ADL HW3 Report

姓名：陳竹欣

系級：資工三

學號：B10902005

Q1: LLM Tuning

Describe

How much training data did you use?

5000筆作為train data，剩下5000筆作為validation data。

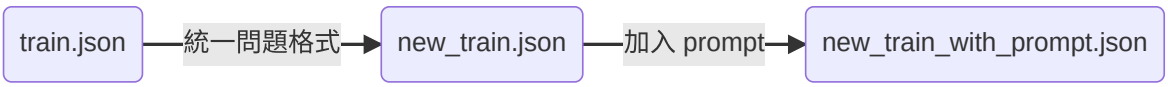
How did you tune your model?

我使用助教課提到的`axoloti` [1] fine-tune 我的 model。

在tune model之前，我有先更改data的問題格式。只要是文言文翻白話文的問題，統一格式為「把這句話翻譯成白話文：{text}」，只要是白話文翻文言文的問題，統一格式為「把這句話翻譯成文言文：{text}」。

接著，再將`instruction`改為`get_prompt(instruction)`，然後設定好`training_args`，就可以進行得到fine tune過後的model。

以下是我的訓練資料的演進圖：



以下是我訓練模型的指令：

```
python -m axolotl.cli.train qlora.yml
```

其中，`qlora.yml`就是我設定的參數所在的檔案。

What hyper-parameters did you use?

About Lora Config

```
lora_r: 32
lora_alpha: 16
lora_dropout: 0.05
lora_target_linear: true
```

我有試過`lora_r=64`，會因為model還是過大，導致out of memory。

About Tokenizer

```
sequence_len: 4096
sample_packing: true
pad_to_sequence_len: true
```

About Training

```
gradient_accumulation_steps: 4
micro_batch_size: 2
num_epochs: 1
optimizer: paged_adamw_32bit
lr_scheduler: cosine
learning_rate: 0.0002

train_on_inputs: false
group_by_length: false
bf16: true
fp16: false
tf32: false
```

以上設定主要是參考了助教課提到的`axoloti` [1]。

`quantization_config`

```
quantization_config=BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_compute_dtype=torch.bfloat16,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_quant_type='nf4',
)
```

以上設定主要參考了投影片提供的`qlora` [2]。

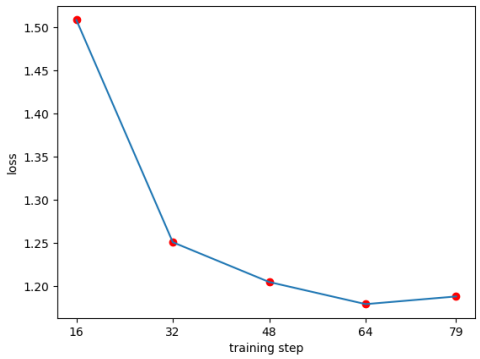
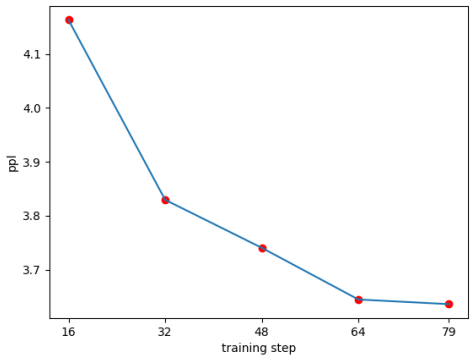
Show your performance

What is the final performance of your model on the public testing set?

`ppl = 3.63534`

Plot the learning curve on the public testing set

在`qlora.yml`中，我將`logging_steps`以及`save_steps`設成`0.2`，所以總共會有5個data point，分別對應到`0.2 epoch`, `0.4 epoch`, ..., `1 epoch`。

train loss	ppl in public test
	

step	16	32	48	64	79
loss	1.5086	1.251	1.2049	1.1793	1.1882
ppl	4.16314	3.82961	3.74060	3.64477	3.63617

Q2: LLM Inference Strategies

Zero-Shot

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

第一種:

```
def get_prompt(instruction: str) -> str:
    return instruction
```

其實就是不加任何東西，直接輸出。

在沒有fine-tune，且**有**將題目格式統一的情況下，`ppl = 10.93533`。

而在沒有fine-tune，且**沒有**將題目格式統一的情況下，`ppl = 7.14981`。

第二種:

```
def get_prompt(instruction: str) -> str:
    return f"你是人工智慧助理，以下是用戶和人工智能助理之間的對話。當用戶希望你將一段文言文或古文翻譯成白話文時，你應該要幫用戶將其翻譯成白話文。反之，當用戶希望你將一段白話文翻譯成文言文或古文時，你應該幫助用戶將其翻譯成文言文。USER: {instruction} ASSISTANT: "
```

