

Applied Deep Learning HW3 Report
學號：R09944010 系所：網媒所 姓名：解正安

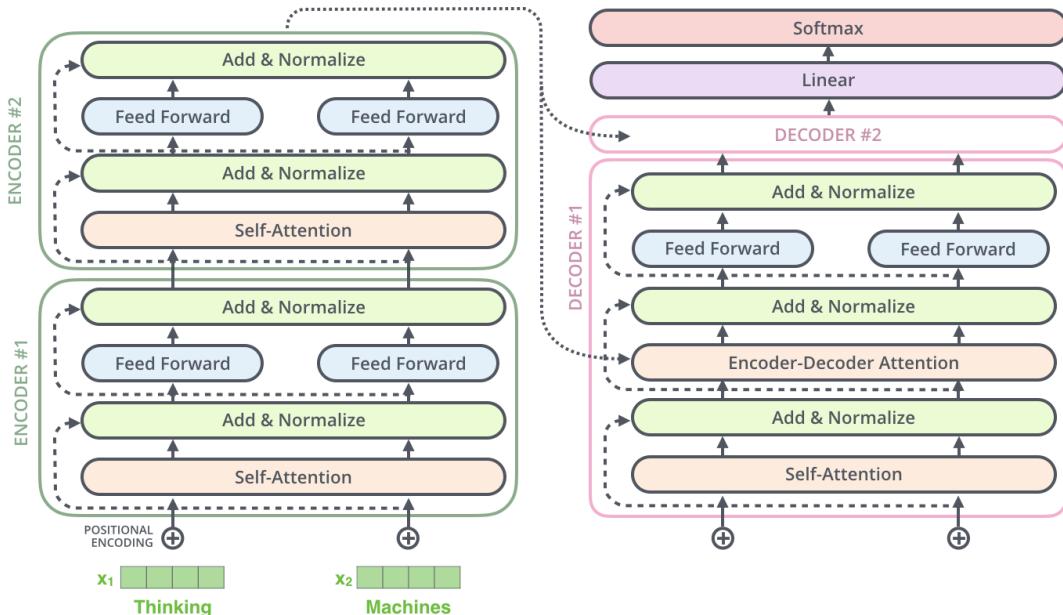
Q1: Model

1. Model

1. Describe the model architecture and how it works on text summarization

本次作業採用 `huggingface` 的 Multilingual T5，模型結構與 T5 相同，差別只在 dataset 是由 101 種語言組成。

模型結構是採用 Encoder-Decoder 的方式，結構如下：



Ref: <https://laptrinhx.com/t5-a-detailed-explanation-782290276/>

Encoder 和 decoder 皆使用到 self-attention:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

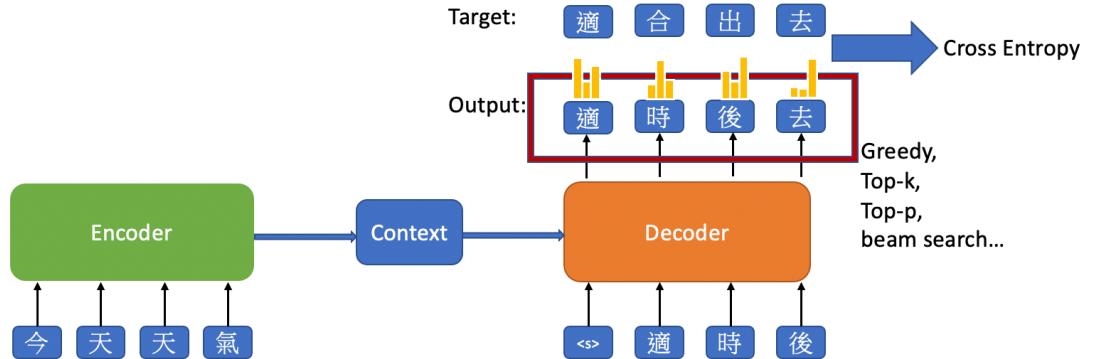
其中 c_i 是 hidden state 的 context vector， h_j 是 hidden state 的 vector。而 α 則是 softmax 的輸出，為 attention score:

$$\text{Attention score } \alpha_{ij} = e^{e_{ij}} / \sum_{k=1}^{T_x} e^{e_{ik}}$$

$$\underline{\text{score}} \quad e_{ij} = a(s_{i-1}, h_j)$$

$e_{i,j}$ 是 alignment model，最簡單算法是把 target hidden state 跟 output hidden state 做 dot product。

而本次作業因為是使用 pretrained model 做 fine-tune，只需要根據 target 跟 output 的 probability 做 cross entropy 即可計算 loss。架構如下：



Model 盡可能學習輸出的機率分佈，而最後產生的字可以依據不同演算法如 greedy, top-k,top-p 等等而有不同的結果。

2. Preprocessing

1. Describe your preprocessing (e.g. tokenization, data cleaning and etc.)

對文章內容使用 regex 去除一些不必要的資訊，像是文章結尾：“延伸閱讀”，“《原文刊登於.》”，“來源”，連結等等這些跟文章主旨並沒有關係的文字，會直接刪除。

Code:

```
context = re.sub(r'延伸閱讀.+', r'', context)
context = re.sub(r'《原文刊登於.+', r'', context)
context = re.sub(r'來源 : .+', r'', context)
context = re.sub(r'來源 : 文 .+', r'', context)
context = re.sub(r'文 .+', r'', context)
context = re.sub(r'看這裡>>.+', r'', context)
context = re.sub(r'看這裡>>.+', r'', context)
context = re.sub(r'本文摘自.+', r'', context)
context = re.sub(r'【.+】', r'', context)
context = re.sub(r'\(.+\)', r'', context)
context = re.sub(r'\n', r'', context)
context = re.sub(r'http\S+', r'', context)
context = re.sub(r'圖 / .+提供', r'', context)
context = re.sub(r'圖 / 翻攝自臉書社團', r'', context)
context = re.sub(r'(.) ', r'', context)
context = re.sub(r'《本文作者.+', r'', context)
context = re.sub(r'地址 : .+', r'', context)
```

