網媒所碩一 R10944028 蔡易儒

ADL HW3 report

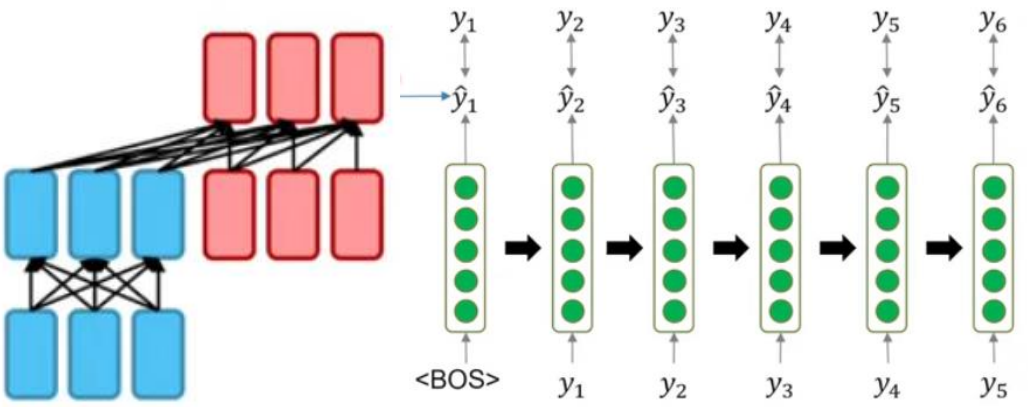
Q1. Model

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

屬於 Encoder-Decoder 的架構，常用於 Sequence to Sequence 的任務，接受 sequence 的輸入，並可以 output 出不同長度的 sequence。Encoder 的部分(下圖藍色部分)可以擁有 bi-directional 的 context 資訊，也就是可以看到整體的文字輸入並使用 fully-visible attention mask，而在 Decoder 的部分(下圖紅色部分)則保有 Auto-regressive 的特性，只能夠透過自己以前的資訊來預測下一次的結果。

使用在 text summarization 的方法是，會先在文章前面加上"summarize:"，將經過 tokenize 過的文章 token x 當作是 encoder 的輸入，而 generation 的部分則是先在 decoder 的開頭輸入一個 start token，接著讓 decoder 藉由y_1到y_{n-1}與之前看過的文章來預測出y_n，也就是計算$p(y_i|y_1,\dots,y_{i-1},x)$如下圖右，以 greedy 為例，會將當下的預測機率最高的 token 當作是下一個時間的 input，繼續做預測。

$$output\_enc = encoder(x)$$
$$y\_enc_t = decoder(output\_enc\ , y_{1:t-1})$$
$$y_t = Strategy(y\_enc_t)$$



Model architecture config:

ADL HW3 report

2. Preprocessing: Describe your preprocessing (e.g. tokenization, data cleaning and etc.)

有做 data 的最大長度限制，如果 context 的長度超過訂定的長度，會將超過的資料 truncated。

Tokenizer 的部分，mt5 是使用 T5 tokenizer，基於 SentencePiece 的一種非監督式的 text tokenizer 與 detokenizer，其詞彙量是訓練前就會先決定好的，並不像一般的非監督式的分詞系統認為詞彙量有無限大，底層使用 subword unit(BPE)與 unigram language model，藉由統計每一個連續 subword pair，選擇機率最高的將其合併一直到 vocab size 滿意為止，但就跟作業 2 一樣若是中文的話應該還是會直接將一個中文字當成一個 token。

