本次作業內容為Named-Entity Recognition

本次使用的環境為以下，套件則為附件中的requirements.txt

Environment types	If local
Running environment	System: Ubuntu 22.04, CPU: Ryzen 7-7800X3D
Python version	Python 3.10.1

一、資料前處理流程

本次作業使用的資料集為 Hugging Face 上的 `ncbi_disease`，資料已以詞（word）為單位斷詞，並標註了 `ner_tags`，採用 BIO tagging scheme，分別為：

- `O`：非實體
- `B-Disease`：疾病名稱開頭
- `I-Disease`：疾病名稱延續詞

處理步驟如下：

1. 使用 `AutoTokenizer`（`bert-base-uncased`）進行斷詞，設定 `is_split_into_words=True`。
2. 對 `ner_tags` 進行對齊處理，對 `subword` 的 label 設定為 `-100`（忽略 loss 計算）。
3. 使用 `.map()` 將轉換應用到整個資料集，並移除原始欄位。
4. 使用 `DataCollatorForTokenClassification` 動態 padding。

二、模型與超參數

使用 Hugging Face Transformers 的 `BertForTokenClassification`：

- **預訓練模型**：`bert-base-uncased`
- **輸出類別數**：3（O, B-Disease, I-Disease）
- **訓練設定**（`TrainingArguments`）：

參數	值
batch size (train/eval)	16 / 32
learning rate	5e-5
optimizer	AdamW
warmup_ratio	0.1
epoch 數	3
gradient_accumulation_steps	2
logging / save strategy	每 epoch
evaluation_strategy	epoch
使用 FP16	是
load_best_model_at_end	是

三、模型訓練與效能提升方法

訓練方式：使用 Hugging Face `Trainer` API，自動管理 optimizer、scheduler、evaluation loop。

效能提升手法：

1. 加入 `warmup_ratio=0.1` 使學習率平滑啟動。
2. 使用 `FP16` 加速訓練並減少記憶體使用。
3. 動態 padding（`DataCollatorForTokenClassification`）降低不必要的 padding 開銷。
4. 加入 `load_best_model_at_end` 以 validation loss 做 early selection。
5. 定義 `compute_metrics` 使用 `seqeval` 評估 `f1`, `precision`, `recall`, `accuracy`。

四、模型效能與主要影響因子

Dataset	F1-score	Precision	Recall	Accuracy
Validation	0.8518	0.8228	0.8829	0.9821
Test	0.8583	0.8366	0.8813	0.9748

> **主要影響因子分析：**
> 子詞分割（subword tokenization）造成 NER 標註不連續問題，是模型準確率與召回率的主因。透過正確處理 `labels` 與設計合理的 batch/padding 策略，可以穩定訓練並達到良好泛化能力。

五、額外補充

- 我嘗試在訓練前以小批量測試 `compute_metrics`，並修正 id2label 映射錯誤問題。
- 在 debug 時，有使用 ChatGPT-4o 協助處理錯誤訊息（如 `evaluation_strategy`、`seqeval.compute` 模組問題），並已於程式碼內註記。
- 這次作業，是我第一次沒使用ChatGPT協助作業，幫助深入理解 Hugging Face `Trainer` 的工作流程與 token classification 的重點轉換邏輯。