而 tokenization 的部分，使用 `huggingface` MT5 預設的 `MT5Tokenizer`，文章內容 `max_length` 為 512，而標題長度在分析過資料後發現集中在 20-60 之間，因此 `max_length` 為 64。Padding 使用 “`max_length`”。

Q2: Training

1. Hyperparameter (1%):

1. Describe your hyperparameter you use and how you decide it.

(1) Train: 2 顆 2080ti gpu

- Max_seq_length:** 512 (model 預設最長)
- Max_summary_length:** 64 (本次 training 資料 title 最長為 86，但因為分佈多位在 20-60 之間，故設為 64)
- Max_epochs:** 40 (大約在 epoch 30 左右就會收斂，但為了觀察後面 training 圖形設定大一些)
- Batch size:** 3 (由 pytorch lightning tune model 所算出)
- RL ratio:** 0.3 (更新 loss 時，RL loss 所佔比例(設定 0.5 和 0.3，在 training 後比較 performance 以 0.3 較佳))
- RL Ratio Power:** 100 (原先 loss 大約只有 0.0002 左右，觀察後設定為 100)
- RL start epoch:** 36 (epoch 36 時，rougeL 分數最佳)

(2) Model:

MT5 model，參數使用 google 訓練時預設的參數，避免不同參數會使 performance 降低。

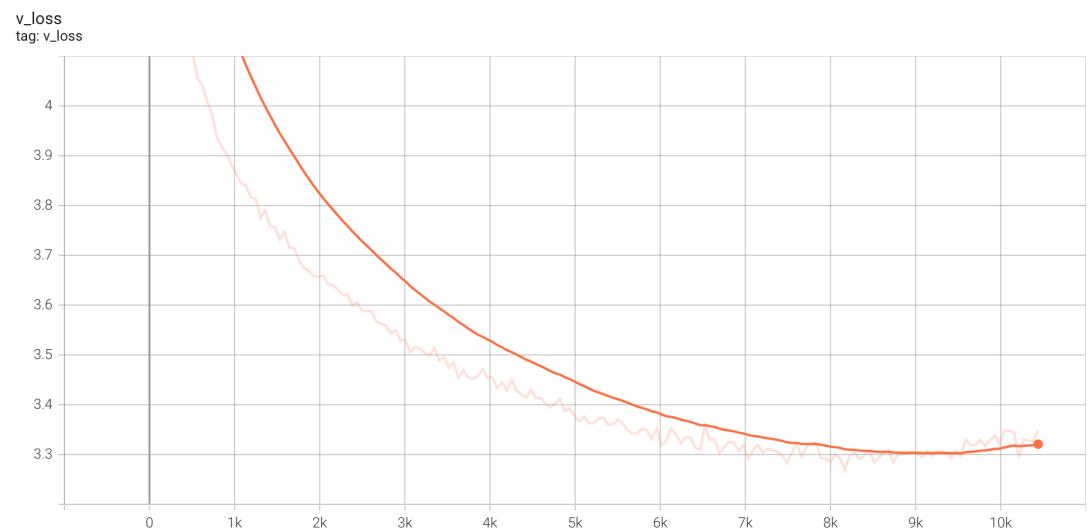
- Activation function:** gated-gelu
- Num_heads:** 6
- Num_layers:** 8
- Dropout:** 0.1
- layer_norm_epsilon:** 1e-06
- Vocab_size:** 250112
- Max length:** 512