網媒所碩一 R10944028 蔡易儒

ADL HW3 report

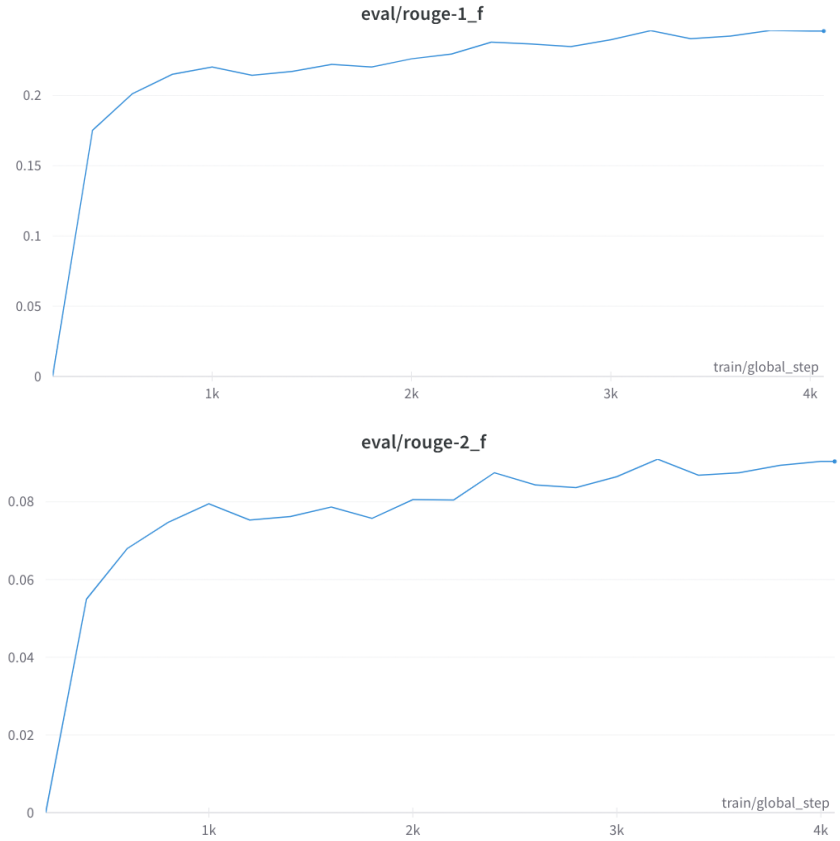
Q2. Training

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

- Pretrained model: google/mt5-small
- Learning rate: 1e-3
- Max source length: 1024
- Max target length: 128
- Effective batch size(per device batch size*gradient accumulate step *# of device): 2*8*1 =16
- Epoch: 3
- Optimizer: Adafactor
- Scheduler warm up ratio: 0.1

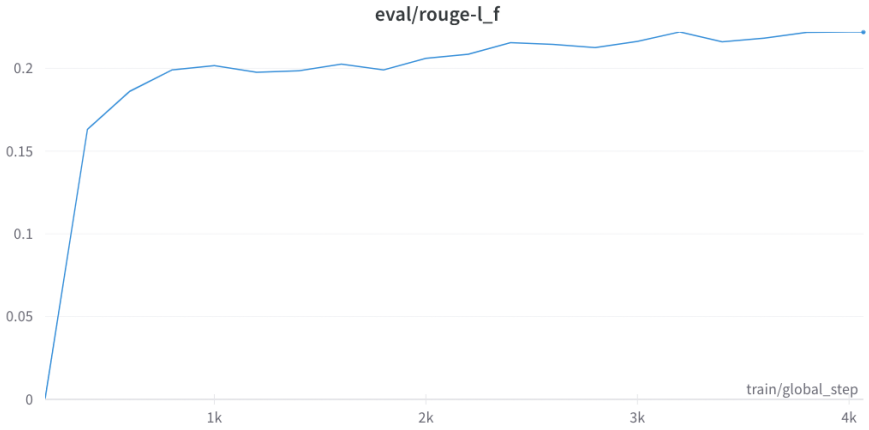
能夠改動的超參數其實不多，比較能改的只有 optimizer、epoch、與 learning rate，而使用預設 3e-5 等級的 learning rate 會發現他成長的速率非常的慢，因此直接將 learning rate 提升，就可以只訓練 3 epoch 就可以有不錯的結果，因此後續就沒有再調整參數了，而改將精力放在 decoding strategy 上。

