ADL HW3

b08902045 資工三 袁紹奇

Q1: Model (2%)

Model (1%)

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

I use `google/mt5-small` from huggingface (https://huggingface.co/google/mt5-small) for text summarization. The model is similar to T5. It is a seq2seq model that is pretrained unsupervised in multilingual data. The structure is an encoder-decoder-based model. The encoder is based on BERT and transformer, so attention is introduced to encoding all the tokens of the maintext for generating the summary. The decoder, however, attends to only the previous generated tokens and it inputs the hidden state for the previous timestamp to output the whole title (the prediction of the model).

Preprocessing (1%)

Describe your preprocessing (e.g. tokenization, data cleaning and etc.)

For the input maintext and the title, I first remove all the "\n", "`", "'" and "\r". After that, the tokenizer of `google/mt5-small` will tokenize the input maintext and title to their corresponding IDs where their max length are set to 512 and 64, respectively. For those sentences that are shorter/longer than 512 and 64, the tokenizer would pad/truncate to their max length.

Q2: Training (2%)

Hyperparameter (1%)

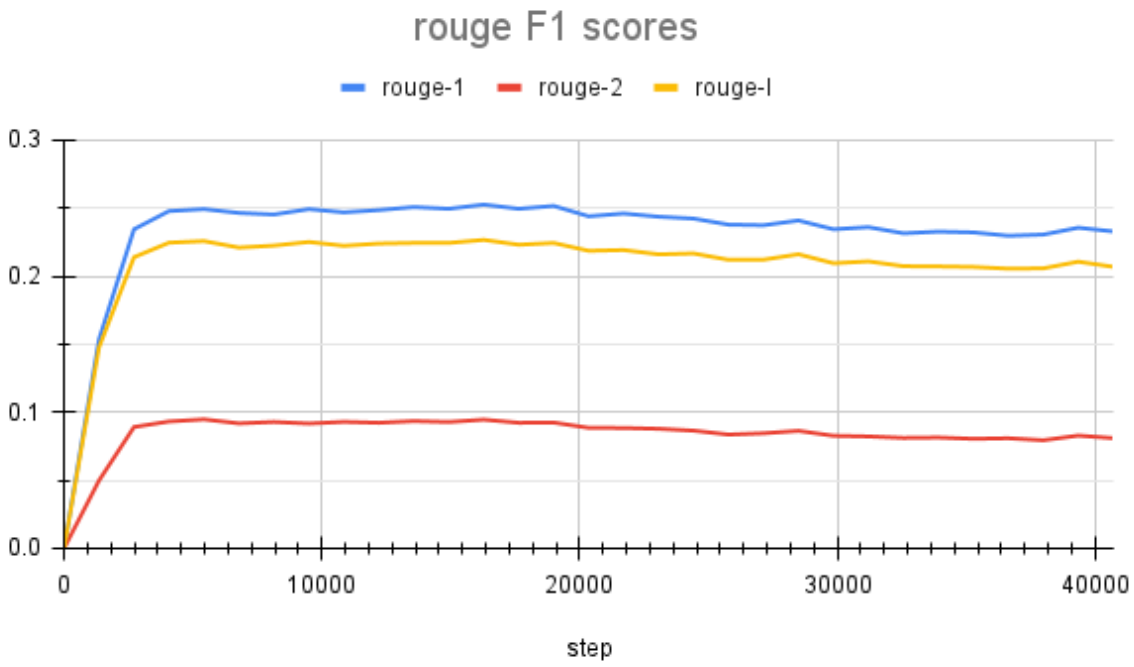
Describe your hyperparameter you use and how you decide it.

input max length	output max length	optimizer	learning rate	weight decay	training batch_size	loss function
512	64	AdamW	0.001	0.01	16	CrossEntropyLoss

Those hyperparameters are chosen based on the hyperparameters set during pretraining stage and efficiency.

Learning Curves (1%)

Plot the learning curves (ROUGE versus training steps)



I chose the F1 scores for the 3 rouge criteria. The data points are selected for each epoch, where each epoch takes 1357 steps.