2. Learning Curves

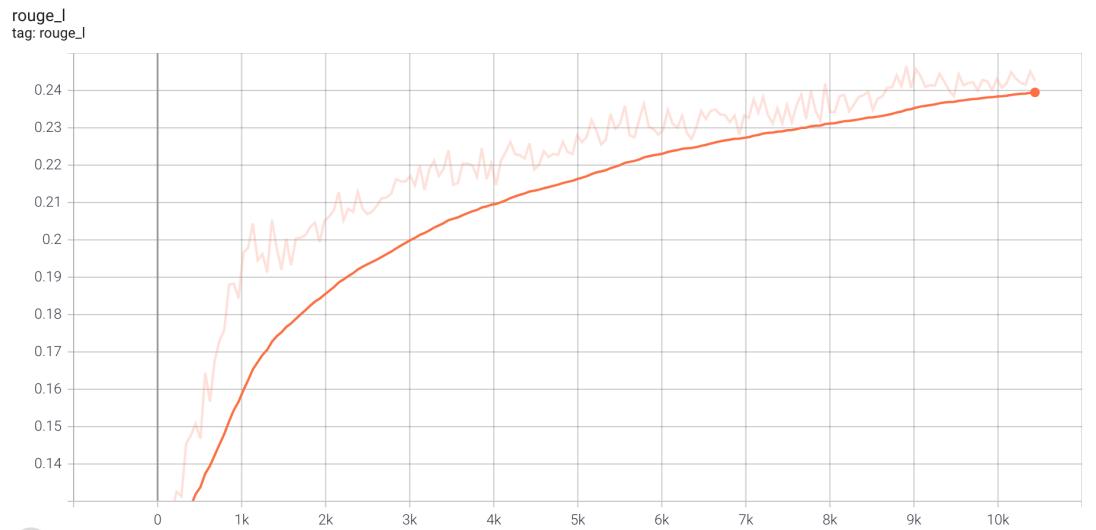
1. Train Loss



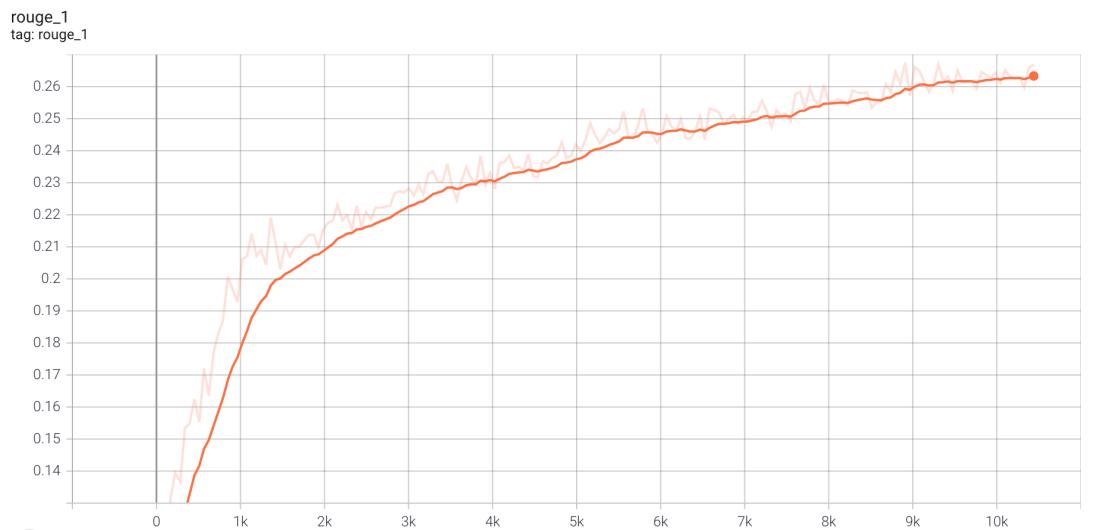
2. Validation loss (Validation 只有取隨機的 500 筆作為測資)



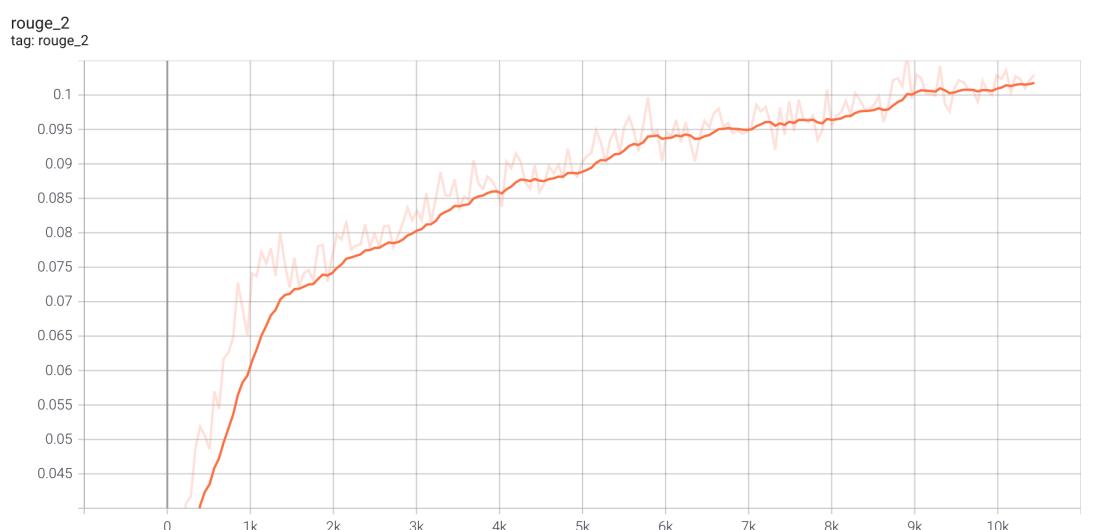
3. Validation rouge-l score



4. Validation rouge-1 score



5. Validation rouge-2 score



Q3: Generation Strategies

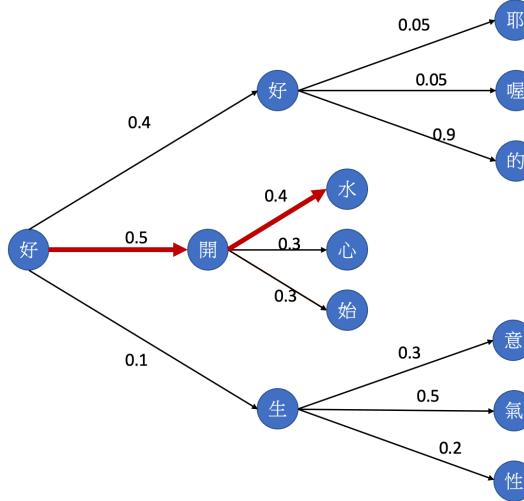
1. Strategies

1. Greedy

每次取最高機率的 words 當作輸出。

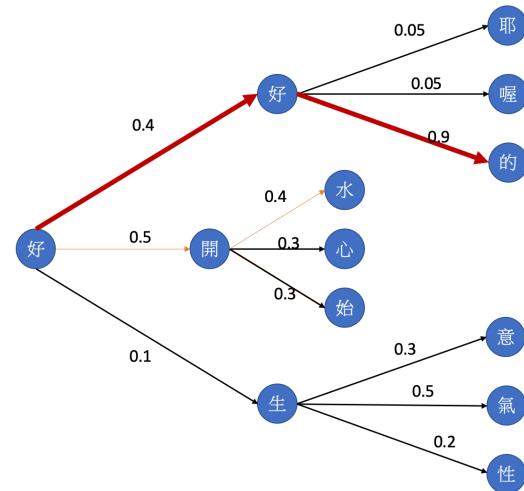
$$w_t = \text{argmax}_w P(w|w_{1:t-1})$$

如下圖，每次都取最高機率出現的字，即為 greedy 得算法。



2. Beam Search

每次追蹤不止一條路徑，可以設定要追蹤幾條路徑。如下圖以 num_beams=2 做追蹤。“好”, “開”機率 0.5 最高，但因為追蹤兩條，第二高(“好”, “好”)機率 0.4 也會追蹤。最後以(“好”, “好”, “的”)0.36 大於(“好”, “開”, “水”)的 0.2。



3. Top-k Sampling

每次從 $P(x|x_{1:t-1})$ 選取機率最高的前 k 個 token，將這些 token 機率加總可以得到 $p' = \sum P(x|x_{1:t-1})$ ，再更新新的機率分佈：

$$P'(x|x_{1:t-1}) = P(x|x_{1:t-1})/p'$$

最後從新的機率分佈 sample 出一個 token 作為 output。該方法缺點是 k 是固定的，但每句話的後面一個字機率分佈不同，像是嫦娥的嫦字，後面娥的機率非常大，但如果 k 設定 10 就仍會找前 10 名機率高的 token，產生一些不該出現的詞。

