[ADL 2023 Fall]
HW3 Report

Student ID: R12945039   Name: 楊瀚博

Q1: LLM Tuning

■ Describe:

How much training data did you use?

In this homework, we used batch size of 32 and trained the model for 200 steps, which means we used **32 * 200** = **6400** samples in the training dataset to apply instruction-tuning on the model.

How did you tune your model?

We refer to the QLoRA fine-tuning code on GitHub "artidoro/qlora/qlora.py", then make some adjustment to make it in line with our task of applying instruction-tuning on the Taiwan-LLaMA-v2-chat model using Chinese datasets.

We first **load** the **pre-trained** Taiwan-LLaMA-v2-chat model weights. Secondly, we **add** the **QLoRA architecture** to the original model using **PEFT** library from Hugging Face (get_peft_model). Finally, after **preprocessing** the **instructions** and corresponding **output** context (including context concatenation and mask adjustment), we could start to apply instruction-tuning on the adapters of this model.

```python
            print(f"adding LoRA modules...")
            modules = find_all_linear_names(args, model)
            config = LoraConfig(
                r=args.lora_r,
                lora_alpha=args.lora_alpha,
                target_modules=modules,
                lora_dropout=args.lora_dropout,
                bias="none",
                task_type="CAUSAL_LM",
            )
            model = get_peft_model(model, config)
```

```python
        elif dataset_format == "Taiwan_LLM_chat":
            print("Formatting Taiwan_LLM_chat dataset")
            dataset = dataset.map(lambda x: {"input": get_prompt(x["instruction"]), "output": x["output"]})
```

What hyper-parameters did you use?

Optimization Algorithm	*paged_adamw_32bit*
Learning Rate	*2e-4*
lr_scheduler_type	*linear*
Warmup Ratio	*0.03*
Batch Size per Device	*2*
Gradient Accumulation Steps	*16*
Effective Batch Size	*32*
Steps	*200*
LoRA Rank	*4*
LoRA Alpha	*16*
LoRA Dropout	*0.05*
num_beams	*3*
do_sample	*False*
load_in_4bit	*True*
bnb_4bit_compute_dtype	*torch.bfloat16*
bnb_4bit_use_double_quant	*True*
bnb_4bit_quant_type	*nf4*

Show your performance

Final performance of the model on the public testing set

The model achieved a mean perplexity score ***3.4555*** on the public testing set. Some generation results are shown below:


```
{
  "id": "07f75449-94b9-4c3b-a525-e62cdbf85382",
  "instruction": "文言文翻譯：\n靈鑒忽臨，忻歡交集，乃迴燈拂席以延之。",
  "output": "答案：靈仙忽然光臨，趙旭歡欣交集，於是他就把燈點亮，拂拭乾淨床席來延請仙女。"
},
{
  "id": "7b7ead70-1353-433f-a59f-7704594cce59",
  "instruction": "希望您以後留意，不要再齣這樣的事，你的小女兒病就會好。\n這句話在古代怎麼說：",
  "output": "以後幸長官留意，勿令如此。"
},
{
  "id": "b8adf597-edb9-46d4-a1a6-074ce9724f07",
  "instruction": "第二年召迴朝廷，改任著作佐郎，直史館，改任左拾遺。\n翻譯成文言文：",
  "output": "明年召還，改著作佐郎，直史館，改左拾遺。"
},
```


Fig.1 Instructions and Labeled Outputs in public testing set

```
{
  "id": "07f75449-94b9-4c3b-a525-e62cdbf85382",
  "output": "答案：靈鑒忽然臨席，忻歡熱絡地交談，於是靈鑒迴燈拂席，暗示延後纔會。"
},
{
  "id": "7b7ead70-1353-433f-a59f-7704594cce59",
  "output": "惟欲留，不復復，妹病當瘉。"
},
{
  "id": "b8adf597-edb9-46d4-a1a6-074ce9724f07",
  "output": "二年復召，累改著作佐郎，直史館，改左拾遺。"
},
```


