ADL2023-HW2 Report

Q1: Model

Model

T5, a.k.a. Transfer Text-to-Text Transformer, is built upon the transformer architecture, which includes an encoder-decoder framework. The special part of T5 is Text-to-Text, this formulates all NLP tasks as text transformation problems. The following are some critical components of it:

Token and Positional Embedding: represent tokens with their positional relationship in a vector space.

Encoder-Decoder Structure: The encoder processes the input text, while the decoder generates the output text, both of them have layers of self-attention, add&normalize, and feed forward.

In the case of text summarization, T5 processes input data in the form of text pairs, where one sequence represents the source document and the other represents the target summary. During training, T5 learns the mapping from the input document to the target summary, enabling it to generate concise, contextually relevant summaries. Leveraging its transformer architecture, T5 captures semantic relationships, allowing it to produce coherent abstractive summaries that encapsulate the essential information from the source text.

Preprocessing

Tokenization Algorithm: sentencepiece, an unsupervised text tokenizer and detokenizer.

Add prefix: I add prefix "summarize: " to the text we want to perform summarization to specify its task.

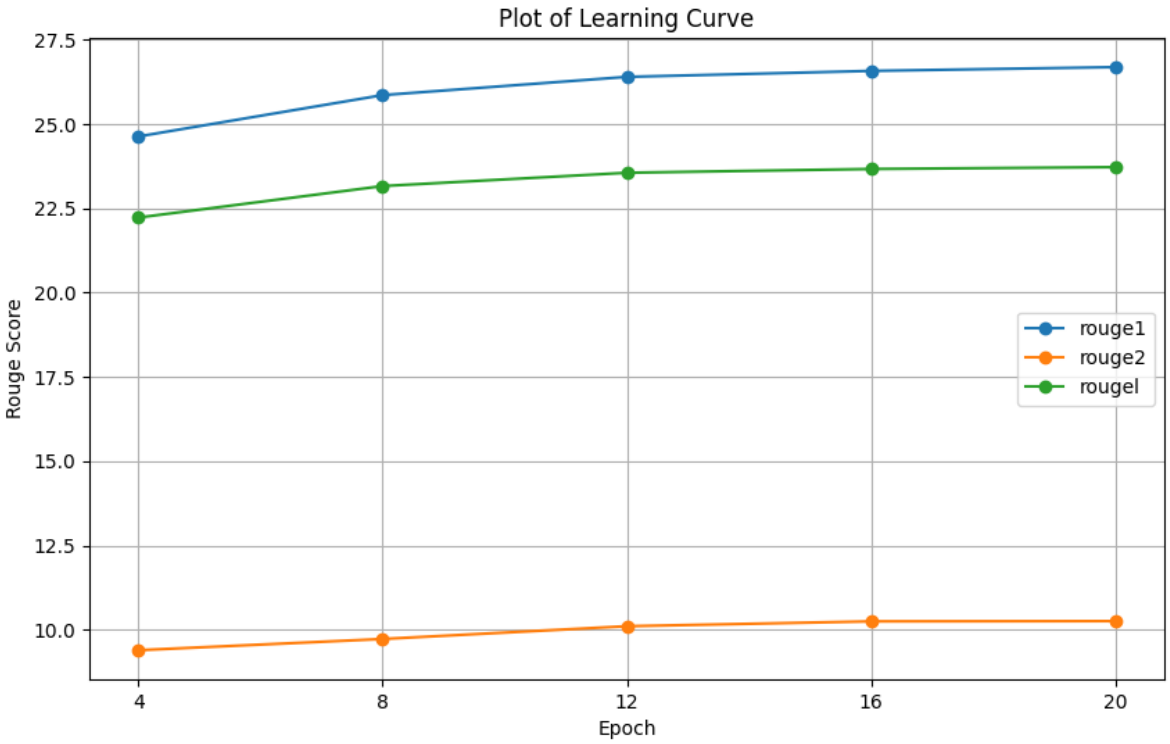
Token length limiting and padding: limit max_source_length to 2048 and max_target_length to 128, and if the input is shorter than 2048, it will pad to max length.

Q2: Training

Hyperparameter

- max_source_length 2048
  I think with longer max source length, the performance would be better, but the training time would be longer too.
- batch_size 4
  I have tried 2, 4, 8, 16, 32. In my experiments, the performance is maximized when batch size set to 4.
- learning_rate 1e-4
  Default is 5e-5, but I think larger learning rate may lead to better performance in this situation.
- num_train_epochs 20
  I have tried 10, 20, 40, 50, and the performance seems stop improving when epoch is larger than 20.

Learning Curves



Q3: Generation Strategies

Strategies

- Greedy
  Selects the most probable next word according to the model, without considering feature consequences. It tends to favor common words and phrases, and often leads to local optimal solution but not global optimal one.
- Beam Search
  For a given beam number b, beam search keeps track b most probable candidates at each step, and selects the most likely sequences base on a scoring function. Beam search is more likely to reach global optimal than greedy.
- Top-k Sampling
  For a given number k, top-k sampling involves sampling from the top k most likely candidates. It allows diversity in generated text while ensuring that the model doesn't consider improbable words. It prevents the model from focusing only on the most probable words and encourages exploration of less likely but still possible choices.
- Top-p Sampling
  Top-p sampling is somewhat similar to top-k sampling, but unlike top-k sampling has k candidates at every step, top-p sampling's candidate set consisting of the smallest possible set of words whose cumulative probability exceeds a predefined threshold p.
- Temperature

Temperature is a scaling factor used to control the randomness of the predictions in the softmax function during sampling. Higher temperature leads to more exploratory and generates more diverse but potentially less coherence text. Lower temperature leads to less randomness but potentially leads to repetitive outputs.

Hyperparameters

Strategy	Rouge 1	Rouge 2	Rouge L
Greedy	26.71	10.22	23.66
Beam 2	27.63	11.08	24.64
Beam 4	27.94	11.45	24.87
Top K 25	23.03	7.95	20.24
Top K 75	21.40	7.31	18.86
Top P 0.75	24.63	8.92	21.73
Top P 0.9	23.44	8.31	20.68
Temperature 0.75	23.10	8.29	20.48
Temperature 1.5	15.67	4.36	13.92

Strategy: Greedy (because we only have 1 hour to execute run.sh)