Contents

Introduction

In this work, I am continuing to work with one of my previous projects. In this previous project, the NPL task of Question-Answering was performed using the BERT language model. To build the QA model, the "Simple Transformers" library was used. The pre-trained model used was '***bert-base-cased***' from Hugging Face: https://huggingface.co/bert-base-cased. This model was fine-tuned and tested using the traditional SQuAD dataset. The metric of Exact Match score was used to evaluate the performance of the model. The achieved EM score was 58.05%. To further improve the EM score of the model, some ideas were proposed as follows:

1. Changing the pre-trained model from BERT-base to BERT-large to have better context understanding. Also, considering other versions of the BERT model, such as RoBERTa, Electra or XLNet can lead to significant improvements in the EM score.
2. Performing hyperparameter optimization to find the best initial configuration of the model.

The goals of this work are as follows:

1. Testing out the solutions proposed in the previous work to see if a higher EM score can be achieved.
2. Evaluating the effect of fine-tuning performed on the pre-trained model regarding the QA task.

The workflow of this project is as follows:

1. Perform hyperparameter optimization.
2. Fine-tune the pre-trained model based on the best hyperparameter values.
3. Evaluate the effect of fine-tuning to the pre-trained model.

1. Hyperparameter optimization

To achieve higher performance the model was changed from BERT to RoBERTa. The pre-trained model of RoBERTa is '***roberta-case***' from Hugging Face: https://huggingface.co/roberta-base.

To perform hyperparameter optimization, the "W&B Sweeps" feature is used through the "Simple Transformers" library. W&B Sweeps helps you to automate hyperparameter search and visualize rich, interactive experiment tracking.

1.1 Configure a W&B sweep

To configure a W&B sweep, the user has to specify three key elements of the hyperparameter optimization process:

a) **method –** Specifies the search strategy (grid, random, bayes).
b) **metric** – Specifies the metric to be optimized and the optimization goal (min or max).
c) **parameters –** Specifies the hyperparameters and their values to explore.

The configuration of the sweep is shown below:

```
# Configuring the sweep

sweep_config = {
    # Specify the search strategy
    "method": "bayes",  # grid, random

    # Specify the metric to be optimized
    "metric": {"name": "train_loss", "goal": "minimize"},

    # Specify the hyperparameters and their values to explore
    "parameters": {
        "train_batch_size": {"values": [64, 128] },  # Discret values
        "learning_rate": {"min": 4e-5, "max": 4e-4}, # Range of values
    },
}
```

As shown above, the search method is ***bayes***, and the metric to be optimized is the ***training loss*** to minimize it. The hyperparameters to be explored are the ***training batch size*** and the ***learning rate***. For the training batch size, there are two discrete values (**64, 128**), and for the learning rate, there is a range of values (**4e-5 – 4e-4**).

1.2 Run the configured W&B sweep

To perform hyperparameter optimization the Colab environment was used. Due to the limitations presented by the training environment, for each pair of hyperparameter values, the model is set to be trained for only 3 epochs and the total number of performed explorations is 25. Also, it is worth mentioning that these limitations limited the batch size to 128 samples. When tried with higher values the environment run out of memory, failing the hyperparameter optimization process.

Based on the achieved training loss, evaluation loss, and the number of correct predictions, 10 explorations were classified as the top 10:

