Final Project Report

Lingxiao Zhao

April 22, 2025

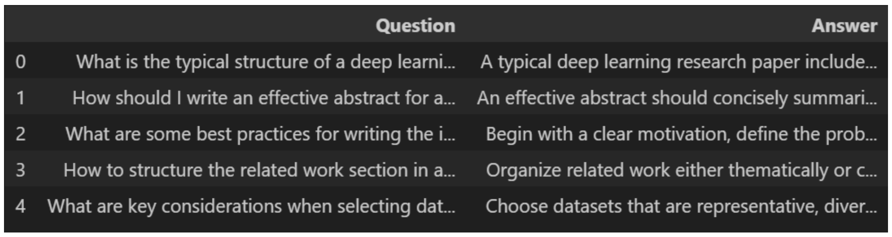
1 Dataset construction

As stated in the requirements, the core goal of this project is to train an open-source large language model that can automatically generate high-quality research methods and strategies for a given research question. To achieve this goal, I used LLM to generate a hundred questions covering computer research fields such as machine learning and artificial intelligence. The dataset is organized in the form of question-answer pairs (QA). These question-answer pairs are automatically generated by a pre-trained large language model, covering one hundred specific questions and corresponding answers in different research fields. After screening, I left the one hundred most representative questions to form the dataset.

The specific process of generating the dataset is as follows:

- Use clearly designed prompts to let large language models generate research-oriented questions and corresponding answers.
- The generated question-answer pairs are manually reviewed to ensure that the generated questions are clear and well-defined, and the answers are accurate and detailed.
- The selected question-answer pairs are all manually selected to ensure that they can effectively promote the model's ability to generate research methods and strategies when training the model.

In general, using LLM to generate data sets covers all issues and has a clear data structure, which is conducive to efficient model learning. Especially for the latter, the clear data format is very convenient for subsequent operations, and the clear field division makes the subsequent data preprocessing, feature extraction, and model training and fine-tuning much clearer. This unified structure reduces the extra burden in data processing.



	Question	Answer
0	What is the typical structure of a deep learni...	A typical deep learning research paper include...
1	How should I write an effective abstract for a...	An effective abstract should concisely summari...
2	What are some best practices for writing the i...	Begin with a clear motivation, define the prob...
3	How to structure the related work section in a...	Organize related work either thematically or c...
4	What are key considerations when selecting dat...	Choose datasets that are representative, diver...

Figure 1: Some selected questions

2 LLM selection (llama-2-7b)

After comprehensive consideration, the basic model selected for this project is LLaMA2-7B. LLaMA2 is a powerful open source large language model developed by Meta. As an open source model, LLaMA2 can be freely modified, fine-tuned and extended by researchers, making it easier to adapt to various research and practical application scenarios.

The 7B (7 billion) parameter selection scale is moderate, which can provide high performance while being relatively economical in computing resource requirements, suitable for general research experiments and rapid iterative development. In fact, in the fine tuning of colab in this project, the GPU RAM was close to the upper limit of colab pro. Using this model is a comprehensive choice in terms of performance and resources.

3 Generative fine-tuning method based on parameter adjustment

3.1 Training details

The first adjustment of this project uses a simplified interface to call the LLaMA2-7B model through the locally deployed Ollama API to adjust the parameters. The default parameters are temperature=0, top_p=0. The temperature is initially set to 0 to control the randomness of the generated results. The top_p is initially set to 0 to limit the probability range of the model's generated output.

The specific training steps are as follows:

The specific training steps are as follows:

- 96 questions were read from the `cs_deep_learning_qa_detailed.csv` file and used as test and evaluation data for the model.
- The lightweight model `bert-base-uncased` was used, and the F1 index of BERT score was calculated and printed to evaluate the effect.
- Different combinations of `top_p` and `temperature` parameters were used for experiments, and the effect was evaluated by BERT score.
- Other parameters such as `max_tokens` were also adjusted in order to find better results.

