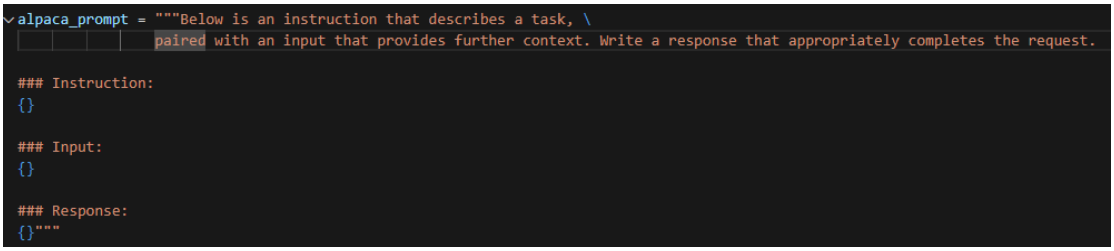
ZH-TW Reading Comprehension Test for LLMs

1. Create DataSets

因為使用 Alpaca_llama3 的 ipynb 做 finetune，因此先將原始數據集整理成 json 格式方便後續訓練時配合讀取成 prompt 格式。

DataSets json:

Prompt(可以自行調整):



2. Training

- 參考連結:
  https://colab.research.google.com/drive/135ced7oHytdxu3N2DNe1Z0kqjyYIkDXp?usp=sharing#scrollTo=e2pEuRb1r2Vg

分別使用不同的模型進行訓練，分別有

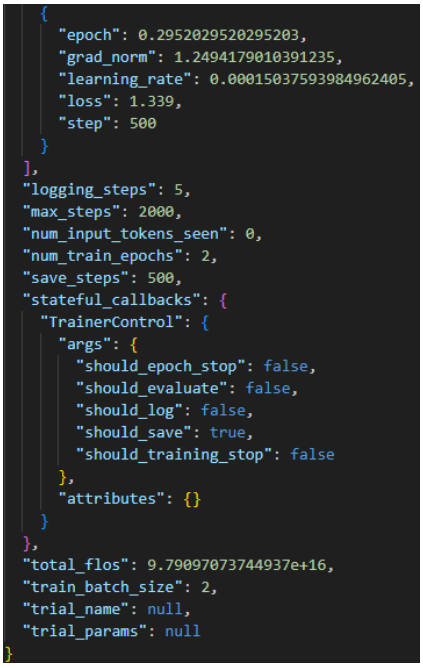
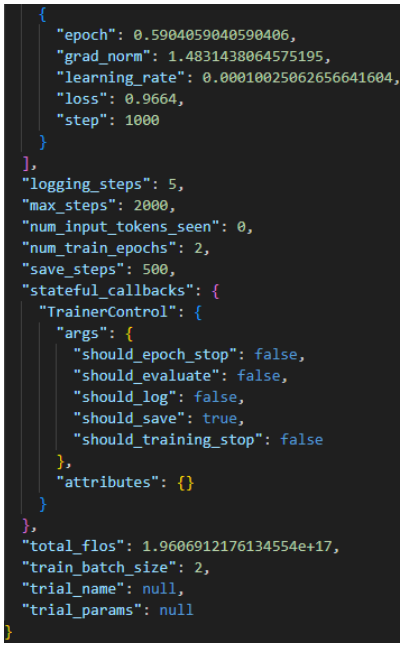
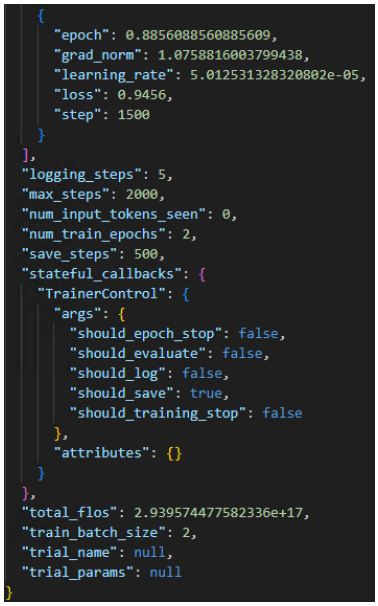
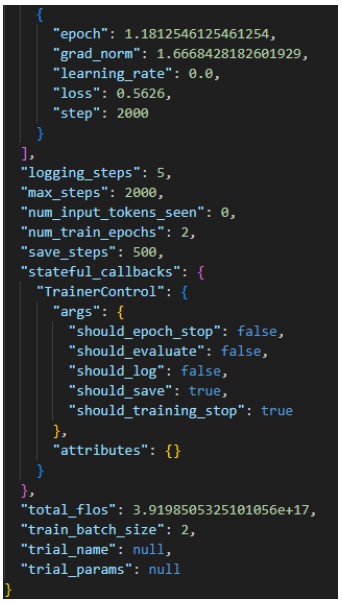
1. "unsloth/mistral-7b-v0.3-bnb-4bit"
2. “yentinglin/Llama-3-Taiwan-8B-Instruct”
3. “yentinglin/Taiwan-LLM-7B-v2.0-base”

