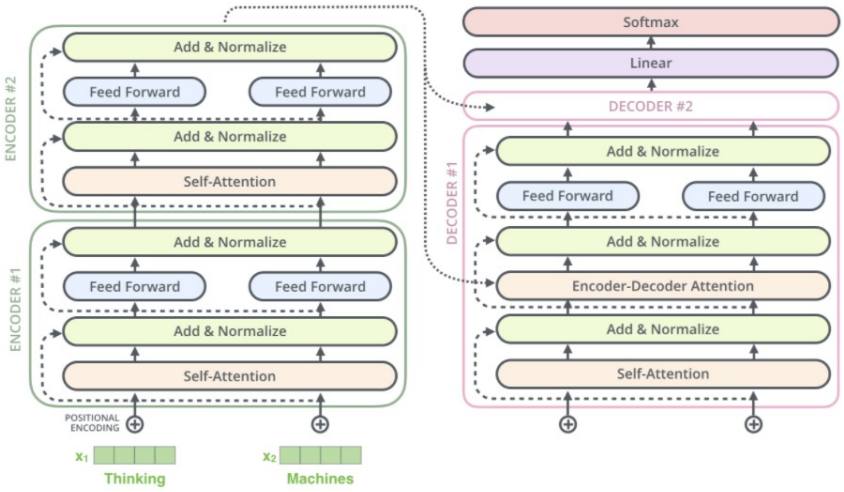
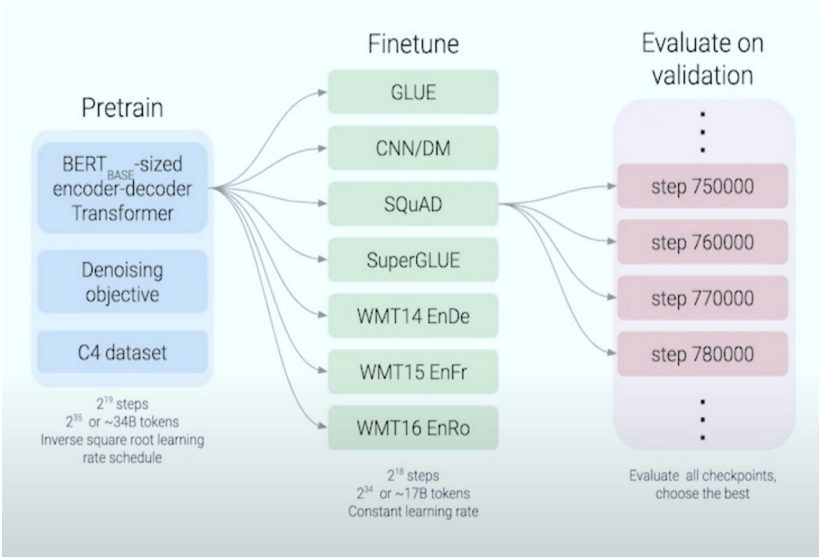
Q1: Model

- Model structure


```
{
  "_name_or_path": "google/mt5-small",
  "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_max_distance": 128,
  "relative_attention_num_buckets": 32,
  "tie_word_embeddings": false,
  "tokenizer_class": "T5Tokenizer",
  "torch_dtype": "float32",
  "transformers_version": "4.18.0",
  "use_cache": true,
  "vocab_size": 250100
}
```



T5 是一種 Transformer Encoder-Decoder 模型,使用 BERT-style 式的 mask 方法,並使用 Replace Span 的 mask 策略,其中 mask 比例為 15 %,長度為 3 最佳。

forward 函式將 source text tokenizer output 當成 encoder input ,並將最後一層 encoder output 當作 decoder hidden states,並將 target text tokenizer output 往右位移,作為 decoder input,經過 attention mechanism 之後產生 decoder output,最後經過 linear 層得到產生每個 subword 的機率。

generate 函式會則是將前面 decode 的結果當作 decoder 的 input, 依序產生結果。

- Preprocessing

tokenization: t5 tokenizer 為 sentencepiece, 是由 byte pair encoding 以及 unigram language model 組成, 會將 sentence 切成 subwords。