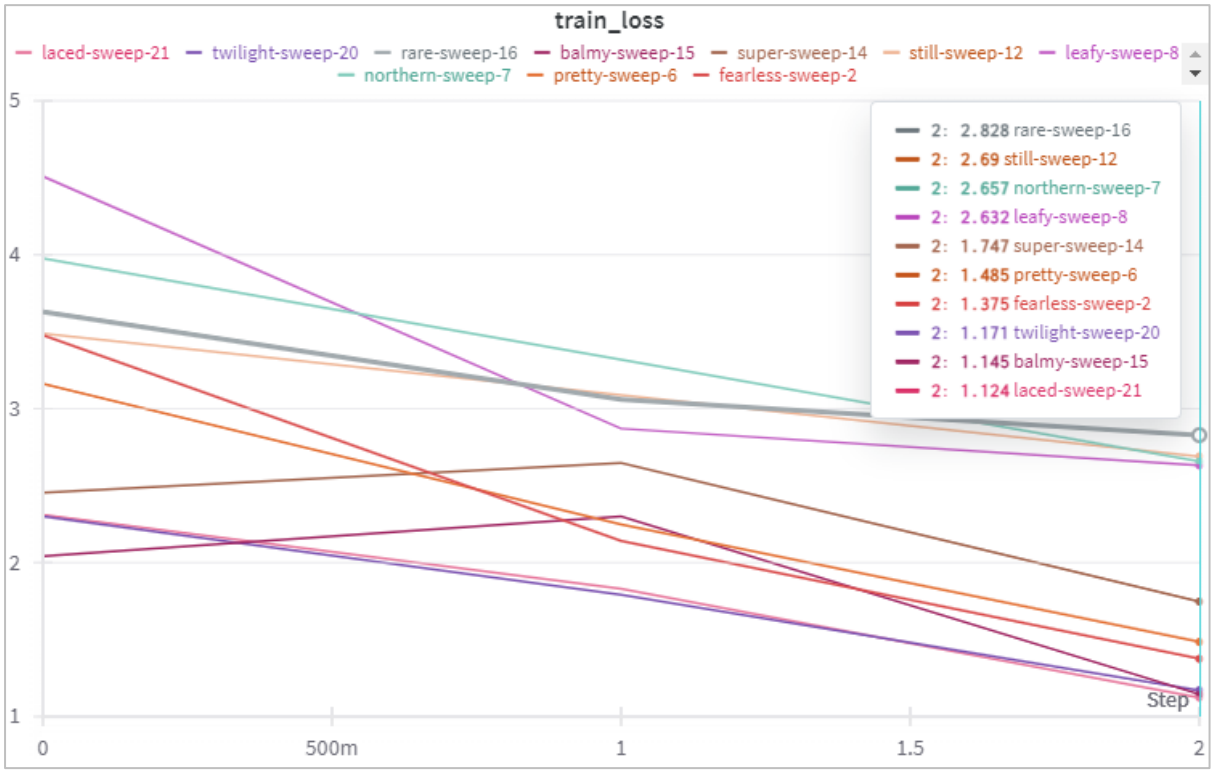

Fig. 1 Training loss for 10 top explorations

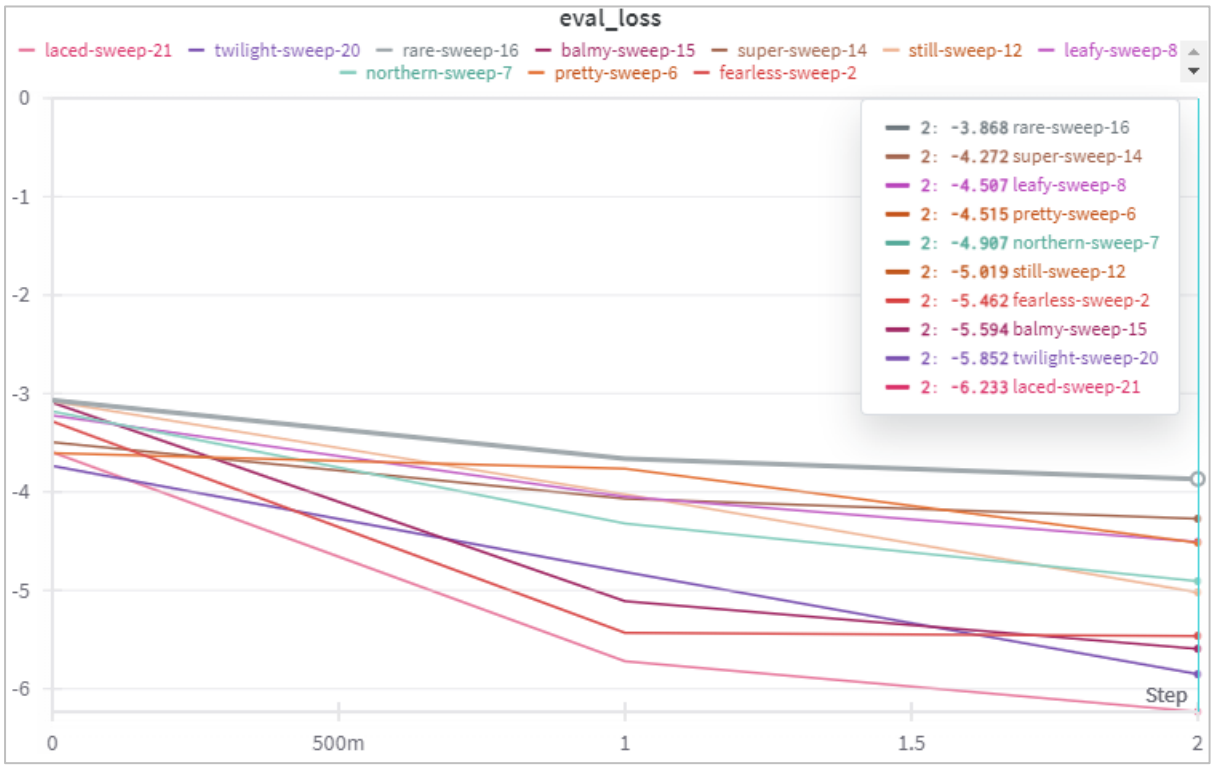

Fig. 2 Evaluation loss for 10 top explorations

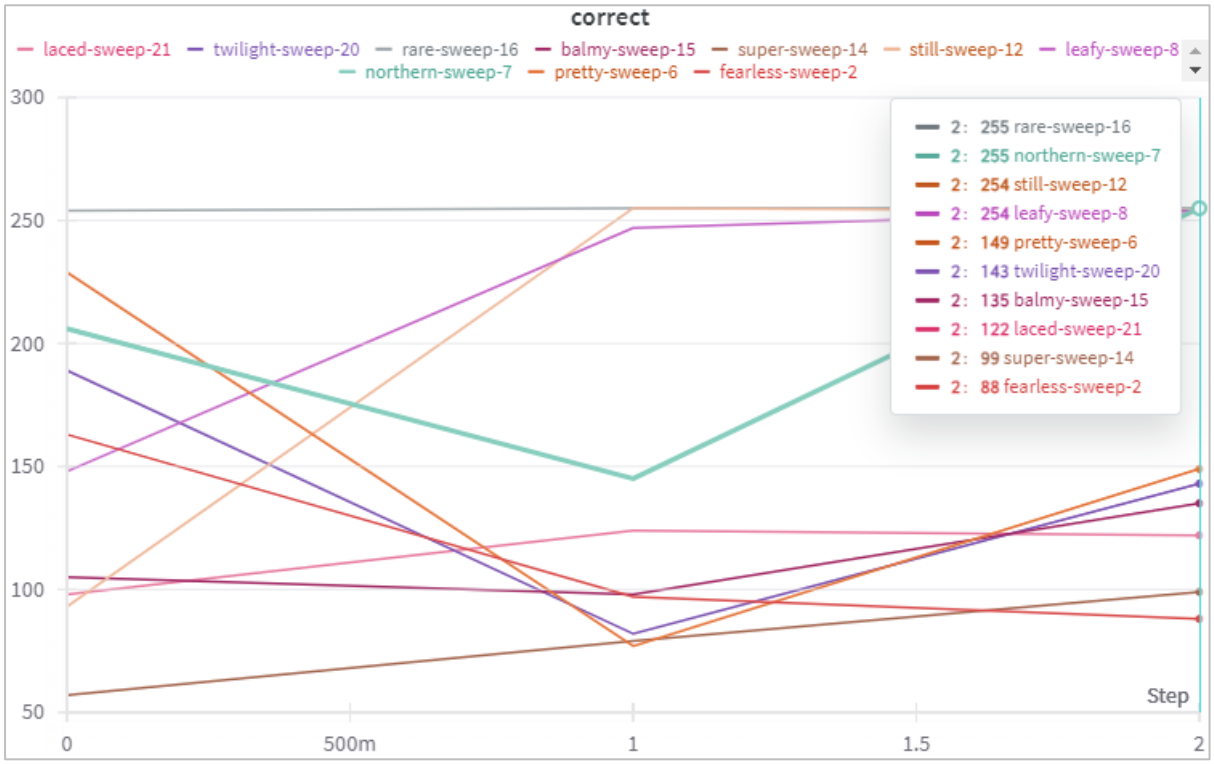

Fig. 3 Correct answers for 10 top explorations

From these top 10 explorations, I chose the two extreme explorations: one with the lowest

training loss and the one with the highest number of correct answers. The hyperparameter values for these two explorations are shown below:

Table 1 The hyperparameter values for two extreme explorations

Exploration	Training batch size	Learning rate	Training loss	Evaluation loss	Correct answers
laced-sweep-21	64	2.045e-4	1.124	-6.233	122
rare-sweep-16	128	3.9598e-4	2.828	-3.868	255

In the following section, these two pairs of hyperparameter values and a random pair will be used to fine-tune the pre-trained RoBERTa model for the QA task.

2. Fine-tune the pre-trained model

In this section, we will show the results of fine-tuning different pre-trained models for the QA task using different hyperparameter values. In total 4 experiments were conducted as shown below:

Table 2 Fine-tuning experiments

Name	Pre-trained model	Training batch size	Learning rate
Roberta	roberta-base	128	3.9598e-4
Roberta 1	roberta-base	64	2.045e-4
Roberta 2	roberta-base	128	4e-5
Bert	bert-base-case	128	4e-5

Fig. 4 shows changes in the EM score of each model after each evaluation during the training process:

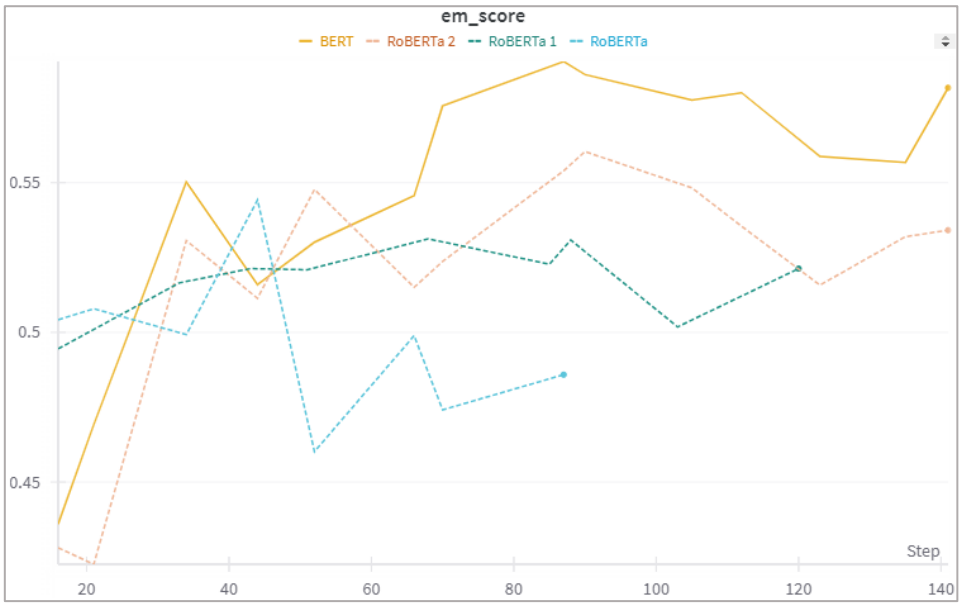

Fig. 4 EM score for each model

As we can see from Fig. 4, the RoBERTa models reach a higher EM score in the beginning phase of the training process, while in a later phase, the BERT model is the one with the highest EM score. In the following table, we summarize the results of all fine-tuning processes:

Table 3 Summarization of the fine-tuning processes

Name	Pre-trained model	Training batch size	Learning rate	Training em_score	Testing em_score
Roberta	roberta-base	128	3.9598e-4	0.5442	0.5451
Roberta 1	roberta-base	64	2.045e-4	0.5312	0.5460
Roberta 2	roberta-base	128	4e-5	0.5604	0.5707
Bert	**bert-base-case**	**128**	**4e-5**	**0.5905**	**0.5805**

Based on the results we got, we will stay with the BERT model to evaluate the effect of fine-tuning to a pre-trained LLM model.