使用 usloth library:

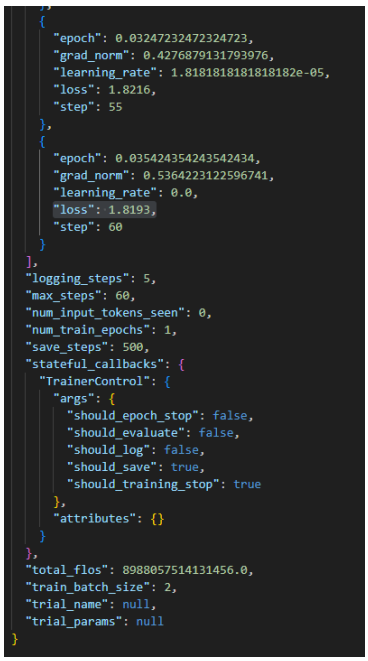
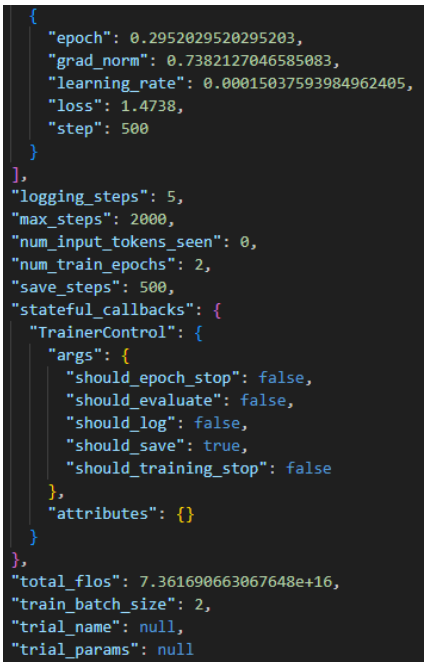
可以更高效率的讀取模型；並寫更高效進行模型訓練。

可以在維持不錯生成回答精確度的同時降低記憶體在訓練時的使用量，目前測試能夠限縮到使用 6GB

1. mistral-7b-v0.3-bnb-4bit

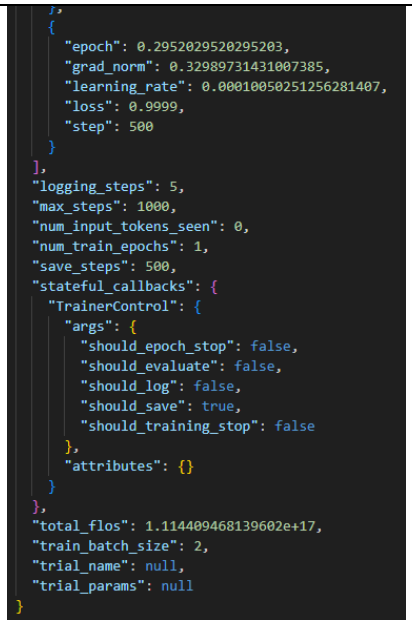
<table>
<tr><td>  </td><td>  </td><td>  </td></tr>
<tr><td>Checkpoint-60</td><td>Checkpoint-500</td><td>Checkpoint-1000</td></tr>
<tr><td>  </td><td>  </td><td></td></tr>
<tr><td>Checkpoint-1500</td><td>Checkpoint-2000</td><td></td></tr>
</table>

