

# NTU ADL 2024 Fall HW3

Deadline: 2024/11/7 23:59:59

# Update Logs

1. Learning curve 請畫在public testing data上面
2. adapter\_checkpoint
  - ├ adapter\_config.json
  - └ adapter\_model.bin -> adapter\_model.safetensors
3. DEADLINE: ~~10/31 23:59:59~~ 11/7 23:59:59

# Links

- Homework 3 files ([link](#))

# Task Description

# Instruction Tuning (Classical Chinese)

- *Example 1*

**Instruction:**

翻譯成文言文：

雅裏惱怒地說：從前在福山田獵時，你誣陷獵官，現在又說這種話。

答案：

**Output:**

雅裏怒曰：昔畋於福山，卿誣獵官，今復有此言。

- *Example 2*

**Instruction:**

議雖不從，天下咸重其言。

翻譯成白話文：

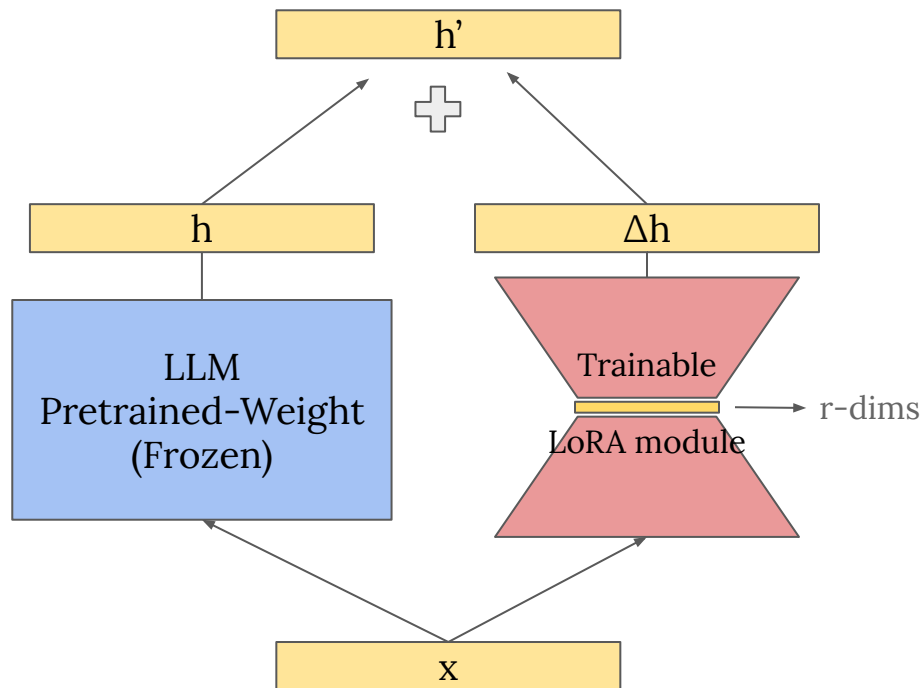
**Output:**

他的建議雖然不被採納，但天下都很敬重他的話。

# Model- Gemma 2

- Model Name: zake7749/gemma-2-2b-it-chinese-kyara-dpo
- Gemma is a open source LLM trained by Google
- The model we used is the one fine-tuned on traditional chinese data
- For more information about the model: [link](#)

# Low Rank Adaptation (LoRA)



# QLoRA Fine-tuning

- QLoRA reduces the memory usage of LLM fine-tuning by employing 4-bit quantization to compress a LLM, while utilizing 16-bit float to perform computations
- [Paper](#)



# Experiments

- Dataset
  - Training (train.json): 10000
  - Testing (Public) (public\_test.json): 250
  - Testing (Private) (private\_test.json): 250
- Evaluation:
  - Perplexity (ppl.py)

