ADL HW3

tags: `write-up`

備註：所有 f1-score 都有乘上倍率 100。

Q1: Model (2%)

Q1.1 Model (1%)

我 pre-trained model 使用了 MT5ForConditionalGeneration 的 "google/mt5-small"，config 部分皆採取原 model 的預設值。

mT5 是建立在 T5 之上，使用更多種語言的資料庫、更多的超參數選擇、改動 activation function 使得下游 (metric, task) 表現更好；T5 使用了 C4 (Common Crawl's web crawl corpus) 資料做訓練，基於 BERT 與 transformer 的模型架構，並加上 Encoder-Decoder 與 mask (填空) 的改動，由於 encoder 的 self-attention 與 decoder 的 auto-regressive，還有 Fine-tuning 階段採取並行 (同步) 更新 encoder & decoder 使得模型表現的較為優秀。

config


```
{
  "architectures": [
    "MT5ForConditionalGeneration"
  ],
  "d_ff": 1024,
  "d_kv": 64,
  "d_model": 512,
  "decoder_start_token_id": 0,
  "dropout_rate": 0.1,
  "eos_token_id": 1,
  "feed_forward_proj": "gated-gelu",
  "initializer_factor": 1.0,
  "is_encoder_decoder": true,
  "layer_norm_epsilon": 1e-06,
  "model_type": "mt5",
  "num_decoder_layers": 8,
  "num_heads": 6,
  "num_layers": 8,
  "pad_token_id": 0,
  "relative_attention_num_buckets": 32,
  "tie_word_embeddings": false,
  "tokenizer_class": "T5Tokenizer",
  "vocab_size": 250112
}
```

Q1.2 Preprocessing (1%)

tokenization 部分使用 AutoTokenizer 的 "google/mt5-small"，策略一樣採取 SentencePiece 做出分詞，其切割出 subword units 使用了 byte-pair-encoding (BPE) [Sennrich et al.] and unigram language model [Kudo.] 兩種方法。

preprocessing 部分，因為起初使用 huggingface transformers 的 summarization sample code 遇到諸多麻煩，所以自己寫了一個 MT5Dataset 繼承自 torch.utils.data.Dataset，做的操作除了 tokenizer 之外就只有將 input/output truncate 至大小為 256/64 長度。

Q2: Training (2%)

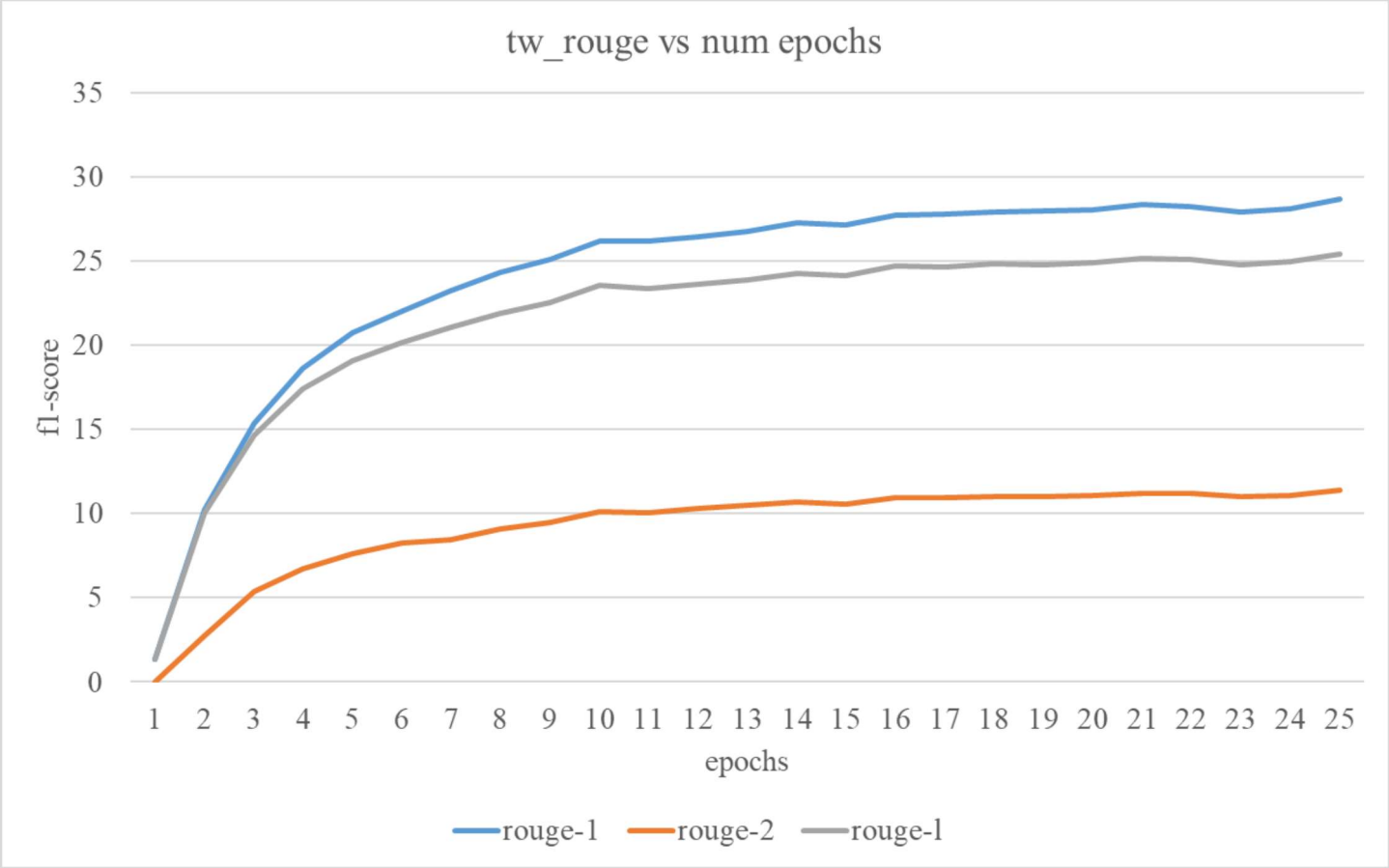
Q2.1 Hyperparameter (1%)