Q2: Training

- Hyperparameter

max_source_length=1024:Hugging Face default

max_target_length=128:Hugging Face default

per_device_train_batch_size=4:此為本次實驗硬體設備之極限。

gradient_accumulation_steps=8

num_train_epochs=5:5 個 epoch 已收斂

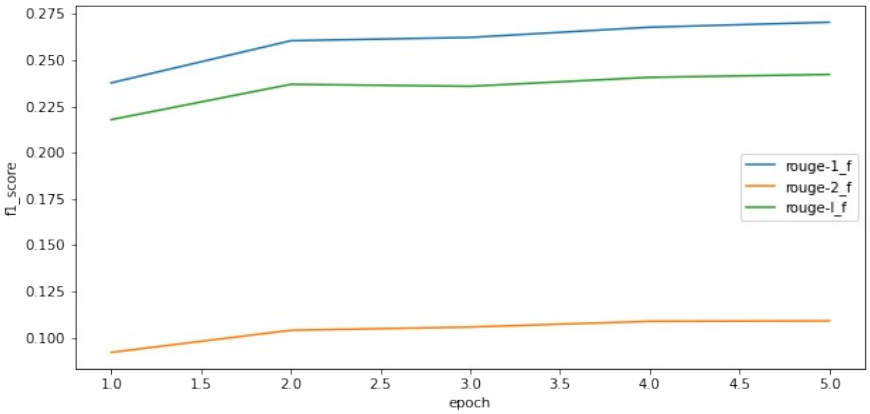
adafactor=True:使用 adafactor 比起 adam 可以降低G P U memory 使用量

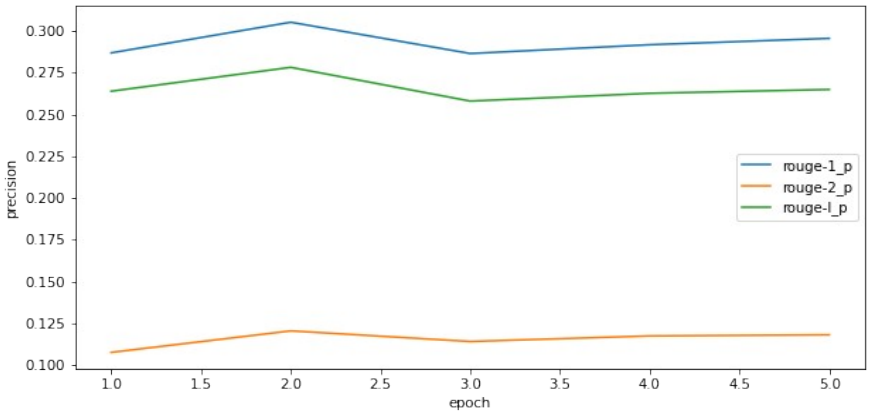
learning_rate=1e-3:設太小(eg.2e-5)訓練結果不好,故設成 1e-3 有不錯的結果