這種和原本`utils.py`給的prompt很像，但我有再強調此次的任務為古文翻譯化白話文翻譯。

在沒有fine-tune，且**有**將題目格式統一的情況下，`ppl = 9.20747`。

而在沒有fine-tune，且**沒有**將題目格式統一的情況下，`ppl = 9.26619`。

Few-Shot (In-context Learning)

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

第一種:

```
def get_prompt(instruction: str) -> str:
    return f"把這句話翻譯成白話文：秦王惡之，後戒左右贊來不得通，贊亦不往，月一至府而已，退則杜門不交人事。 >> 秦王討厭他，後來告誡手下人劉贊來瞭不得通報，劉贊也不去，每月去王府一次罷瞭，迴來後就閉門不齣，不和人交往。把這句話翻譯成文言文：老虎發怒，直嚮皇上撲來，左右均退避，昭袞棄馬，翻身躍上虎背揪住虎的雙耳，老虎大驚，想要逃走。>> 虎怒，奮勢將犯蹕。左右闢易，昭袞捨馬，捉虎兩耳騎之。{instruction} >>"
```

這種是基於zero-shot的第一種prompt，再加上一個文言文翻白話文的例子，以及一個白話文翻文言文的例子，也就是2-shot。問題和答案中間使用`>>`隔開，和上課時的投影片相同。

在沒有fine-tune + 2-shot，且**有**將題目格式統一的情況下，`ppl = 4.76968`。

而在沒有fine-tune + 2-shot，且**沒有**將題目格式統一的情況下，`ppl = 5.12961`。

第二種：

```
def get_prompt(instruction: str) -> str:
    return f"你是人工智慧助理，以下是用戶和人工智能助理之間的對話。當用戶希望你將一段文言文或古文翻譯成白話文時，你應該要幫用戶將其翻譯成白話文。反之，當用戶希望你將一段白話文翻譯成文言文或古文時，你應該幫助用戶將其翻譯成文言文。例如，當用戶說「翻譯成白話文：秦王惡之，後戒左右贊來不得通，贊亦不往，月一至府而已，退則杜門不交人事。」，你應該回答「秦王討厭他，後來告誡手下人劉贊來瞭不得通報，劉贊也不去，每月去王府一次罷瞭，迴來後就閉門不齣，不和人交往。」。當用戶說「翻譯成文言文：老虎發怒，直嚮皇上撲來，左右均退避，昭袞棄馬，翻身躍上虎背揪住虎的雙耳，老虎大驚，想要逃走。」，你應該回答「虎怒，奮勢將犯蹕。左右闢易，昭袞捨馬，捉虎兩耳騎之。」。USER: {instruction} ASSISTANT: "
```

這種是基於zero-shot的第二種prompt，再加上一個文言文翻白話文的例子，以及一個白話文翻文言文的例子。問題和答案中間有加上較為流暢的文字敘述，例如：「當用戶說...，你應該回答...」，比較有對話的感覺。

在沒有fine-tune + 2-shot，且**有**將題目格式統一的情況下，`ppl = 8.35623`。

而在沒有fine-tune + 2-shot，且**沒有**將題目格式統一的情況下，`ppl = 8.05039`。

第三種：

```
def get_prompt(instruction: str) -> str:
    return f"把這句話翻譯成白話文：秦王惡之，後戒左右贊來不得通，贊亦不往，月一至府而已，退則杜門不交人事。 >> 秦王討厭他，後來告誡手下人劉贊來瞭不得通報，劉贊也不去，每月去王府一次罷瞭，迴來後就閉門不齣，不和人交往。把這句話翻譯成白話文：又胥女為楚王延壽後弟婦，數相饋遺，通私書。 >> 另外，劉胥的女兒是楚王劉延壽之後的弟婦，經常相互鎖贈禮物，私下書信往來。把這句話翻譯成文言文：老虎發怒，直嚮皇上撲來，左右均退避，昭袞棄馬，翻身躍上虎背揪住虎的雙耳，老虎大驚，想要逃走。 >> 虎怒，奮勢將犯蹕。左右闢易，昭袞捨馬，捉虎兩耳騎之。把這句話翻譯成文言文：大夫祭祀三個與人們日常生活密切相關的神，即族厲之神，門神，路神。 >> 大夫立三祀：曰族厲，曰門，曰行。{instruction} >> "
```

