ADL HW3 Report R11944004

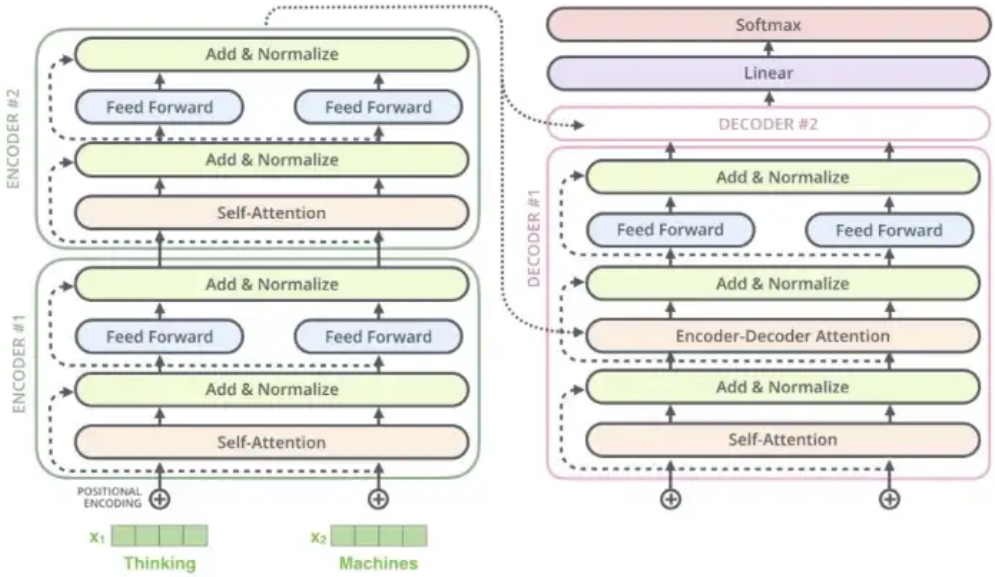
Student ID: R11944004

Name: 李勝維

Department: GINM11

Q1: Model

Model: Describe the model architecture and how it works on text summarization



Architecture

I use multilingual T5 (Xue et al. 2020) as my model. The architecture of mT5 is very similar to vanilla transformer encoder-decoder, but the activation function is gated-gelu and no dropout is used during pretraining instead. The variant I use is mT5-small, which consists of 8 encoder layers and 8 decoder layers with hidden size of 512.

Text Summarization

Since the mT5 is pretrained on the mC4 dataset, a large-scale, web-crawled, multilingual dataset, and a text-to-text pretraining objective, It can conduct text-to-text generation naturally. Paragraph summarization tasks can be viewed as a text-to-text generation: generate the summarization conditioning on the input paragraph. I further fine-tuned the mT5 on the udn.com news and optimize the model with teacher-forced maximum likelihood.

Preprocessing: Describe your preprocessing

Tokenization

I use the pretrained mT5 tokenizer to tokenize the data. The tokenizer splits the input Chinese sequence into characters and then transforms each character into its corresponding ids based on its dictionary.

After tokenization, the tokenizer truncates/pads the sequence and appends [BOS] and [EOS] tokens into the sequence.

Data Cleaning

After obtaining the generated titles, I clean other columns in the data and save the predictions with their id and title.

Q2: Training

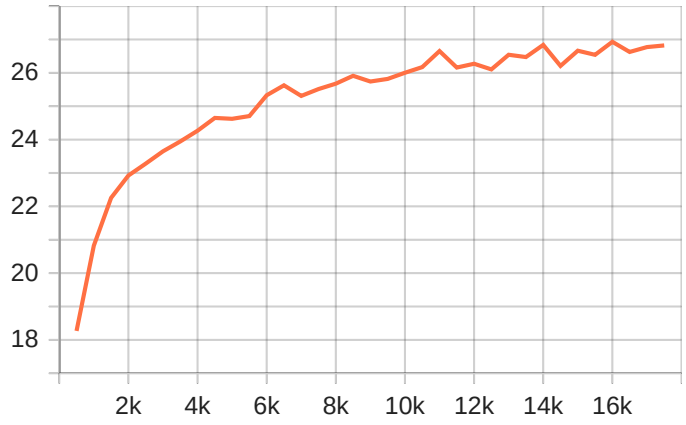
Hyperparameters:

- Training epochs: 15 → I have worked on text-summarization tasks before, empirically, these tasks require longer training time
- Batch size: 8 x 4(accumulation steps) → Usually larger batch size makes the model converges faster.
- Learning rate: 5e-5 → Since I train the model for a long time, it is better to use a smaller lr.
- Linear warm-up: 300 steps → Warm-up helps the model to converge faster.
- Auto mixed precision training with bfloat16 → Faster training and inference speed.
- Validation metric: I created a metric, “rouge-combined,” to select models. The details will be described in the RL(bonus) section as it is also the reward function I use.