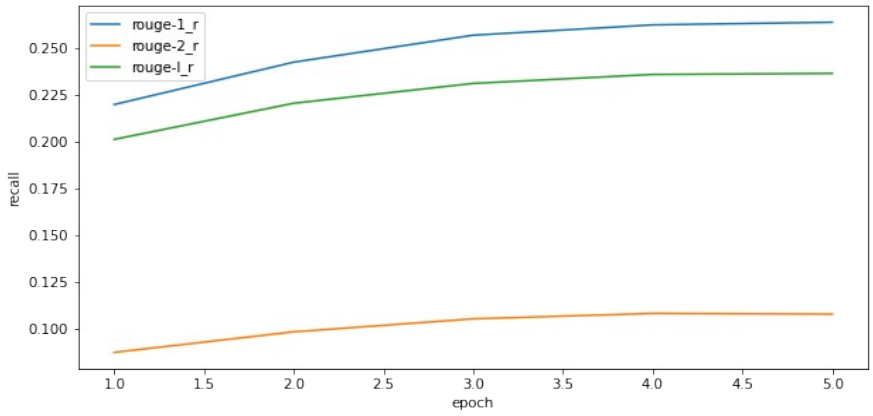
warmup_ratio=0.1

num_beams=5:再往上設時間複雜度太高,且效果提昇有限

- Learning Curves







Q3: Generation Strategies

- Describe the detail of the following generation strategies
  - Greedy:選擇機率最高的 word
  - Beam Search:每個 time step 都選前 n_beams 機率最大的候選，下一個 time step 則從這一個 time step 的候選中找出前 n_beams 機率最大的候選
  - Top-k Sampling:從機率為前 k 個高的 word 中 sample 下一個 word
  - Top-p Sampling:從機率高累加到低直到>=p 之 word set 當中 sample 下一個 word
  - Temperature:對 softmax 做 sharpen 或 smoothing，小於 1 為 sharpen，大於 1 則為 smoothing
- Hyperparameters
  - 比較所有 decode algorithm 在不同設置之下之結果

	Greedy	Beam Search(n_beams=3)	Beam Search(n_beams=5)	Top-k Sampling(k=25)	Top-k Sampling(k=50)	Top-p Sampling(p=0.7)	Top-p Sampling(p=0.9)	Sampling(Temperature=1e-5)	Sampling(Temperature=1.0)	Sampling(Temperature=10.0)	Sampling(Temperature=1e5)
rouge-1_f	0.257	0.267	0.270	0.214	0.203	0.213	0.187	0.255	0.164	0.002	0.002
rouge-1_p	0.293	0.294	0.295	0.231	0.219	0.230	0.198	0.294	0.170	0.001	0.001
rouge-1_r	0.243	0.259	0.264	0.210	0.200	0.209	0.188	0.240	0.169	0.010	0.007
rouge-2_f	0.095	0.106	0.109	0.070	0.067	0.074	0.063	0.095	0.050	0.000	0.000
rouge-2_p	0.106	0.116	0.118	0.075	0.071	0.079	0.066	0.106	0.052	0.000	0.000
rouge-2_r	0.092	0.105	0.108	0.070	0.066	0.074	0.064	0.091	0.052	0.000	0.000
rouge-l_f	0.229	0.239	0.242	0.190	0.181	0.190	0.168	0.229	0.147	0.002	0.002
rouge-l_p	0.261	0.264	0.265	0.205	0.196	0.205	0.177	0.264	0.152	0.001	0.001
rouge-l_r	0.216	0.232	0.237	0.186	0.179	0.186	0.168	0.215	0.151	0.009	0.007

    - overview:由圖可見 Greedy,Beam Search,和 Sampling(Temperature=1e-5)時有過 baseline,其他演算法皆過不了 baseline,可以見得非 Sampling 的 algorithm 比 Sampling algorithm 來的好,因為 Sampling algorithm 隨機性太大了。整體而言演算法表現排序如下:Beam Search>Greedy>Top-p Sampling>Top-k Sampling>Sampling

- Greedy vs Beam Search(no greedy)

  Greedy 每次只取機率最高者為答案,但如果某個 time step 正確答案並非機率最高者,則會連帶影響後面 time steps。而 Beam Search 保留了 n_beam 個答案,雖然因此時間複雜度較高,不過解決了 Greedy 的問題,可以選到更正確的答案。

- Top-k Sampling vs Top-p Sampling vs Sampling

  Top-k Sampling 當中,k 越大,下一個 word 的可能性越多,diversity 越高,結果越差;同理,Top-p Sampling 當中,p 越大,下一個 word 的可能性越多,diversity 越高, 結果越差。

  Top-k Sampling 每次取固定數量的 word set 做選擇,而 Top-p Sampling 會依照 next word distribution 動態調整 word set 大小, 為 Top-k Sampling 之改良版,由結果可以看出 Top-p 確實比 Top-k 好一點。

  Sampling 則隨機性太高,因此在三者之中結果最差。

- Temperature

  Temperature<1,會將機率 sharpen, 造成機率大的更大, 小的更小,導致機率大的更加容易取到,若取趨近於 0 的值,則類似 greedy(eg.Sampling(Temperature=1e-5));而 Temperature>1 則會將機率 smooth, 使機率趨於平均分佈,若取趨近於無限大的值,則類似 Random Sampling(eg.Sampling(Temperature=1e5)),結果非常差。

- Final generation strategy: Beam Search (n_beams = 5)