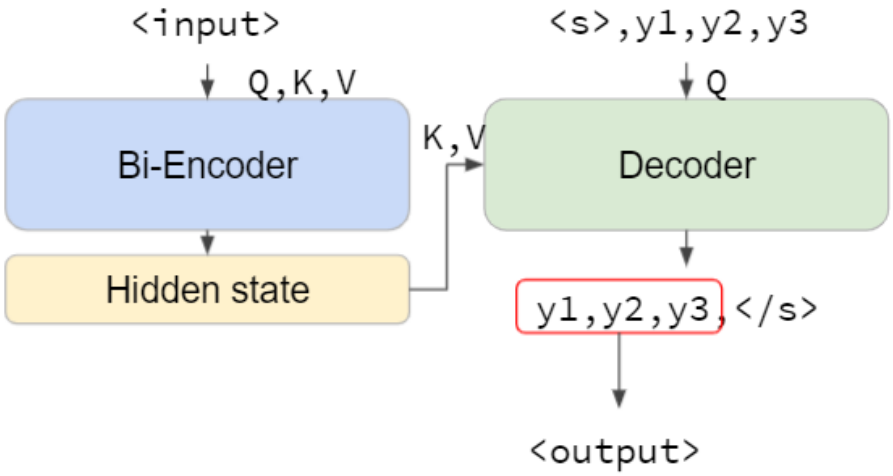
Applied Deep Learning Homework 3 report

M10915045 施信宏

Q1: Model

1.1 model :

採用 google 的 mt5-small model，可處理多國語言的 T5 模型。T5 剛推出便在 NLG 領域造成轟動，比當時其他的模型來的更加強大，可用於機器翻譯、文章總結等等的任務。架構如助教的作業投影片的圖:



前者的 Bi-Encoder 如 BERT 架構，但比 BERT 多了一層 Decoder，為 seq2seq 的模型。並且採用 sentencepiece 編碼，也打破了 BERT 的最大輸入限制，並可以輸入超過 512 個字的向量，只要硬體資源是可以承受的情況。

1.2 Preprocessing:

先下載 jsonlines 及 sentencepiece 套件，讀取 jsonl 檔案和使用 T5 模型。接著讀取 train,public jsonl 中的 maintext , title, id，以 transformers datasets 的形式讀取，在處理 maintext 中，將換行符號'\n'皆給去除，只保留原文的部分，訓練階段，將 maintext 前面補上 "summarize :"，之所以加上此 prefix，是 T5 model 中建議使用，可以分辨此任務是哪一類，將翻譯、總結等任務切分，使訓練成效更佳。接著經過 tokenizer 的處理，轉成 T5 model 可以輸入的向量，而在這便不關注<pad>的部分，若該單字為 <pad>，在計算 loss 時會跳過，讓學習效果提升。

Q2: Training

2.1 Hyperparameter:

Learning rate = 5e-5
Epoch = 10
Batch_size = 8
Input_seq_len = 512
Target_len = 128

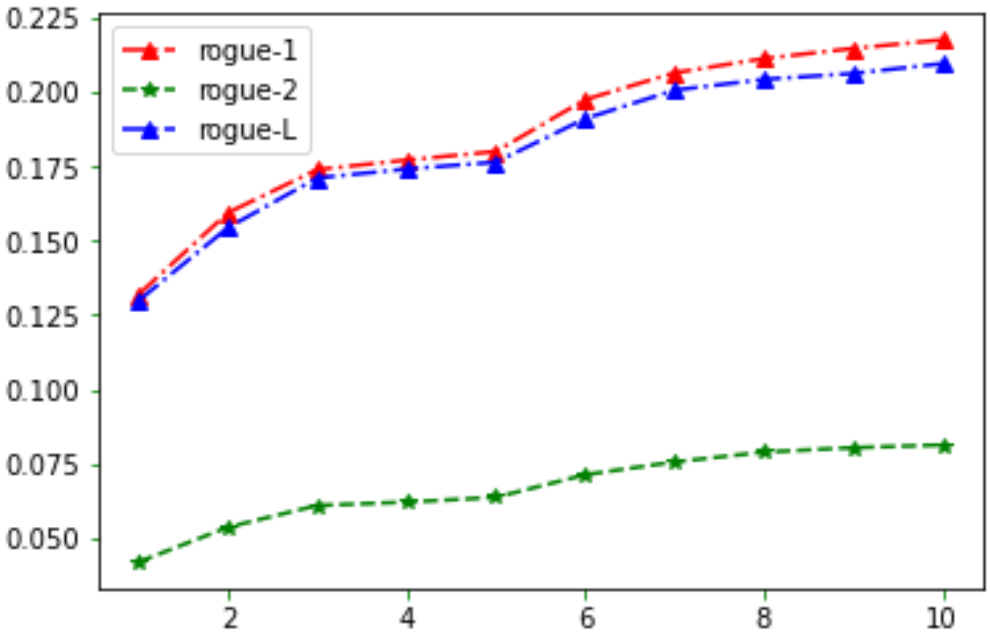
並有採用 learningRate_scheduler，無採用 fp16 進行訓練。

