## ADL HW3 Report

### Q1

1. Model architecture

本次使用 transformers 套件中的 MT5ForConditionalGeneration(mt5-small)來作為執行 summarization 的模型，其 config 如下：


```
{
  "_name_or_path": "google/mt5-small",
  "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_max_distance": 128,
  "relative_attention_num_buckets": 32,
  "tie_word_embeddings": false,
  "tokenizer_class": "T5Tokenizer",
  "torch_dtype": "float32",
  "transformers_version": "4.18.0",
  "use_cache": true,
  "vocab_size": 250112
}
```


此模型是 google 提出的 seq2seq 的 pretrained model，為 encoder 和 decoder 的組合，可以以文字訊息(經過 tokenize.encode 處理成 token tensor)當作輸入，經過如同 transformer 內部的 attention 等機制後，輸出另一個 sequence。T5 model 可以透過開頭提示詞(prompt)來 specify 任務內容，而此次作業的輸出為摘要。

2. Pre-processing

在將資料餵進模型前，要先對其進行預處理。我參考 huggingface 提供的 pre-process 作法，將每個文章開頭都先加上”summarize: ”這個起始 token，再透過 T5tokenizer 進行 tokenize。另外，網路上有看到一些參考資料會將\n、\r、空格等符號移除，但我認為這些字元對內容影響不大，故未特別做 cleaning。

而至於 tokenizer 的部分，我使用的是 T5Tokenizer，並且設定文本最大長度(max_input_length)為 512、摘要最大長度(max_target_length)為 128，最後做 padding 到最大長度為止。

**Q2**

1. Hyperparameter

使用 huggingface 的 Trainer 來做訓練，hyperparameter 如下：

Learning rate = 2e-5

Batch size = 8

weight decay = 0.01

gradient accumulation step = 2

num_epoch = 20

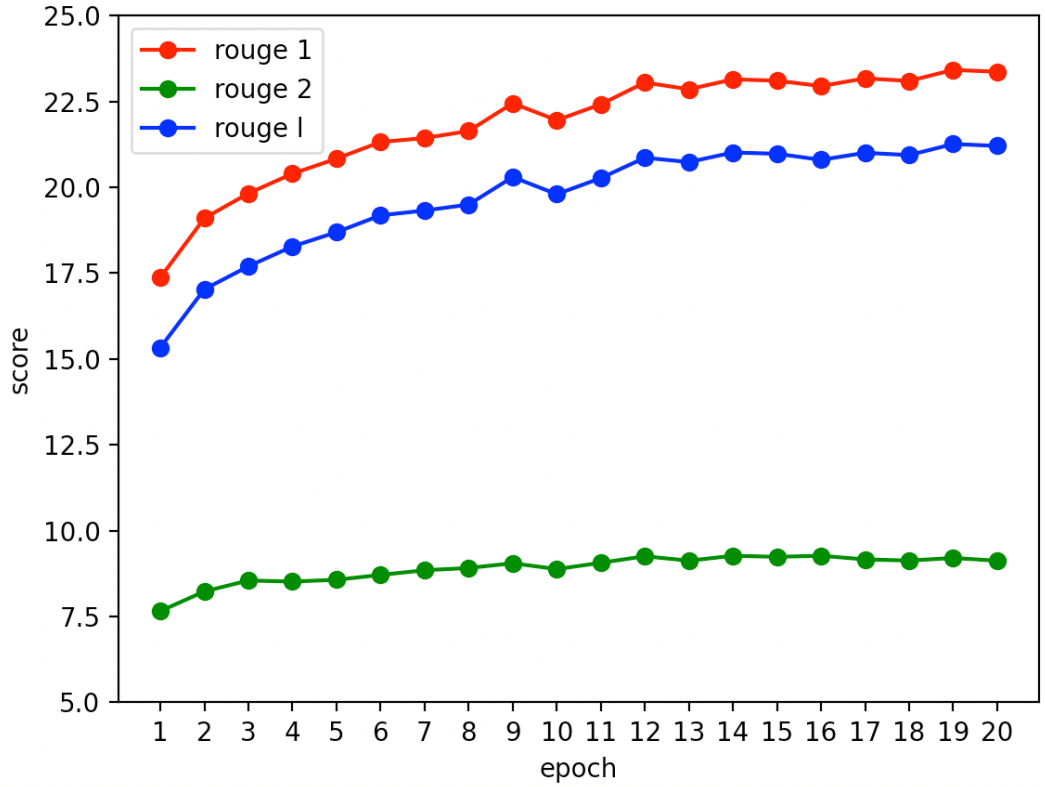
optimizer = adamw

criterion = crossentropy

對於 batch_size 的選擇是根據 gpu ram 的量來做選擇的(8*2 的大小在 colab 上算足夠)，而 num epoch 則是抓 4-5 小時的 training time，20 個 epoch 落在這個區間中且 loss 和 rouge score 已經接近收斂。其他針對 lr、weight decay 等超參數，則是參考 huggingface 提供的範例來訂定的。

2. Learning Curve

epoch(1357 steps) vs. rouge score (f1 score)



**Q3**

1.