# Data Format

- json format

```
[
  {
    "id": "0094a447412998f6",
    "instruction": "高祖初，為內秘書侍禦中散。\\n翻譯成現代文：" ,
    "output": "高祖初年，任內秘書侍禦中散。"
  },
  {
    "id": "01b15dfad04577c4",
    "instruction": "它的旁邊有一顆小星，名叫長沙星，星不宜明，若與軫宿的四顆星一樣明亮，五顆星進入軫宿，錶示將有大的戰爭發生。\\n這句話在中國古代怎麼說：" ,
    "output": "其旁有一小星，曰長沙，星星不欲明；明與四星等，若五星入軫中，兵大起。"
  },
  ...
]
```

# Rules

# What You Can Do:

- LLM checkpoint:
  - [zake7749/gemma-2-2b-it-chinese-kyara-dpo](#)
- Packages:
  - torch==2.4.1, transformers==4.45.1, bitsandbytes==0.44.1, peft==0.13.0
  - Others: no limitation

# What You Can NOT Do

- Use external training data
- Any means of cheating or plagiarism, including but not limited to:
  - Directly apply others' published / unpublished code..., including the codes written your classmates or public on the internet
  - Give/get trained model/predictions to/from others.
  - Give/get report answers or plots to/from others.
  - Publish your code before deadline.
- Violations may cause zero/negative score and punishment from school.

Submission

# File Layout - Before downloading

- Zip your folder, which should be named as your student id (lower-cased) (ex. r11000000) and submit the .zip to NTU Cool.

r11000000

|— download.sh

|— utils.py

|— prediction.json

|— run.sh

|— report.pdf

|— README.md

|— code/script (all the code/script you used to train, predict, or plot report figures should be included)

# download.sh

- **download.sh** should download or create a folder called **adapter\_checkpoint** which contains peft configuration file (adapter\_conf.json) and weights (adapter\_model.bin).
  - **!!DO NOT download original model checkpoint!!**
- Do not modify your file after deadline, or it will be seen as cheating.
- Keep the URLs in download.sh valid for at least **3 weeks** after deadline.
- You can download at most **4G**, and download.sh should finish within 1 hour. (At csie dept with maximum 10MB/s bandwidth)
- Do not do things more than downloading. Otherwise, your **download.sh** may be killed.
- Do not pip install ANYTHING in your **download.sh**, you are not allowed to modify the testing environment
- We will execute **download.sh** before predicting scripts.



# File Layout - After downloading

- After we run download.sh, the **adapter\_config.json** and **adapter\_model.bin** should be in **adapter\_checkpoint**.

r11000000

```
|— download.sh
|— utils.py
|— prediction.json
|— run.sh
|— report.pdf
|— README.md
|— code/script (all the code/script you used to train, predict, or plot report figures should be included)
└— adapter_checkpoint
    |— adapter_config.json
    └— adapter_model.bin -> adapter_model.safetensors
```

# utils.py

- Two functions are required:
  - `def get_prompt(instruction: str) -> str:`
    - return the prompt to input into the LLM
  - `def get_bnb_config() -> BitsAndBytesConfig:`
    - return your bnb configuration
- TAs will include “`from utils import get_prompt, get_bnb_config`” when testing.
- Make sure that your function's input and output fields are correct and can be used accurately

# README.md

- README.md should contain step-by-step instructions on how to setup your environments and how to train your model with your codes/scripts.
- You will get a **-2** penalty if you have no or empty README.md.
- If necessary, you will be required to reproduce your results based on the README.md.
- If you cannot reproduce your result, you may lose points.

