Q1: LLM Tuning

How much training data did you use? (2%)

用 other/select_data.ipynb 從 train.json 中過濾出 3400 筆 instruction 和 output data，其中 1700 為文言文轉白話文，另外 1700 為白話文轉文言文。

How did you tune your model? (2%)

按照規定使用 Bits and Bytes 和 QLoRA 對 Taiwan-LLaMa 做 Instruction Tuning。為了讓大型 LLM 也可以在有限的 GPU 資源上 fine tune，使用 BitsAndBytesConfig 減少多餘的精確度，大幅降低 memory 使用量。用 QLoRA 對 target modules 進行 fine tune，減少需要調整的參數量。

為了使 Taiwan-LLaMa 能夠應付專門的文言文翻譯，運用成對翻譯的 instruction 和 output data，用 qlora 對 target modules 做調整，使其學習翻譯的方法。

因為這次需要文言文到白話文的雙向翻譯，因此在設計 prompt 的時候使用一個文言文到白話文的翻譯和一個白話文到文言文的翻譯做示範，並在後面加上該筆 data 的 instruction，期望 model 可以生出對應的 output。

```
def get_prompt(instruction: str) -> str:
    '''Format the instruction as a prompt for LLM.'''
    return f"以下為翻譯指令，盡量簡短翻譯，並輸出翻譯結果。\n\n \
    ### 翻譯指令：翻譯成文言文：\n尚未赴任時，便又改任金州知州。\n\n### 翻譯結果：未行，改金州。\n\n \
    ### 翻譯指令：讜雖不從，天下毓重其言。\n翻譯成白話文：\n\n### 翻譯結果：他的建議雖然不被采納，但天下都很敬重他的話。\n\n \
    ### 翻譯指令：{instruction}\n\n### 翻譯結果："
```

What hyper-parameters did you use? (2%)

1. Pretrained weights: yentinglin/Taiwan-LLM-7B-v2.0-chat
2. Max source length: 512
3. Max target length: 256
4. Batch size: 6(per_device)*3(gradient_accumulation_steps)
5. Learning rate: 0.0002
6. Steps: 190
7. warmup_ratio: 0.03
8. lora_r: 64,
9. lora_alpha: 32
10. lora_dropout": 0.05
11. target_modules": [
    "gate_proj",
    "down_proj",
    "v_proj",
    "q_proj",
    "o_proj",
    "up_proj",
    "k_proj"

]

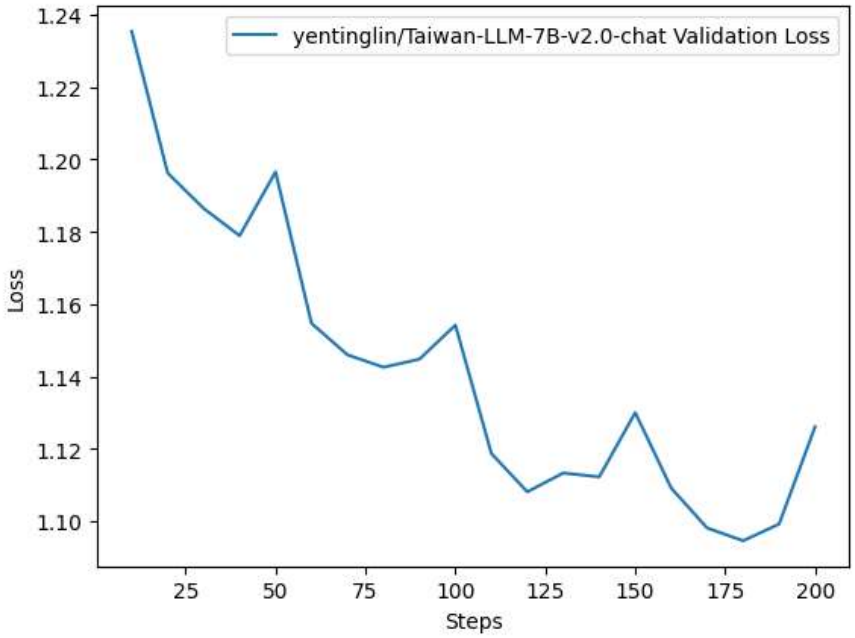
Show your performance:

What is the final performance of your model on the public testing set? (2%)

mean_perplexity : 3.5067630133628844

Training Loss: 1.049900 / Validation Loss: 1.099149

Plot the learning curve on the public testing set (2%)



Q2: LLM Inference Strategies

Zero-Shot

What is your setting? How did you design your prompt? (1%)

因為 training data 中的 instruction 本身就已經包含翻譯的指令和要翻譯的原文，因此 zero shot 沒有加上額外的翻譯指令，單純使用 data 中的 instruction 當做 prompt。

```
    return f"{instruction}"
```

Few-Shot (In-context Learning)

What is your setting? How did you design your prompt? (1%)

