ADL hw3 Report

R10546017 朱瑋民

Q1: Model

1. Training 的 model 是使用 mT5，pretrained weight 是 google/mt5-small，model 架構如下：


```
 "architectures": [
      "MT5ForConditionalGeneration"
  ],
  "d_ff": 1024,
  "d_kv": 64,
  "d_model": 512,
  "decoder_start_token_id": 0,
  "dropout_rate": 0.1,
  "eos_token_id": 1,
  "feed_forward_proj": "gated-gelu",
  "initializer_factor": 1.0,
  "is_encoder_decoder": true,
  "layer_norm_epsilon": 1e-06,
  "model_type": "mt5",
```

"num_decoder_layers": 8,

"num_heads": 6,

"num_layers": 8,

"pad_token_id": 0,

"relative_attention_num_buckets": 32,

"tie_word_embeddings": false,

"tokenizer_class": "T5Tokenizer",

"transformers_version": "4.11.3",

"use_cache": true,

"vocab_size": 250112

}

mT5 是一個可以用在多國語言的 text to text 的 transfer transformer，不同語言但是語意相似的句子可以 fine tune 成類似的 embedding，可以做到在不同語言之間的 transfer learning， 因為是一個 Encoder Decoder 的架構，在 text summarization 這個 case 輸入的 input 為將原始文章經由 tokenized 後的 input _ id 跟 attention mask，放入 encoder，在 decoder 端用 teacher forcing 的方式來產生 output，並與正確的 title 做比對來算 loss 做 backpropagation，最後將 model 的 output 做 decode 便可以得到最終的 output title。

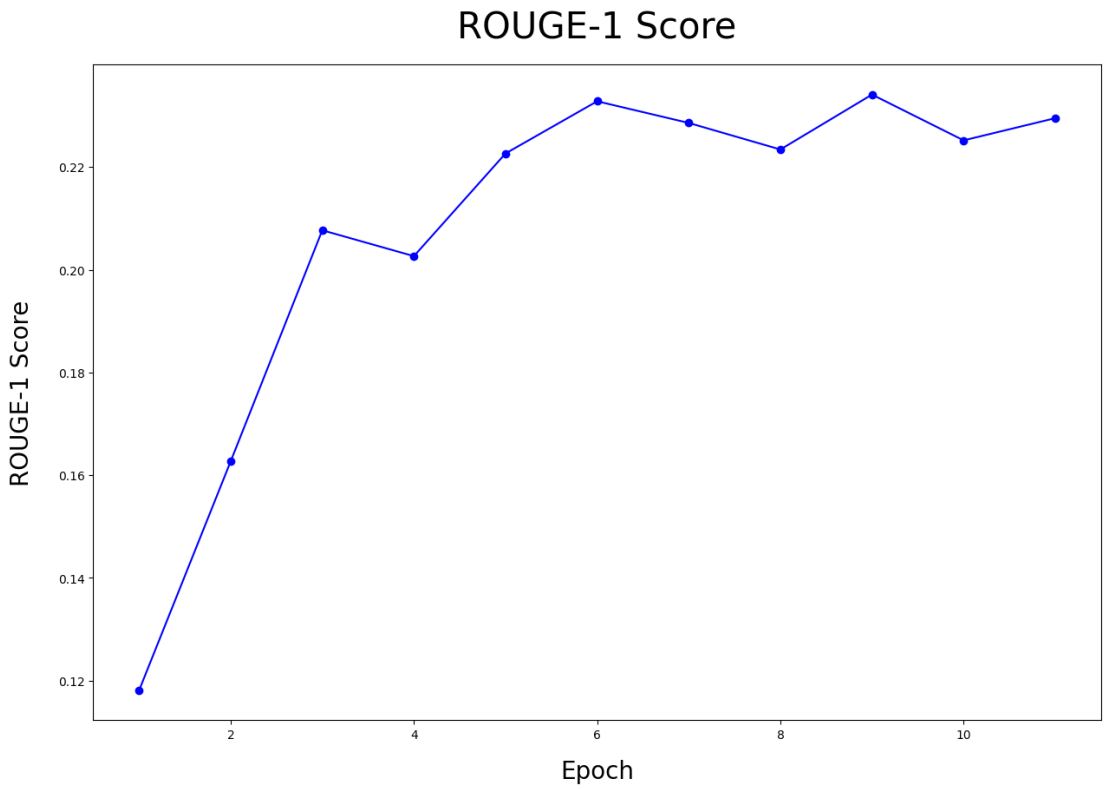
2. 用 T5 的 tokenizer，pretrained 為 google/mt5-small，max source length 設為 1280，代表 input 可以取到最多 1280 個字，而 tokenize 為將字詞對應到 pretrained 的 vocabulary 的 250112 個字之中，來代表 input 的 id，並將 label 的 tokenized 做 padding 到跟 label 的最長（max length）一樣長，padding 的值為-100，來做到 ignore padding 所造成的 loss。