2. Learning Curves: Plot the learning curves (ROUGE versus training steps)



網媒所碩一 R10944028 蔡易儒

ADL HW3 report



ADL HW3 report

Q3. Generation Strategies

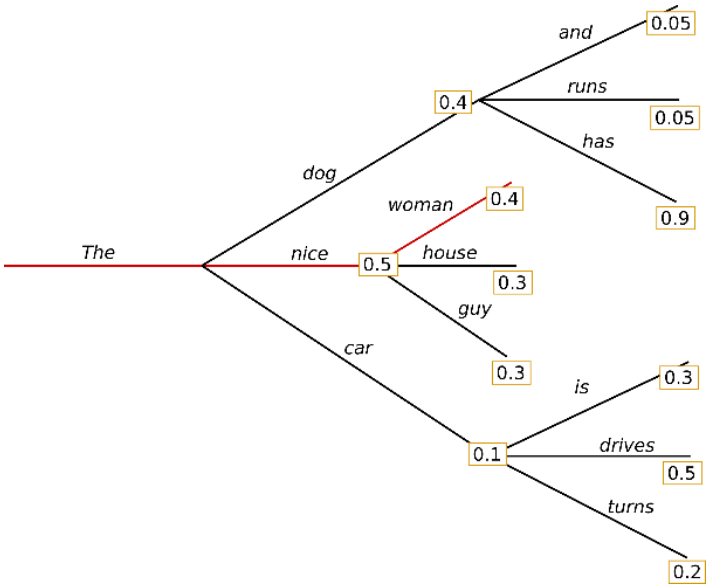
1. Strategies: Describe the detail of the following generation strategies:

- Greedy

  選擇當前 step 機率最大的 token 來當作這一輪預測的 token。

  $$w_t = argmax_w P(w|w_{1:t-1}) \; at \; each \; timestep \; t$$

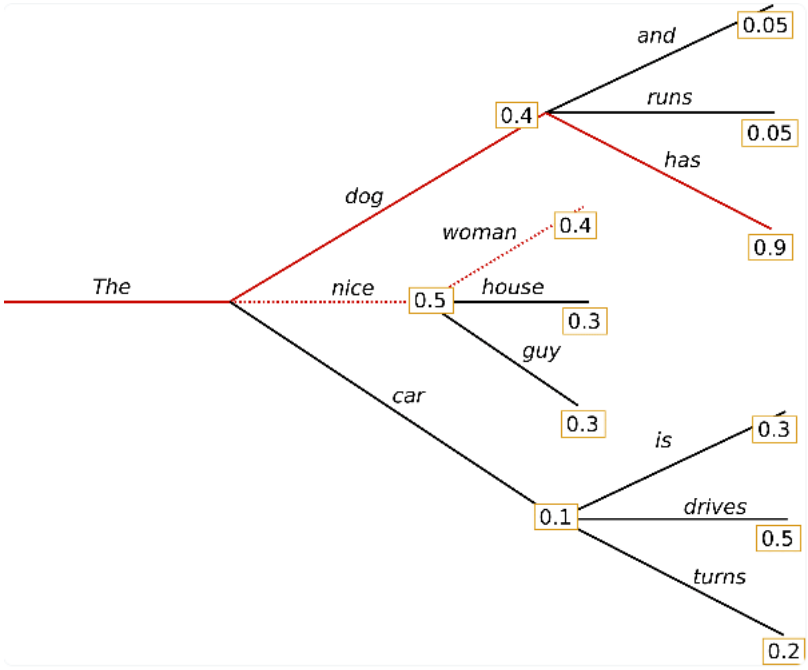
  以下圖為例，從”The”開始，會選擇機率最大的”nice”，接著再選擇基於前面選擇的下一輪機率最大的”woman”。總體的機率為 0.5*0.4=0.2。由於不需要考慮其他 step 的資訊，只需要選擇此輪的最大值，因此此演算法算是 decoding strategy 中最有效率且效果不差的了。



- Beam Search

  與 greedy 演算法相比，Beam Search 能夠降低錯過其他最終機率較高的可能性，Beam Search 與 greedy 其實十分類似，只是相較於 greedy 只會保留一個最大機率的可能，Beam Search 會保留 num_beams 個最大機率的選擇，最後選擇 overall 機率最高的當作最後的輸出。以 num_beams=2 與下圖為例，此時會保留”dog”與”nice”作為可能的輸出，接著下一輪(“dog”, “has”)的機率為 0.36，(“nice”, woman”)的機率為 0.2，而因為(“dog”, “has”)的 overall 機率最高，因此最後的輸出就會為”The nice woman”，與 greedy 相比就可以發現，此方法能夠捕捉到 greedy 漏掉但其實 overall 機率較大的可能。

