mT5 model for chinese news title prediction

Contents

Q1: Model (2%)

1. Model (1%)

本次作業使用模型為 Google 發表的語言模型 Multilingual T5，該架構為 Encoder-decoder(如圖 1)，mT5 – small 的參數量大約為 300M。

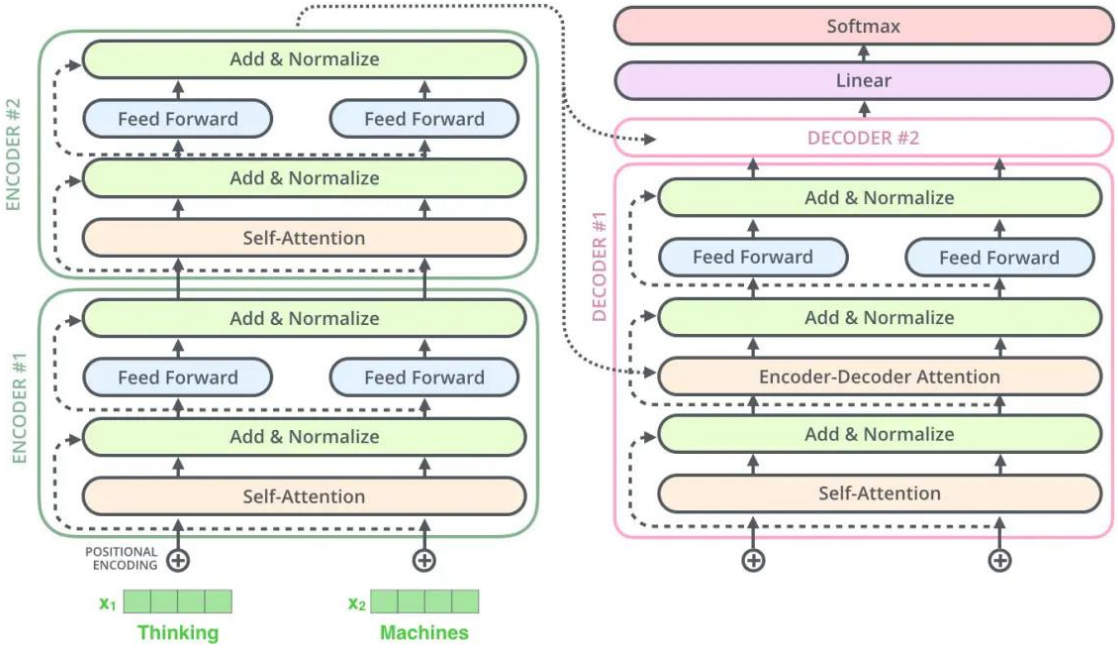

圖 1、T5 encoder-decoder 模型架構(參考資料: https://www.jianshu.com/p/627d4643f7a7)

mT5 預訓練於 mC4 的資料集上，該資料集大小在 TensorFlow 上為 38.49 TiB，共有 101 種語言，由 Common Crawl 在網頁上蒐集 71 個月而成。製作 mT5 的團隊針對 mC4 資料做了清洗：他們去除不好的字眼；利用 Compact Language Detector v3 (CLD3)模型偵測資料集語言，並去除模型判定不足 70%信心水準的資料。mT5 預訓練方法分為兩種：第一種方式和 BERT 類似，將某些單字去除，並要模型預測挖除的字詞；另一種是 NLP 監督任務，例如：情感分類、主題分類等。

2. Preprocessing (1%)

在資料處理上我沒有剃除任何資料，所有 train.jsonl 中的資料做為模型訓練資料，public.jsonl 作為驗證資料。我使用 MT5Tokenizer 做為本次任務的 tokenizer，在 train.jsonl 及 public.jsonl 檔案中 'maintext' 和 'title' 的內容文字資料需要做 tokenization。

MT5Tokenizer 採用 SentencePiece 的方法，基本上是將語言拆成 subword units，SentencePiece 將句子視為 Unicode 字符序列，沒有依賴於語言的邏輯，該 tokenizer 有三種特別的 token："</s>"作為一個序列的結束；"<unk>"用於未

知的 token；"<pad>"用於做 padding。

資料做完 tokenization 後，出現兩個向量：'input_ids'及'attention_mask'，input_ids 為輸入的每個文字轉換成字典中的 ID； attention_mask 中只會出現 0 或 1，1 代表原始句子的 token，0 代表額外 padding 的 token。我將輸入模型的資料長度限制最長為 256，輸出限制最長為 64。

Q2: Training (2%)

1. Hyperparameter (1%)

- ✓ Optimization algorithm: AdamW。
- ✓ Learning rate: $5*10^{-5}$。
- ✓ Effective Batch size: 1(batch size: 1, gradient accumulation steps: 1)。

Effective Batch size 我嘗試分別調 1、8 和 64，試驗下來後發現 **1 的訓練效果最好**，64 的訓練效果最差。Learning rate 我嘗試調 $1*10^{-4}$ 及 $5*10^{-5}$，訓練結果差異不大，最終**使用 $5*10^{-5}$ 做模型訓練**。