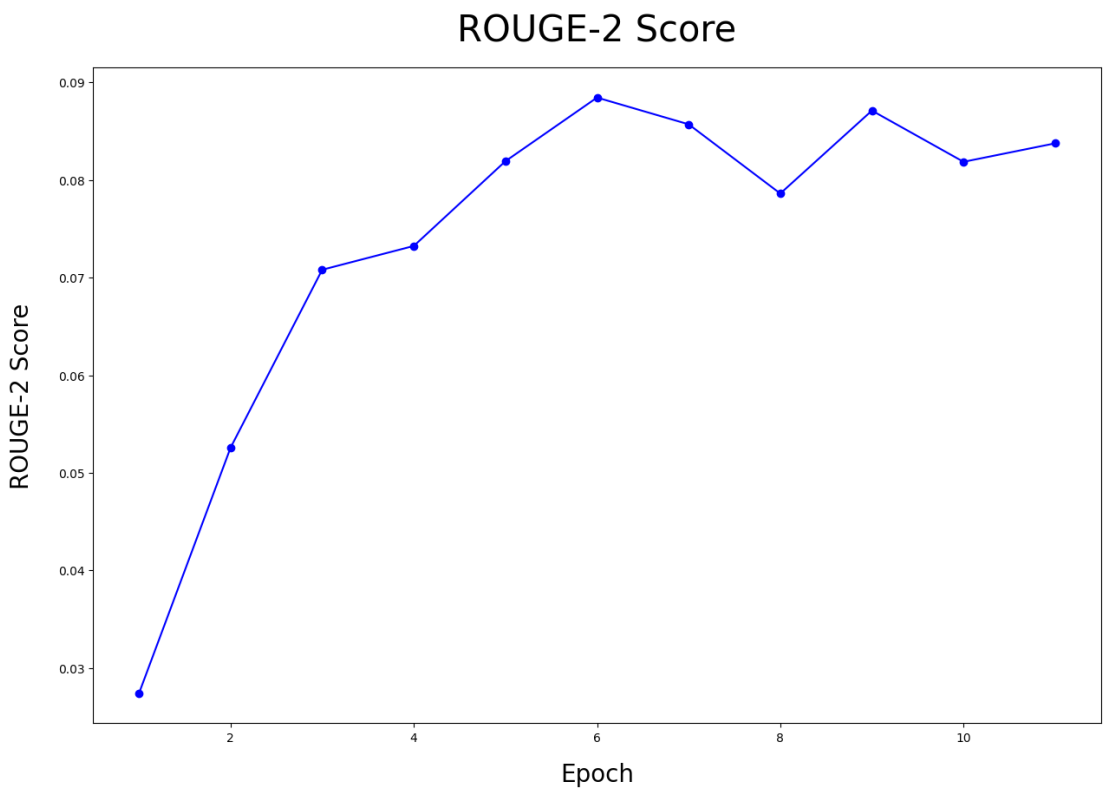
Q2: Training

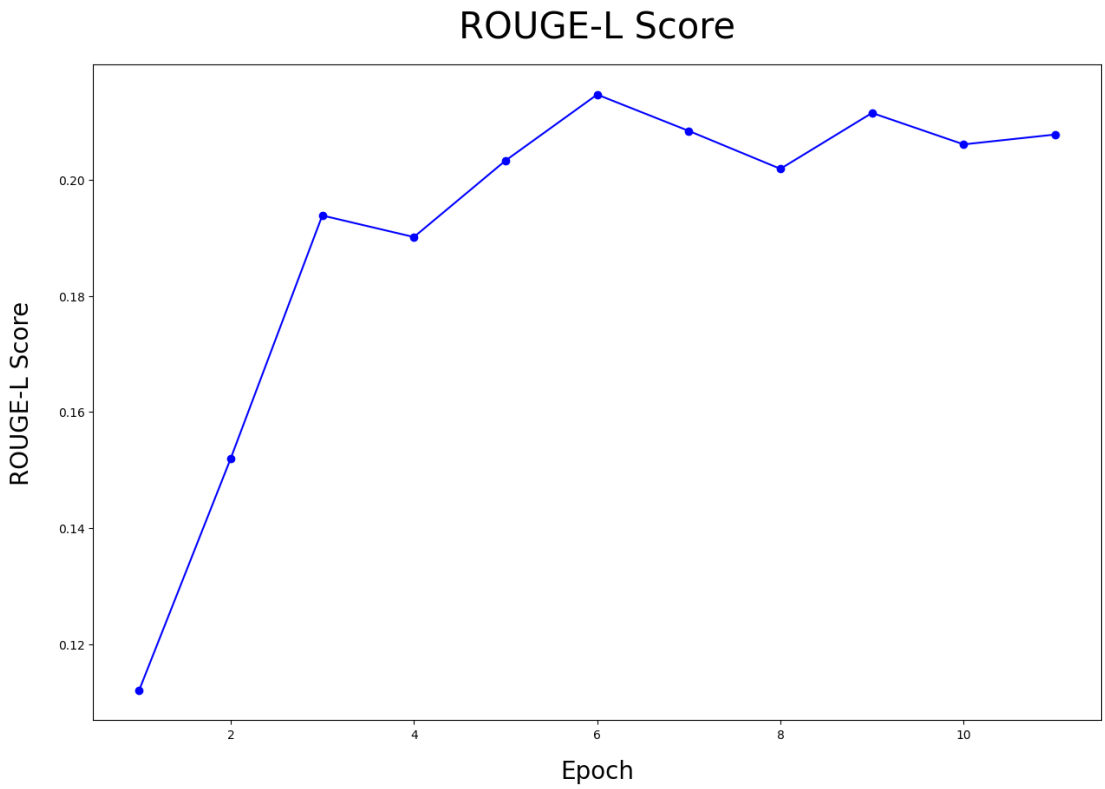
1. 訓練的 batch size 設為 1，max source length 設為 1280，train 在 12 個 epoch 上面，有取 5% 的 data 來做 validation，validation 使用的 metric 為 ROUGE ，用 ROUGE 來看每個 epoch 的好壞，optimizer 使用的是 AdaFactor，具有省內存和自適應學習率的特性，loss function 為 Crossentropy loss。

2.

