Q1: Model (2%)

Model (1%)

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

根據作業指引使用 mT5 transformer model，並在 mT5 model 的 decoder 上多加了一個 linear layer，計算下一個 word 的 logits。

mT5 Encoder 由多個相同 layer 組成，每個 Encoder layer 中有兩個 sub-layers，用來做 multi-head self-attention mechanism 和 position-wise fully connected feed-forward network，並配合使用 residual connection 和 layer normalization。

mT5 Decoder 除了和 Encoder 相似的技術外，還插入了對 Encoder 的 output 做 multi-head attention 的 sub-layer。

Encoder 先將 source sequence map 成 sequence representation。在 decode 時，decoder 用 encoder output 和之前生成的 sequence 來生成新的 sequence，交給多加的 linear layer 的計算 logits，再用自訂的 strategy 來找到下一個 word。

Preprocessing (1%)

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

用 SentencePiece 將 maintext 和 title tokenize、truncate 並 padding，確保 source_len 和 target_len 不超過設定的最大長度(按照作業指引設為 256, 64)。在算 loss 時，因為會忽略 target 中的 [PAD]，所以不用額外 data cleaning。

Q2: Training (2%)

Hyperparameter (1%)

Describe your hyperparameter you use and how you decide it.

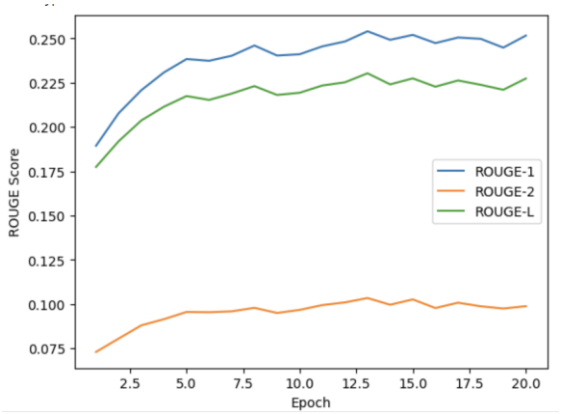
觀察 learning curves 變化、參考作業規定和 https://github.com/huggingface/transformers/blob/main/examples/pytorch/summarization/run_summarization_no_trainer.py 後設置參數如下：

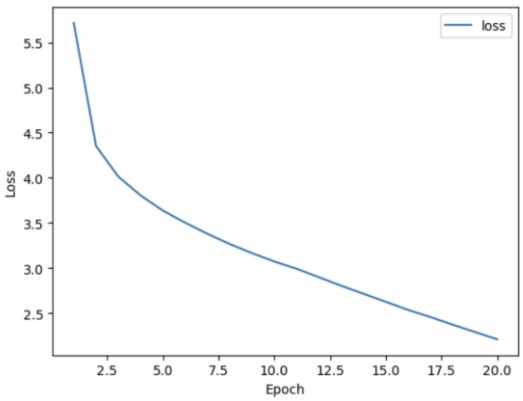
1. Pretrained weights: google/mt5-small
2. Max source length: 256
3. Max target length: 64
4. Batch size: 16(per_device)*4(gradient_accumulation_steps)
5. Learning rate: 1e-3
6. Optimizer: AdamW (weight decay: 1e-2)
7. Epoch: 20
8. Scheduler: linear

Learning Curves (1%)

Plot the learning curves (ROUGE versus training steps)

使用 validation file (20% train.jsonl)，generation strategy 為 beam_size=5。





Q3: Generation Strategies(6%)

Strategies (2%) Describe the detail of the following generation strategies:

Greedy

最簡單的 generation strategy，根據 model 的機率分佈，每次都選擇機率最高的下一個 word。成果通常連貫性較好，但可能過於確定，缺乏創造性。

Beam Search

和 Greedy 相比，Beam Search 尋找下一個 word 時保留 Beam size 個最有可能的序列。用這些序列判斷下個 word 的機率，繼續留下最有可能的序列，持續選擇直到滿足終止條件。Beam size 越大，搜索範圍越廣，但計算成本也越高。因為保留多個可能，所以在保持一定相關性時成果會更多樣化，但會較耗費資源。

Top-k Sampling

和 Greedy 相比，Top-k Sampling 不一定選擇機率最高的 word，而是對 k 個最可能的 word 做 Sampling，使成果有隨機和多樣性，同時保持一定相關性。k 的大小要小心設置，過大可能降低連貫性，過小則會損失多樣性。

Top-p Sampling

和 Top-k Sampling 差別在 Top-p Sampling 不是從前 k 個 word 中選，而是從累積機率超過 p 以上的 word 集合中選擇。每次可選的 word 數量不一定，改善 Top-k Sampling 在機率分布型態不同時可能遺漏高機率的選項或考慮了機率太低的選項，使機率分布 narrow 或 broad 時 model 都可以有較好的表現。

Temperature

Temperature τ 用來縮放 softmax 函數機率分佈的形狀，控制成果的多樣性。$P(w)$ 是 token w 調整後的機率分布，其中 s_w 為 token w 的 logits，V 代表

token set。
溫度高使分佈更平均，成果較隨機、有創意。
溫度低使分佈更尖銳，容易選到最可能的選項，成果較集中、沒有變化。

$$P(w) = \frac{\exp(s_w/\tau)}{\sum_{w' \in V} \exp(s_{w'}/\tau)}$$

Hyperparameters (4%)
Try at least 2 settings of each strategies and compare the result.
使用 validation file (20% train.jsonl)

	rouge-1	rouge-2	rouge-l	time
Greedy	23.05731783	8.3923354880	20.896218226	2:43
Beam search 3	24.45480801	9.7243275008	22.230738921	4:59
Beam search 5	24.64798020	9.9736772603	24.144357647	6:29
Beam search 7	25.20804530	10.521611910	22.809700441	7:15
Top-k 0	14.61336112	4.1491943073	13.141396307	2:39
Top-k 5	21.15120515	6.9488496801	18.872863400	2:52
Top-k 10	19.92741751	6.3815116478	17.740644756	2:41
Top-k 20	19.16295215	6.0681355429	17.093973697	2:48
Top-p 0.25	22.65759623	8.1375212681	20.430410462	4:56
Top-p 0.5	21.79308162	7.6285634955	19.627071347	4:52
Top-p 0.75	20.40351208	6.8392651348	18.188811368	4:55
Temperature 0.5	21.96469761	7.6646348483	19.726985254	2:46
Temperature 0.7	20.07524198	6.6833779756	17.919448909	2:45
Temperature 1	17.55655752	5.3576767325	15.738347779	2:48

What is your final generation strategy? (you can combine any of them)
經過實驗後和 Greedy 相比，sampling 並不會增加 rouge score，反而會影響表現，因此決定不使用 sampling，而是使用 Greedy。
Beam search 可以增加 model 的表現，但是隨著 beam size 的增加，消耗的資源也會跟著增加，而且效益沒有顯著提高，甚至倒退。因此，在效能和資源中間取得平衡，決定使用 beam size = 5。