ADL HW3 Report

學號：B08502141　　姓名：石旻翰　　系級：電機四

1 Model

1.1 Model Architecture

The model I use is mT5 which basically follows standard encoder-decoder architecture with multiple stack of attention blocks. The pretrained model I used is google/mT5-small, which is loaded from transformers, and is fine-tuned on the task of this homework. The encode-decoder architecture enables model to be easily applied on summarization. During training, the padded mini-batch text is fed into T5-Encoder with expected title fed into the T5-Decoder. In inference, the T5-Encoder takes the text and the T5-Decoder would generate the predicted logits.

1.2 Preprocess

The preprocess program follows the original recipe of T5. I use sentencepiece tokenizer, and each context is truncated or padded to 256 , and each title is trucated to 64. No data cleaning techniques are applied.

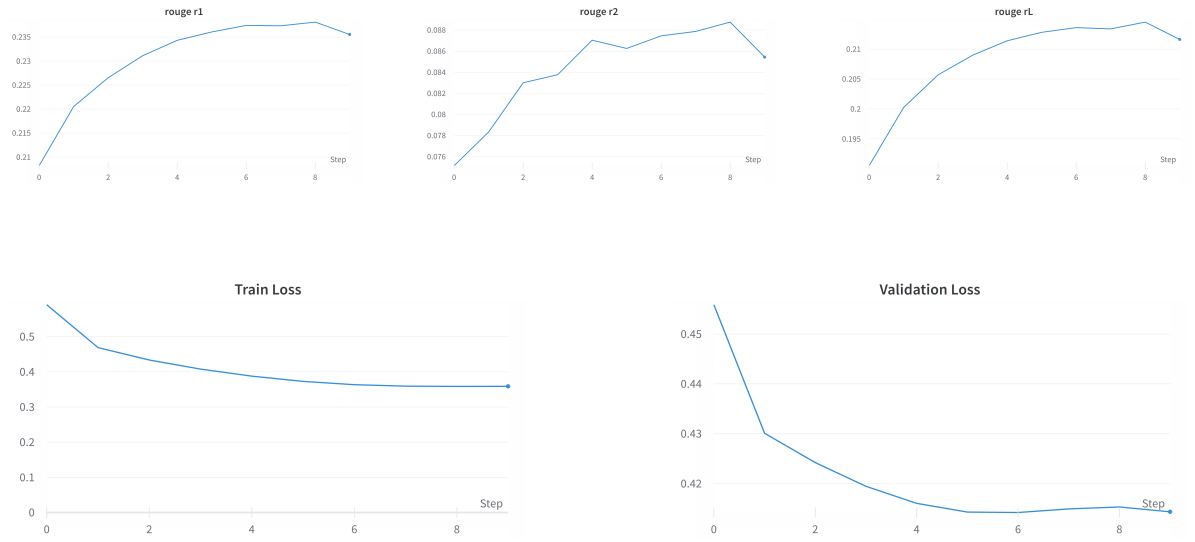
2 Training

2.1 Hyperparameter

- Optimizer: `Adafactor` from transfromers
- Scheduler: `get_cosine_schedule_with_warmup` from transfromers
- Seed: 1126
- Use fp16: True
- Learning rate: 0.0003
- Weight decay: 0.0001
- Batch size: 1
- Accumulation steps: 8
- Epochs: 10
- Max source length: 256
- Max target length: 64

2.2 Learning Curves

I use `public.jsonl` as my validation set, and through calculating the mean score between rouge-r1, rouge-r2, and rouge-rL, the model with the highest mean score will be saved.



3 Generation Strategies

3.1 Stratgies

3.1.1 Greedy

Greedy decoding chooses the most probable word for each preceding word. However, the output could be poor for the lack of backtracking, and generally, the disadvantage of greedy algorithm is that some path with higher probability may not be explored.

3.1.2 Beam Search

Beam search is similar to greedy, if we set `num_beams` = 1, then it is the same with what greedy does. However, if we set `num_beams` > 1, it will keep tracking of `num_beams` most probable sequences to find a better solution than greedy does. Setting small `num_beams` might lead to incorrect or weird sentences, but setting large `num_beams` makes the procedure computational expensive, so the trade-off is essential when we select the setting of `num_beams`.

3.1.3 Top-K Sampling

Top-K sampling samples the word via distribution but restrict to the k probable words. If we set k = 1, it is the same with what greedy does. If we set k > 1, the output could be more diverse when k increasing.

3.1.4 Top-P Sampling

Top-P sampling samples from a subset of vocabulary with the highest probability. Denote selected word as w_i, and the algorithm can be writen in:

$$w_i \sim V^{(p)}, \text{where } V^{(p)} = \sup_{V' \subset V} \sum_{x \in V'} P\left(x \mid w_1, \ldots, w_{i-1}\right) \geq p \quad [3.1]$$

3.1.5 Temperature

Temperature is applied to the **Softmax**, to control the diversity which is applicable to any decoding algorithm. Output can be more diverse as temperature increases.

3.2 Hyperparameters

3.2.1 Results

Greedy
{'num_beams': 1, 'top_k': 200, 'top_p': 1.0, 'temperature': 1.0}

Use Greedy	Rouge-1	Rouge-2	Rouge-L
True	0.2382	0.0887	0.2146
False	0.1712	0.0536	0.1542

Table 1: Rouge score of applying Greedy or not

Beam Search
{'do_samples':False, 'top_k': 200, 'top_p': 1.0, 'temperature': 1.0}

num_beams	Rouge-1	Rouge-2	Rouge-L
1	0.2382	0.0887	0.2146
3	0.2492	0.0991	0.2243
5	0.2483	0.1001	0.2236
10	0.2477	0.1006	0.2231

Table 2: Rouge score of different settings of beam size

Top-K Sampling
{'do_samples':False, 'num_beams': 1, 'top_p': 1.0, 'temperature': 1.0}

K	Rouge-1	Rouge-2	Rouge-L
50	0.2412	0.0902	0.1996
100	0.2401	0.0881	0.2016
200	0.2382	0.0887	0.2146

Table 3: Rouge score of different settings of k

Top-P Sampling
{'do_samples':False, 'num_beams': 1, 'top_k': 200, 'temperature': 1.0}

P	Rouge-1	Rouge-2	Rouge-L
0.3	0.2329	0.0811	0.2011
0.5	0.2268	0.0792	0.2033
0.7	0.2177	0.0761	0.1998
1.0	0.2382	0.0887	0.2146

Table 4: Rouge score of different settings of p

Temperature
{'do_samples':False, 'num_beams': 1, 'top_k': 200, 'top_p': 1.0}

Temperature	Rouge-1	Rouge-2	Rouge-L
0.1	0.2382	0.0869	0.2093
0.3	0.2381	0.0887	0.2091
0.5	0.2380	0.0881	0.2144
0.7	0.2381	0.0897	0.2146
1.0	0.2382	0.0887	0.2146

Table 5: Rouge score of different settings of temperature

3.2.2 Conclusion

I use beam search with num_beams=10 as my final generation strategy.

4 Bonus:Applied RL on Summarization

4.1 Algorithm

4.2 Compare to Supervised Learning

References

[1] https://huggingface.co/google/mt5-small

[2] https://github.com/cccntu/tw_rouge

[3] https://huggingface.co/docs/transformers/main_classes/text_generation