3. Evaluate the effect of fine-tuning to the pre-trained model

To evaluate the effect of fine-tuning on the BERT pre-trained model, we will test the fine-tuned model and non-fine-tuned model on the same dataset and will compare the results of these tests. Moreover, we will try to answer questions with both models and will compare the responses of each model.

3.1 Test the models

Both the fine-tuned BERT model and non-fine-tuned BERT model were tested using a testing dataset from the following link: https://rajpurkar.github.io/SQuAD-explorer/. The testing dataset contains 6425 data samples. The testing results are as shown below:

Table 4 Testing results

Model	Correct answers	Similar answers	Incorrect answers	**EM score**	Modified EM score
BERT (non-fine-tuned)	0	3957	2468	**0.0**	0.1245
BERT (fine-tuned)	3730	2504	191	**0.5805**	0.6696

By looking at the EM score of each model in Table 4, we can easily observe the effect of fine-tuning. Before fine-tuning, even though the model has obtained a general context-aware word embedding and understands the connection between words and sentences, still it is not enough

for the model to perform the QA task. The model is so general that it cannot even predict a single correct answer. This outcome shows that this pre-trained LLM should be trained to be specialized for a specific task. Without fine tuning the model understands words but does not have the logic to 'play' with words.

3.2 Answer questions with the models

In this section, the models are used to answer different questions from different domains. The model takes as the input the question and the context from where to extract the answer and has to provide, at maximum, three possible answers. For each answer, the prediction probability or the confidence score of the model is provided as well. The first answer is the answer for which the model is most confident. The asked questions and their corresponding answers are shown below:

ID	Context		Vin is a Mistborn of great power and skill.
0	Question		What is Vin's speciality?
	BERT (fine-tuned)	Best answer	'Mistborn of great power and skill'
		Confidence score	0.4951
	BERT (non-tuned)	**Best answer**	**'empty'**
		Confidence score	**0.5997**

ID	Context		Bidirectional Encoder Representations from Transformers (BERT) is a family of language models introduced in 2018 by researchers at Google.
1	Question		When was BERT introduced?
	BERT (fine-tuned)	Best answer	'2018'
		Confidence score	0.9705
	BERT (non-tuned)	**Best answer**	**'empty'**
		Confidence score	**0.6483**

ID	Context		Earth is the third planet from the Sun and the only place known in the universe where life has originated and found habitability.
2	Question		What is Earth?
	BERT (fine-tuned)	Best answer	'the third planet from the Sun'
		Confidence score	0.6599
	BERT (non-tuned)	**Best answer**	**'empty'**
		Confidence score	**0.6585**

ID	Context		Earth is the third planet from the Sun and the only place known in the universe where life has originated and found habitability.
3	Question		Where has life originated?
	BERT (fine-tuned)	Best answer	'Earth'
		Confidence score	0.9090
	BERT (non-tuned)	**Best answer**	**'empty'**
		Confidence score	**0.6371**

From the answers of the non-tuned model, we can easily conclude that it is unpractical at all. Every time it predicts an 'empty' answer with a confidence score of 0.6. These results confirm one more time the need for fine-tuning the pre-trained LLM. Once we performed fine-tuning, we got much better results with the model able to predict correct answers with high confidence. This new behavior of the model best shows the effect of fine-tuning the pre-trained LLM. Before fine-tuning the model cannot answer questions, after fine-tuning it can answer questions.

4. Discussion and Conclusion

In this work, the project of building a Question Answering model was continued. Firstly, hyperparameter optimization was performed and then the best hyperparameter values were used to fine-tune the pre-trained model. RoBERTa and BERT models from Hugging Face were used, with the BERT model achieving the highest EM score of 0.5905. To observe the effect of fine-tuning on LLM, the non-fine-tuned BERT model, and the fine-tuned BERT model were tested on the same testing dataset and were used to answer questions. The achieved EM score of the non-fine-tuned model was 0.0 and it was unable to predict any meaningful answer. When the same tests were conducted on the fine-tuned model, the achieved EM score was 0.5805, and it was able to answer the questions, with high confidence. This big difference in the behavior of the two models shows how strong is the effect of fine-tuning and how important it is in specializing a general pre-trained LLM model for a specific task.