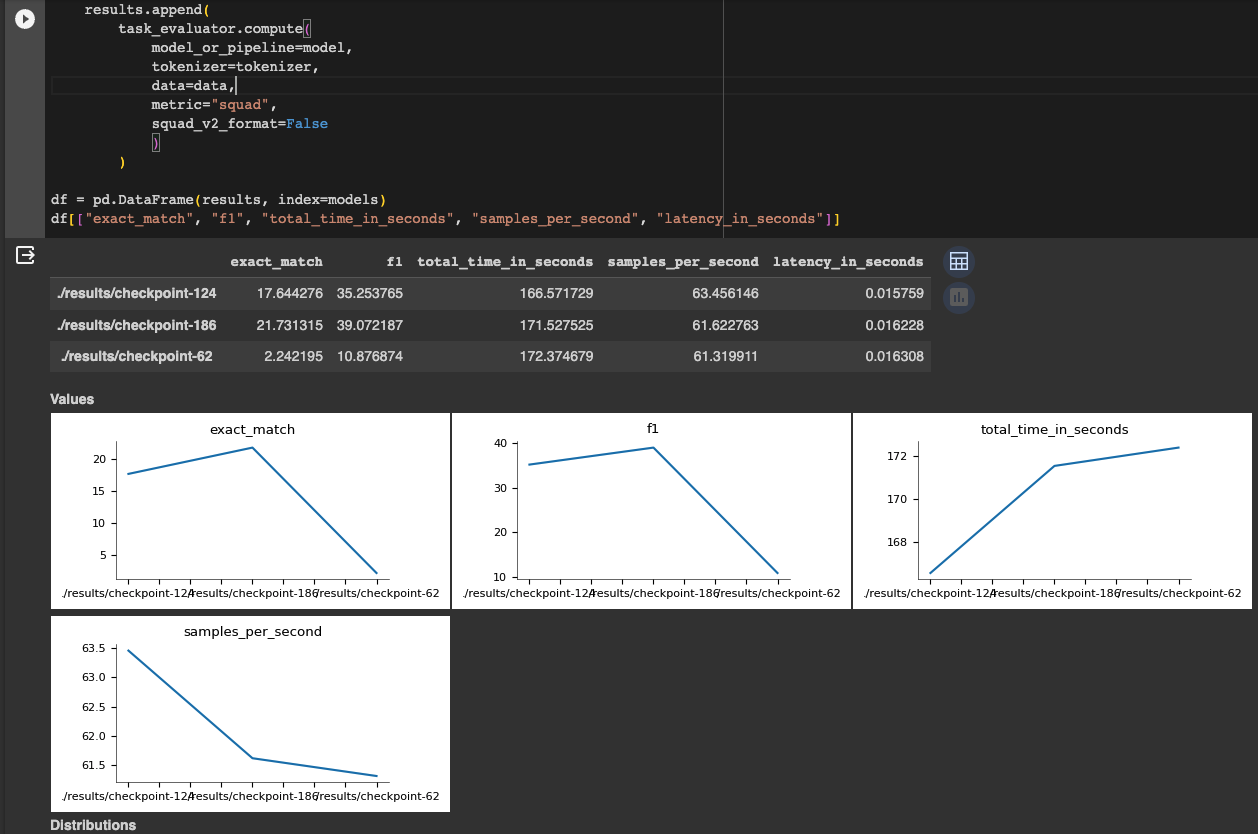
Fine tuned BERT with CPGQA by Using PEFT P-Tuning Method, evaluated on SQuAD:



Only 0.5 percent better f1 score than the base model.



Fine-tuned BERT with CPGQA by Using Full Fine Tuning Method, evaluated on SQuAD:



For the best model, the best f1 score of 40 percent is not bad when we consider that we are only using 983 sets of training data and for the validation we are totally using a different dataset.