根據上述兩種prompt，顯然第一種prompt好很多，所以我再加上一個文言文翻白話文的例子，以及一個白話文翻文言文的例子，也就是4-shot。問題和答案中間一樣使用`>>`隔開。

在沒有fine-tune + 4-shot，且**有**將題目格式統一的情況下，`ppl = 7.23622`。

而在沒有fine-tune + 4-shot，且**沒有**將題目格式統一的情況下，`ppl = 7.21978`。

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

在以上的例子中，我有試過2-shot以及4-shot。在2-shot的情況下，我選擇的是一個文言文翻白話文的例子，以及一個白話文翻文言文的例子，在4-shot則是各兩個。

在挑選例子時，我是從`train.json`中，盡量找翻譯前和翻譯後差異較大的例子。

以下是我認為**不好**的例子：

```
{
  "id": "7b215019-026c-4b56-a506-8877861e9eeb",
  "instruction": "後二年，移鄧州，又徙襄州。\n把這句話翻譯成文言文：",
  "output": "後二年，徙鄧州，又徙襄州。"
}
```

在這個例子中，只有「移」被翻成「徙」，如果把這個當成例子的話，在預測的時候可能會以為只要把前句改一個字變成後句即可，導致分數不佳。

再舉一個文言文翻白話文**不好**的例子：

```
{
  "id": "a6b20dca-17b7-47b7-afe2-3523e6369f92",
  "instruction": "鄭餘慶\n翻譯成現代文：",
  "output": "鄭餘慶"
},
```

在這個例子中，前句和後句甚至沒有任何差異，如果把這個當成例子的話，應該沒有提供任何有用的資訊吧。

最後舉個我覺得**好**的例子：

```
{
  "id": "f1e8b632-6c88-4c93-822e-0bdcd4a2d527",
  "instruction": "翻譯成現代文：\n秦王惡之，後戒左右贊來不得通，贊亦不往，月一至府而已，退則杜門不交人事。",
  "output": "秦王討厭他，後來告誡手下人劉贊來瞭不得通報，劉贊也不去，每月去王府一次罷瞭，迴來後就閉門不齣，不和人交往。"
},
```

在這個例子中，幾乎前句的每一個詞語都有被翻譯成現代文，而且句子也算長，能夠提供不少有用的資訊。

Comparison

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

在0-shot, 2-shot, 4-shot中，我發現2-shot的表現最好，我認為0-shot表現不太好的原因可能在於沒有提供例子，資訊相對較少。至於4-shot表現不如2-shot，甚至和0-shot差不多的原因可能在於語句過長，或者資訊太多吧。

而在有沒有統一問題格式的部份，在0-shot並沒有特別突出的表現，甚至出現沒有統一比統一好的情況。但在2-shot的時候，統一格式的表現就明顯比沒有統一還要好，在few-shot的第一種prompt中，`ppl`從`5.12961`進步到`4.76968`，算是有非常顯著的提升。

至於在有使用LoRA fine-tune，以及沒有使用LoRA fine-tune中，顯然有使用LoRA fine-tune的表現又更上一層樓。在我的final performance中，我選擇的prompt是few-shot的第一種prompt，`ppl`從`4.76989`進步到`3.63534`，算是有非常顯著的提升。

Q3: Bonus: Other methods

Experiments with different PLMs

Describe your experimental settings and compare the results to those obtained from your original methods

我使用的PLM為`hfl/chinese-alpaca-2-7b` [3]，由於模型過大，所以我沒有進行fine-tune。

我使用的prompt為few-shot的第一種prompt，也就是我最終使用的prompt，test data的題目也有統一格式。

	Taiwan-LLM-7B-v2.0-chat	hfl/chinese-alpaca-2-7
`ppl`	4.76968	15.01981

雖然沒有經過fine-tune，但很顯然後者的表現和前者的表現差蠻多的，於是我看了一下用後者預測的文字，發現有很多簡體字出現，或許這就是原因吧。

Reference:

[1] https://github.com/OpenAccess-AI-Collective/axolotl/tree/main

[2] https://github.com/artidoro/qlora

[3] https://huggingface.co/hfl/chinese-alpaca-2-7b