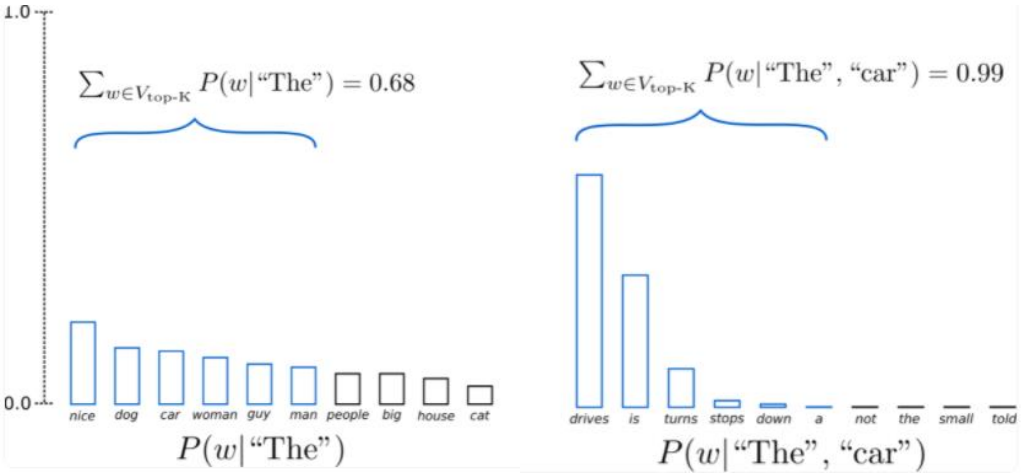
ADL HW3 report



- Top-k Sampling

使用 Top-k Sampling 的時候，只會將機率前 k 高的 token 保留下來並依據原本的機率重新分配他們的採樣機率。如下圖左所示，在計算”The”的下一個字的時候，有”nice”, “dog”, “cat”, “woman”, “guy”, “man”, “people”, “big”, “house”, “cat”這麼多可能，但此時我們設定 k 為 6，因此採樣的時候只可能從”nice”, “dog”, “cat”, “woman”, “guy”, “man”中選出下一個字。

然而這種做法仍然存在著一些問題，那就是無法動態的調整 k 的數值，有些分佈可能原本就很集中在少數的字，這時候 k 就可能沒辦法發揮很大的作用，像是下圖右所示，雖然”down” 與”a”的機率都很小但仍然是前 k 高因此不會被剔除掉，但這個方法還是能夠很有效的過濾掉“not”, “the”, “small”, “told”這些明顯不合理的選項。此方法也有可能限制模型的創意能力，像是下圖左的 “people”, “big”, “house”, “cat”其實都是十分通順的，但因為限制了 k 的數量，就可能讓模型永遠只能從比較 popular 的字來做選擇。