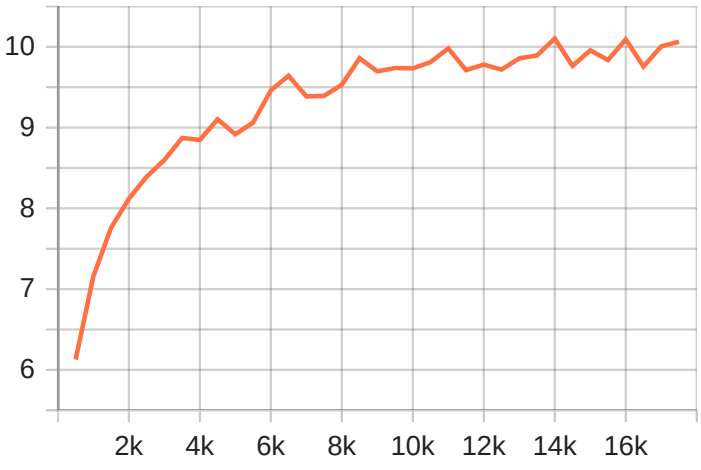
Learning Curves:

(Unit: 100 * f1-score for rouge-*)

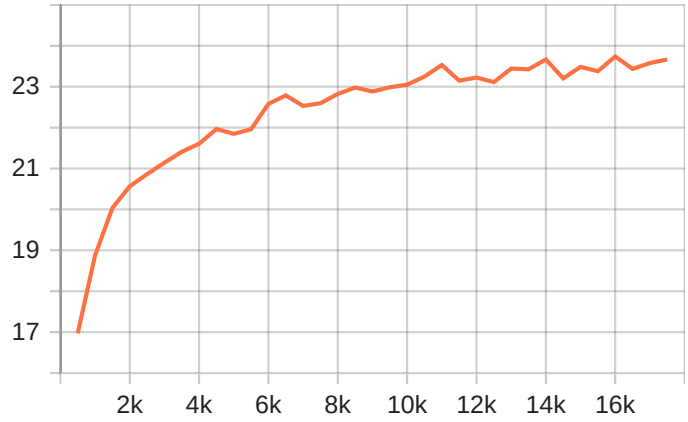
Rouge-1



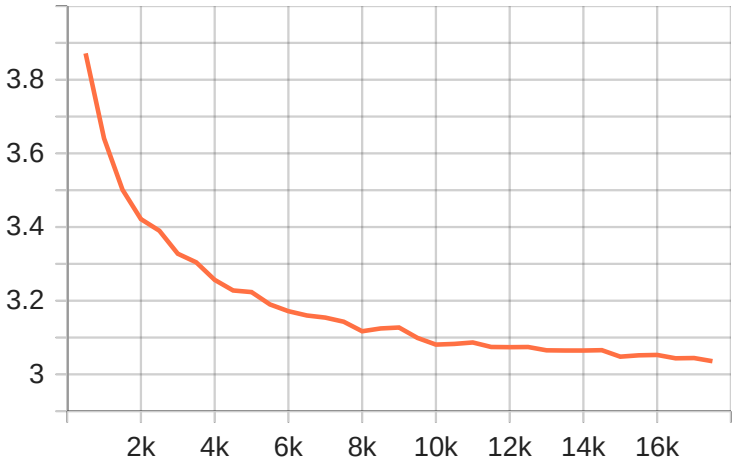
Rouge-2



Rouge-L



Validation Loss



Q3: Generation Strategies

Strategies: Describe the detail of the following generation strategies:

Greedy

Greedy search simply selected the word with the highest probability at each step.

Beam Search

Beam search keeps the most likely k candidates at each time step and chooses the sequence that has the overall highest probabilities eventually. Beam search with k of 1 degrades to greedy search.

Beam search selects the top k words greedily, this allows beam search to reduce the risk of missing higher probability word sequences with low probability starting words.