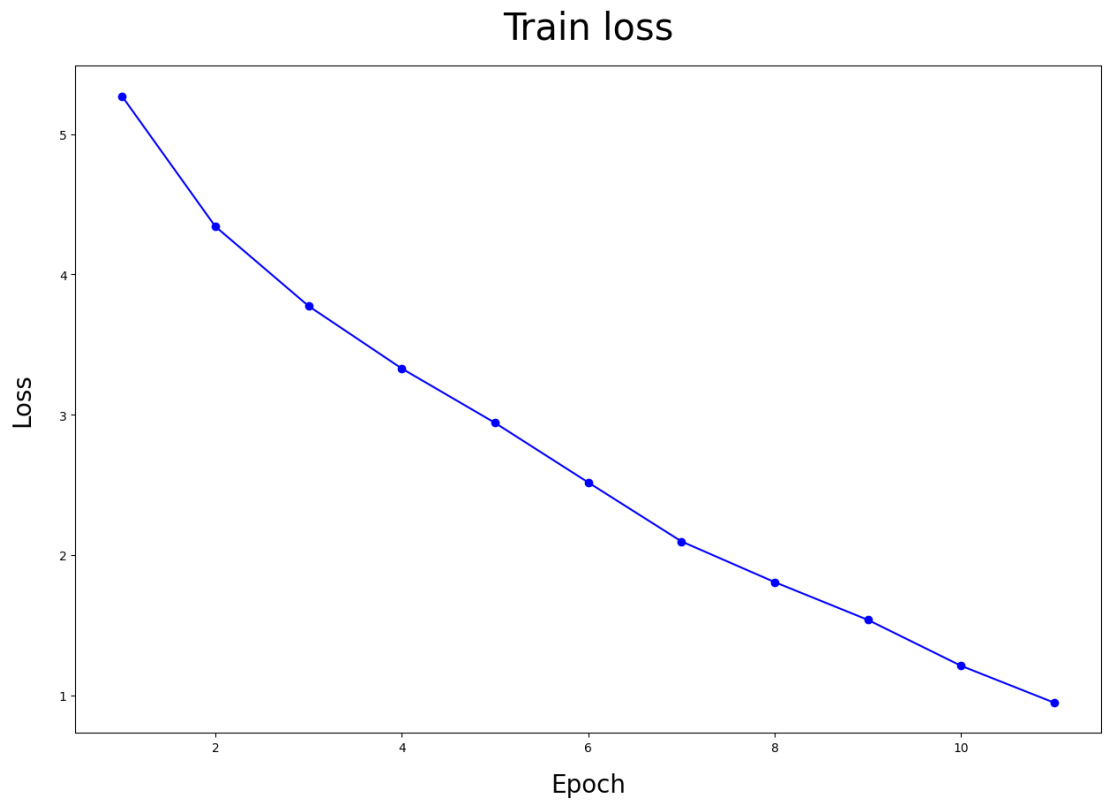
用 5% 的資料來當作 validation set，在每個 Epoch train 完後用 ROUGE Score 來看在 validation 的 ROUGE score，紀錄 11 個 epoch 的結果，如下所示：


ROUGE-1 Score
ROUGE-1 Score
0.22
0.20
0.18
0.16
0.14
0.12
2
4
6
8
10
Epoch


ROUGE-2 Score
ROUGE-2 Score
0.09
0.08
0.07
0.06
0.05
0.04
0.03
2
4
6
8
10
Epoch



Loss 的 每個 epoch 的值變化如下：



Q3: Generation Strategies

（a）

1. **Greedy**：針對 output 的機率分佈，取機率最大的來當作 decode 出來的答案並作為下一個 input，因為不是針對整個 sequence 做 optimization，只有對當前的字詞做 optimization，且因為 complexity 複雜沒辦法找到 global optimal。

2. **Beam search**: 每個時間點都去保留 k 個機率最大的 sequence 做保留（從原來的 greedy 追蹤一條變現在追蹤多條），Small beam size，越接近 greedy，產生的東西可能會有 ungrammtical 跟 unnatural 的問題，而 Large beam size，計算複雜，在特定 cases (ex: chatting 的 case) 不會希望一直講一樣 的東西，一直重複的問題。

   而 Sampling （Sampling-Based Decoding）是用隨機的方式從原來的 distribution 選擇下一個字詞，而不是用 Argmax 的方式找最大的，雖然後面被 sample 出來的字會跟當前跑出來的有關係，有可能會因此被影響而離題。