決定將文章長度加長，較能輸入整篇的文章，512 的長度較能符合情況，並將目標輸出上限設定為 128，但以標題長度來說，64 應以可處理。

訓練環境為 google colab ，GPU : Tesla T4

2.2 Learning Curves:

以 public.jsonl 作為驗證集，產生文字策略為 Greedy，所以會稍稍低於 baseline，並觀察 Rouge 中的 F1-score。總共訓練 10 個 epoch。



Q3: Generation Strategies

- Greedy:

  輸出每個時段 t 中，機率值最大的單字，便根據輸出的單字繼續找尋下一個單字，但只關注到局部最佳，沒有考慮到全部，生成文字不一定是最佳解。

- **Beam Search:**

  相比 greedy，考慮的較周全，每個時間點 t，選取 beam 個數量的單字，繼續往下找尋，會找尋到前面機率值不高，但後續生成結果佳的內容。

- **Temperature**

  使 P(w|w1:t−1) 分布較明顯(增加高機率的 likelihood 及降低低機率的 likelihood)，如同 softmax 的效果。

  公式:

$$P(w_t) = \frac{e^{s_w/\tau}}{\sum_{w\prime \in V} e^{s_{w\prime}/\tau}}$$

  temperature hyperparameter τ

  Higher temperature : $P(wt)$ becomes more uniform →more diversity

  Lower temperature: $P(wt)$ becomes more spiky →less diversity

- **Top-k Sampling**

  Sampling: wt~P(w|w1:t−1)，隨機選取下一個單字(from a distribution)，top-k sampling 則是將選擇機率單字固定在前 K 個，從前 K 個單字內隨機選取輸出。

- **Top-p Sampling**

  與 top-K 不同，以機率值總和去選擇單字，找尋到機率總和為 p 的所有單字，從這些單字中去選取輸出。0 < top_p < 1

每個 Strategies 的實驗結果: 每個 Strategies 的評價在最後。

Greedy:


```
{
  "rouge-1": {
    "f": 0.2186704099715676,
    "p": 0.23946731362745238,
    "r": 0.21548042057611427
  },
  "rouge-2": {
    "f": 0.08185972333876676,
    "p": 0.0893114153734875,
    "r": 0.0814023189757906
  },
  "rouge-l": {
    "f": 0.20908279949526107,
    "p": 0.24657930153727764,
    "r": 0.19521725758617822
  }
```


Beam search:

Beam = 3 Beam = 5

```
"rouge-1": {
  "f": 0.2352623479931667,
  "p": 0.24600088059353833,
  "r": 0.24071919586004006
},
"rouge-2": {
  "f": 0.09412758193691852,
  "p": 0.09840442699036714,
  "r": 0.09681109340699896
},
"rouge-l": {
  "f": 0.2236907579230562,
  "p": 0.24994251148288923,
  "r": 0.2173568189574954
}
```

```
"rouge-1": {
  "f": 0.23482327656137536,
  "p": 0.24272088869365843,
  "r": 0.24288286023970684
},
"rouge-2": {
  "f": 0.09519254324724821,
  "p": 0.09851131207973937,
  "r": 0.09888762897790421
},
"rouge-l": {
  "f": 0.22257874117745677,
  "p": 0.24537066420138084,
  "r": 0.21876920441542327
}
```