# run.sh

- **run.sh** should perform text generation using your trained models and output predictions on testing file (.json)
- arguments
  - \${1}: path to the model checkpoint folder
  - \${2}: path to the adapter\_checkpoint downloaded **under your folder**
  - \${3}: path to the input file (.json)
  - \${4}: path to the output file (.json)
- TA will predict testing data as follow:
  1. bash ./download.sh
  2. bash ./run.sh /path/to/model-folder /path/to/adapter\_checkpoint \ /path/to/input.json /path/to/output.json
- run.sh should finish within 2 hours. (See [environment details](#))

# run.sh - cont.

Command:

```
bash run.sh \  
  /path/to/`zake7749/gemma-2-2b-it-chinese-kyara-dpo` \  
  /path/to/adapter_checkpoint/under/your/folder \  
  /path/to/input \  
  /path/to/output
```

○ example:

```
bash run.sh \  
  /home/ \ `zake7749/gemma-2-2b-it-chinese-kyara-dpo` \  
  /home/hw3/r11922000/adapter_checkpoint \  
  /home/data/public_test.json \  
  /home/output/r11922000_output.json
```

# prediction.json

- Testing set (Private)
- DO NOT include any special tokens (<s>, </s>, ...) and your prompt in your output

```
[
  {
    "id": "0094a447412998f6",
    "output": "高祖初年，任內秘書侍禦中散。"
  },
  {
    "id": "01b15dfad04577c4",
    "output": "其旁有一小星，曰長沙，星星不欲明；明與四星等，若五星入軫中，兵大起。"
  },
  ...
]
```

# Execution Environment

- We will run the testing codes on the computer with
  - Ubuntu 20.04
  - 32GB RAM, GTX 3070 **8GB VRAM**, and 20GB disk space available
- Python3.10
- Packages
  - torch==2.4.1, transformers==4.45.1, bitsandbytes==0.44.1, peft==0.13.0

# Grading

- Model Performance (5%)
  - Public baseline: ppl = 17.500↓ (2%)
  - Private baseline: ppl = 24.000↓ (2%)
  - prediction.json: Human evaluation (1%)
- Report (15% + 2%)
- Format
  - You may lose (some or all) of your model performance score if your script is at wrong location, causes any error, etc.



# Grading - ppl.py

- TA will use our own ppl.py (which is the same as we published), so you don't have to upload ppl.py
- Command:

```
python3 ppl.py \  
  --base_model_path /path/to/`zake7749/gemma-2-2b-it-chinese-kyara-dpo` \  
  --peft_path /path/to/adapter_checkpoint/under/your/folder \  
  --test_data_path /path/to/input/data
```

- example:

```
python3 ppl.py \  
  --base_model_path /home/`zake7749/gemma-2-2b-it-chinese-kyara-dpo` \  
  --peft_path /home/hw3/r11922000/adapter_checkpoiint \  
  --test_data_path /home/data/public_test.json
```

# Late Submission

- Late submission penalties:
  - 0 day < late submission  $\leq$  1 day: original score \* 0.95
  - 1 day < late submission  $\leq$  3 day: original score \* 0.90
  - 3 day < late submission  $\leq$  4 day: original score \* 0.75
  - 4 day < late submission  $\leq$  5 day: original score \* 0.50
  - 5 day < late submission  $\leq$  6 day: original score \* 0.25
  - 6 day < late submission: original score \* 0.00
- Late submission is determined by the last submission.
  - Update your submission after deadline implies that you will get penalty.

# Report

# Q1: LLM Tuning

- Describe:
  - How much training data did you use? (2%)
  - How did you tune your model? (2%)
  - What hyper-parameters did you use? (2%)
- Show your performance:
  - What is the final performance of your model on the public testing set? (2%)
  - Plot the learning curve on the public testing set (2%)

## Q2: LLM Inference Strategies

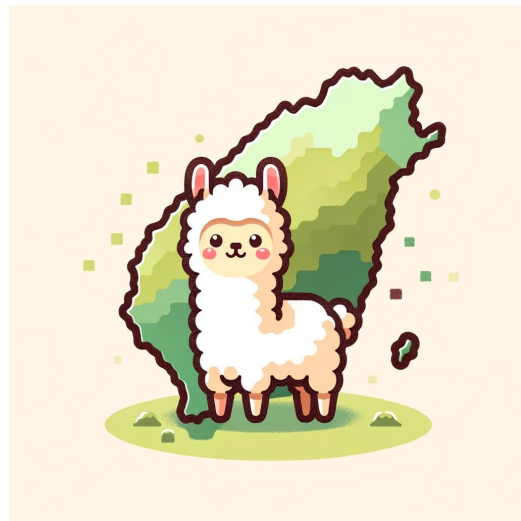
- Zero-Shot
  - What is your setting? How did you design your prompt? (1%)
- Few-Shot (In-context Learning)
  - What is your setting? How did you design your prompt? (1%)
  - How many in-context examples are utilized? How you select them? (1%)
- Comparison:
  - What's the difference between the results of zero-shot, few-shot, and LoRA? (2%)