3.2 Evaluation metric and experiments

First, for evaluation, let's look at the performance of the model with the default parameters for preliminary experiments:

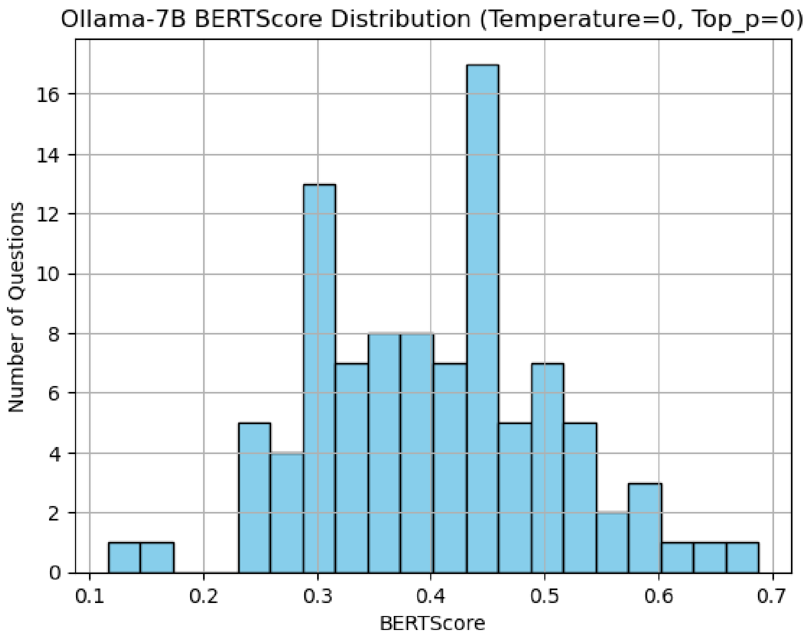


Figure 2: Initial parameter performance

We can see that the BERT scores are mainly concentrated around 0.3 and 0.45, which is not an ideal result. Next, we first tune different parameters of top_p and analyze the results.

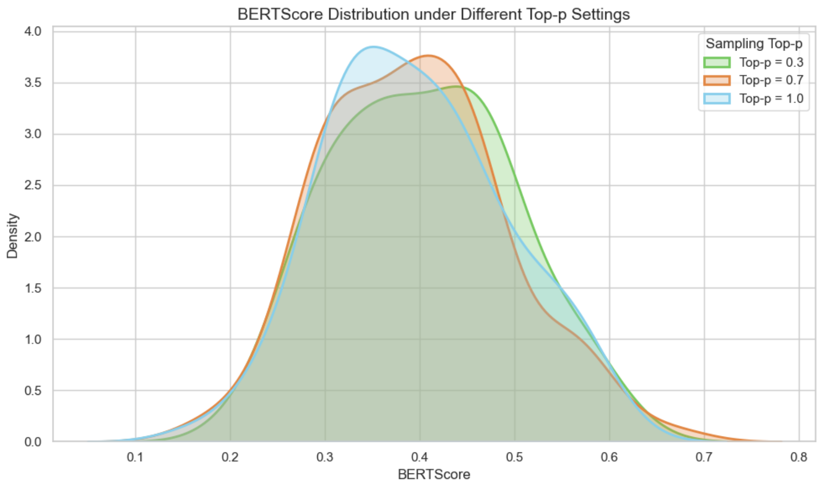


Figure 3: top_p parameter tuning

We can see that the higher the top_p, the more concentrated the data, but the accuracy is not as good as the case of top_p=0. This shows that the larger the top_p, the more the model tends to output more common results, but the F1 score will decrease. However, when we relax the restrictions on top_p, the F1 score will increase.

Then we consider tuning the temperature. The temperature parameter is an important parameter in the language model generation process, which controls the randomness and diversity of the generated text. The text generated by a low temperature value tends to choose the words with the highest probability, which is more in line with common patterns and logic. We also analyze different temperature indexes and finally calculate their BERT scores.

