ADL Hw3

R11944014 戴靖婷

tags: `ADL` `Python` `Report`

Q1: Model

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

- **google/mt5-small**:
    - Multilingual T5 (mT5) is a text-to-text, encoder-decoder transformer with attention mechanism, which means it figures out the relationship between all words in a sentence.
    - Also, it is pretrained on ultilingual Common Crawl corpus (mC4).
    - The folloing is the content of config.json.

```
{
  "_name_or_path": "google/mt5-small",
  "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": "4.22.2",
  "use_cache": true,
  "vocab_size": 250100
}
```

- For text summarization,

- It can be seen as a machine translation task by translating a text into a shorter summarization.
- To specify the task of mt5, add a prompt prefix –*source_prefix* "*summarize:* ".
- I made use of the sameple code from the hugginface github. https://github.com/huggingface/transformers/tree/t5-fp16-no-nans/examples/pytorch/summarization (https://github.com/huggingface/transformers/tree/t5-fp16-no-nans/examples/pytorch/summarization)

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

- Tokenization:
  - Tokenizer class: T5Tokenizer
  - T5Tokenizer is based on SentencePiece method.
    - It treats the input as the raw input including space because not every language separate sentences by a space, such as Chinese.
    - Then, it constructs vocabulary by the BPE or unigram tokenization algorithms.
    - Therefore, it is suitable for multilingual T5 and also builds an independent system for each language end-to-end.