Top-k Sampling

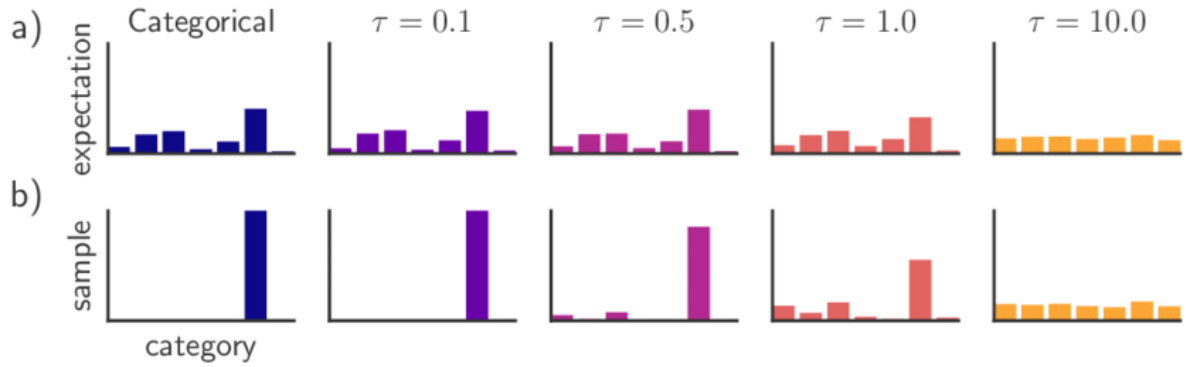
Similar to greedy search, top-k sampling selects one word at each step, but it samples a word from the top-k words in the output distribution instead of choosing the highest probability one.

Top-p Sampling

Similar to top-k sampling, top-p sampling samples the top-n words which have a probability sum exceeding p instead of the top-k words.

Temperature

an illustration of applying temperature:



What temperature does is it divides the numerator and the denominator during the softmax procedure:

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

This changes the distribution of SoftMax output: A small temperature makes the distribution “harder,” and a large temperature makes the distribution “softer.”

The temperature affects the result of top-k and top-p sampling strategies since it changes the shape of the distribution.

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

Decoding Strategy	Parameters	Rouge-1 (f1 score)	Rouge-2 (f1 score)	Rouge-L (f1 score)
Greedy	N/A	0.25642752528908475	0.09626477489706607	0.227963398822016
Beam	beams=3	0.26864378003137906	0.10695499746170836	0.23894097304377968
Beam	beams=5	0.2691455388529504	0.10810981521506378	0.23938370505963916
Beam	beams=7	0.2688296883863027	0.10829266532983617	0.23916674711631575
Top-k	k=50	0.2071020454795221	0.06735157424886933	0.18234690816337815
Top-k	k=100	0.19968123536447188	0.0649358007587434	0.17679397256116322
Top-p	p=0.80	0.2298666300121739	0.08054754037289177	0.20328263452853101
Top-p	p=0.92	0.21869647391676378	0.07446454797125682	0.19163321441336628
Temperature + Top-p	temp=0.7 p=0.92	0.24280042160148757	0.08760499965720947	0.21454306588181118
Temperature + Top-p	temp=2.0 p=0.92	0.12631527080848837	0.026390372551564175	0.10923342641491517

Since text summarization is more of a deterministic process, all sample-based methods do not perform well. Instead, we can apply sample-based methods on more stochastic tasks such as dialog systems.

Beam search outperforms all sample-based methods, while also surpasses greedy search by searching on a larger search space.

Top-p sampling with high temperature produces unreasonable sentences since the distribution almost degrades to a uniform distribution.

Example sentence produced by temperature of 2: "讓愛生子的孩子都滿懷 廖曉喬看見極細的「優雅白媽、我的妻兒」. 滿懷小孩"

What is your final generation strategy?

I use beam search with beam = 5 as my final decoding strategy since it suppresses other methods in this instance.

Bonus: Applied RL on Summarization

The code for is part is `RL_Trainer.py` and `RL_Summarization.py`

Algorithm: Describe your RL algorithms, reward function, and hyperparameters.