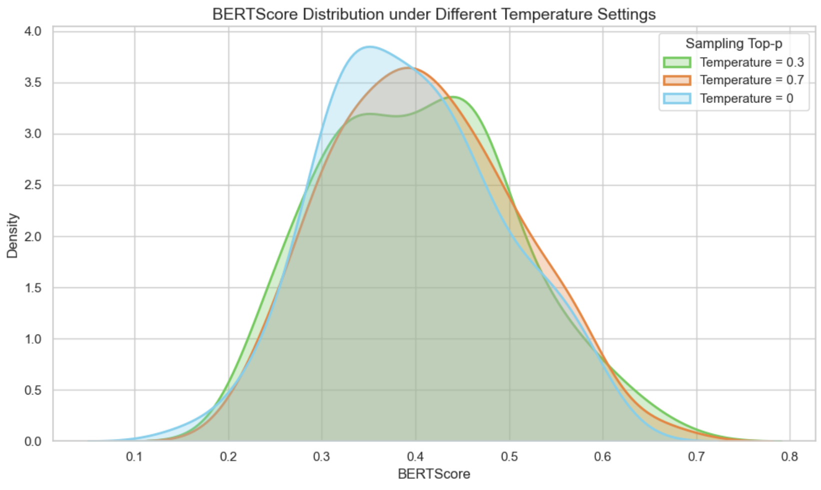


Figure 4: temperaturer parameter tuning

In general, in the process of tuning the parameters of temperature and top_p, we found that top_p and temperature are a pair of parameters with similar action mechanisms. When the restrictions on their parameters are relaxed, they both have the hope of achieving a higher BERT score. Next, we will explore the best parameter combination between top_p and temperature.

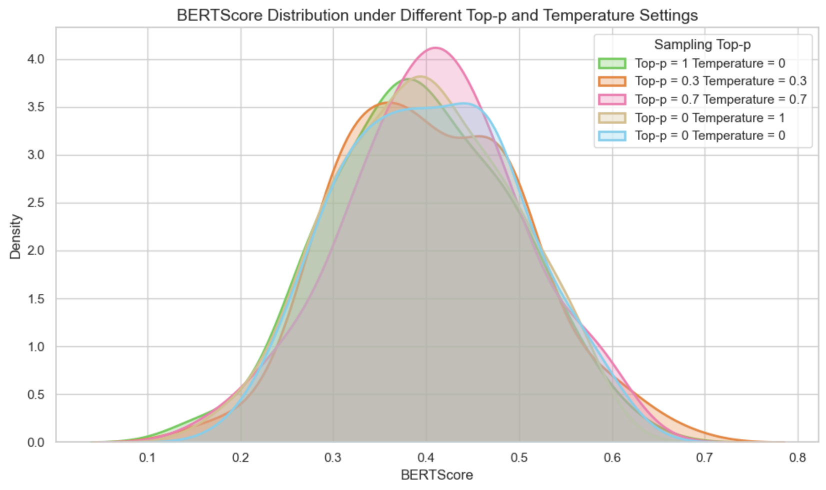


Figure 5: top_p and temperaturer parameter tuning

Overall, top_p=0.7 and temperature=0.7 seem to be the best among the five parameter settings in terms of score concentration and prediction results, but there was no significant improvement. We found better parameters than the current default settings in the above tuning. After confirming top_p=0.7 and temperature=0.7, we tuned other parameters. We first focus on adjusting max_tokens.

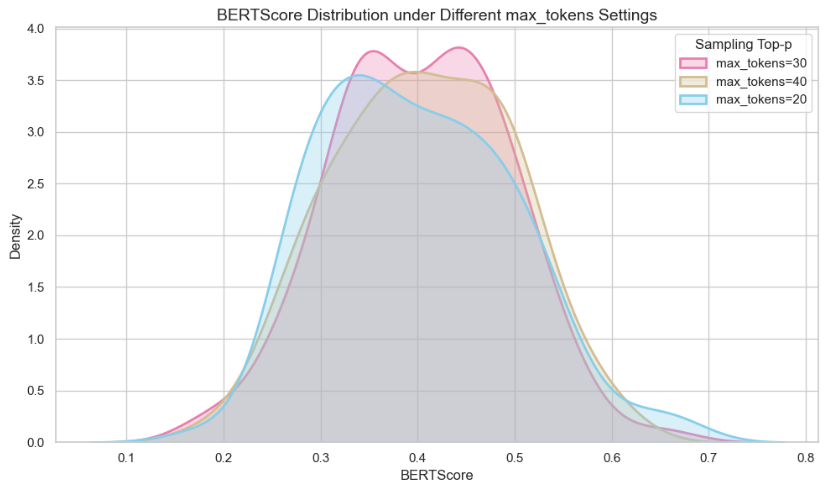


Figure 6: max_tokens parameter tuning

max_tokens is not necessarily better the longer it is. A length of 30 is concise and accurate enough for the answer. On the contrary, increasing max_tokens to 40 makes the answer vague and the overall performance decreases. Now we look at the parameter adjustment of the penalty term. In the previous experiment, we fixed the penalty term to 1.1. Now we redefine the function in order to change the parameter of the penalty term.