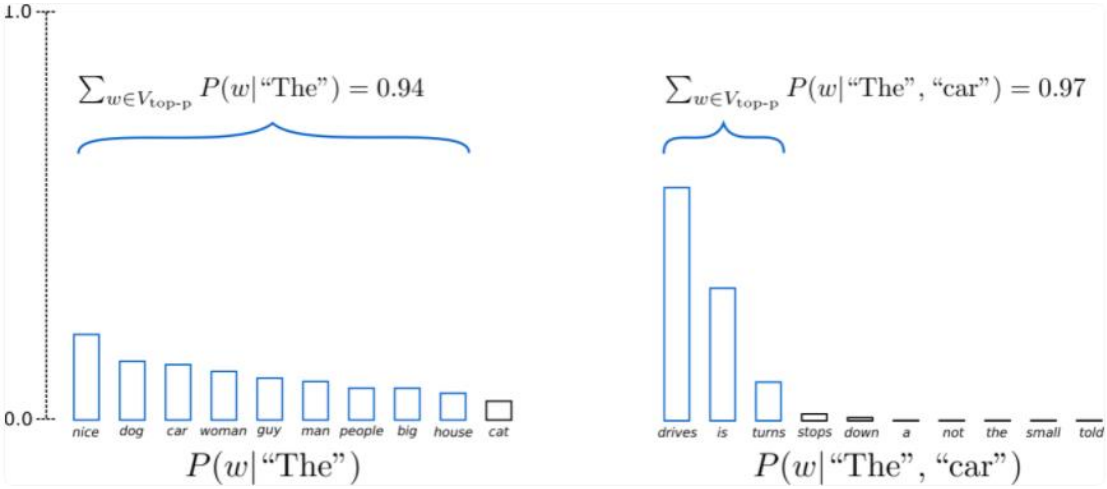


網媒所碩一 R10944028 蔡易儒

ADL HW3 report

- Top-p Sampling

與 Top-k Sampling 不同，Top-p 是想辦法取得一個最小的 token set，使其從機率最大開始累加的機率能夠大於預先設定好的 p，如下圖左所示，若 p 設定成 0.94，那麼就會從機率大的 token 開始累加機率來增加 token 的 set 大小，直到這個 set 所有的 token 機率和大於 p。接著重新分配 set 內的 token 機率已滿足機率和為 1 的設定。如此一來 Top-p Sampling 就可以解決了 Top-k 無法根據分佈的情況動態調整的問題，如下圖左右對照，在左邊較為平均的分布中，取的範圍就可以較寬一點，而在右邊較為集中的分佈，則可以將範圍取的小一點。



- Temperature

Temperature 則是一個可以調整 token 機率分佈集中與平均的一個超參數，較高的 temperature 會使得分布較為平均，decoding 出來的結果可能會較有較多的 diversity，而較低的 temperature 會使分布更加集中，diversity 會降低。
下式的τ就是 temperature。

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

網媒所碩一 R10944028 蔡易儒

ADL HW3 report

2. Hyperparameters:

以下 output 皆以 id: 21710 為例。

Target: Anker 新款真無線藍牙耳機 Liberty Air 2 Pro 引進台灣市場

a. Try at least 2 setting of each strategy and compare the result.

	1-p	1-r	1-f	2-p	2-r	2-f	l-p	l-r	l-f
Greedy	0.2896	0.2260	0.2450	0.1035	0.085	0.0902	0.2622	0.2040	0.2213
output	Anker 推出真無線藍牙耳機 確定引進台灣市場								
Sample	0.2105	0.1899	0.1934	0.0661	0.0615	0.0616	0.1874	0.1687	0.1718
output	Anker 預購新款真無線藍牙耳機 正式引進台灣市場								

可以明顯發現使用 Greedy 的結果比 Sample 還要好非常多，這個原因也並不意外，因為使用最基礎的 Sample 就有可能採樣到機率較低，而明顯不符合自然人語氣的詞。

Num_beams	1-p	1-r	1-f	2-p	2-r	2-f	l-p	l-r	l-f
Beam 2	0.2933	0.2414	0.2562	0.1116	0.0945	0.0989	0.2651	0.2176	0.2310
output	Anker 推出真無線藍牙耳機 確定引進台灣市場								
Beam 4	0.2953	0.2495	0.2616	0.1162	0.1002	0.1040	0.2673	0.2253	0.2364
output	Anker 推出真無線藍牙耳機 確定引進台灣市場								

使用 Beam Search 的結果也如意料之中都會比 Greedy 還要好，Greedy 其實可以當作 num_beams=1 的特例，而 num_beams 越大，其結果應該就會越佳，因為較不會漏掉前面機率較低但整體機率較高的選項，而最後能選到整體機率較大的答案。但是 num_beams 越大也會使 inference 的時間越長，因為必須維護與記憶的資訊也較多。

Top_k	1-p	1-r	1-f	2-p	2-r	2-f	l-p	l-r	l-f
10	0.2468	0.2109	0.2206	0.081	0.072	0.0737	0.2196	0.1878	0.1963
Output	Anker 打造真無線藍牙耳機 確定引進台灣市場								
20	0.2293	0.1994	0.2066	0.0725	0.0665	0.0665	0.2042	0.1774	0.1837
Output	Anker 打造真無線藍牙耳機 確定正式引進台灣市場								
40	0.2145	0.1921	0.1967	0.0672	0.0624	0.0626	0.1911	0.1710	0.1751
output	Anker 預購新款真無線藍牙耳機 正式引進台灣市場								

一開始先做了 k=20 與 k=40 的實驗，雖然使用 Top-k 都有比一般的 Sample 還要好，但還是無法跟 Greedy 一樣好，且隨著 k 的提升，表現還會下降，因此很有可能代表至少在此 dataset 上，只需要考慮機率前幾高的 token 就可以了，因此又再做了 k=10 的實驗，發現其結果是當中最好的，稍微印證在這個 dataset 上取小一點的 pool size 的表現會較佳的想法。