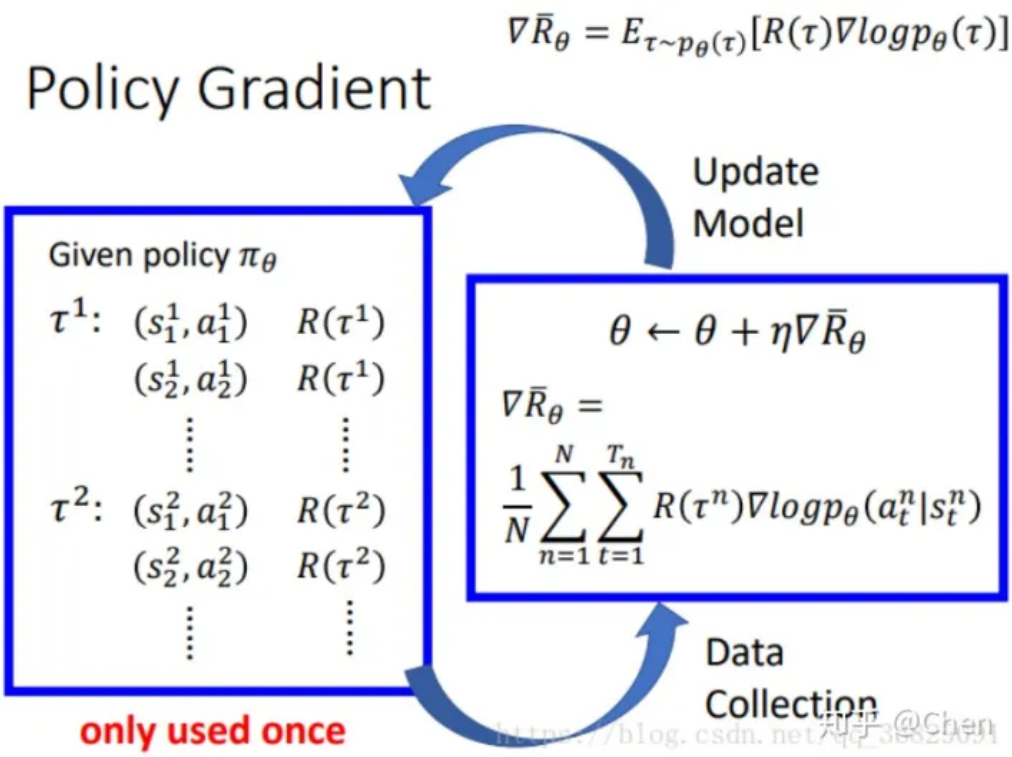
Experiment Settings

In this section, I will introduce three different settings to evaluate the effectiveness of applying RL on text summarization

1. Supervised finetuning from mT5 (Q2):
   This is the setting finetunes the pretrained mT5 by optimizing MLE objective via cross-entropy.
2. Intermediate supervised finetuning from mT5 + RL finetuning:
   This model is initialized from the results of (1), and further finetuned by RL algorithm.
3. RL finetuning from mT5:
   This model is initialized from the pretrained mT5 and finetuned by RL algorithm.

I use beam search with beam = 5 as the decoding strategies for all three settings.

RL Algorithm



To optimize the model, I use policy gradient (Sutton et al. 2000) with REINFORCE (Williams et al. 1992), which samples a large amount of (s, a) pairs, and update the model according to each trajectory's reward.

In other words, the policy gradients tends to maximize (or minimize) the probability of an action with given state according to the reward value.

Reward Function

The objective is to maximize the rouge-1, rouge-2, and rouge-L scores. Thus, I designed a metric called “rouge-combined” which considers all three metrics:

$$\text{rouge\_combined} = \frac{\text{rouge-1}}{b_i} + \frac{\text{rouge-2}}{b_2} + \frac{\text{rouge-L}}{b_L}$$

where b represents the baseline for each metrics.

```
baseline = {
  'rouge-1': 0.220,
  'rouge-2': 0.085,
  'rouge-L': 0.020,
}
```

Hyperparameters

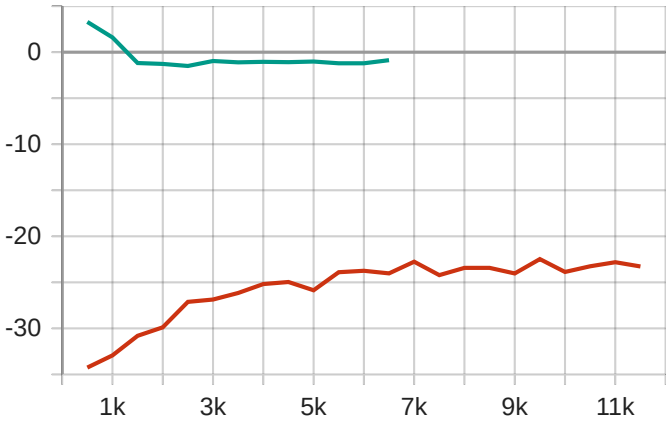
1. Supervised finetuning from mT5: omitted.
2. Intermediate supervised finetuning from mT5 + RL finetuning:
    - Training epochs: 10
    - Batch size: 4 x 8(accumulation steps)
    - Learning rate: 1e-5
    - Linear warm-up: 300 steps
    - Reward: rouge_combined
    - Reward discount factor: 0.999
3. RL finetuning from mT5
    - Training epochs: 10
    - Batch size: 4 x 8(accumulation steps)
    - Learning rate: 1e-4
    - Linear warm-up: 1000 steps
    - Reward: rouge_combined
    - Reward discount factor: 0.900

