Applied Deep Learning - Homework 3

Written By: Wu-Jun Pei (B06902029)

Q1: Model (2%)

1) Model

Following the guides on the slides, I mainly use a Seq2seq-style T5, including a T5Encoder, T5Decoder and a fully-connected layer on top of decoder to output the probability of each token. More formally, let

- e_i to be the encoder input token id for the i-th encoder input token, and d_i to be the decoder input token id for the i-th decoder input token
- the hidden state of each encoder input h_{e_i} can be obtained by $h_{e_{1..L}} = \text{T5Encoder}(e_{1..L})$
- the hidden state of each decoder input h_{d_i} can be obtained by $h_{d_{1..L'}} = \text{T5Decoder}(d_{1..L'}, h_{e_{1..L}})$
- finally, the output token o_i is obtained using the fully-connected head, $o_i = \text{LMHead}(h_{d_i})$

The pretrained weight I used is `google/mt5-small` following the suggestions from TAs.

2) Preprocessing

- We first tokenize the main text to tokens. After that, the token sequence is padded and truncated to length = 384.
- We first tokenize the title into tokens. After that, the token sequence is padded and truncated to length = 96.

Q2: Training (2%)

1) Hyperparameters

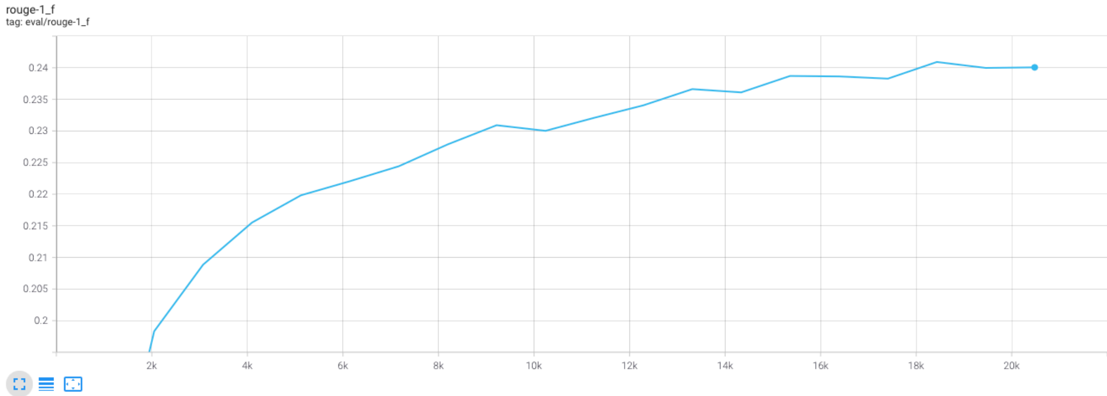
- **Optimizer**: Adam with learning rate 1e-4 and weight decay 5e-5 with linear learning rate scheduler (min_lr = 0)
- **Epochs**: 20 (the total training time is about 6 hours, and about one third of total training time is for evaluation)
    - The model mistakenly stopped at 20k steps because the disk space remaining on my device was not adequate, leading to a
- **Batch size**: 2 for training and 4 for evaluation

- **Gradient accumulation**: the gradient is accumulated for 8 steps
- **FP16**: activated

