魏廷儒 r10521601
CSIE : Applied Deep Learning
November 30, 2022

ADL HW3 Natural Language Generation

Question 1. Model

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

我採用 google 的 mt5/small 做爲我的 pretrained model，本模型用的用 bert-encoder-decoder 的架構，decoder 出來的結果再經過 linear layer 得到每個 subword 預測的機率，最後我們在用 greedy, top-k, top-k, beam-search 等策略去選擇我們要預測的 subword。


```
{
  "_name_or_path": "./tmp/tst-summarization",
  "architectures": [
    "MT5ForConditionalGeneration"
  ],
  "d_ff": 1024,
  "d_kv": 64,
  "d_model": 512,
  "decoder_start_token_id": 0,
  "dense_act_fn": "gelu_new",
  "dropout_rate": 0.1,
  "eos_token_id": 1,
  "feed_forward_proj": "gated-gelu",
  "initializer_factor": 1.0,
  "is_encoder_decoder": true,
  "is_gated_act": 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": "4.24.0",
  "use_cache": true,
  "vocab_size": 250112
}
```


FIGURE 1. Model config google mt5/small.

2) Preprocessing

MT5Tokenizer 會先將 input 文本做分詞做 tokenization。

Question 2. Training

使用 google mt5/small 做爲 pretrained model

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

   max source length 我設爲預設的 1024，max length 也是預設的 128。經由測試我發現 learning rate 設在 1e-4 會有不錯的表現，另外我也有嘗試不同 batch size 與 gradient accumulation 的組合，發現調大 batch size 與 gradient accumulation 並不會有比較好的 performance，不過因爲我 train 了 20 個 epochs，考量到時間因素，因此我最後將 batch size 設爲 4，gradient accumulation 設爲 1。

2) Plot the learning curves.

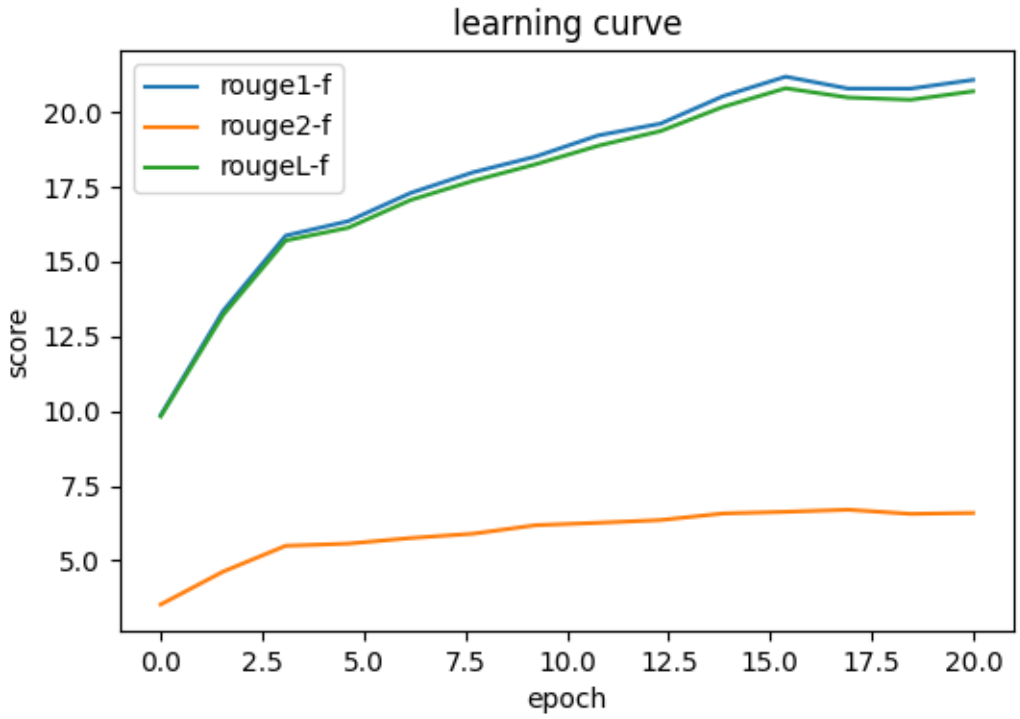

FIGURE 2. Learning curve.

