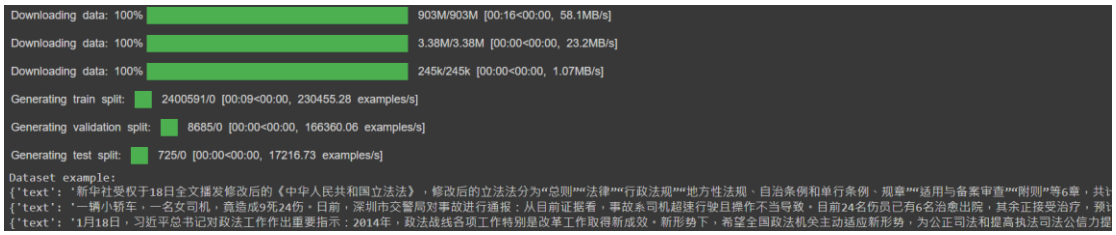
作業二 b 報告

資訊 114　H44091196 洪茂菘

1. Data:

✓ Load the "train" and "validation" splits of data (15 pts). You can use pandas, Dataset and Dataloader.

```
class CommonGenDataset(Dataset):
    def __init__(self, split="train") -> None:
        super().__init__()
        assert split in ["train", "validation", "test"]
        dataset = load_dataset("hugcyp/LCSTS", split=split, cache_dir="./cache/").to_pandas()
        self.data = []
        for index, row in dataset.iterrows():
            text = row["text"]
            summary = row["summary"]
            self.data.append({"text": text, "summary": summary})
```



由上兩張圖可以看到有成功將資料分為 train ,validation ,test 三個集合並將前三筆新聞內容顯示出來。

✓ Tokenize the text (15 pts). You can design your own tokenizer or use any API (recommended).

```
t5_tokenizer = T.T5Tokenizer.from_pretrained("google/flan-t5-base", cache_dir="./cache/")
```

如上圖，這邊我使用的是 t5 tokenizer。

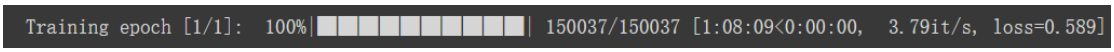
2. Generation Models

✓ Model design (15 pts). Unlike Project 2.a, you should use transformer-based model. (Huggingface API)

```
t5_model = T.T5ForConditionalGeneration.from_pretrained("google/flan-t5-base", cache_dir="./cache/").to(device)
```

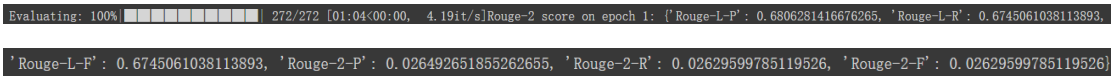
我使用 Transformes (一個由 Hugging Face 提供的開源機器學習庫) 來載入一個預訓練的模型，在這次作業中，是使用 T5 模型的一個變體，名字為 "FLAN-T5"。

✓ Train(finetune) the model (10 pts).



如上圖，因為訓練一次要花超級久的時間，且 colab 有時還會因 TPU 用量限制而強制跳停，所以我將 epoch 設為 1，只做一次訓練。

✓ Evaluate your model when you are training. (5 pts)



我的 evaluate 結果如上兩張圖所呈現。

3. Analysis

Model (20 pts)

✓ Intrioduce what model you have used in your code (10 pts). Compare the T5 model with GPT2 (10pts) and describe the differences between T5 and GPT2.

如上圖所示，由於我是第一次做這類型的作業，因此我決定選擇和助教提供的範例程式碼一樣使用 T5 模型作為本次作業的訓練模型。

根據我在網路上查到的資料顯示，T5 是將所有自然語言處理問題視為文本到文本的轉換，採用統一模型來處理各種任務，如翻譯、摘要、問答等。它通過預訓練和針對特定任務的微調來提升性能，適應不同的文本轉換任務，強調任務適應性和靈活性。

而 GPT-2 則是透過大規模的無監督學習，專注於生

成連貫的文本。這個模型主要用於開放式的文本生成，如故事創作、對話生成等，優勢在於創造性內容的生成和開放式對話應用。

從以上的結果來看，我認為還是 T5 模型較適合本次的作業。

Dataset (5 pts)

- ✓ Briefly describe your methods to process the data and how to input them into the model.