Beam 較大的情況，在 rouge-2、rogue-l 上表現較好，推估 beam 較大能生成比較符合訓練集原始的標題，整句的意思學習較好。

Top-K sampling

K = 30


```
"rouge-1": {
  "f": 0.1785368777359569,
  "p": 0.18688529135952572,
  "r": 0.1829281131286481
},
"rouge-2": {
  "f": 0.05739974844642312,
  "p": 0.059518479407187574,
  "r": 0.05955347919234715
},
"rouge-l": {
  "f": 0.1648420377777561,
  "p": 0.1777599894873206,
  "r": 0.16376247014488227
}
```


K = 50


```
"rouge-1": {
  "f": 0.1732205689086487,
  "p": 0.18055650164689052,
  "r": 0.17899149374076792
},
"rouge-2": {
  "f": 0.05477344246547213,
  "p": 0.056815790528414616,
  "r": 0.05728658499967056
},
"rouge-l": {
  "f": 0.15954818858900188,
  "p": 0.17107329226625811,
  "r": 0.16003913504156847
}
```


K 值比較小時，分數較高，從最高機率分布的前 K 個取一個作為時間 t 的輸出，k 值較小的情況，較能選出最合適的答案，雖分數不高，但文法上比 greedy 方法有判讀性。

Top-p sampling

p = 0.8


```
"rouge-1": {
  "f": 0.1939014631570233,
  "p": 0.20529212626852456,
  "r": 0.19636730201818972
},
"rouge-2": {
  "f": 0.06623475550685383,
  "p": 0.06952747861151046,
  "r": 0.06790436294129361
},
"rouge-l": {
  "f": 0.17945536088226477,
  "p": 0.19654379803731267,
  "r": 0.1758112352456356
}
```

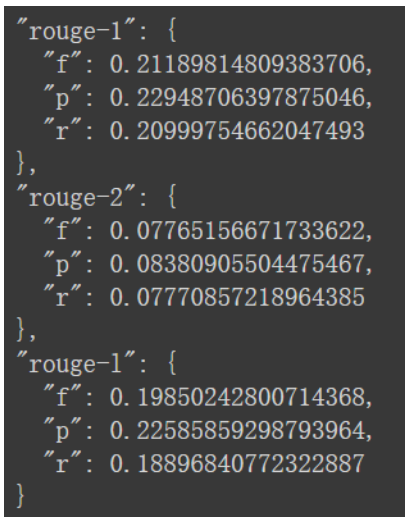

p = 0.9


```
{
  "rouge-1": {
    "f": 0.19344908300121752,
    "p": 0.20487731350112673,
    "r": 0.19542475411362184
  },
  "rouge-2": {
    "f": 0.06533706591924832,
    "p": 0.06913195902588737,
    "r": 0.06642971423965546
  },
  "rouge-l": {
    "f": 0.17887542925163377,
    "p": 0.19623509317590382,
    "r": 0.1749240434057731
  }
```

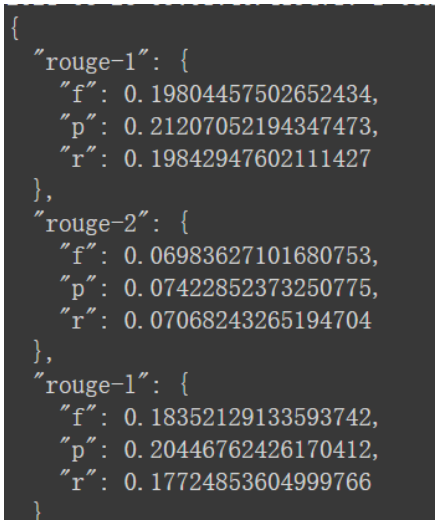
如同 top-K 的解釋，p 值比較小時，分數較高，越小的機率總和表示能選擇的字將會更少，能選到時間 t 較為合適的詞作為輸出。

Temperature

0.5



0.7



Temperature 越小，其輸出越接近 greedy，故 temperature 較高分數則下降。

可以發現 beam search 在 rogue 分數上表現較佳，可以考慮多種路徑的最佳情況，生成的文字相較 greedy 來的合理，且重複的無意義文字也較少。至於 sampling 的方法應該需要搭配起來使用，如將 temperature 搭配 top-k 或 top-p，應能生成較佳的文句，但在分數上表現就不得而知，故最終是使用 beam-search , beam = 5 的輸出作為本次作業的結果。