Compare to Supervised Learning: Observe the loss, ROUGE score, and output texts, what differences can you find?

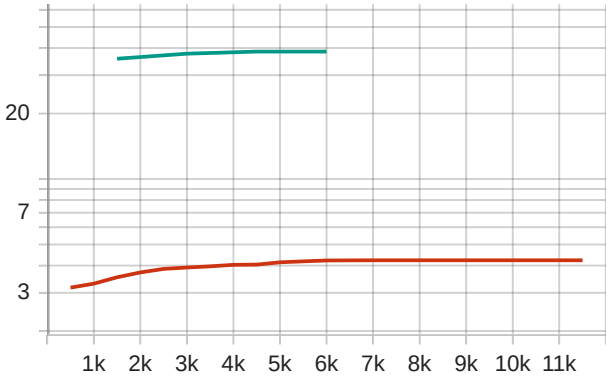
Comparing with/without Intermediate Supervised Finetuning

Red/Green: With/without intermediate finetuning

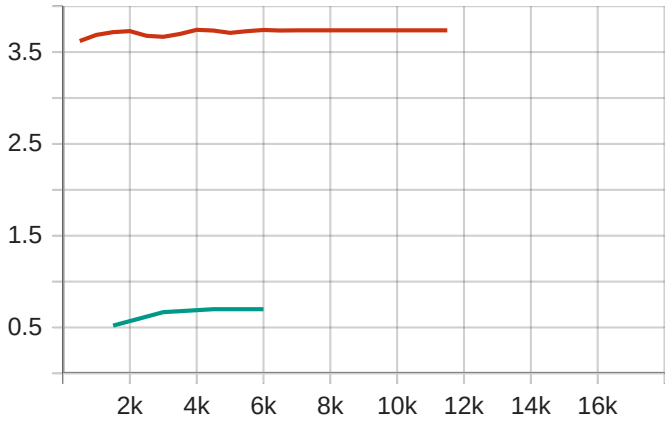
Training Loss:



Evaluation Loss:



Rouge Combined:



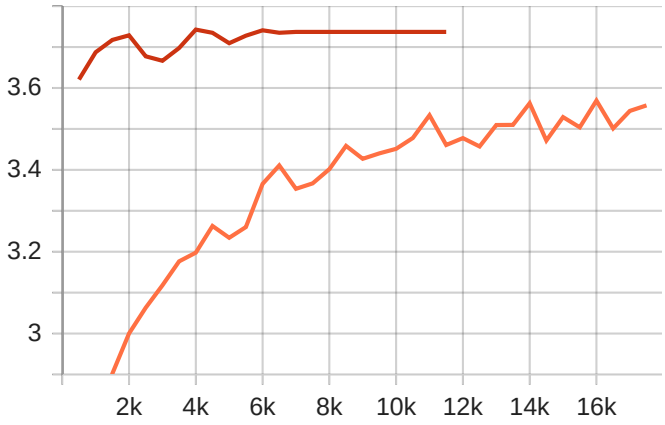
We can see that with supervised intermediate finetuning, the model starts training from a more optimal position and thus the procedure is more stable.

Comparing with/without Reinforcement Learning Finetuning

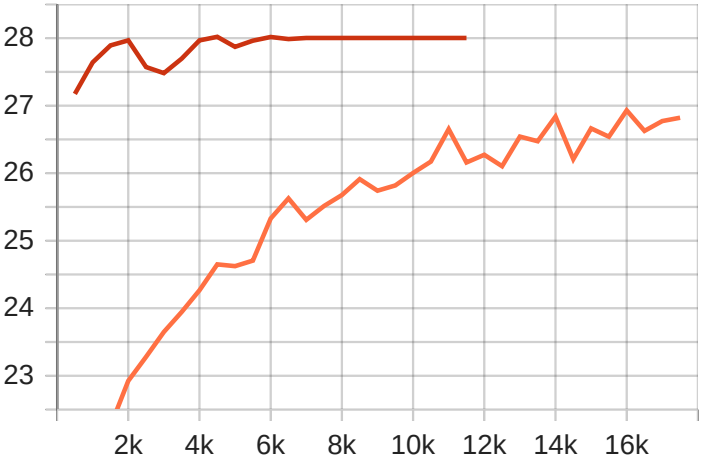
Red/Orange: With/without reinforcement finetuning