首先，我透過 CommonGenDataset 這個 class 來加載並預先處理 LCSTS 數據集，將文本和摘要轉換為模型可處理的形式；接著再使用 t5_tokenizer.batch_encode_plus 將文本批量編碼為張量，並通過 DataLoader 將 Batch 做處理；最後在模型訓練時，對這些 Batch 數據進行迭代，計算損失並更新權重。最後，使用 ROUGE 指標來評估模型性能。

Train (10 pts)

- ✓ Describe how do you train(tune) your model.

```python
for ep in range(epochs):
    t5_model.train()
    total_loss = 0.0
    pbar = tqdm(lcsts_train)
    pbar.set_description(f"Training epoch [{ep+1}/{epochs}]")
    for inputs, targets in pbar:
        optimizer.zero_grad()
        loss = t5_model(input_ids=inputs, labels=targets).loss
        loss.backward()
        optimizer.step()
        total_loss += loss.item()
        pbar.set_postfix(loss=loss.item())
    avg_loss = total_loss / len(lcsts_train)
    print(f"Avg. Loss on epoch {ep+1}: {avg_loss}")
    # Evaluate the model after each epoch
    rouge_scores = evaluate(t5_model, t5_tokenizer, lcsts_validation, rouge)
    print(f"Rouge-2 score on epoch {ep+1}:", rouge_scores)
```

如同上圖的程式碼所示，我的訓練過程如下:首先會遍歷每一個訓練週期（epoch）。然後在每個週期中，將模型設置為訓練模式。使用帶進度條（tqdm）的迴圈迭代訓練數據集。

在每次迭代中，我使用 optimizer.zero_grad() 清除舊的梯度。通過模型前向傳播計算損失（loss = t5_model(input_ids=inputs, labels=targets).loss）。

接著再調用 loss.backward() 來進行反向傳播和計算梯度。最後使用 optimizer.step() 來更新模型的權重。

Evaluation (5pts)

- ✓ Select evaluation metrics (BLEU, rouge, ...) and show the scores.

  我的 rouge score 如下:

```
'Rouge-L-P': 0.6806281416676265, 'Rouge-L-R': 0.6745061038113893, 'Rouge-L-F': 0.6745061038113893,
```

'Rouge-2-P': 0.026492651855262655, 'Rouge-2-R': 0.02629599785119526, 'Rouge-2-F': 0.02629599785119526}

根據查詢，ROUGE 是一種常用於自然語言處理中評估自動文本摘要和機器翻譯的工具，而 Rouge-2 則是衡量兩個連續字的重疊（bigrams）。各數值所代表的資訊如下:

Rouge-L-P (精確率):代表生成的摘要中，有 68.06%的單詞是正確地出現在參考摘要中。

Rouge-L-R (召回率):代表參考摘要中有 67.45%的單詞被生成摘要正確地覆蓋了。

Rouge-L-F (F1 分數):為精確率和召回率的調和平均，通常用來給出單一的性能指標。

Rouge-2-P (精確率):意味著在生成的摘要中，只有 2.65%的 bigrams 是正確出現在參考摘要中。

Rouge-2-R (召回率):表示在參考摘要的 bigrams 中，只有 2.63%被生成摘要覆蓋。

Rouge-2-F (F1 分數):反映了 bigrams 的總體匹配質量。

總而言之，從本次作業產生的這些數據可以看出，模型在處理較長的字串（如句子）時表現尚可，但在處理較精細的詞組層面（bigrams）時表現較為不佳。這可能代表模型在捕捉更細微的語言結構上有所欠缺。