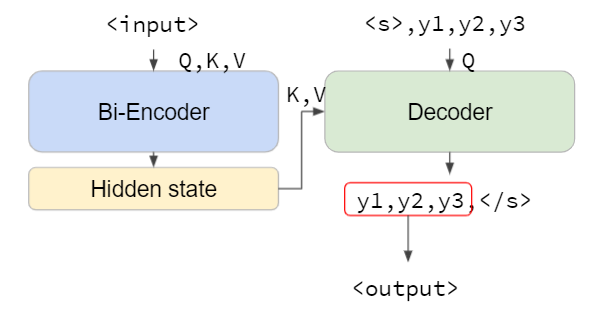
Applied Deep Learning Homework 3 report

M10915045 施信宏

Q1: Model

* 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. 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

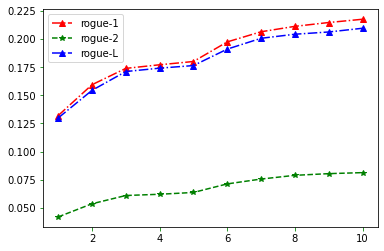
並有採用learningRate\_scheduler，無採用fp16進行訓練。

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。

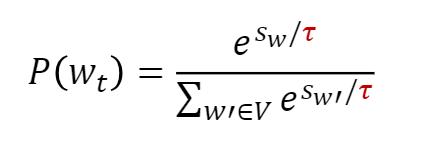
* Beam Search:

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:



temperature hyperparameter 𝜏

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

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

* Top-k Sampling

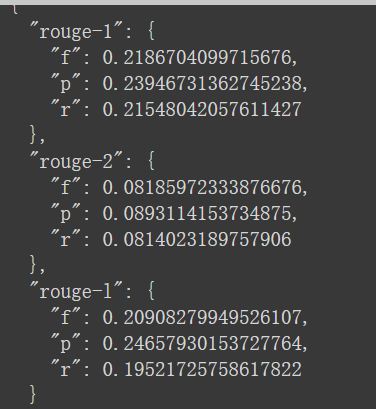
Sampling: wt∼P(w|w1:t−1)，隨機選取下一個單字(from a distribution)，top-k sampling則是將選擇機率單字固定在前K個，從前K個單字內隨機選取輸出。

* 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

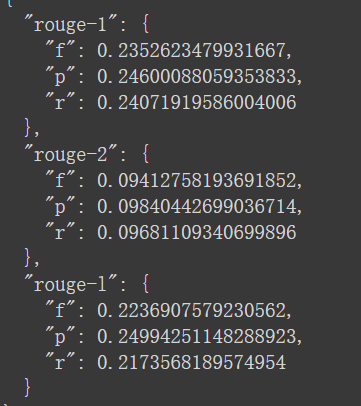
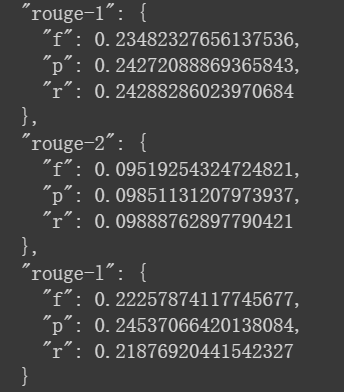
每個Strategies的實驗結果: 每個Strategies的評價在最後。

Greedy:



Beam search:

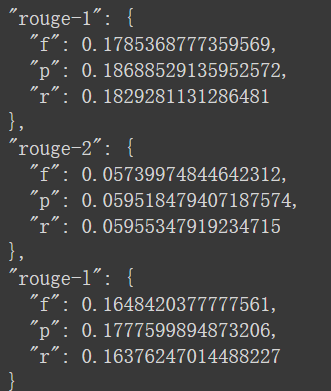
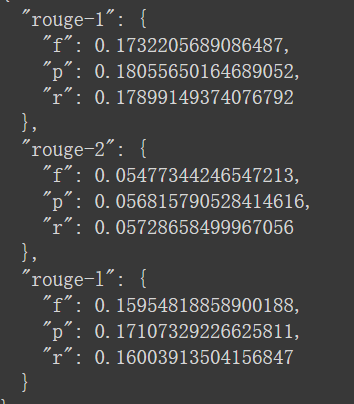
Beam = 3 Beam = 5

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

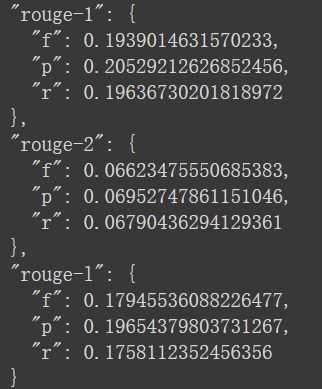
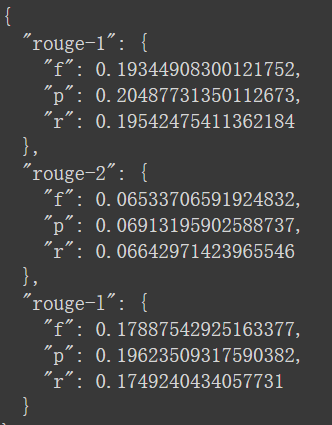
K = 30 K = 50

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。

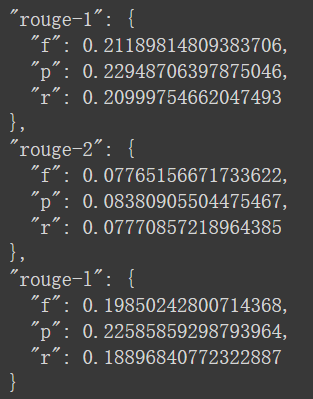
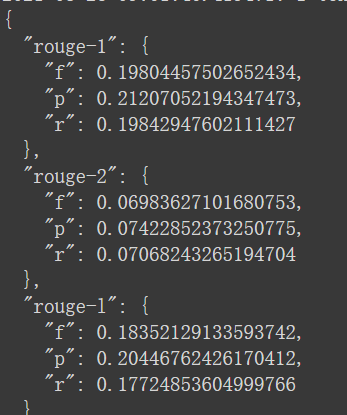
Top-p sampling

p = 0.8 p = 0.9

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。