2. Learning Curves (1%)

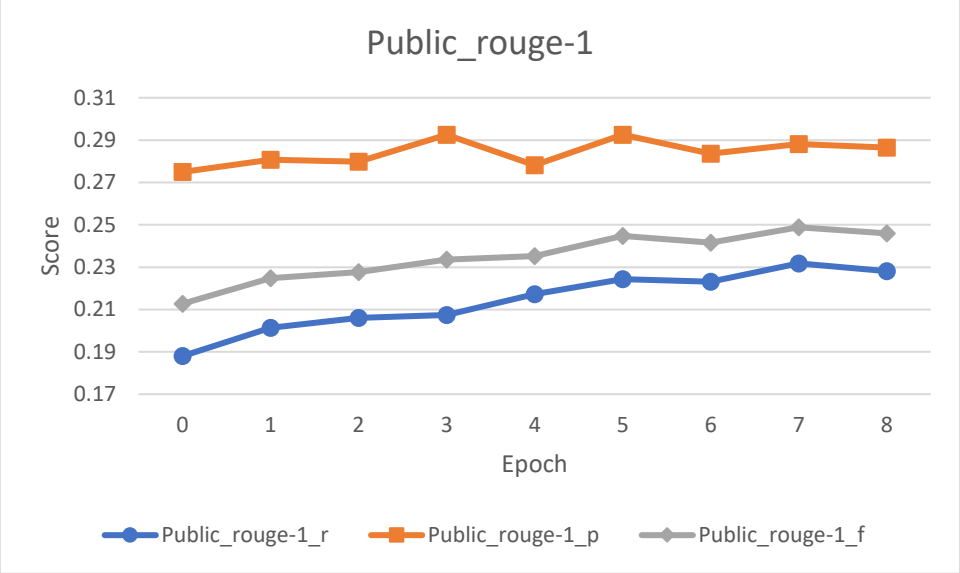

圖 2、rouge-1 分數於 Public dataset

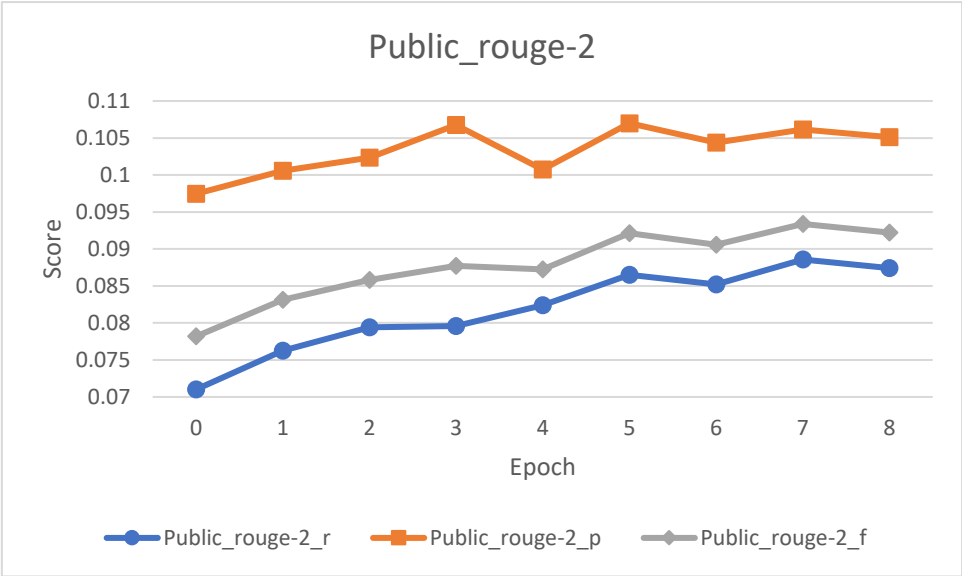

圖 3、rouge-2 分數於 Public dataset

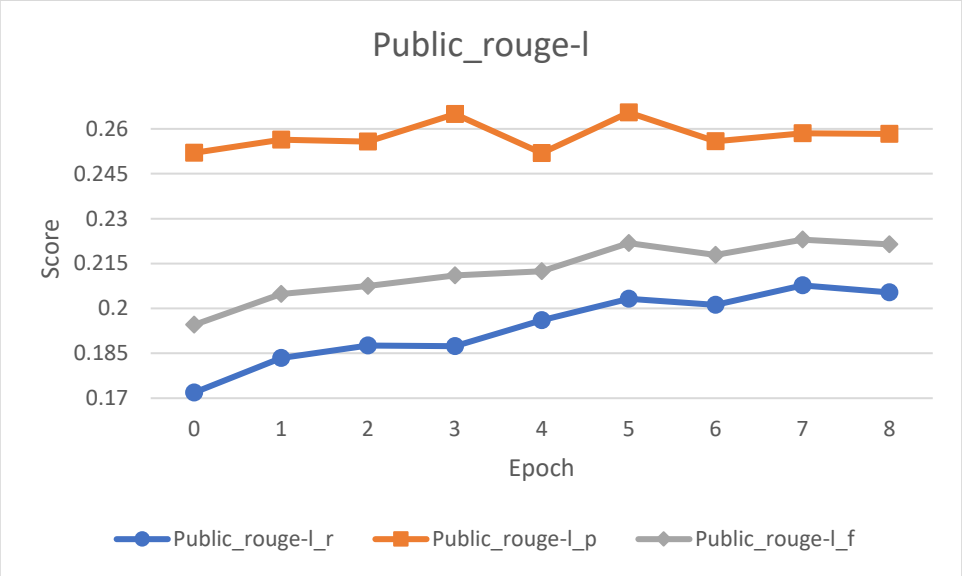

圖 4、rouge-l 分數於 Public dataset

由圖 2、圖 3、圖 4 可知模型預測 Public dataset 三個指標的 f score 有穩定上升，第 8 個 epoch 有稍微往下降一點，但下降幅度不高，繼續往下訓練仍有機會提升表現。

Q3: Generation Strategies(6%)

1. Strategies (2%)

- **Greedy**

  Greedy search 是指在每個 t 時刻選擇下一個詞時，會根據 $w_t = argmax\ P(w|w_{1:t-1})$ 公式選擇機率最高的詞。用此方式生的句子不會考慮到前後詞的機率狀況。

- **Beam Search**

  Beam search 利用參數 num_beams 設定，在每個時間點記錄機率最高的前幾個路徑，在下一個時間點可以有多個路徑同時搜索，最後選擇多個路徑中相成機率最高的路徑。此方法可以避免錯過隱藏的高機率詞彙。當 num_beams 設定較小時，產生的句子容易專注在主題上，但敘述會較常產生錯誤；當 num_beams 設定較大時，產生的句子敘述較正確，但容易離題。

- **Top-k Sampling**

  Sampling 是指從已有詞序列的條件機率分佈中隨機選擇下一個詞彙，數學式為：$w_t \sim P(w|w_{1:t-1})$。Top-K sampling 是選出前 K 個條件機率最高的詞，並將條件機率重新分配到這前 K 個詞中。當 K 設定越大時，產生的句子傾向越多元但越可能產生錯誤的詞彙；當 K 設定越小時，產生的句子傾向越安全但越容易產生較一般的詞彙。

- **Top-p Sampling**

  Top-p sampling 是從候選的詞彙中選擇個數最少且累計機率超過機率 p 的詞彙集合，這麼做的好處是選擇的詞彙數量會根據詞彙的機率分布決定，當詞彙的條件機率分佈分散時，代表下一個詞不易預測，詞彙集合的範圍大，可從越多詞彙中做 sampling；當詞彙的條件機率分佈集中時，代表下一個詞容易預測，詞彙集合的範圍小，容易 sampling 到條件機率大的詞。

- **Temperature**

