**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：

一張含有 文字 的圖片

自動產生的描述

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。

1. 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

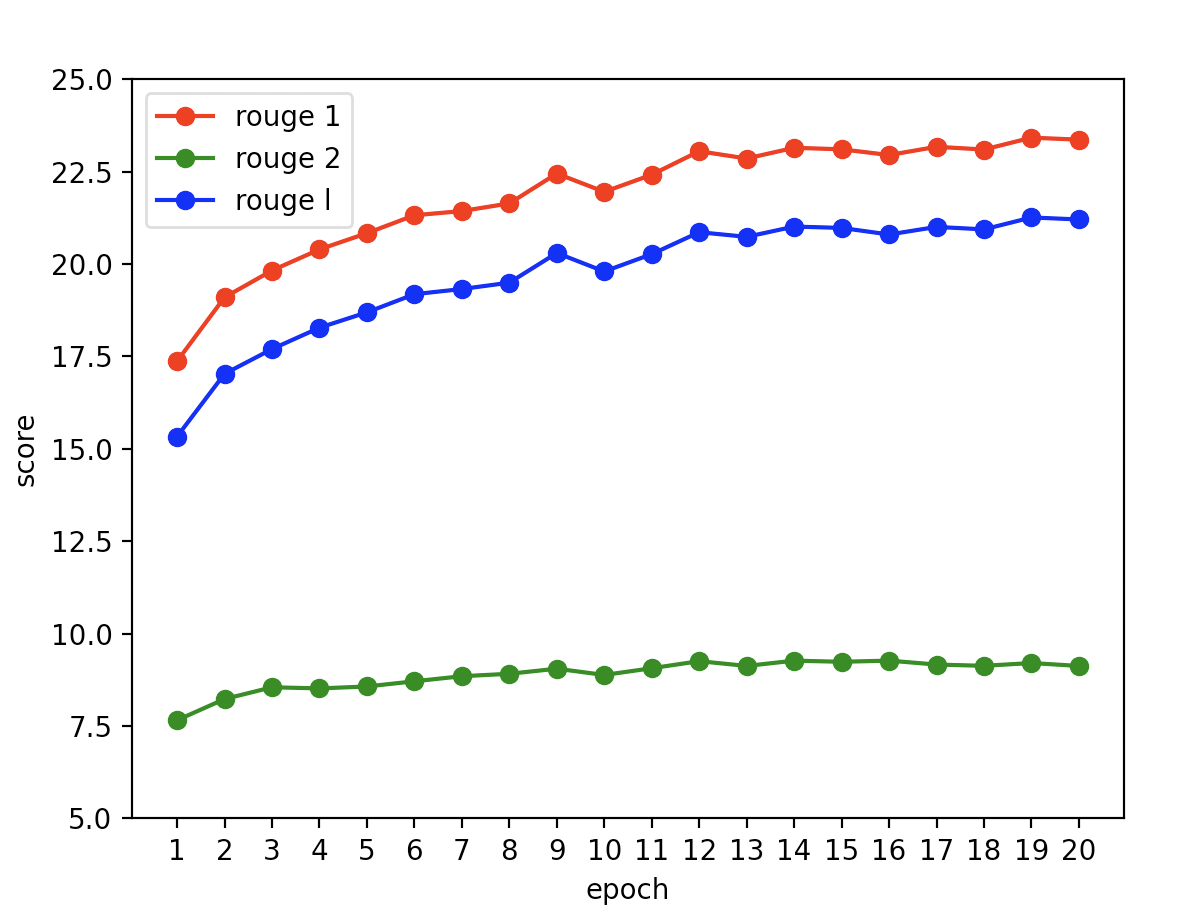
optimizer = adamw

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。

1. Learning Curve

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



**Q3**

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。

1. 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)。

1. 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。

1. 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。

1. Temperature sampling

一張含有 文字 的圖片

自動產生的描述在取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。

這裡只取f1 score做比較

1. Greedy

rouge-1: 0.24147644343076358

rouge-2: 0.0837110801256496

rouge-l: 0.20665567130745988

1. 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。

1. 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。

1. 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。

1. 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。

1. 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)