Q3: Generation Strategies(6%)

Stratgies (2%)

Describe the detail of the following generation strategies:

- Greedy
  The strategy continues to select the token with the highest probability as the output sentence. The procedure stops once the `<END>` token is generated.
- Beam Search
  Beam Search has a hyperparameter B to choose the number of the beam that the model should maintain. Suppose that we have V number of vocabularies. The candidate of the next token would be chosen from the size of $B * V$. The model would choose the highest B of these candidates as the output sequence. Note that when $B = 1$, the case is the same as the Greedy search.
- Top-k Sampling
  We set the hyperparameter K. For each $P(x|S)$, that is, the probability of predicting token x given the previous sequence s, we collect those tokens with top-k probabilities. Then the model samples from the set with size K randomly based on the distribution. Note that when $K = 1$, the case is the same as the Greedy search.
- Top-p Sampling
  The sampling strategy is similar to top-k sampling. The main difference is instead of considering a fixed size k of candidates, the model considers all $P(x|S)$ that has the probabilities that are larger than pre-set $p \in [0, 1]$. Therefore the size of candidates varies for different sequences, and the model performs better than top-k sampling when the distribution is narrow or broad.
- Temperature
  The softmax temperature is a method to broaden or narrow the probability distribution. The hyperparameter τ should be set depending on whether we want more general or diverse outputs. Given the probability of predicting token x, we have

$$P(x) = \frac{e^{S_w}}{\sum_{w' \in V} e^{S'_w}}$$

The probability adjusted by temperature is

$$P(x) = \frac{e^{S_w/\tau}}{\sum_{w' \in V} e^{S_{w'}/\tau}}$$

We can then apply different sampling strategies to the shifted distribution.

Hyperparameters (4%)

Try at least 2 settings of each strategies and compare the result.

These are the results of different generation strategies. The scores are f1-scores of each critic scaled by 100.

Method	Rouge-1	Rouge-2	Rouge-l
Greedy	23.414	8.1076	20.992
Beam Search (b=3)	25.026	9.5880	22.525
Beam Search (b=10)	**25.217**	**9.9120**	**22.693**
Sample from distribution	15.726	4.6581	14.091
Top-k Sampling (k=50)	18.486	5.5539	16.383
Top-k Sampling (k=500)	16.298	4.8106	14.543
Top-p Sampling (p=0.5)	22.247	7.5805	19.934
Top-p Sampling (p=0.8)	20.997	6.9990	18.667
Temperature (tau=0.5)	22.242	7.4984	19.956
Temperature (tau=0.8)	20.351	6.5455	18.160

From the table above, we can see that greedy is better than all sampling methods. As in this task, we do not need much diversity while sampling gives us some tokens that are more unlikely to appear. The sampling-from-distribution method performs worst among all the other strategies because once the model predicts the tokens with low probability, the prediction of the next token would go south resulting in the low score of the whole sequence. Also, though a larger beam in generating the sequence has better performance, the cost of memory and time is much larger.

What is your final generation strategy? (you can combine any of them)

From the experiment, a larger beam size would result in high usage of memory and longer time, while the performance gain is rather small. In my opinion, the sampling strategy is not necessary for this task since we do not need much diverse or creative output. Therefore, I chose Beam Search with beam size=3 in this homework.

Bonus: Applied RL on Summarization (2%)

Algorithm (1%)

Describe your RL algorithms, reward function, and hyperparameters.

I use gradient policy as the RL algorithm, and the reward function is the f1 score of rouge-l divided by 0.205. That is,
`reward = rouge['rouge-l']['f']/0.205`.
The hyperparameters are basically the same as the original finetuning algorithm.

input max length	output max length	optimizer	learning rate	weight decay	training batch_size
512	64	AdamW	0.001	0.01	16

Compare to Supervised Learning (1%)

Observe the loss, ROUGE score, and output texts, what differences can you find?

The training goal of RL is slightly different from supervised learning. The original finetuning method is to fit the generation of each token in the training data. Whereas the RL method is to fit the rouge score of the generated sentences. The following experimental results are tested with beam size = 3, and top_k = 50. The scores are f1 scores and scaled by 100.

Method	Rouge-1	Rouge-2	Rouge-l
Supervised Learing	23.414	8.1076	20.992
Reinforcement Learning	25.493	9.5805	22.928

After observing the output of RL and finetuning, I found that the output title of RL is more reasonable and suitable than the other. In my experiment, the RL algorithm gained significant performance in the rouge scores, and the titles of RL's output are also better for human readers.