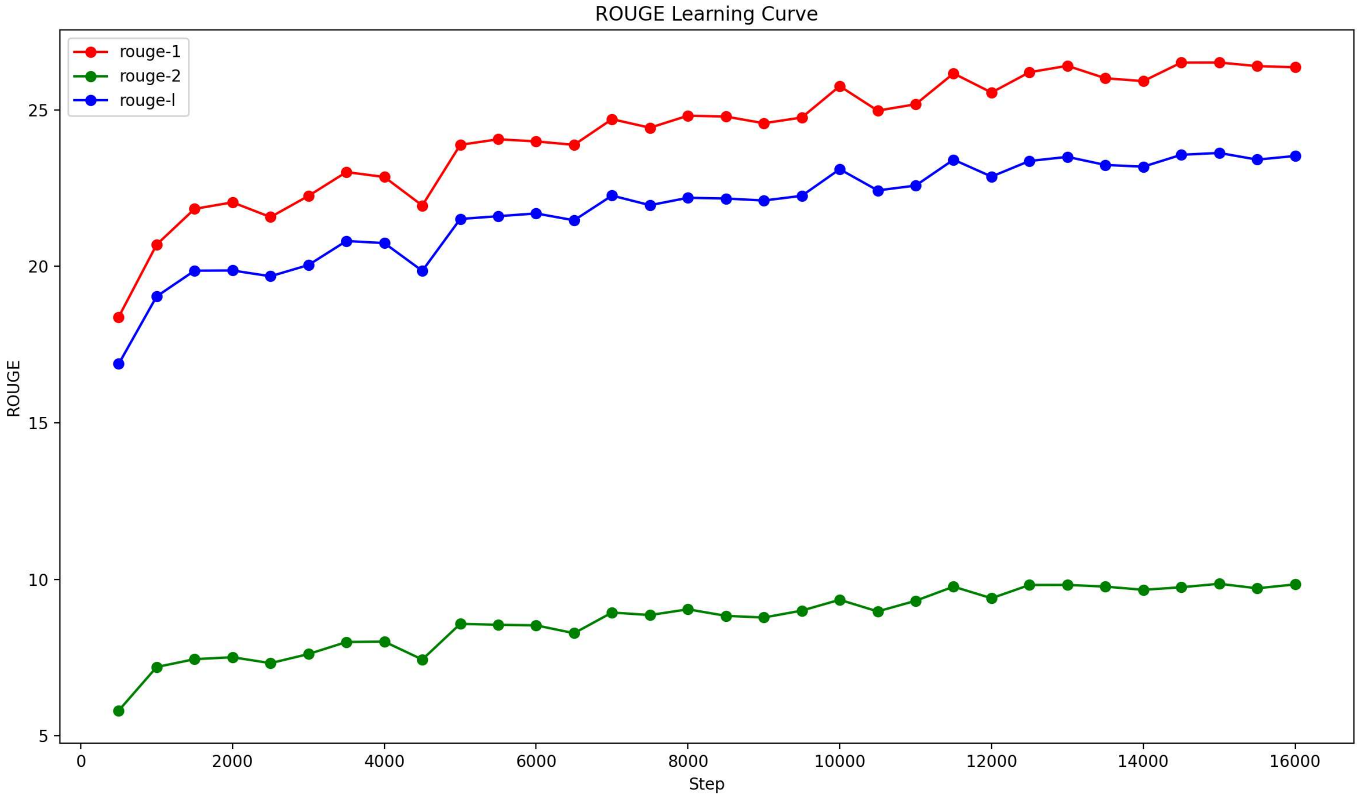
Q2: Training

Hyperparameter: Describe your hyperparameter you use and how you decide it.

```
python ./train_summarization.py \
    --model_name_or_path google/mt5-small \
    --do_train \
    --do_eval \
    --train_file ./data/train.jsonl \
    --validation_file ./data/public.jsonl \
    --source_prefix "summarize: " \
    --text_column maintext \
    --summary_column title \
    --per_device_train_batch_size 4 \
    --per_device_eval_batch_size 4 \
    --optim adafactor \
    --learning_rate 5e-4 \
    --max_source_length 256 \
    --max_target_length 64 \
    --output_dir ./ckpt/ \
    --overwrite_output_dir \
    --predict_with_generate \
    --evaluation_strategy steps \
    --eval_steps 500 \
    --logging_steps 500
```

- I trained my model with 1024 and 256 at first. However, it used more than 8G memory, so I truncated the text length.

Learning Curves: Plot the learning curves (ROUGE versus training steps)



- This result is generated by greedy method with fp32.

Q3: Generation Strategies

Stratgies: Describe the detail of the following generation strategies:

- Greedy
  - For the next word, select the word with the highest probability.
  - $w_t = \mathrm{argmax}_w P(w|w_{1:t-1})$
- Beam Search
  - Keep the most likely num_beams of possible words at each time and choose the one with overall highest probability.
  - It avoids missing the hidden high probability word sequences behind a relatively lower probability word.
- Top-k Sampling
  - Keep only the K most likely next words for sampling from the reconstructed distribution.
  - And, sampling means picking the next word based on its conditional probability distribution. $w_t \sim P(w|w_{1:t-1})$
- Top-p Sampling
  - Choose the next word from the set of words whose cumulative probability exceeds the probability p.
- Temperature
  - Temperature changes the probability of every word.

- The range of temeperature is between 0.0 to 1.0.
  - When temperature is closer to 0.0, it is closer to greedy.
  - When temperature is closer to 1.0, each word has the same probability to be generated as the next word.
- Lower the temperature of the softmax makes the distribution P sharper, which means increasing the likelihood of the words with high probability and decreasing the likelihood of the words with low probability.

- Reference: https://huggingface.co/blog/how-to-generate (https://huggingface.co/blog/how-to-generate)

Hyperparameters

- Try at least 2 settings of each strategies and compare the result. The ROUGE result is evaluated by --do_eval in run_summarization.py instead of eval.py (http://eval.py).
  - Greedy
    - The more accurate the floating point number (fp32) is, the better the result is.
    -

	rouge-1	rouge-2	rouge-l
fp32	27.1872	10.1801	24.2615
fp16	26.9455	9.981	23.9738

  - Beam Search
    - Adjust num_beams and set early_stopping=True.
    - The larger the num_beams is, the more possible predictions it keeps. So the result is better when the num_beams is larger.
    -

	rouge-1	rouge-2	rouge-l
num_beams = 2	28.0982	11.0138	25.0553
num_beams = 5	28.3986	11.4707	25.3744
num_beams = 10	28.3385	11.5820	25.2885

  - Top-k Sampling
    - Adjust top_k and set do_sample=True.
    - The result is not good if using a large set of likely next words, so we may narrow down the set of the likely next words.
    -

	rouge-1	rouge-2	rouge-l
top_k = 20	23.0773	7.2718	20.1300
top_k = 10	24.2822	8.0891	21.3037
top_k = 5	25.5530	8.7149	22.3640

- Top-p Sampling
  - Adjust top_p and set top_k=0, do_sample=True.
  - The result is better when p is smaller, which means using less likely word set is better. The trend of top_p is the same as the trend of top_k.
  -

	rouge-1	rouge-2	rouge-l
top_p = 0.95	18.7501	5.6598	16.5373
top_p = 0.50	24.2476	8.5073	21.3852
top_p = 0.30	25.7993	9.2286	22.7769

- Temperature
  - Adjust temperature and set top_k=0, do_sample=True.
  - It is better to set temperature closer to 0, it is closer to greedy.
  -

	rouge-1	rouge-2	rouge-l
temperature = 0.7	24.1415	8.4414	21.2756
temperature = 0.5	25.9870	9.4473	22.9505
temperature = 0.3	26.9834	9.9318	24.0372

- What is your final generation strategy? (you can combine any of them)
  - Over the experiments, I found that the result of beam search is the best, so my final generation strategy is **num_beams=5** with early_stopping=True and the test file is predicted from it.