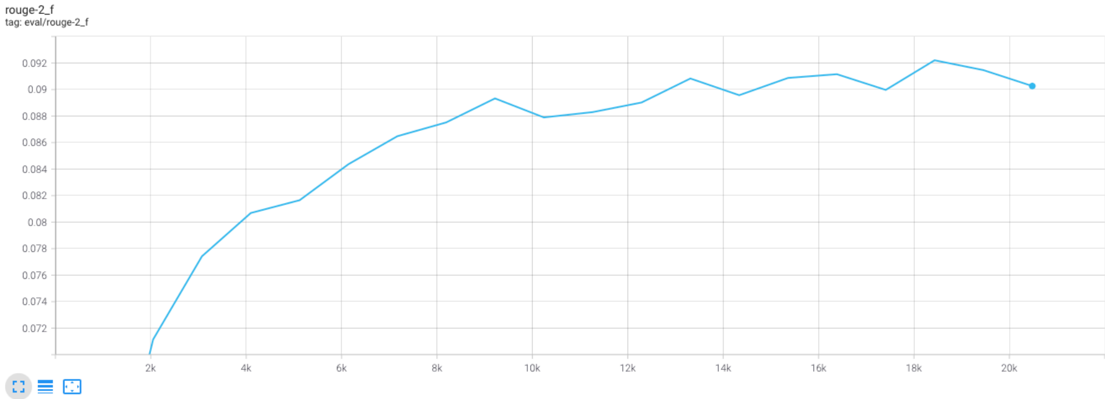
2) Learning Curve

> *The evaluation is performed every 1024 training steps. The generation strategy is greedy as it's the default strategy.*

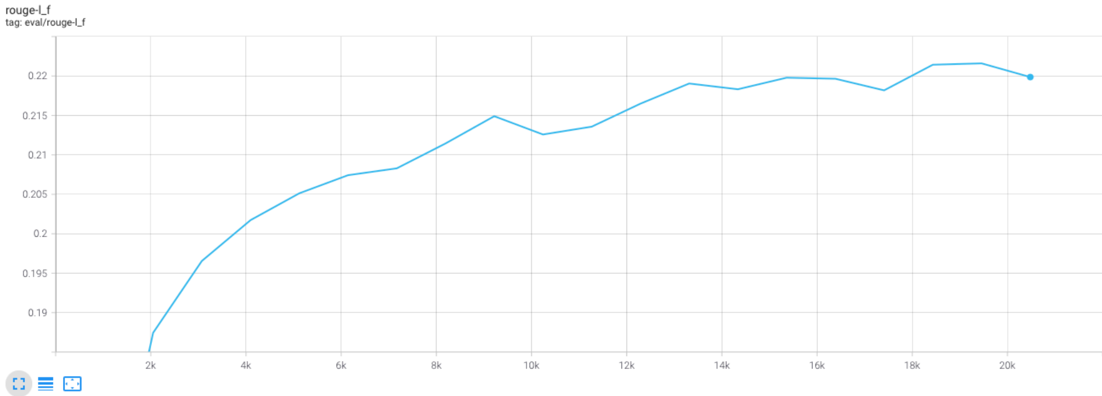
- **Rouge-1**



- **Rouge-2**



- **Rouge-L**



Q3: Generation Strategies (6%)

1) Strategies

- **Greedy**: Always use the token with the highest probability
- **Beam search**: Maintain the top *num_beams* sequences with top probability. Greedy can be considered a special case of beam search when *num_beams = 1*
- **Top-k sampling**: Sample only from the top *k* tokens.
- **Top-p sampling**: Sample only from the top tokens where the total probability of token pool is larger than *p*.
- **Temperature sampling**: Controls the diversity of the outcome from sampling. The *temperatrure* is actually the denominator of logits before calculating softmax.

2) Hyperparameters

Description	Rouge-1	Rouge-2	Rouge-L	Time (mm:ss)
Greedy	23.8308	9.0921	22.3980	01:38
Beam search (num_beams = 3)	25.1437	10.2419	23.4102	04:30
Beam search (num_beams = 5)	25.3738	10.4971	23.6096	07:09
Sampling baseline	19.4593	6.5109	17.7335	02:01
Sampling (num_beams = 3)	24.6182	9.7992	22.8299	05:26
TopK sampling (k = 10)	21:5361	7.5299	19.7783	02:01
TopK sampling (k = 25)	20.1094	6.7752	18.4633	02:01
TopP sampling (p = 0.25)	23.6406	8.9559	22.1158	05:14
TopP Sampling (p = 0.5)	22.9461	8.5789	21.2561	04:58
Temperature (t=0.5)	23:0771	8.6066	21.3575	01:59
Temperature (t=2)	10.6780	1.9175	9.5607	02:31

3) Final Strategy

My final strategy is beam search (num_beams = 3). With this strategy, I achieved the following metrics on validation dataset (`public.jsonl`).

- Rouge-1: 25.1437
- Rouge-2: 10.2419
- Rouge-L: 23.4102

The entire prediction process takes about 04:30 on my PC equipped with a 8G 2070 Super GPU.