使用一個文言文到白話文的翻譯：「翻譯成文言文：\n 富貴貧賤都很尊重他。\n 答案：貴賤並敬之。\n\n」和一個白話文到文言文的翻譯：「議雖不從，天下鹹重其言。\n 翻譯成白話文：他的建議雖然不被采納，但天下都很敬重他的話。\n\n」，並在後面加上該筆 data 的 instruction 做為 prompt。

```
return f"\
翻譯成文言文：\n富貴貧賤都很尊重他。\n答案：貴賤並敬之。\n\n\
議雖不從，天下鹹重其言。\n翻譯成白話文：他的建議雖然不被采納，但天下都很敬重他的話。\n\n\
{instruction}"
```

How many in-context examples are utilized? How you select them? (1%)

測試時發現 4 個 examples 並沒有得到更好的效果，因此使用 2 個 examples，一個翻譯成白話文，一個翻譯成文言文，因為想讓 model 了解有 2 個翻譯的可能。在選擇時 examples 時，挑選翻譯前後差異較大的作為區分。

Comparison:

What's the difference between the results of zero-shot, few-shot, and LoRA? (2%)

	zero-shot	two-shot	four-shot	LoRA+two-shot
mean_perplexity	7.143351150	6.919742969	6.930564581	3.506763013

Q3: Bonus: Other methods (2%)

Experiments with different PLMs

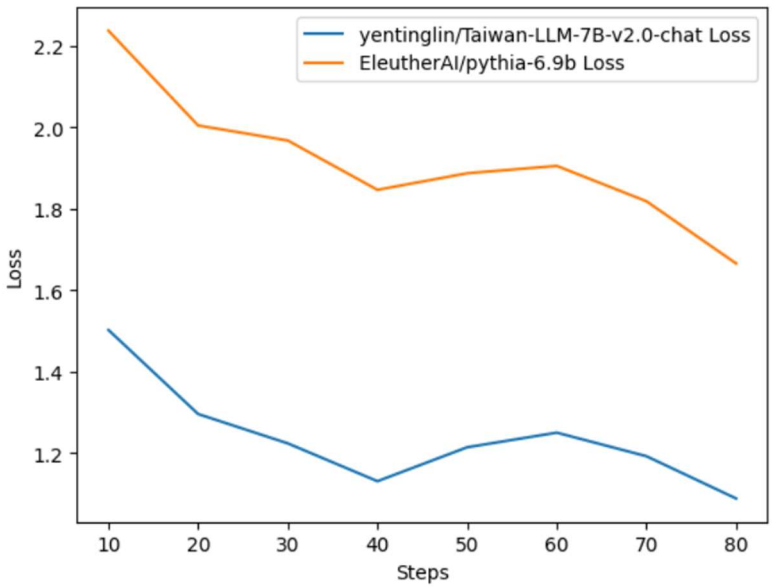
選擇參數數量和 Taiwan-LLaMa 相差不多的 EleutherAI/pythia-6.9b 來做測試，為 Transformer 結構、6.9B 參數、以英文為主的 model，想了解不同 pretrained data 語言對 model 的影響。

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

兩個 model 的參數都相同，和 Q1 參數差異處如下：

1. Steps: 80
2. lora_r: 64,
3. lora_alpha: 16
4. lora_dropout": 0.0

用一樣的方法，輸入中文 instruction，用 Qlora fine tune target modules，參數和 prompt 設置也相同，但是由於適用語言的不同，因此效果比 Taiwan-LLaMa 差。下圖為兩者的 learning curve，發現在同樣的 training steps 下，EleutherAI/pythia-6.9b 的 loss 比 Taiwan-LLaMa 高約 0.6。



用相同的參數和 prompt 生成翻譯內容，發現 EleutherAI 雖然可以正常生成繁體中文，但是時常輸出原來的內容，只是將翻譯指令刪除，並沒有經過翻譯。以下為生成範例。

問題	從唐末幽、薊二州割據以來，戍守的軍隊廢置散失，契丹因而得以齣來攻陷平、營二州，而幽、薊二州的人每年深受契丹侵犯掠奪之苦。\n這句話在中國古代怎麼說：
EleutherAI	從唐末幽、薊二州割據以來，戍守的軍隊廢置散失，契丹因而得以齣來攻陷平、營二州，而幽、薊二州的人每年深受契丹侵犯掠奪之苦。
Taiwan-LLaMa	自唐末幽、薊割據，軍隊廢，契丹因而齣來攻營，幽、薊人每年苦為契丹犯奪。

問題	秦領是都人，從江夏都尉升任南陽太守，上任時經過宜城城內，看見一傢朝東的房子，他停車觀看，說：這個住人的地方可以築墓。\n翻譯成文言文：
EleutherAI	秦領是都人，從江夏都尉升南陽太守，上任時經過宜城城內，見一傢朝東的房子，他停車觀看，說：這個住人的地方可以築墓。
Taiwan-LLaMa	秦領都人，從江夏都尉升南陽，上者馳宜城，見一傢朝東，說：此墓之地。

使用助教提供的 ppl.py 測試兩個 model，EleutherAI 表現明顯不如 Taiwan-LLaMa。

	Taiwan-LLaMa	EleutherAI
mean_perplexity	3.866767275	9.08478125