Question 3. Startgies AND Hyperparameters Describe the detail of the following generation strategies:

1) greedy

Choosing the most probable word (argmax)

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

	greedy
rouge_1_r	0.215236
rouge_1_p	0.224150
rouge_1_f	0.213975
rouge_2_r	0.073004
rouge_2_p	0.074073
rouge_2_f	0.071512
rouge_L_r	0.190272
rouge_L_p	0.198087
rouge_L_f	0.189032

FIGURE 3. greedy approach.

2) beam search

Keeping track of the k most probable sequences and finding a better one. Beam search reduces the risk of missing hidden high probability word sequences by keeping the most likely num beams of hypotheses at each time step and eventually choosing the hypothesis that has the overall highest probability.

我們可以很明顯的看出採用 beam search 的策略後，performance 會比 greedy 好非常多，並且當 num beam 愈大，performance 會愈好，但是預測的時間也會需要愈多，因此如何在效率與模型表現之間做權衡將會是這個策略的重點。

	num_beams=3	num_beams=5	num_beams=10	num_beams=50
rouge_1_r	0.267452	0.270683	0.270686	0.272793
rouge_1_p	0.288383	0.288358	0.284328	0.282079
rouge_1_f	0.270349	0.272021	0.270332	0.270317
rouge_2_r	0.108517	0.109671	0.110147	0.111980
rouge_2_p	0.114429	0.114675	0.113730	0.114224
rouge_2_f	0.108325	0.109074	0.108900	0.110083
rouge_L_r	0.237924	0.240315	0.240686	0.242562
rouge_L_p	0.256476	0.255984	0.252707	0.251048
rouge_L_f	0.240372	0.241385	0.240222	0.240368

FIGURE 4. Beam search.

3) top k

Randomly sample the word via the probability distribution instead of argmax. Sample the word via distribution but restricted to the top-k probable words. 使用 top k 可以看到在 k=50 會比原本 greedy 的 performance 好，另外如果 temperature 設爲小於 1 的值，也可以提高模型的表現。而當 temperature 大於 1 時，也就是對預測結果做 smoothing，反而會使得 performance 下降。

	top_k=50	top_k=50, temperature=0.7	top_k=50, temperature=1.4	top_k=100
rouge_1_r	0.217448	0.235708	0.183220	0.211468
rouge_1_p	0.224119	0.255577	0.175979	0.214970
rouge_1_f	0.214846	0.239165	0.174887	0.207977
rouge_2_r	0.073185	0.086317	0.054490	0.071277
rouge_2_p	0.074128	0.090336	0.051243	0.071013
rouge_2_f	0.071491	0.086032	0.051274	0.069293
rouge_L_r	0.191953	0.208613	0.160687	0.186400
rouge_L_p	0.197755	0.226169	0.154458	0.189987
rouge_L_f	0.189537	0.211575	0.153367	0.183478

FIGURE 5. Top k.

4) top p

Instead of sampling only from the most likely K words, in Top p sampling chooses from the smallest possible set of words whose cumulative probability exceeds the probability p. 使用 top p 策略會比 greedy 的 performance 好，另外如果 temperature 設爲小於 1 的值，可以提高模型的表現。而當 temperature 大於 1 時，也就是對預測結果做 smoothing，反而會使得 performance 下降。

	top_p=0.9	top_p=0.95	top_p=0.95, temperature=0.7	top_p=0.95, temperature=1.4
rouge_1_r	0.228495	0.224378	0.239083	0.190844
rouge_1_p	0.240319	0.234775	0.261451	0.185355
rouge_1_f	0.228435	0.223777	0.243617	0.183160
rouge_2_r	0.080259	0.077518	0.087695	0.057796
rouge_2_p	0.082407	0.080299	0.092582	0.055319
rouge_2_f	0.079109	0.076793	0.087742	0.054927
rouge_L_r	0.201298	0.197908	0.212482	0.166771
rouge_L_p	0.211759	0.207063	0.232146	0.162343
rouge_L_f	0.201178	0.197303	0.216319	0.160172

FIGURE 6. Top p.

5) Temperature

A trick is to make the distribution sharper or lower (increasing the likelihood of high-probability words and decreasing the likelihood of low-probability words), which is so-called temperature of softmax.

6) What is your final generation strategy?

考量上述 startgies 與參數的組合，其中表現最好的是 beam search (n=50) 的策略，但是這組參數在 predict 時太耗時，我用 v100 32g 的顯卡預測完 public.jsonl 須費時 58 分鐘，在預測效率的考量之下，我最後選擇的策略爲 beam search (n=5)。