網媒所碩一 R10944028 蔡易儒

ADL HW3 report

Top_p	1-p	1-r	1-f	2-p	2-r	2-f	l-p	l-r	l-f
0.6	0.2637	0.2191	0.2321	0.0922	0.080	0.0828	0.2364	0.1962	0.2078
output	Anker 推出真無線藍牙耳機 加入 App 設定手勢								
0.8	0.2415	0.2060	0.2156	0.0808	0.071	0.0731	0.2159	0.1841	0.1926
output	Anker 打造真無線藍牙耳機 確定引進台灣市場								
0.92	0.2292	0.2000	0.2074	0.0739	0.067	0.0679	0.2039	0.1779	0.1845
output	Anker 打造真無線藍牙耳機 確定引進台灣市場								

可以發現使用 Top-p 的結果比使用 Top-k 還要好，而從理論上來看，能夠動態調整 pool size 的 Top-p 也的確應該要比 Top-k 要好。另外，與 Top-k 的結論類似，使用較低的 Top-p，可以讓 pool size 變小，因此在這個 dataset 上的表現就會較佳，尤其是取到 p=0.6 時，可能 k=1 或 2 就可以達到了累加的機率大於 0.6。

Temperature	1-p	1-r	1-f	2-p	2-r	2-f	l-p	l-r	l-f
1.5	0.1444	0.1504	0.1422	0.034	0.037	0.034	0.1270	0.1319	0.1247
output	Anker 預賣新款真無線藍牙耳機 正式搶進台灣市場								
1.2	0.1820	0.1731	0.1719	0.052	0.051	0.049	0.1614	0.1530	0.1521
output	Anker 預購新款真無線藍牙耳機 正式加入台灣市場								
0.8	0.2411	0.2052	0.2152	0.081	0.070	0.0729	0.2144	0.1823	0.191
output	Anker 推出真無線藍牙耳機 加入 App 設定手勢								
0.5	0.2694	0.2192	0.234	0.094	0.079	0.0832	0.2407	0.1956	0.209
output	Anker 推出真無線藍牙耳機 加入 App 設定手勢								

調整 Temperature 影響的是整體分佈的集中或平均性，而能夠發現降低 Temperature 能夠讓 performance 變好，而將 Temperature 大於 1 則會讓表現變得更差，雖然這樣是讓機率分佈變得集中，但是可能過度地壓低了其他選項的機率，而導致整個表現十分的差。若將 Temperature 設的低一點反而能夠讓一些過度集中的機率分佈稍微平均一些，而若是原本機率就比較平均的，影響也不會太大，有可能是因為這樣 Temperature 設成 0.5 能夠讓 decoding 的表現有所提升。

b. What is your final generation strategy?

最後選擇直接使用 Beam search，且設定 num_beams=4，雖然 decoding 的時間較長，但是最後的結果相較於其他不 deterministic 的方法更為穩健。

網媒所碩一 R10944028 蔡易儒

ADL HW3 report

Bonus: Applied RL on Summarization

- Algorithm: Describe your RL algorithms, reward function, and hyperparameters.

我使用的方式是在原本 supervised 的 training 中，將 compute_loss 的地方加入另外一個 loss component，並使用 rl_ratio 來調整使用原本 loss 與 rl_loss 的比例。

RL 的 algorithm 是 policy gradient，由於不同時間 decode 出來的字在算 rouge 的時候是一樣重要的，因此是等到完整的句子生成完才計算 rouge，也沒有使用 discounting 的 reward，rl_loss 是將預測的 logits 做 decode 與 detokenize 得到完整的句子，並將其與原本的 ground truth summary 算 ROUGE score，將 rouge-1-f、rouge-2-f、rouge-l-f 三者平均當作 reward，並算出 logits 的 maximum-likelihood loss，將兩者相乘當作是 rl_loss。為了模型的穩定，算 reward 時是一次算一個 batch 的，以避免零星的極端值擾亂訓練的過程。

$$Reward = Rouge_{1f}(y_t^*, y_t) + Rouge_{2f}(y_t^*, y_t) + Rouge_{lf}(y_t^*, y_t)$$

$$L_{ml} = -\sum_{t=1}^{n'} \log p(y_t^* | y_1^*, \ldots, y_{t-1}^*, x)$$

$$rl_{loss} = -Reward * L_{ml}$$

$$total\_loss = loss * (1 - rl\_ratio) + rl\_loss * rl\_ratio$$

網媒所碩一 R10944028 蔡易儒

ADL HW3 report

Hyperparameters:

- Pretrained model: google/mt5-small
- Learning rate: 1e-3
- Max source length: 1024
- Max target length: 128
- Effective batch size(per device batch size*gradient accumulate step *# of device): 2*8*1 =16
- Epoch: 3
- Optimizer: Adafactor
- Scheduler warm up ratio: 0.1
- rl_ratio: 0.6

- Compare to Supervised Learning: Observe the loss, ROUGE score and output texts, what differences can you find?