2. Llama-3-Taiwan-8B-Instruct

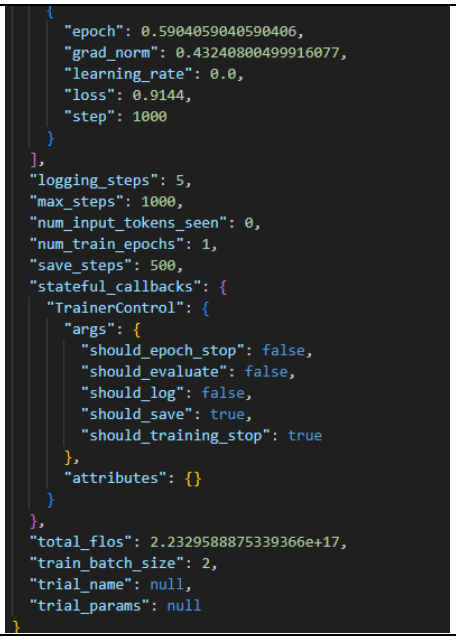
<table>
<tr>
<td> 
</td>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>Checkpoint-60</td>
<td>Checkpoint-500</td>
<td>Checkpoint-1000</td>
</tr>
<tr>
<td> 
</td>
<td> 
</td>
<td></td>
</tr>
<tr>
<td>Checkpoint-1500</td>
<td>Checkpoint-2000</td>
<td></td>
</tr>
</table>

3. Taiwan-LLM-7B-v2.0-base

<table>
<tr>
<td>



</td>
<td>



</td>
</tr>
<tr>
<td>Checkpoint-500</td>
<td>Checkpoint-1000</td>
</tr>
</table>

3. Inference

使用 usloth 載入模型:

```
# 加載最佳模型和 Tokenizer
model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = model_dir,
    max_seq_length = max_seq_length,
    dtype = dtype,
    load_in_4bit = load_in_4bit,
    # token = "hf_...", # use one if using gated models like m
)

# # 调用 for_inference
FastLanguageModel.for_inference(model)
```

用與 train 相同的 prompt 進行 test 的讀取。

```
def formatting_prompts_test(examples):
    instructions = examples["instruction"]
    inputs       = examples["question"]
    outputs      = examples["output"]
    texts = []
    for instruction, input, output in zip(instructions, inputs, outputs):
        text = alpaca_prompt.format(instruction, input, output)
        texts.append(text)
    return { "text" : texts }
```

最後按照平常的格式進行文字生成

```python
with torch.no_grad():
    print("Start inference.")
    results = []
    for test_dataset in tqdm(test_datasets, desc="inference: "):

        id = int(test_dataset['id'])
        input_text = str(test_dataset['text'])

        # 生成回答
        inputs = tokenizer(input_text,return_tensors="pt").to("cuda")
        generation_output = model.generate(**inputs, max_new_tokens = 64, use_cache = True)

        s = generation_output[0]

        output = tokenizer.decode(s,skip_special_tokens=True)
        response = output.split("### Response:")[1].strip()

        answer = response.split(')')[0]

        # print(f"======={id}=======")
        # print(f"Input: {input_text}\n")
        # print(f"Response: {response}\n")

        results.append({"id":id,"answer":answer})

    print("End inference.")
```

P.S

機器學習:

【分類問題】: 需要 label 正確答案，作為參照。

【文字生成】: 不需要 label。需要將所有文字文章、問題、選擇 1~4、答案，利用自行設定的 prompt 串接成一個 text，作為 input。Encode/decode 都使用 tokenizer，不過 Encode 部分可以包還在 SFTTrainer、Trainer 訓練器做轉換，不須在資料前處理的時候就先手動轉換 Encode。避免發生不必要的轉換錯誤。