3. **Top-k Sampling**：只選 top k 個機率比較高的字詞來做 sample，設定 1 為 greedy，設定全部則為 pure sampling，完全隨機從從原來的

distribution 選擇下一個字詞，k 大得到較多樣的 output，k 小得到較一般的 output。

4. **Top-p Sampling（Nucleus Sampling）**: given 前面已經產生的字，去看下一個字出現的機率，如果累加起來的機率大於 P，就把他放進一個 subset 裡面，根據一個給定的機率值 P，來決定現在可以做 sampling 的 set，也就是可以動態的去改變 Top-k sampling 的 k 的多寡。

5. **Temperature：** 將 Softmax 的 output 的 Probability Distribution 加上一個 temperature 的 hyperameter 在 exponential 指數的地方，如下圖所示：

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

t 大則較為 uniform，較平，最高分的會比較平緩一點，差異不會那麼大，diversity 較大，相反的 t 小，diversity 較小，差異較大

（b）

Greedy 的結果如下：

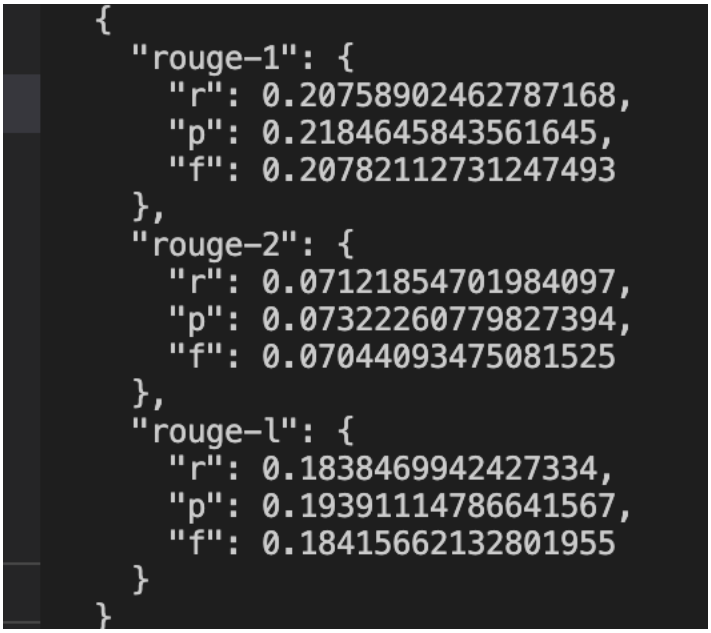

```
{
  "rouge-1": {
    "r": 0.2444645926070898,
    "p": 0.28017110815064383,
    "f": 0.2546006515963482
  },
  "rouge-2": {
    "r": 0.09260744037546884,
    "p": 0.10157469430839579,
    "f": 0.09442420627386743
  },
  "rouge-l": {
    "r": 0.21838051917487583,
    "p": 0.2503634619873717,
    "f": 0.22736642887083464
  }
}
```


Beam search　將保留的 k 設為 6 (num_beams = 6)的結果如下：


```
{
  "rouge-1": {
    "r": 0.25439268478416666,
    "p": 0.29426751801869355,
    "f": 0.266442255779271
  },
  "rouge-2": {
    "r": 0.10277481449793485,
    "p": 0.11662326844268914,
    "f": 0.1066291513729974
  },
  "rouge-l": {
    "r": 0.22739399382109407,
    "p": 0.2633873663605968,
    "f": 0.23821420652966052
  }
}
```

Top-k Sampling，top k 的 k 設為 40 :

Top-p Sampling (Nucleus Sampling) 將機率 P 值設為 0.92 ：



可以發現 beam search 的結果比 Top-p Sampling 跟 Top-k Sampling 都還要來得好，蠻意外的結果，也可能是因為參數的設定所造成的結果。