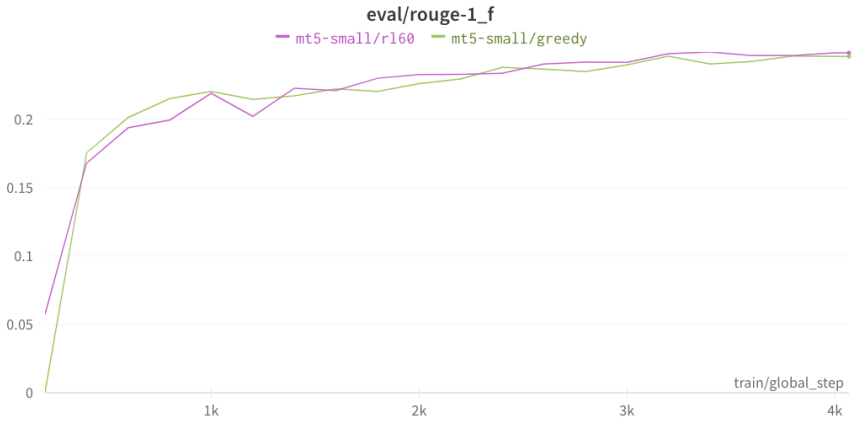
	1-p	1-r	1-f	2-p	2-r	2-f	l-p	l-r	l-f
Supervised	0.2896	0.2260	0.2450	0.1035	0.085	0.0902	0.2622	0.2040	0.2213
RL	0.2910	0.2292	0.2480	0.1049	0.0662	0.0914	0.2621	0.2060	0.223

為了能夠更好的比較兩者的差距，兩者都使用最基本且 deterministic 的 decoding strategy，也就是 Greedy。其實在訓練的時候，由於兩者的超參數幾乎都一樣，只加了 rl_ratio 的部分，因此兩模型的表現極為相似，且設計的 rl_loss 與 supervised 的 loss 也有許多類似的部分如L_{ml}其實就是原本 supervised 的 loss。total_loss 其實可以簡化為

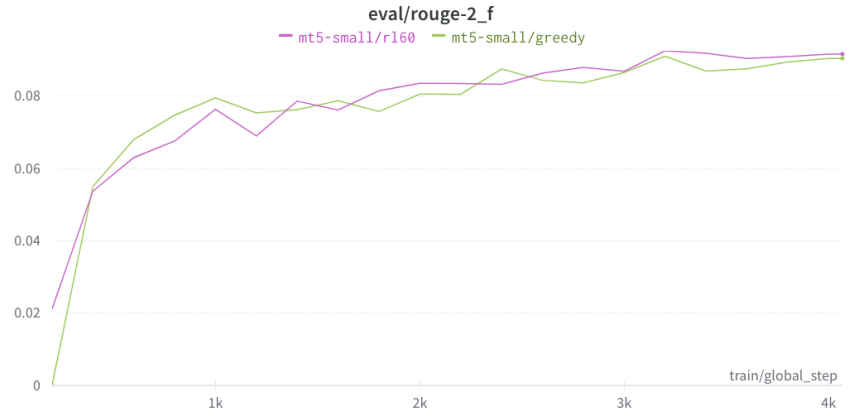
$$total_{loss} = (1 - rl_{ratio}) * loss - Reward * rl\_ratio * loss$$
$$= loss * (1 - rl_{ratio} - Reward * rl\_ratio)$$