  Temperature 參數(τ)用於調整 Softmax 函式，數學式為： $p(w_t) = \frac{e^{\frac{s_w}{\tau}}}{\sum_{w' \in V} e^{\frac{s_{w'}}{\tau}}}$，加入此參數可調整詞彙經過 Softmax 後的條件機率分佈：當τ越高，詞彙的機率分佈會傾向越平滑；當τ越低，詞彙的機率分佈會傾向越尖銳。參數(τ)的調整可以適當增加少樣本類別被 sample 的機率，減少多樣本類別被 sample 的機率。

2. Hyperparameters (4%)

表 1、Settings of each strategy and their tw_rouge f score on the public dataset

Settings of each strategy	rouge-1_f	rouge-2_f	rouge-l_f
Greedy	0.2508	0.0944	0.2256
num_beams = 3	0.2631	0.1054	0.2365
num_beams = 8	0.2637	0.1074	0.2366
num_beams = 13	0.2640	0.1078	0.2366
num_beams = 18	0.2647	0.1083	0.2369

num_beams = 23	0.2639	0.1080	0.2361
top_k = 10	0.2266	0.0770	0.2005
top_k = 25	0.2111	0.0712	0.1875
top_p = 0.5	0.2410	0.0881	0.2149
top_p = 0.9	0.2188	0.0750	0.1950
top_p = 0.5 **temperature = 1.5**	0.2235	0.0781	0.1982
top_p = 0.5 **temperature = 0.5**	0.2496	0.0933	0.2239

由表 1 可知，採用 Sampling 策略生成的標題在 tw_rouge 上表現都沒有比 Greedy 還要好，我做出最好的 Sampling 策略是 top_p = 0.5、temperature = 0.5；最差的是 top_k = 25。使用 Beam Search 產生的標題在 tw_rouge 上有比 Greedy 高分，因此我多試了幾組 num_beams 的設定，最終 num_beams = 18 表現最好。本次的任務對於使用 Sampling 生成標題並不適合，且 Sampling 相關參數設置越小，Sampling 可選擇的詞彙越少，傾向 Greedy 方式生成標題，表現會越好，隨機性越大反而降低模型生成標題的表現。

我最終只設定 **num_beams = 18** 做為我模型的 generation strategy。