**Note:**

**Please conduct zero-shot and few-shot experiments on Original Model that has not been fine-tuned with QLoRA**

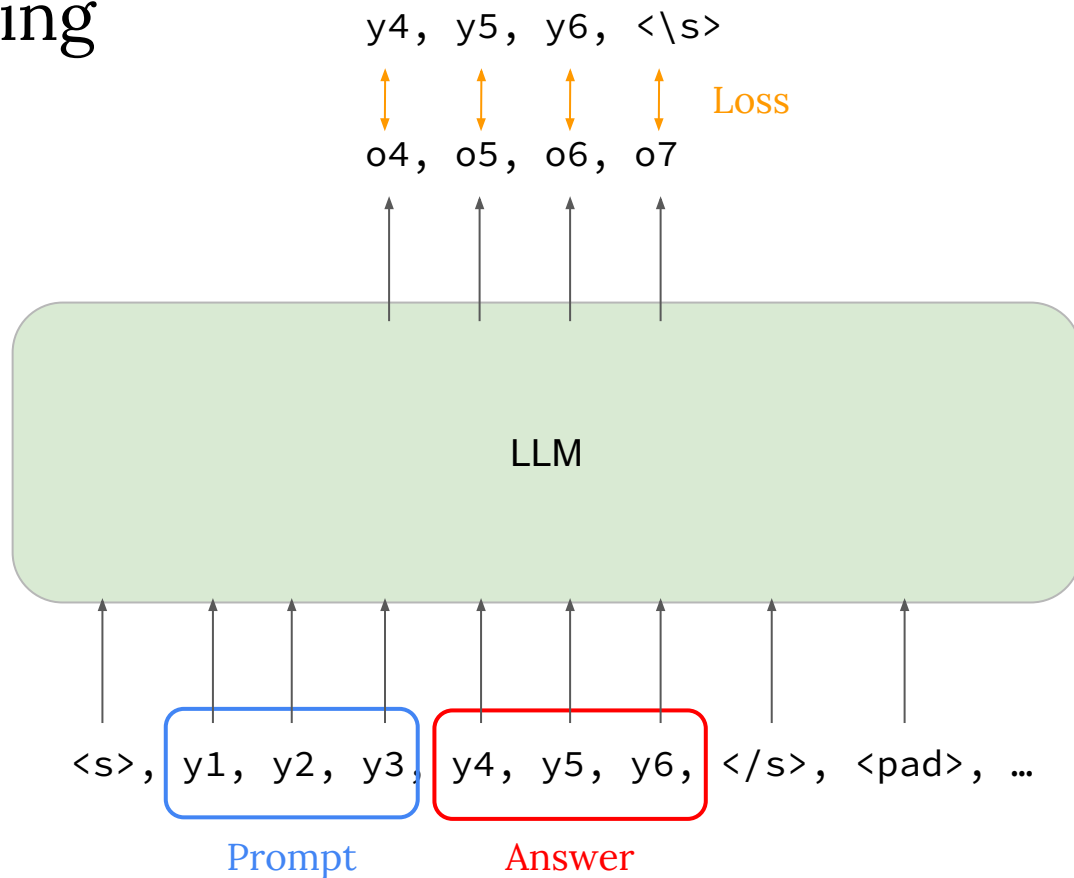
## Q3: Bonus: Try Llama3-Taiwan (8B) (2%)