(1) Greedy

對於 output logit，每一次都取 argmax，也就是當中機率最大的 vocab 直到最後一個詞。

(2) Beam search

給定 num_beam = n 之後，根據這個值每次維護 n 條線(句子)，亦即每次 decode 時先取前 n 大的值，根據現有的 n^2 種結果挑選機率前 n 大的句子組合，並維持此方法直到 decode 結束(原則上還是 greedy 的做法)。

(3) Top-K sampling

決定好 K 值之後，對於 decode 的 output logit 每次只取前 K 大的 vocab(要過 softmax)，且將剩餘的機率都改成 0，之後根據此 distribution 進行 sampling，所得的單詞即為此回合生成的答案。

(4) Top-P sampling

與 Top-K 類似，只是這次取的 p 為一 0 和 1 之間的機率值，在 sample 時只取累積機率在 p 值以下的 candidate 們，即為所求。Top-K 和 Top-P 都為隨機採樣的方法，當 K 或 p 值高時，會增加採樣的多樣性(sample 到冷門字的機率)；相反的，K 或 p 值低時，因為選擇變少，會越來越接近 Greedy 的方法。

(5) Temperature sampling

在取 softmax 時加入一個參數 temperature(如下圖)，當 temperature 越高時，此 distribution 會接近 uniform distribution，亦即每個詞 sample 到的機率會越接近；當 temperature 越低時，會使得原本機率高的詞和低的詞機率差更大，且當 temperature 趨近於 0 時，此方法即相當於 greedy search。

$$P(x|x_{1:t-1}) = \frac{exp(u_t/temperature)}{\sum_{t'} exp(u_{t'}/temperature)}, temperature \in [0, 1)$$

2.

這裡只取 f1 score 做比較

(1) Greedy

rouge-1: 0.24147644343076358

rouge-2: 0.0837110801256496

rouge-l: 0.20665567130745988

(2) Beam search

num_beam = 5

rouge-1: 0.2544603860444196

rouge-2: 0.10091753337241269

rouge-l: 0.22444280381985574

num_beam = 10

rouge-1: 0.2544497845242141
rouge-2: 0.10255793502691916
rouge-l: 0.2238448047900607
可以看出來 beam search 在 num_beam = 5 or 10 時表現差不多，原因可能是因為在維護 5 或 10 個句子時，在 num_beam = 5 時機率最大的句子很高機率在 num_beam = 10 時還是最大，故大部分會 output 相同的答案。
(3) Top-K sampling
K = 10
rouge-1: 0.2057761472180823
rouge-2: 0.06263092704968105
rouge-l: 0.17572530290568508
K = 20
rouge-1: 0.19600507929796904
rouge-2: 0.05995326665597388
rouge-l: 0.1669386923349455
比較 K = 10 or 20，會發現 K = 10 的三項分數都較高，原因是當 K 值變高，可能會 sample 到較冷門的字，與目標 label 的差距就更大，導致分數變低。將其與 greedy(K = 1)比較也可以發現結果比 K = 10 or 20 都好上不少。
(4) Top-p sampling
p = 0.7
rouge-1: 0.21091822548599057
rouge-2: 0.06836705670926606
rouge-l: 0.18102642181673317
p = 0.92
rouge-1: 0.18970161956458778
rouge-2: 0.057525267485928594
rouge-l: 0.16286998774603817
此方法與 Top-K 相似，都是屬於 random sampling，因此當 candidate 多時容易 sample 到遠離正解的詞，因此一樣可以觀察到 p = 0.92 的 score 小於 p = 0.7。
(5) Temperature sampling
Temperature = 2
rouge-1: 0.21091822548599057
rouge-2: 0.06836705670926606
rouge-l: 0.18102642181673317
Temperature = 0.5
rouge-1: 0.22641788216030903
rouge-2: 0.07624912488949591
rouge-l: 0.1937508365973977

對於隨機採樣的方式而言，如上一題所講，溫度越高越接近 uniform sampling，也就會讓每個詞被 sample 到的機率變近。根據先前的推論，摘要的重點不是創意性而是重要性，故 sample 到冷門詞很有可能讓分數降低，故可以發現溫度為 0.5 時(較接近 greedy)的分數較為 2 時高。

3. Final generation strategy

根據上述的分數結果，我選用 beam search(num_beam = 5)來當作最後策略，同時，因為這種方法有時會遇到重複的結果，我將 repetition_penalty 設為 3.0 來避免重複的詞出現。(max length 依舊保持 128)