GAI Project 2b 資訊 113 F74096027 吳瀚惟

1. Model

在本次 project 中，我選用的 pre-trained model 是從 huggingface call 下來的 ozcangundes/mt5-small-turkish-summarization，它是基於 Google 的 mT5-small(多語言 T5 小型 model)，利用 MLSUM 土耳其新聞數據集進行了微調，專門用在生成新聞文章的摘要。這個模型使用 Transformer 架構，具有大約三億個參數。它的優點是具有支援簡體中文的 tokens，以及可以生成土耳其文呈現的文本摘要，我基於模型的這個能力，嘗試將它微調成可以生成中文呈現的文本摘要，以符合本次作業需求。

至於 GPT-2，我採用 huggingface 上的 benjamin/gpt2-wechsel-chinese。這個模型是基於 GPT-2 架構的中文文本生成模型，它使用了 WECHSEL 方法來實現跨語言遷移，即通過利用多語言的靜態詞嵌入，tokenize 英文模型的字詞並嵌入遷移到中文的環境，這種方法是為了減少訓練大型語言模型所需的計算資源，並同時降低環境的影響。

關於 T5 和 GPT-2，兩個模型都是基於 Transformer 模型去改進的，都比最初的 Transformer 要強大很多。他們兩者最大的區別是 GPT-2 只有 Decoder，而 T5 同時具有 Encoder 和 Decoder。理論上，T5 應該比較擅長”對於給定輸入，產生對應輸出”相關的任務，例如：翻譯、文本摘要等。而 GPT-2 則比較擅長”自由創作”的相關任務，例如：文本生成。因此，根

據我對 T5 和 GPT-2 的理解，我推測 T5 在本次作業”中文文本摘要”的任務表現會比 GPT-2 來的更好。稍後，我也將嘗試以實驗數據佐證這個推測。

2. Dataset

T5 的資料處理方法：

(1) 首先載入 hugcyp/LCSTS dataset，並將其轉為 pandas dataframe。接著將訓練資料量縮減至 15000 筆，驗證資料量則控制在 1000 筆，進行適當的切割以減少訓練和驗證的時間。

(2) 使用 t5_tokenizer 對文本和摘要做 tokenize，添加 padding 和 truncation，以確保所有輸入文本和摘要的長度統一。這生成了適合模型訓練的輸入格式。

(3) 在 get_tensor 函式中，model inputs 為 tokenize 過後的文本，而 model outputs 則 tokenize 過後的摘要。之後他們被進一步的處理並載入，以供模型訓練使用。

GPT-2 的資料處理方法：

(1) 首先加入 pad token，再調整模型嵌入層的大小，以包含<pad>。

(2) 載入 hugcyp/LCSTS dataset，並將其轉為 pandas dataframe。接著將訓練資料量縮減至 15000 筆，驗證資料量則控制在 1000 筆，進行適當的切割以減少訓練和驗證的時間。

(3) 新增一個 text_summary 格式，它是將 text 與 summary 整合在一起，兩者之間用<eos>做連結，此外在結尾也手動加上<eos>。

(4) 使用 gpt2_tokenizer 對文本摘要(結合格式)和摘要做 tokenize，添加 padding 和 truncation，以確保所有輸入都有一致的最大長度。這生成了適合模型訓練的輸入格式。

(5) 檢查 label 長度是否與輸入長度一致，並進行必要的填充，確保訓練過程不會出現錯誤。

(6) 在 get_tensor 函式中，model inputs 為 tokenize 過後的文本摘要以及 attention mask(用以在訓練時省略 padding)，而 model outputs 則 tokenize 過後的摘要。之後他們被進一步的處理並載入，以供模型訓練使用。

3. Train (Finetune)

在訓練 T5 時，我使用的 epochs=2, lr=1e-4, optimizer=AdamW, batch size=300。訓練的過程大致如下：在每個 epoch 中遍歷 train dataset 的 batch -> optimizer 歸零 -> 模型接受 inputs(text)和 labels(summary)，計算 loss -> 後向傳播，根據 loss 計算梯度 -> 應用梯度更新模型參數 -> 顯示當前 batch 的 loss -> 在每個 epoch 結束後評估模型性能，print rouge 結果。

在訓練 GPT-2 時，我使用的 epochs=2, lr=1e-3, optimizer=AdamW, batch size=200。訓練的過程大致如下 : 在每個 epoch 中遍歷 train dataset 的 batch -> optimizer 歸零 -> 模型接受 inputs(text_summary)，計算 loss，並輸入 mask(用以在訓練時省略 padding) -> 後向傳播，根據 loss 計算梯度 -> 應用梯度更新模型參數 -> 顯示當前 batch 的 loss -> 在每個 epoch 結束後評估模型性能，print rouge 結果。

4. Evaluation

我選擇的 evaluation metrics 有 rouge-1, rouge-2, rouge-L, 以及 rouge-Lsum。接下來，我將簡單說明四個指標各別的意義，再對比 T5 與 GPT-2 兩個模型的評估結果。

Rouge-1: 測量摘要中的單詞與參考摘要中單詞的重疊程度，這個指標主要著重在評估提取的內容是否包含了原文的主要概念。

Rouge-2: 測量摘要中兩個連續單詞與參考摘要的重疊程度，這個指標更能反映語句之間的連貫性與結構。

Rouge-L & Rouge-Lsum: 這兩個指標是基於 LCS(最長公共子序列)，能夠評估生成摘要的流暢性和文法結構，其中 Rouge-Lsum 是對整個摘要而不只是句子進行評估的版本。

在這四個指標中，我分析並 print 出了他們各自於最佳狀態、正常狀態、

以及最差狀態的 precision, recall, 和 F1。以下，我將以**正常狀態的 F1 分數**作為主要的評估依據(因為它同時考慮了 precision 和 recall，算是一個綜合分數)，來對比 T5 和 GPT-2 的評估結果。

T5 評估結果：

Evaluation results:

rouge1: F1: 0.0822

rouge2: F1: 0.0102

rougeL: F1: 0.0819

rougeLsum: F1: 0.0824

GPT-2 評估結果：

Evaluation results:

rouge1: F1: 0.0742

rouge2: F1: 0.0086

rougeL: F1: 0.0747

rougeLsum: F1: 0.0758

從結果可以看出 **T5 在四項評估指標數值都大於 GPT-2**，這也證明了第一點我提到過的推測：T5 確實在中文文本摘要的任務中表現的較 GPT-2 來得理想。