- Llama-3-8b trained by traditional Chinese data
- Tune this model on the classical chinese data
- Describe your experimental settings and compare the results to those obtained from your original methods



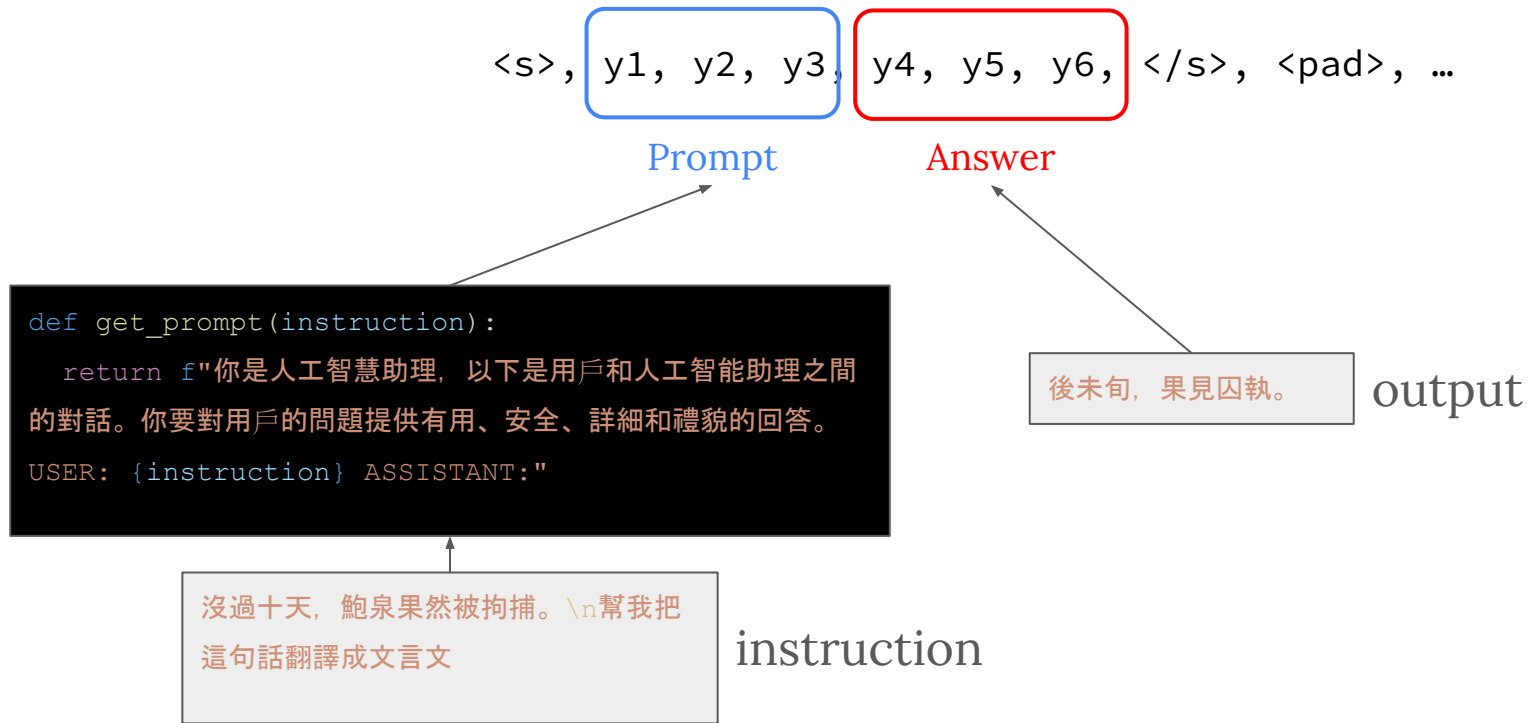
# Guides

# Instruction Tuning

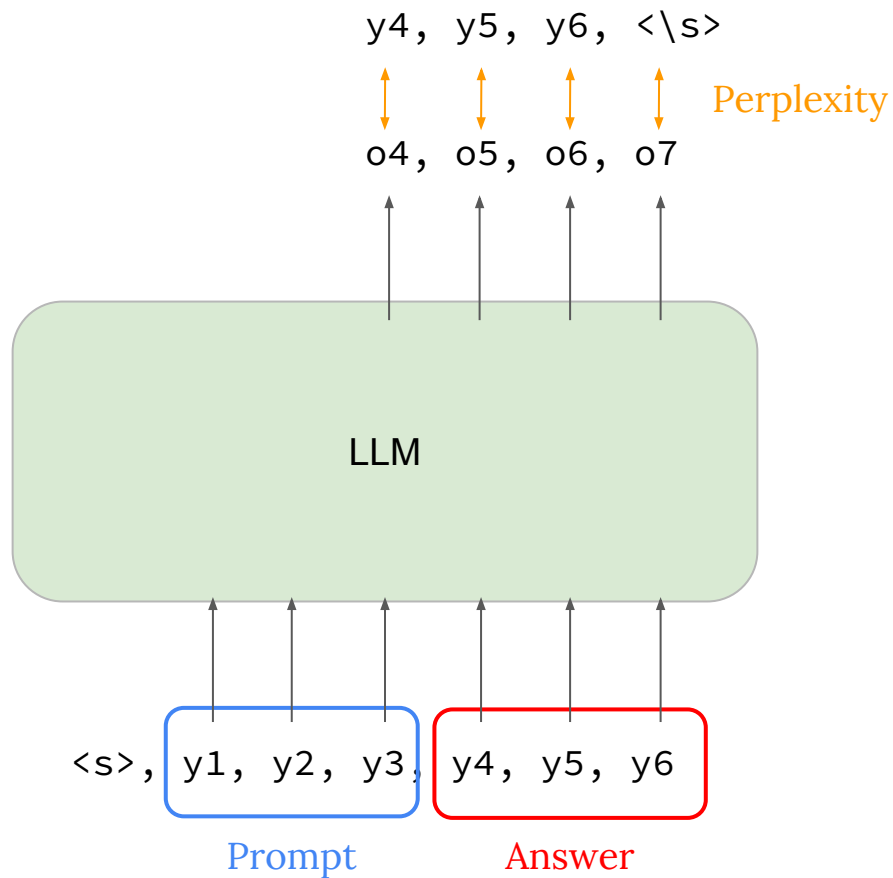




# Instruction Tuning



# Perplexity



# Quantization (4 bit)

```
from utils import get_bnb_config
from peft import prepare_model_for_kbit_training

bnb_config = get_bnb_config() # return a BitsAndBytesConfig

model = AutoModelForCausalLM.from_pretrained(
    model_name_or_path,
    quantization_config=bnb_config
)

if training:
    model = prepare_model_for_kbit_training(model)
```

Ref:

1. <https://huggingface.co/blog/4bit-transformers-bitsandbytes>
2. [https://huggingface.co/docs/transformers/main\\_classes/quantization#advanced-use-cases](https://huggingface.co/docs/transformers/main_classes/quantization#advanced-use-cases)
3. <https://github.com/artidoro/qlora>

# Reference

1. <https://huggingface.co/docs/peft/index>
2. <https://github.com/huggingface/peft>
3. <https://huggingface.co/blog/4bit-transformers-bitsandbytes>
4. [https://huggingface.co/docs/transformers/main\\_classes/quantization#advanced-use-cases](https://huggingface.co/docs/transformers/main_classes/quantization#advanced-use-cases)
5. <https://github.com/artidoro/qlora>
6. <https://github.com/huggingface/trl>

# Any questions

- NTU COOL discussion
- Email:
  - [adl-ta@csie.ntu.edu.tw](mailto:adl-ta@csie.ntu.edu.tw)
- TA hours
  - Tue. 11:00 ~ 12:00 google meet
  - Thur. 14:00 ~ 15:00 @ 徳田 524