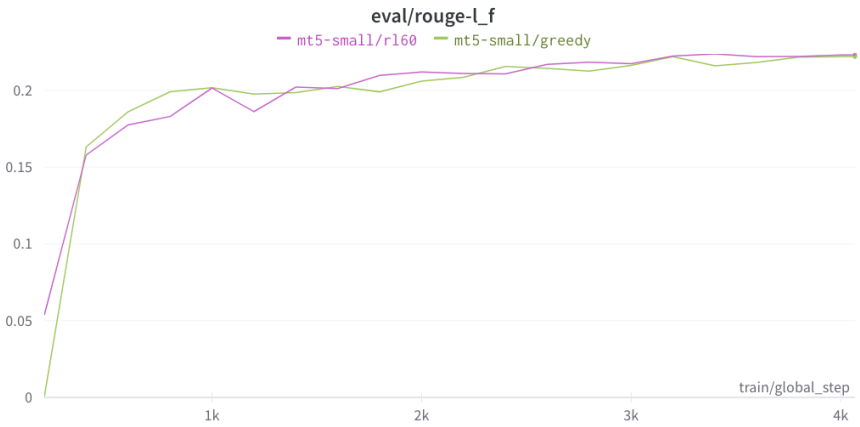
因此加上了 RL loss 之後，其實只是變成動態的調整當次更新的 loss 大小，某種程度上可以視為動態的調整 learning rate。

Learning curve:



ADL HW3 report





從 Learning curve 來看，使用 rl_loss 的結果就如同一開始預料的，並沒有相差太多。另外使用 rl_loss 在最後都有稍微超過沒有使用 rl 的版本，可能是因為在最後的微調階段，因為 rl_loss 而導致的 reweighting 將表現微調上去了。

Output text:

1. id: 21952

Ground truth: "萬寶龍簡化設計的 Summit Lite 系列智慧手錶 售價新台幣 27300 元"

w/o RL: "萬寶龍推出新款新款 打造智慧手錶\n"

w/ RL: "萬寶龍推出新款智慧手錶 採用簡化設計\n"

在此範例可以看到兩者都有抓到“萬寶龍”、“智慧手錶”的概念，但是使用了 RL 的結果有多捕捉到了“簡化設計”的概念。而兩者在“萬寶龍”都會接下去“推出新款”因此“採用簡化設計”的機率可能在沒有使用 RL 的情況下較低，低於<EOS>的機率，導致沒有生成出來。而“簡化設計”可以提高整體的表現，雖然 loss 可能也會偏低，但因為有 Reward 的幫助，最後有救回來。

2. id: 22873

Ground truth: "黑嘉嘉甜穿 K-SWISS 防潑水老爹鞋 網：「我也想要一雙」"

w/o RL: "黑嘉嘉搶先演繹 網友:我想要一雙\n"

w/ RL: "黑嘉嘉搶先演繹!K-SWISS 新款老爹鞋 網友直呼:我想要一雙\n"

ADL HW3 report

從這個例子可以看到兩者其實都有抓到大部分的語意，然而使用 RL 會為了要得到更多的 Reward 而盡量將每個詞捕捉近來提高 Recall，像是“K-SWISS”、“老爹鞋”，都是使用了 RL 才捕捉近來的詞彙。

3. id: 23065

Ground truth: " 2 月全台活動總整理！武陵櫻花季、月影燈季...春節+228 假期重點玩法"

w/o RL: "全台「全台活動」懶人包!\n 全台「3 大展覽」懶人包\n"

w/ RL: "全台「全台活動」全台!\n 全台「七天年假」、七天年假、櫻花林必訪\n"

從這個範例來看，可以發現使用使用 RL 仍然會有許多重複的詞，推測原因可能是因為若模型認為詞屬於答案，那麼即使我多輸出幾次，分數也並不會掉太多，這可能是因為使用 ROUGE 的算法算是 Recall oriented 的評估方式，因此多輸出幾次答案，並不會造成 Recall 的損失，以上述範例計算 ROUGE score。

w/o RL:

{'rouge-1': {'r': 0.1579, 'p': 0.25, 'f': 0.19355},
 'rouge-2': {'r': 0.1053, 'p': 0.1429, 'f': 0.1212},
 'rouge-l': {'r': 0.1579, 'p': 0.25, 'f': 0.19355}}

w/ RL:

{'rouge-1': {'r': 0.21053, 'p': 0.3077, 'f': 0.25},
 'rouge-2': {'r': 0.10526, 'p': 0.1053, 'f': 0.1053},
 'rouge-l': {'r': 0.21053, 'p': 0.3077, 'f': 0.25}}

可以明顯算出沒有使用 RL 的 f score 平均下來仍會較有使用差，即使有使用 RL 生成出來的字有明顯的重複且明顯不符合一般人類的語言邏輯。