Since the loss functions are completely different, I only compare the rouge-* curves

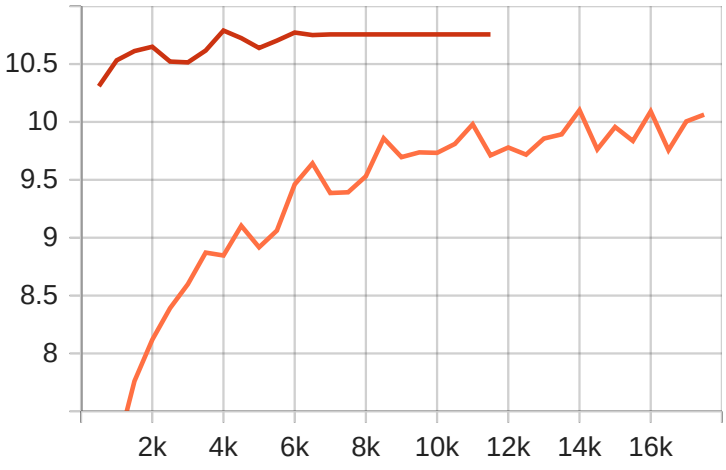
Rouge combined



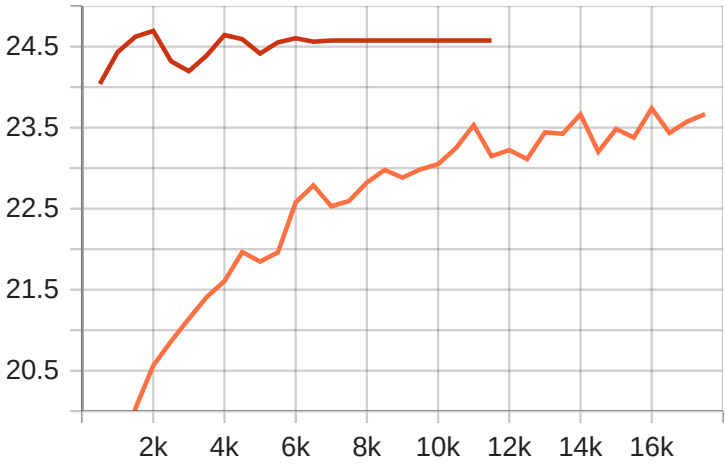
Rouge-1



Rouge-2



Rouge-L



Final Rouge score comparison

Method	Rouge-1	Δ%	Rouge-2	Δ%	Rouge-L	Δ%
Supervised Finetune from mT5	0.2691	N/A	0.1081	N/A	0.2393	N/A
RL Finetune from (1)	0.2712	+0.8%	0.1070	-1.0%	0.2411	+0.8%
RL Finetune from mT5	0.0452	-83.2%	0.0007	-99.3%	0.0446	-81.4%

After further RL optimization on rouge combined, the model performs even better compared to only conducting supervised MLE optimization.

Output text

I grabbed some titles generated by the three models.

a. Supervised finetuning from mT5

b. Intermediate supervised finetuning from mT5 + RL finetuning

c. RL finetuning from mT5

1. ID: 21735

   Ground Truth: 從台積電換三星…小米11首發驍龍888傳出過熱 原因出在三星的5nm？

   a: 台灣評測者:小米11發熱高於小米10 實測功耗有多高?

   b: 小米11發熱問題 小米11發熱高於小米10

   c: <extra_id_0>!」的!」!」

2. ID: 21754

   Ground Truth: 因疫情失業！彰縣釋出公部門670個職缺 時薪160元

   a: 彰投分置2個就業服務中心 每小時工資160元、每月收入最多可增加12800元

   b: 彰化縣政府釋出「安心即時上工計畫」 每月收入最多可增加12800元

   c: <extra_id_0>!」的

Model c failed to converge and collapsed. It generates unreasonable outputs.

Model a and b produces similar results, but model b tends to reduce the probability of words that are not in the ground truth titles, this makes the rouge score even higher. This conclusion also supports the idea of reinforcement learning.