4. Top-p Sampling

為了解決 Top-k 的問題所提出，先設定累加機率 p 作為

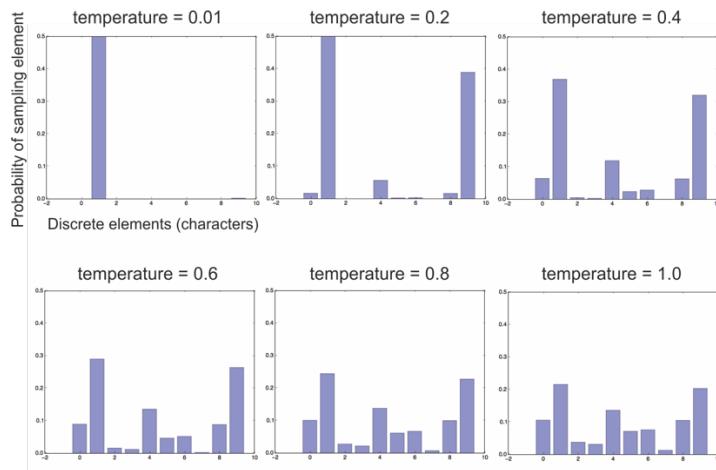
threshold，每次根據 $p' = \sum P(x|x_{1:t-1})$ 取 token 時，會取前幾名 token 累加機率直到 p 為止，這樣就能動態的取不同數量的 token。

5. Temperature

改變原本 softmax 的算法，如下列公式：在原本的 logits 多除以一個 T(Temperature)，可以讓機率分佈更加的極端。

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

如果 temperature 設定的越低，越容易產生固定的句子，因為機率高的字出現的機率更高，機率低的字出現機率更低。此方法可以搭配其他方法如 top-k 使用。



Ref: <https://livebook.manning.com/book/deep-learning-with-python/chapter-8/34>

2. Hyperparameters

- Try at least 2 settings of each strategies and compare the result.

	Greedy	Beam search (k=4)	Beam search (k=8)	Top-k (k=20)	Top-k (k=35)	Top- (p=0.5)	Top-p (p=0.9)	Top-k (k=20)	Top-p (p=0.9)	Top-P:0.6 Top-k:20 temp.:0.9
Rouge1	24.70	26.21	26.33	22.00	21.50	23.51	20.79	22.44	22.22	24.04
Rouge2	9.54	10.40	10.64	7.71	7.46	8.75	7.25	8.01	7.97	9.08
RougeL	23.03	23.44	23.62	20.12	19.62	21.61	19.08	20.54	20.37	22.20

Original:

"title": "台灣中油給您奉茶 重現古早人情味的科技減塑行動",

Greedy:

{"title": "台灣中油攜手Circu Plus 推50處據點、地點、地點", "id": "21784"}

Beam search(k=4):

{"title": "台灣中油攜手Circu Plus 推50處「奉茶行動」", "id": "21784"}

Beam search(k=8):

{"title": "台灣中油攜手Circu Plus 用創新思維做溫暖貼心", "id": "21784"}

Top-k (k=20):

{"title": "台灣中油攜手Circu Plus 提供50處據點、地點、設水服務", "id": "21784"}

Top-k (k=35):

{"title": "台灣中油攜手Circu Plus 讓民眾在即找到「奉茶」", "id": "21784"}

Top-p (p=0.5):

{"title": "台灣中油攜手Circu Plus 試營運40處據點", "id": "21784"}

Top-p (p=0.9):

{"title": "3大科技通通!台灣中油研發App「奉茶行動」,100處據點一次看", "id": "21784"}

Top-k (k=20 temp=0.8):

{"title": "台灣中油攜手Circu Plus 幫50處提供水瓶就能找到", "id": "21784"}

Top-p (p=0.9 temp=0.8):

{"title": "台灣中油攜手Circu Plus 推50處服務時間、地點、服務", "id": "21784"}

Top-k with top-p (k=20,p=0.6,temp=0.9):

{"title": "台灣中油攜手Circu Plus 用創新思維做溫暖貼心", "id": "21784"}

從上面的結果會發現，greedy 容易產生重複的詞，因為這些詞跟主題最有關係，機率相對也較大。而 top-k 和 top-p 的部分，k 值或是 p 值越大越容易產生不一樣的結果，但相對也容易產生不通順的句子，如果值越小則會越像 greedy。而隨著 temperature 的降低，也讓字更加靈活，語句變化性增加，如 Top-k (k=20 temp=0.8) 比起 Top-k (k=20)，“提供水瓶就能找到”是較口語的

說法，不像 title 會有的內容，但如果是為了分數就不適合。而 beam search 方法分數最高，是因為考量整體句子的機率，設定 k 值越大越容易找到機率更大的句子，但相對執行時間也較長。如果為了本次作業，應選擇 beam search，如果為了產生語句的靈活性，則要嘗試調整 k,p 以及 temperature 的數值。

以下附上另外 2 筆實驗結果：

Original:

["title": "Lucasfilm Games再度回歸 但可能只是《星際大戰》遊戲內容所屬品牌識別"]

Greedy:

{"title": "《印地安納瓊斯》系列新作將回歸 可能在《星際大戰》相關遊戲", "id": "21803"}

Beam search(k=4):

{"title": "《印地安納瓊斯》系列遊戲新作將回歸 可能在《星際大戰》相關遊戲", "id": "21803"}

Beam search(k=8):

{"title": "《印地安納瓊斯》系列遊戲新作將回歸 可能在《星際大戰》相關遊戲", "id": "21803"}

Top-k (k=20):

{"title": "「印地安納瓊斯」新作將來臨?遊戲將在《星際大戰》相關遊戲內容", "id": "21803"}

Top-k (k=35):

{"title": "《印地安納瓊斯》相關遊戲新作將回歸?遊戲將在《星際大戰》相關遊戲內容", "id": "21803"}

Top-p (p=0.5):