Fig.2 ***Corresponding Generated Outputs from the Tuned Model***

Learning curve on the public testing set

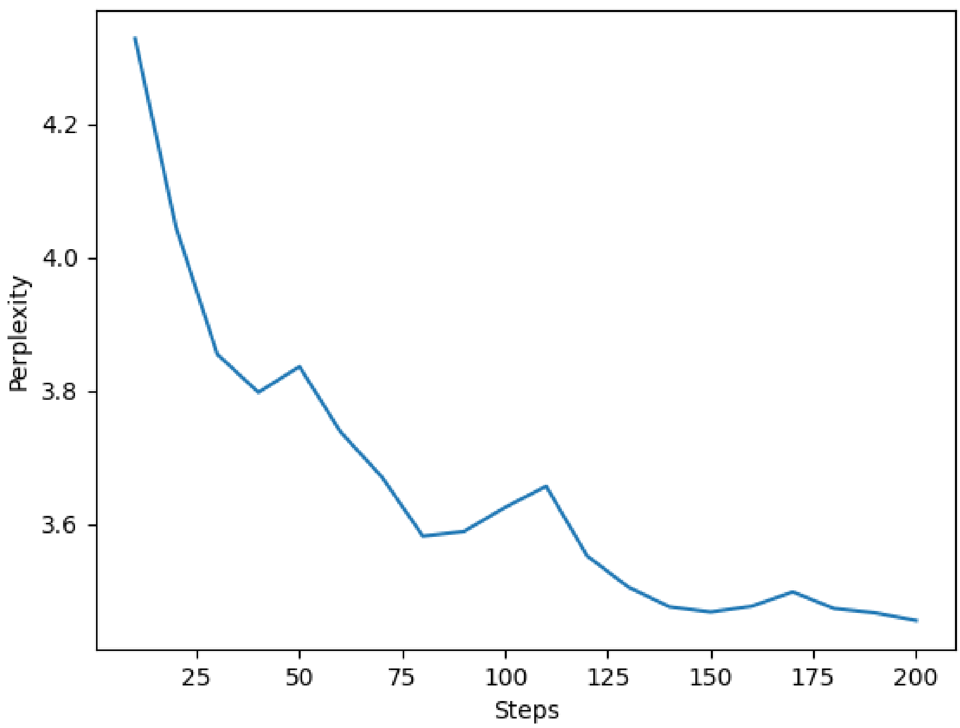

Fig.3 ***Learning Curve on the Public Testing Set***

*(Vertical Axis: **Perplexity Score**, Horizontal Axis: **Training Steps**)*

Q2: LLM Inference Strategies

■ Zero-Shot

What is your setting? How did you design your prompt?

In zero-shot inference, we use **greedy algorithm** to generate the output. The prompt we used in zero-shot inference is:

```python
prompt = (
    f"你是專業古文學者，以下是用戶和專業古文學者之間的對話。你要對用戶的問題提供精確、安全、詳細的回答。"
    f"用戶: {instruction} 專業古文學者:"
)
```

Fig.4 Zero-Shot Prompt

which has only slightly differences compared to the original prompt provided by TAs. This prompt is designed by utilizing the domain knowledge we already knew: the task is mainly about translating sentences between “文言文” and “現代文”.

Several samples from the generation results are shown below:

```json
{
  "id": "07f75449-94b9-4c3b-a525-e62cdbf85382",
  "instruction": "文言文翻譯：\n靈鑒忽臨，忻歡交集，乃迴燈拂席以延之。",
  "output": "答案：靈仙忽然光臨，趙旭歡欣交集，於是他就把燈點亮，拂拭乾淨床席來延請仙女。"
},
{
  "id": "7b7ead70-1353-433f-a59f-7704594cce59",
  "instruction": "希望您以後留意，不要再齣這樣的事，你的小女兒病就會好。\n這句話在古代怎麼說：",
  "output": "以後幸長官留意，勿令如此。"
},
{
  "id": "b8adf597-edb9-46d4-a1a6-074ce9724f07",
  "instruction": "第二年召迴朝廷，改任著作佐郎，直史館，改任左拾遺。\n翻譯成文言文：",
  "output": "明年召還，改著作佐郎，直史館，改左拾遺。"
},
```

Fig.5 Instructions and Labeled Outputs in Public Testing Set

```json
{
  "id": "07f75449-94b9-4c3b-a525-e62cdbf85382",
  "output": " 靈鑒忽臨，忻歡交集，乃迴燈拂席以延之。"
},
{
  "id": "7b7ead70-1353-433f-a59f-7704594cce59",
  "output": " 你的小女兒病就會好。"
},
{
  "id": "b8adf597-edb9-46d4-a1a6-074ce9724f07",
  "output": " 第二年召回朝廷，改任著作佐郎，直史館，改任左拾遺。"
},
```

Fig.6 Corresponding Generated Outputs using Zero-Shot Prompt

■ Few-Shot (In-context Learning)

What is your setting? How did you design your prompt?