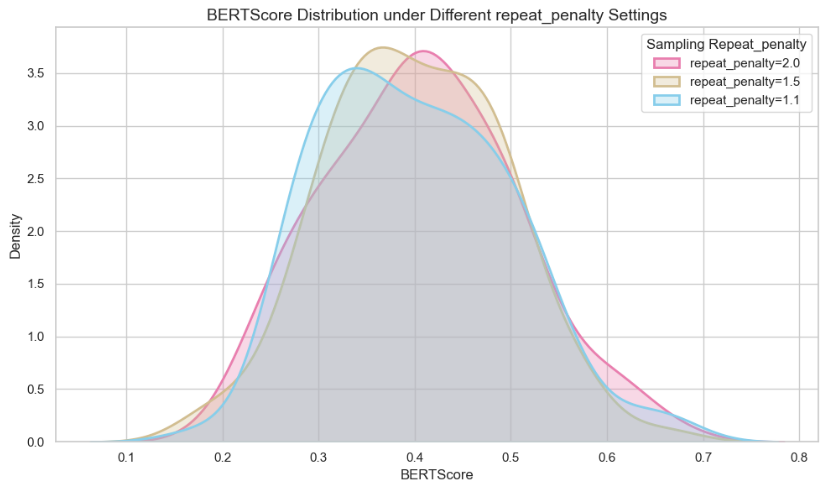


Figure 7: repeat_penalty parameter tuning

The increase of penalty terms can suppress the tendency of the model to repeatedly generate the same word or phrase, allowing the model to improve the diversity and creativity of the output from multiple angles, and comprehensively show a better BERT score.

Overall, the basic parameter adjustment model has reached its upper limit and there is no obvious improvement. Therefore, we consider using LoRA and QLoRA for further comprehensive tuning.

4 Generative fine-tuning method based on LoRA and QLoRA

4.1 Training details

The core idea of LoRA (Low-Rank Adaptation) is to freeze the weights of the original pre-trained model and only add trainable low-rank matrices to some layers (such as Query and Value of the Attention layer) for learning. QLoRA (Quantized LoRA) is a further optimization of LoRA, which quantizes the main weights of the large model and combines the low-rank fine-tuning method of LoRA to achieve efficient fine-tuning with extremely low resource overhead.

In terms of the dataset, we use exactly the same dataset for training. Since we use the load_dataset method in transformer, the acceptable training sample parameters must be in json format. Therefore, we divide the training set and test set into a ratio of 0.8 and 0.2.

Next, we start to find the optimal parameters. We use optuna to simulate grid search to find the optimal parameter values. In this step, we set 10 attempts to find the optimal parameters and the number of training rounds is 30. After that, the optimal parameters found are saved and read as model input to train LlaMa2-7b. From the training loss value print and loss curve, the model has converged around step=300 (epoch=30), so the number of training rounds with epoch=30 is sufficient.

As before, our final evaluation standard is still Bert score, and we also use F1 score for evaluation. F1 score is an indicator that combines the two dimensions of Precision and Recall, and has a strong symbolic effect.