{"title": "《印地安納瓊斯》系列新作將回歸!未來將使《星際大戰》相關遊戲新作回歸", "id": "21803"}

Top-p (p=0.9):

{"title": "《印地安納瓊斯》系列遊戲新作 明年將使《星際大戰》品牌回歸", "id": "21803"}

Top-k (k=20 temp=0.8):

{"title": "《印地安納瓊斯》新作將回歸 原來將再回歸", "id": "21803"}

Top-p (p=0.9 temp=0.8):

{"title": "《印地安納瓊斯》系列新作確定將重啟!將使《星際大戰》相關遊戲內容回歸", "id": "21803"}

Top-k with top-p (k=20,p=0.6,temp=0.9):

{"title": "《印地安納瓊斯》系列新作將回歸!目前已經在《星際大戰》相關遊戲", "id": "21803"}

Original:

["title": "不去會後悔？！在南投仁愛「精英野溪溫泉」泡湯、悠閒野餐"]

Greedy:

{"title": "南投「精英溫泉」 南投「布卡山溫泉」 超簡單好愜意", "id": "21810"}

Beam search(k=4):

{"title": "精華影輯/南投仁愛鄉精英村「布卡山溫泉」 悠閒野餐享受野溪泡湯", "id": "21810"}

Beam search(k=8):

{"title": "精華影輯/南投仁愛鄉精英村「布卡山溫泉」 悠閒野餐享受野溪泡湯", "id": "21810"}

Top-k (k=20):

```
{"title": "南投仁愛鄉精英溫泉「布卡山溫泉」 山巒秘境的超愜意的愜意", "id": "21810"}
```

Top-k (k=35):

```
{"title": "北投「精英溫泉」 泰雅族原住民守護秘境", "id": "21810"}
```

Top-p (p=0.5):

```
{"title": "影/南投「精英溫泉」 南投玩法 超簡單的野溪溫泉", "id": "21810"}
```

Top-p (p=0.9):

```
{"title": "南投和平區「精英溫泉」 5大精選玩法!全台「蠻剛剛到」 體驗溪谷超愜意", "id": "21810"}
```

Top-k (k=20 temp=0.8):

```
{"title": "南投南投遊客超簡單! 精華「布卡山溫泉」 坐擁山巒賞野溪泡湯", "id": "21810"}
```

Top-p (p=0.9 temp=0.8):

```
{"title": "訓練心得/南投精英溫泉 坐擁山巒賞野溪泡湯", "id": "21810"}
```

Top-k with top-p (k=20,p=0.6,temp=0.9):

```
{"title": "南投仁愛鄉精英溫泉「布卡山溫泉」 超簡單好愜意", "id": "21810"}
```

2. What is your final generation strategy?

最後選擇 Search Beam(K=8)，原因是在 validation 的分數較高。

Bonus: Applied RL on Summarization

1. Algorithm

1. Describe your RL algorithms, reward function, and hyperparameters

本次作業的做法是參考 [R Paulus, et al. 2018](#) 的論文，如下列公式所示：

$$L_{rl} = (r(\hat{y}) - r(y^s)) \sum_{t=1}^{n'} \log p(y_t^s | y_1^s, \dots, y_{t-1}^s, x)$$

r 代表 reward function，這裡以 Rouge-l 的 f1 分數作為參考。而 \hat{y} 是每次用 greedy 所跑出來的結果， y^s 則是 random sample 的結果。

$\sum_{t=1}^{n'} \log p(y_t^s | y_1^s, \dots, y_{t-1}^s, x)$ 是 random sample 對 greedy 的 logits 取 log。

因此 model 要做的事情是盡可能讓 greedy 的 rouge-l score 高一些，得到 positive 的 reward。

最後在更新 loss 時，除了原先的 loss，可以加入一定比例的 RL-loss，公式如下：

$$L_{mixed} = \gamma L_{rl} + (1 - \gamma) L_{ml}$$

γ 是本次作業的 RL-ratio，設定為 0.3，而 RL_start_epoch 是指在第幾個 epoch 後開始計算 rl loss，本次作業設定為 36，讓 model 在訓練達到一定 baseline 後才引入 RL loss。

2. Compare to Supervised Learning

RL 因為訓練時間長，一個 epoch 約 1 小時半(2 顆 2080ti)。為了節省訓練時間，本作業的方式是先利用一般 ML 的 loss 訓練到 epoch36 收斂後，再使用 policy gradient 看是否能提升 performance。而每個項目會有兩張圖片是因為 train 到 epoch39 的時候發現 validation loss 變化不大，但 rouge score 有上升，因此中斷訓練 load checkpoint，改存 model 的方式為 score 大的。

結果如圖：

Train loss:

(1)EPOCH 36-38

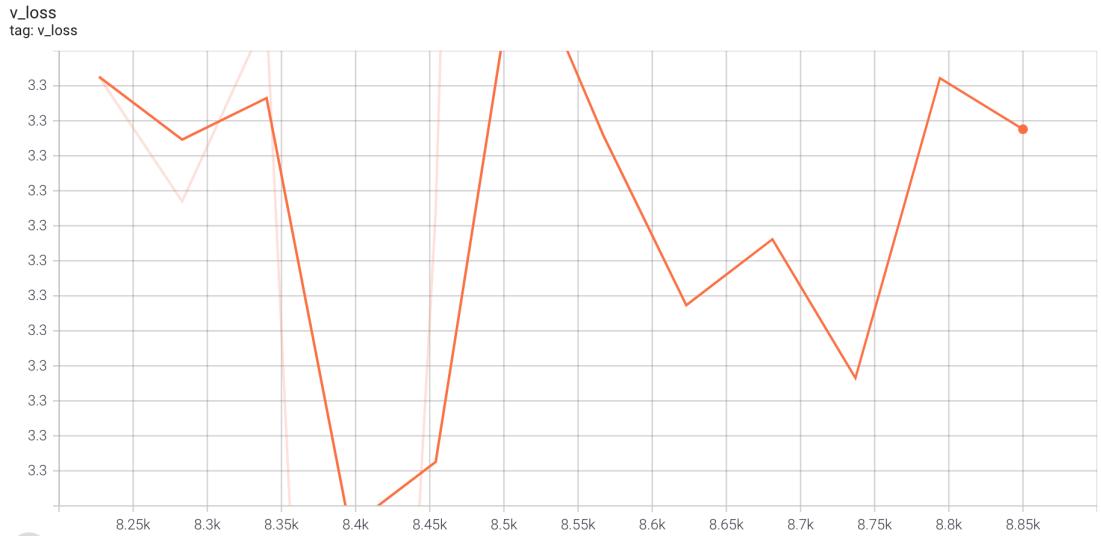


(2)EPOCH 39-50

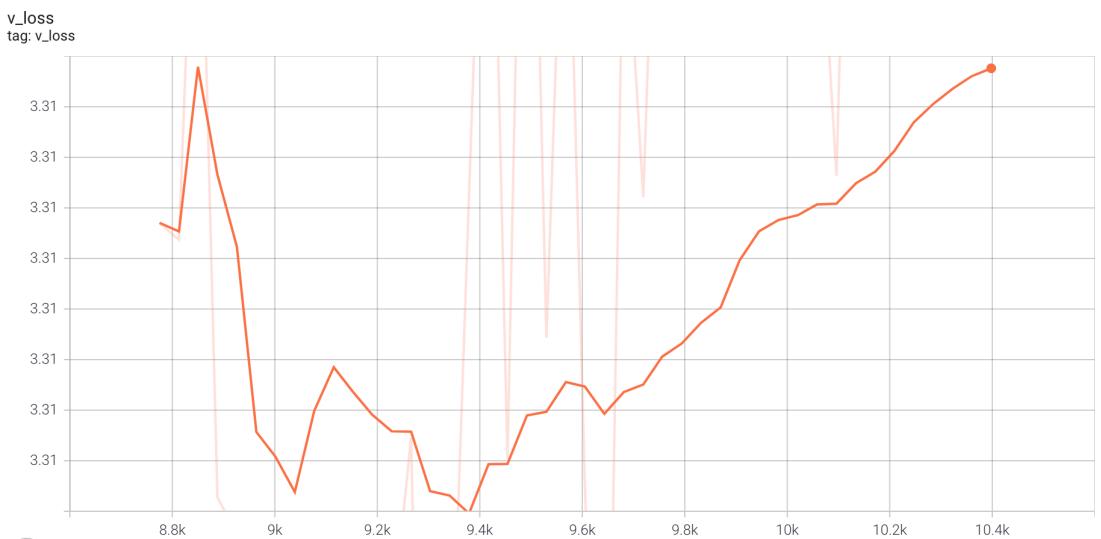


Validation loss:

(1) EPOCH 36-38



(2) EPOCH 39-50



RL loss:

(1) EPOCH 36-38

rl_loss
tag: rl_loss



(2) EPOCH 39-50

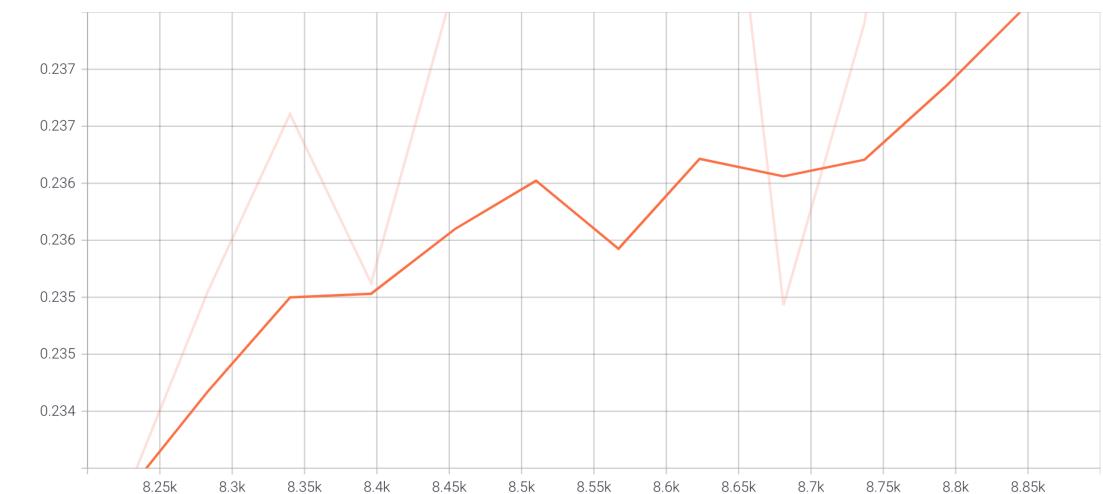
rl_loss
tag: rl_loss



Rouge-I score:

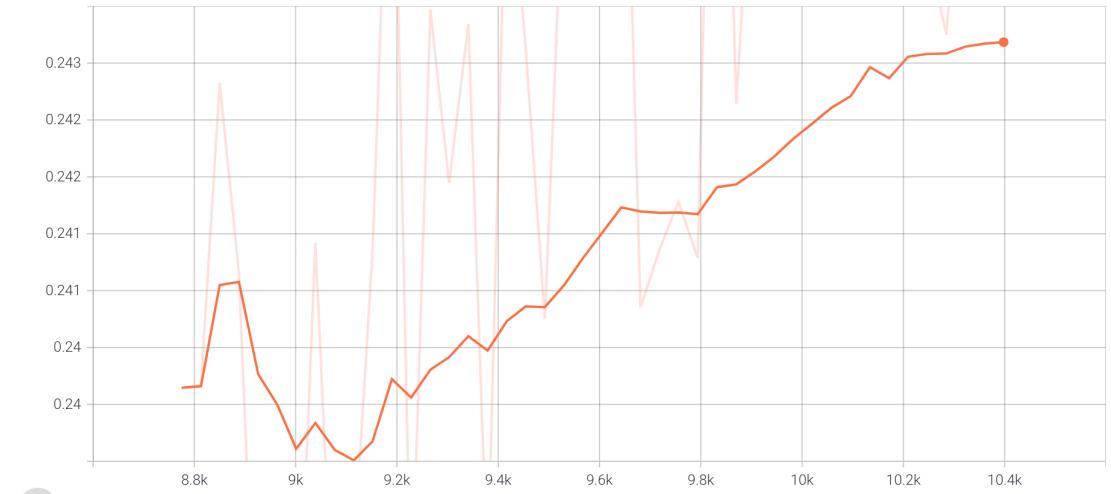
(1) EPOCH 36-38

rouge_I
tag: rouge_I



(2)EPOCH 39-50

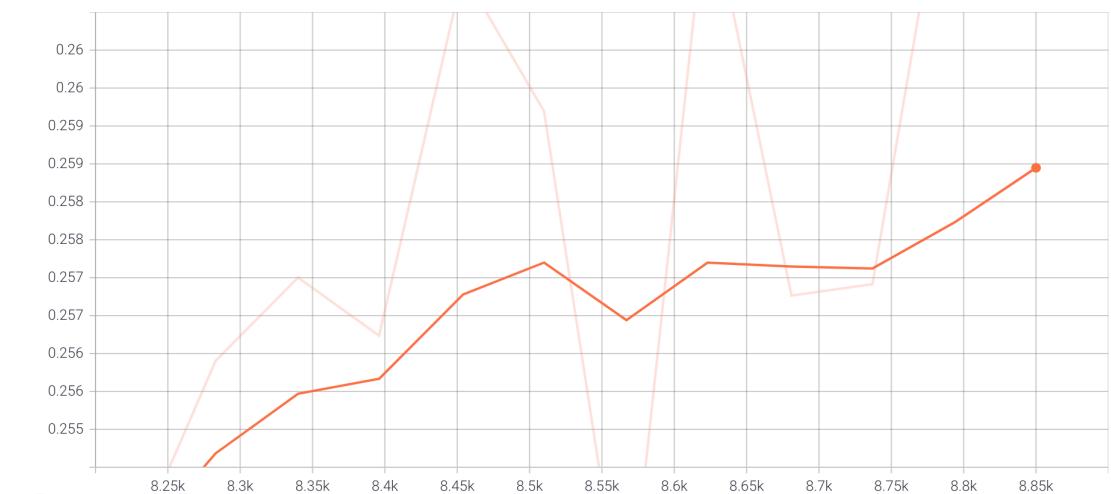
rouge_l
tag: rouge_l



Rouge-1 score:

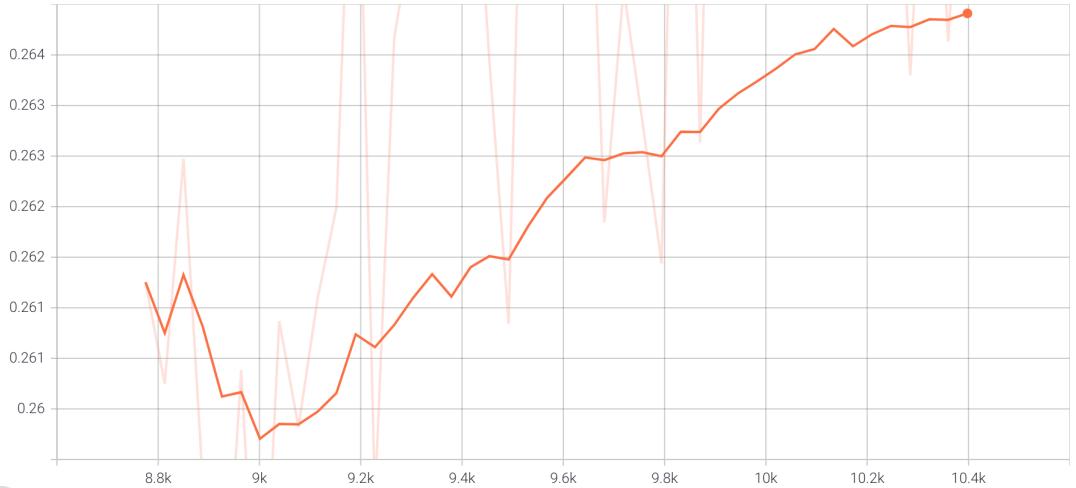
(1)EPOCH 36-38

rouge_1
tag: rouge_1



(2)EPOCH 39-50

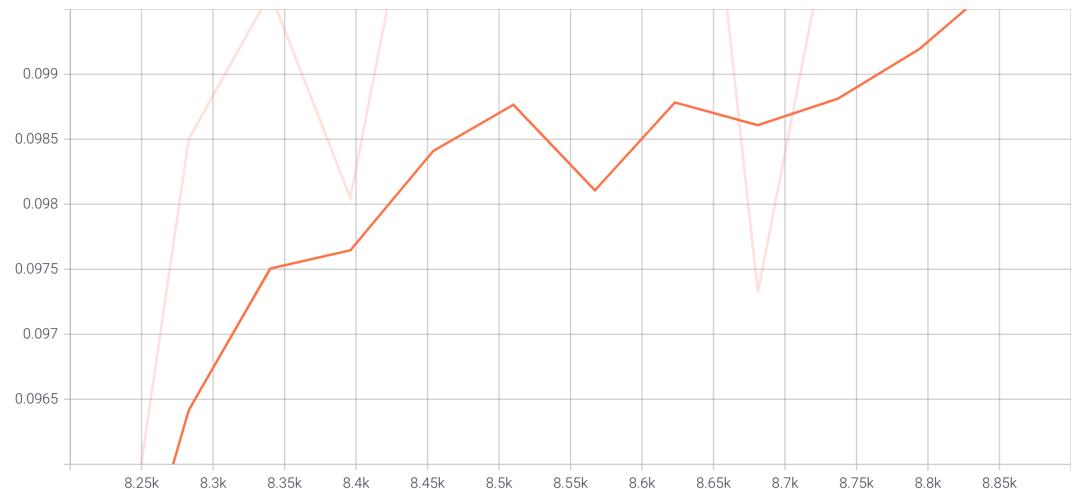
rouge_1
tag: rouge_1



Rouge-2 score:

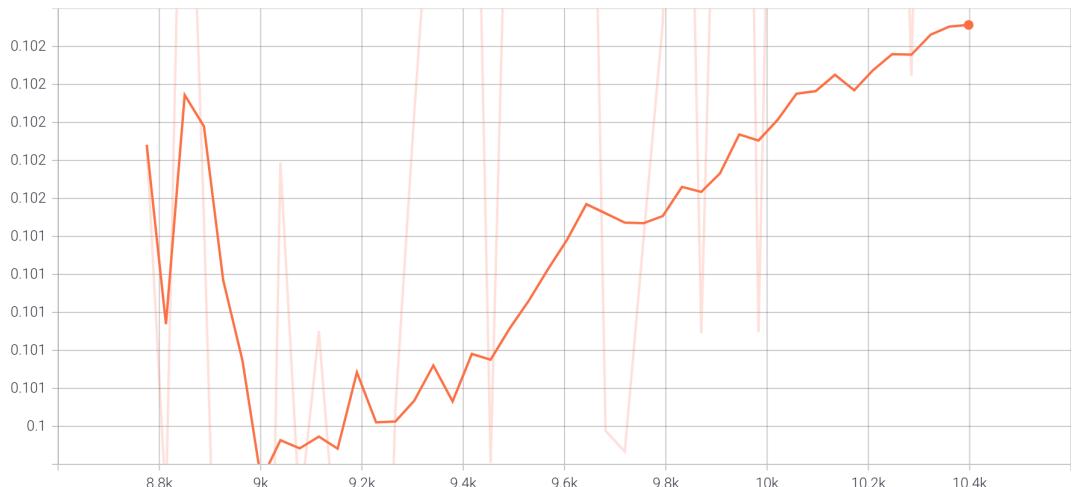
(1) EPOCH 36-38

rouge_2
tag: rouge_2



(2) EPOCH 39-50

rouge_2
tag: rouge_2



從圖中的結果可以看到，validation loss 一開始不太會變化，但隨著 training loss 不斷下降最後 validation loss 會上升。但 rouge score 因為 policy gradient 有明顯的上升，而 RL loss 也有下降。

Rouge 分數比較：

F1-Score*100 Strategy	原先最好 model			加上 policy gradient		
	Greedy	Beam(k=4)	Top-P:0.6 Top-k:20 temperature:0.9	Greedy	Beam(k=4)	Top-P:0.6 Top-k:20 temperature:0.9
Rouge1	24.04	25.91	23.71	24.70	26.21	24.04
Rouge2	9.10	10.31	8.81	9.54	10.40	9.08
RougeL	22.57	23.24	21.96	23.03	23.44	22.20

產生結果：

Original: 原句子

```
"title": "台灣中油給您奉茶 重現古早人情味的科技減塑行動",
"title": "Lucasfilm Games再度回歸 但可能只是《星際大戰》遊戲內容所屬品牌識別"
"title": "不去會後悔？！在南投仁愛「精英野溪溫泉」泡湯、悠閒野餐"
```

Greedy:

Without RL loss:

```
{"title": "台灣中油攜手Circu Plus 推50處據點、地點、地點、交通資訊", "id": 21784}
{"title": "《星際大戰》相關遊戲新作 可能在《星際大戰》相關遊戲內容", "id": 21803}
{"title": "南投「精英溫泉」 南投仁愛鄉精英村「布卡山溫泉」 超簡單的野溪泡湯", "id": 21810}
```

With RL loss:

```
{"title": "台灣中油攜手Circu Plus 推50處據點、地點、地點", "id": 21784}
 {"title": "《印地安納瓊斯》系列新作將回歸 可能在《星際大戰》相關遊戲", "id": 21803}
 {"title": "南投「精英溫泉」 南投「布卡山溫泉」 超簡單好愜意", "id": 21810}
```

在 greedy 會發現，title 長度經過 RL loss 後變短，model 可能認為短一點的 title 分數會較高。此外，內容也較不容易產生重複，在 rouge 分數會較高。

Beam:

Without RL loss:

```
{"title": "台灣中油攜手Circu Plus 用創新思維做溫暖貼心", "id": 21784}  
{"title": "《星際大戰》相關遊戲新作將回歸 可能在《印地安納瓊斯》", "id": 21803}  
{"title": "精華影輯/南投仁愛鄉精英村「布卡山溫泉」 輕鬆愜意泡湯", "id": 21810}
```

With RL loss:

```
{"title": "台灣中油攜手Circu Plus 推50處「奉茶行動」", "id": "21784"}  
{"title": "《印地安納瓊斯》系列遊戲新作將回歸 可能在《星際大戰》相關遊戲", "id": "21803"}  
{"title": "精華影輯/南投仁愛鄉精英村「布卡山溫泉」 悠閒野餐享受野溪泡湯", "id": "21810"}
```

Beam 的變化較大，會抓到更多的重點，像是第一句奉茶“行動”對應原先 title 的減塑“行動”。還有第三個悠閒野餐本來沒有，抓進來後更符合 title。

Top-p with top-k(temperature = 0.9):

Without RL loss:

```
{"title": "台灣中油攜手Circu Plus 推50處「奉茶行動」", "id": 21784}  
{"title": "《星際大戰》遊戲新作 可能在《星際大戰》相關遊戲內容", "id": 21803}  
{"title": "南投仁愛鄉精英村「精英溫泉」 超簡單好去處", "id": 21810}
```

With RL loss:

```
{"title": "台灣中油攜手Circu Plus 用創新思維做溫暖貼心", "id": "21784"}  
{"title": "《印地安納瓊斯》系列新作將回歸!目前已經在《星際大戰》相關遊戲", "id": "21803"}  
{"title": "南投仁愛鄉精英溫泉「布卡山溫泉」 超簡單好愜意", "id": "21810"}
```

第二句會發現 model 將感嘆句加上驚嘆號，此外第三句因為已經有精英 兩字，相較未加上 rl loss 之前，後來的句子較不容易產生重複的句子。