In few-shot inference, we also use **greedy algorithm** to generate the output. The prompt we used in few-shot inference is shown below:

```
prompt = (
    f"你是專業古文學者，以下是用戶和專業古文學者之間的對話。你要對用戶的問題提供精確、安全、詳細的回答。 "
    f"用戶: 幫我進行以下句子的文言文翻譯：契丹主以陽城之戰為彥卿所敗，詰之。彥卿曰： 臣當時惟知為晉主竭力，今日死生惟命。 "
    f"專業古文學者:契丹主因陽城之戰被符彥卿打敗，追問符彥卿，彥卿說： 臣當時隻知為晉主竭盡全力，今日死生聽你決定。 "
    f"用戶: 將下面句子翻譯成文言文：等脩行師到達，腹背攻擊他，脩行師大敗，因而乞求投降，陸子隆同意他投降，將他送於京師。 "
    f"專業古文學者:及行師至，腹背擊之，行師大敗，因乞降，子隆許之，送於京師。 "
    f"用戶: 臧霸自亡匿，操募索得之，使霸招吳敦、尹禮、孔觀等，皆詣操降。幫我把這句話翻譯成現代文 "
    f"專業古文學者:臧霸自己逃到民間隱藏起來，曹操懸賞將他捉拿，派他去招降吳敦、尹禮、孫觀等，這些人全都到曹操營中歸降。 "
    f"用戶: 因為郊祀禮重大，不宜由臣子代行，請求等到聖上身體康復，改換占蔔吉日行禮。這句話在中國古代怎麼說 "
    f"專業古文學者:蓋以郊祀禮重，不宜攝以人臣，請俟聖躬痊，改蔔吉日行禮。 "
    f"用戶: 翻譯成文言文：於是，廢帝讓瀋慶之的堂侄、直將軍瀋攸之賜瀋慶之毒藥，命瀋慶之自殺。 "
    f"專業古文學者:帝乃使慶之從父兄子直閣將軍攸之賜慶之藥。 "
    f"用戶: {instruction} 專業古文學者:"
)
```

Fig.7 Few-Shot Prompt

which has only slightly differences compared to the original prompt provided by TAs and few examples from the public test set are given. This prompt is designed by utilizing the domain knowledge we already knew: the task is mainly about translating sentences between “文言文” and “現代文”.

Several samples from the generation results are shown below:

```
{
  "id": "07f75449-94b9-4c3b-a525-e62cdbf85382",
  "instruction": "文言文翻譯：\n靈鑒忽臨，忻歡交集，乃迴燈拂席以延之。",
  "output": "答案：靈仙忽然光臨，趙旭歡欣交集，於是他就把燈點亮，拂拭乾淨床席來延請仙女。"
},
{
  "id": "7b7ead70-1353-433f-a59f-7704594cce59",
  "instruction": "希望您以後留意，不要再齣這樣的事，你的小女兒病就會好。\n這句話在古代怎麼說：",
  "output": "以後幸長官留意，勿令如此。"
},
{
  "id": "b8adf597-edb9-46d4-a1a6-074ce9724f07",
  "instruction": "第二年召迴朝廷，改任著作佐郎，直史館，改任左拾遺。\n翻譯成文言文：",
  "output": "明年召還，改著作佐郎，直史館，改左拾遺。"
},
```

Fig.8 Instructions and Labeled Outputs in Public Testing Set

```json
{
  "id": "07f75449-94b9-4c3b-a525-e62cdbf85382",
  "output": "靈鑒忽臨，忻歡交集，乃迴燈拂席以延之。"
},
{
  "id": "7b7ead70-1353-433f-a59f-7704594cce59",
  "output": "希望你以後留意，不要再齣這樣的事，你的小女兒病就會好。"
},
{
  "id": "b8adf597-edb9-46d4-a1a6-074ce9724f07",
  "output": "宣帝第二年召迴朝廷，改任著作佐郎，直史館，改任左拾遺。"
},
```

Fig.9 Corresponding Generated Outputs using Few-Shot Prompt

How many in-context examples are utilized? How you select them?

There are **5 in-context examples** are utilized in our few-shot inference. We randomly choose several samples with different length and user instruction context type, trying to make the model more generalized on the test set.

Comparison

What's the difference between the results of zero-shot, few-shot, and LoRA?

One sample results of zero-shot, few-shot, and LoRA are shown below:

```json
"id": "b8adf597-edb9-46d4-a1a6-074ce9724f07",
"instruction": "第二年召迴朝廷，改任著作佐郎，直史館，改任左拾遺。\n翻譯成文言文：",
"output": "明年召還，改著作佐郎，直史館，改左拾遺。"
```

Fig.10 Instructions and Labeled Outputs in Public Testing Set

```json
"output": " 第二年召回朝廷，改任著作佐郎，直史館，改任左拾遺。"
```

Fig.11 Corresponding Generated Outputs using Zero-Shot Prompt

```json
"output": "宣帝第二年召迴朝廷，改任著作佐郎，直史館，改任左拾遺。"
```

Fig.12 Corresponding Generated Outputs using Few-Shot Prompt

```json
"output": "二年復召，累改著作佐郎，直史館，改左拾遺。"
```

Fig.13 Corresponding Generated Outputs from the LoRA Model

According to the results shown above, it's obvious that the **LoRA** model has the **best performance**, generating a **reasonable output**. On the other hand, the generated output using **zero-shot** prompt actually just **copy** the **input context** to the output. Similarly, the generated output using few-shot prompt encounter the **same issue** with a **slightly difference** from the input instruction.

In summary, instruction-tuning using **LoRA** is a very **efficient** way to achieve **significantly better** model performance. It requires much lower computation resource compared to fine-tuning the whole LLM weights while reaching lots better performance.