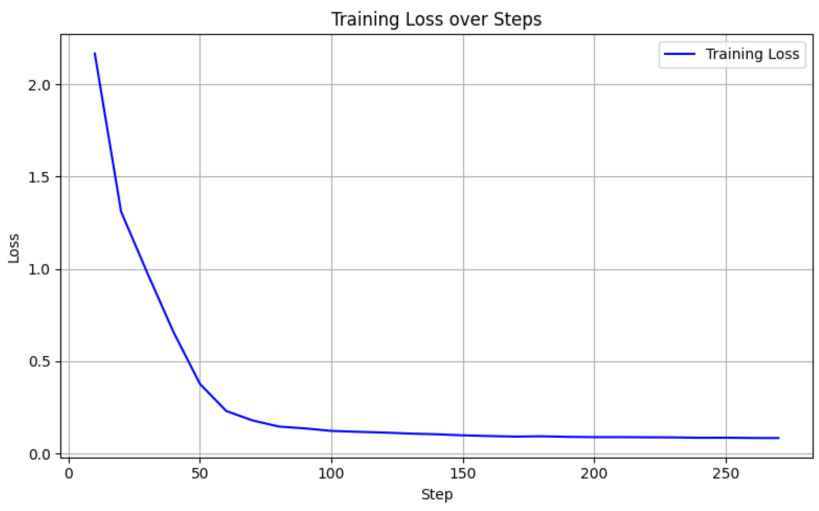


Figure 8: Training loss

4.2 Evaluation metric and experiments

After the previous step, we export the trained model, test the generated text in the test set and score it, and use the BERT score for evaluation. Finally, we plot the score into a frequency histogram and visualize it.

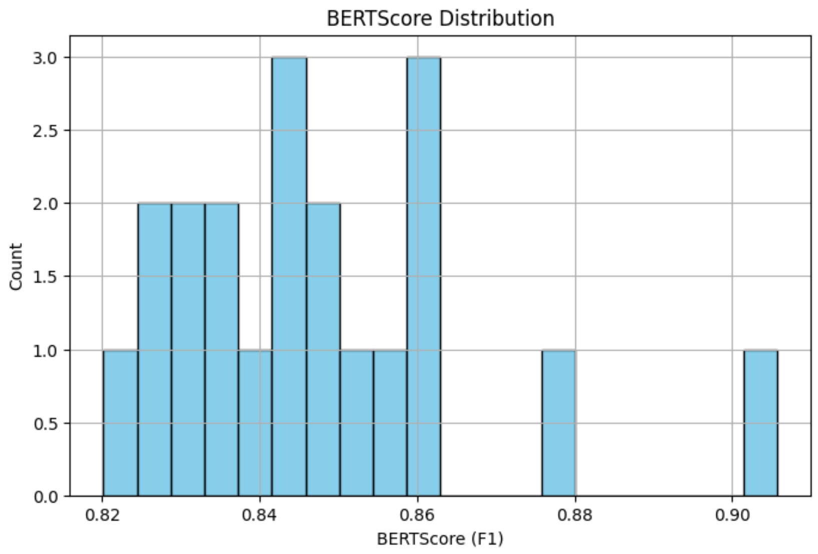


Figure 9: LoRA + QLoRA tuning

We can see that after a short training, the model's performance on the test set is much better than simple parameter tuning, concentrated between 0.82-0.86, which is much better than parameter tuning concentrated between 0.4-0.5.

5 Learned and lessons

For me, my biggest gain is that I have practiced the LLM optimization process from simple parameter adjustment to advanced fine-tuning methods. I have a basic understanding of the fine-tuning methods and their practical application scenarios in the fine tuning process.

I was surprised that simple parameter adjustment did not have much effect and that the subsequent QLoRA method could achieve significant improvement, because based on my previous project's experience in parameter adjustment on GPT-3.5, top_p and temperature should be enough to significantly improve the final generated content. Considering that top_p and temperature are decoding parameters when the model is generated, they can only adjust the diversity of the output results, which is a big disadvantage under the unified BERT score standard.

Objectively speaking, the BERT score is not sensitive enough to parameter fine-tuning (top_p, temperature), and the simple semantic similarity DUIBI is difficult to keenly capture the small but important differences in content production after parameter changes. In this project, professional human raters may be introduced to capture the quality differences of responses more fairly.

Another problem encountered is more specific in model training. After switching to the API on Hugging Face, I deployed the code locally as before, but there was an out of memory problem locally. I switched to colab to continue training. After opening Colab Pro, the training process almost filled up the entire 22.5GB of GPU RAM. LLaMA2-7B has about 7 billion parameters, and just loading these parameters is a big challenge for the GPU resources allocated by Colab, and once it enters the training phase, the memory requirements will be further expanded. During the training process, I was worried that the code would not continue to run even after opening the colab pro.

Although QLoRA is a parameter-saving training method, it is still not enough in this project. I need to consider using more efficient parameter representations to reduce redundancy in parameters, or consider model pruning to reduce the number of intermediate activations.