ADL Homework III

Wei-Hsiang Huang
B09202011

2023 年 11 月 29 日

1 LLM Tunning

1.1 Describe

- First 3000 training data is used for fine-tuning.
- I tuned my model by first setting the prompts of the input. Specifically,

$$inputs = get\_prompt(instructions) + answers$$

$$outputs = mask(get\_prompt(instructions)) + answers$$

  Then, I tokenized them. After that, I set a LoRA framework together with the model. Then, give the trainer "inputs" and ask them to predict "output". While doing the back-propogation, the only weights can be adjusted are the ones inside the adapter.

- Hyperparameters:
  - Learning Rate: 3×10^{-5}.
  - Batch Size (per_device_train_batch_size x gradient_accumulation_steps): 2×8.
  - Learning Rate Scheduler: Linear.
  - LoRA alpha: 16.
  - LoRA dropout: 0.1.
  - LoRA r: 64.
  - Generation beams: 1 or 3 or 5.
  - Generation top_p: 0.9.

- Generation top_k: 10.
- Generation temperature: 1.2

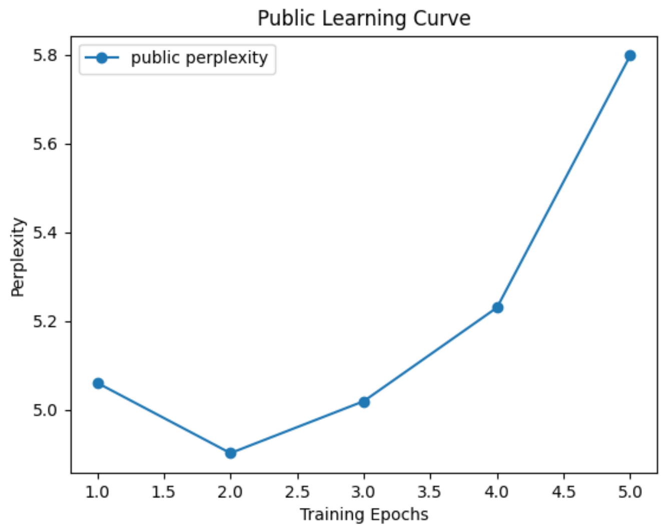
For the epochs, I trained in 1+1+3 iterations. (i.e: I trained it three times. Each time, I start from the checkpoint before.)
I attempted two generation ways to my final model. One is beam=3 and the other is beam=5.
Although using beam=5 seems to predict better, it consumed almost 8GB VRAM, making it a high chance to exceed the computing usages. Therefore, I decided to use beam=3 as my final generation strategy.

1.2 My performance

Here is the performance of my model

Epochs	1	1+1	1+1+3
Public Perplexity	5.060	4.902	5.739

For my human evaluation, I think my prediction for the last epochs is good. The awful perplexity is due to some predictions which contain a very high public perplexity. Therefore, despite for the worst perplexity, I chose to choose the one in the end to try to get the point of human evaluation.
Here is the learning curve on public dataset. where I trained the model 1+1+1+1+1 epochs. It seems that it is easy to overfit for the public dataset.



Public Learning Curve, with beam=1

2 LLM Inference Strategies

2.1 Zero-Shot

For doing the zero-shot, I used the same prompt as the default prompting. And then directly put it into the machine. For the generation strategy, I used the same parameters in Section 1, with beam=1. The resulting performance is very bad. The model will sometimes answer the qeustion in English. Here is the performance when I used ppl to estimate it:

Method	Perplexity
Best Qlora	4.9
Zero Shot	5.5

2.2 Few Shots

For doing the few-shots, I applied 3-shots and used the same prompt as the default prompting, with some modification. Specifically, my inputs are:

$$\begin{aligned} \text{inputs} = \; & \text{default prompt} + \text{以下是範例:} \\ + \; & \text{1. USER: Instruction1 ASSITANT: Answer1} \\ + \; & \text{2. USER: Instruction2 ASSITANT: Answer2} \\ + \; & \text{3. USER: Instruction3 ASSITANT: Answer3} \\ + \; & \text{請作答: USER: Instruction ASSITANT:} \end{aligned}$$

I selected the three template instructions and answers by using the last three data in the training json file. For the generation strategy, I used the same parameters in Section 1, with beam=1. I found some predictions that even generating the fourth question by themselves. (For example, 4: 請將下列語言翻成英文: USER: ......, in fact, Cool!!) But overall, by my human evaluation, the prediction is not bad. And for ppl estimation, the result is as followed:

Method	Perplexity
Best Qlora	4.9
Three Shots	4.8

2.3 Comparison

In the aspect of the tuning, QLoRA resembles the traditional fine-tuning with just a little extra weight that can be fine-tuned. On the other hand, for zero-shot and few-shots, it fixed the weight in the model and tried to get more powerful answers through the prompt. And in the aspect of the performance, from my human evaluation, QLoRA has a better performance than zero-shot and 3-shots. And the performance is very bad for zero-shot. However, with just three prompts, the performance increases a lot, even making the ppl performance greater than the best QLoRA.