> test tw_rouge under num_beams = 2

Hyperparameter	Value
batch_size	16
max_input	256
max_output	64
num_epoch	25
optimizer	torch.optim.AdamW
lr	1e-4
weight_decay	5e-5

為了能夠在手邊的 GPU 能夠訓練與 loss value 可以下降，還有在有限時間內通過 baseline，所以選擇了上述的 hyperparameters。

Q2.2 Learning Curves (1%)



	rouge-1 f1	rouge-2 f1	rouge-l f1
Public baseline	22.0	8.5	20.5
my best score	26.670	10.699	23.809

Q3: Generation Strategies(6%)

Q3.1 Stratgies (2%)

Greedy

num_beams = 1，顧名思義，只記錄目前最大可能性的結果。

Beam Search

num_beams > 1，如此可以保存前 num_beams 個最佳可能，避免在遍歷 tree 時 missing hidden high probability word sequences。

Top-k Sampling

根據 probability 做排序，超過 top_k 的選擇將被丟棄，如此再次計算每個選擇的機率，換句話說，只考慮前 top_k 最有可能的結果。

Top-p Sampling

在 top_k 設定之下，選擇字詞 probability 超過 top_p 且字詞長度最短的選擇。

Temperature

溫度又好比熱力學的狀態，又或者可以理解成 "創造力"，Temperature 設定越靠近 0，表示輸出越趨向於 model 的 argmax (aka. max likelihood)；而 Temperature 越大，表示輸出越趨向於其他可能，例如機率第二大 & 第三大的可能字詞。

Q3.2 Hyperparameters (4%)

Q3.2.1 compare the result

> for Top-k Sampling: num_beams = 5, top_p = 1.0, temperature = 1.0
> for Top-p Sampling: num_beams = 5, top_k = 150, temperature = 1.0
> for Temperature: num_beams = 5, top_k = 150, top_p = 1.0

Stratgies	rouge-1 f1	rouge-2 f1	rouge-l f1
greedy (num_beams = 1)	25.143	9.446	22.489
beam_search (num_beams = 2)	26.194	10.320	23.428
beam_search (num_beams = 3)	26.442	10.541	23.663
beam_search (num_beams = 5)	26.670	10.699	23.809
beam_search (num_beams = 7)	26.609	10.748	23.767
Top-k Sampling (top_k = 25)	26.207	10.260	23.436
Top-k Sampling (top_k = 50)	26.047	10.301	23.273
Top-k Sampling (top_k = 75)	26.108	10.339	23.322
Top-k Sampling (top_k = 100)	26.216	10.369	23.380
Top-k Sampling (top_k = 125)	26.091	10.318	23.370
Top-k Sampling (top_k = 150)	26.174	10.268	23.329
Top-p Sampling (top_p = 0.75)	26.293	10.383	23.509
Top-p Sampling (top_p = 0.8)	26.128	10.238	23.345
Top-p Sampling (top_p = 0.85)	26.135	10.312	23.344
Top-p Sampling (top_p = 0.9)	26.030	10.249	23.250
Top-p Sampling (top_p = 0.95)	26.060	10.243	23.342
Top-p Sampling (top_p = 1.0)	26.174	10.268	23.329

Stratgies	rouge-1 f1	rouge-2 f1	rouge-l f1
Temperature (temperature = 0.8)	25.134	9.789	22.678
Temperature (temperature = 1.0)	26.174	10.268	23.329
Temperature (temperature = 4.0)	5.238	0.138	4.493

Q3.2.2 final generation strategy

經過測試發現 beam_search (num_beams = 5) 還是表現比較好。

Bonus: Applied RL on Summarization (2%)

Algorithm (1%)

我採取助教的建議，使用 policy gradient 計算 num_beams=5 的 ROUGE-L F1-score 作為 reward (loss 的 multiplier)；其餘參數則與 Q2.1 Hyperparameter 描述相同。

Compare to Supervised Learning (1%)

> all test under num_beams = 5

type	rouge-1 f1	rouge-2 f1	rouge-l f1
supervised-learning	26.670	10.699	23.809
reinforcement learning	26.859	10.940	24.020

reinforcement learning model 使用從 fine-tune mT5 supervised-learning 最好表現的 model checkpoint，之後再訓練 10 個 epoch 找出表現最佳的 RL 模型。

由於 supervised-learning fine-tune model 表現已經不錯 (接近 overfit)，所以在 loss & output texts 看不出多大的差異。但是因為 RL policy gradient 以 ROUGE-L F1-score 作為 reward (學習對象)，所以其 f1-score 還可以再推升大約 1 個單位。