HW2

學號：D13922021

姓名：林焜詳

Q1. Model

Model

Architecture

This assignment requires the use of the mT5 model (Xue et al., 2020). The architecture of mT5 is very similar to the standard Transformer Encoder and Decoder, with the main differences being the use of gated-gelu for activation and the absence of dropout during the pretraining phase. The mT5 model is used here, and detailed parameters can be found in the Hyperparameters section.

Hyperparameters

```
{
  "_name_or_path": "google/mt5-small",
  "architectures": [
    "MT5ForConditionalGeneration"
  ],
  "classifier_dropout": 0.0,
  "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": "4.34.1",
  "use_cache": true,
  "vocab_size": 250112
}
```

How it works on Text Summarization

The mT5 model is a multilingual text-to-text generation model based on the Transformer Encoder to Decoder architecture. By leveraging the model's generation capabilities, it helps us perform text summarization tasks.

Here are the steps to apply it to text summarization tasks:

1. **Pretraining**: mT5 is pretrained on the mC4 dataset, which contains a large amount of web-scraped text in multiple languages. The goal of pretraining is text-to-text generation, where the model learns to transform one form of text (e.g., a question) into another form of text (e.g., an answer).
2. **Understanding and Encoding**: In text summarization tasks, the model first needs to understand the input paragraph. This involves using its encoder part to encode the text, capturing the semantics and context of the sentences.
3. **Summarization Generation**: The decoder part of mT5 then generates the summary based on the encoder's output. It generates new words step by step, considering the previously generated content each time to ensure the coherence and relevance of the summary.
4. **Conditioned Generation**: During summary generation, the model is "conditioned" to focus on the input paragraph, meaning the decoder tries to maintain the main idea and key points of the input content while generating the text.
5. **Fine-tuning**: Although mT5 has been pretrained on the mC4 dataset, it needs to be fine-tuned on domain-specific data to perform specific tasks like news summarization better. In this example, mT5 is fine-tuned on news data from udn.com, which helps the model better understand the structure and common language of news articles.

By following these steps, we can train a good summarization model for future use.

Preprocessing

I used a pretrained mT5 tokenizer to tokenize the data. This tokenizer splits the input Chinese sequence into individual characters and then converts each character into its corresponding ID based on its dictionary.

After tokenization, the tokenizer truncates or pads the sequence and adds [BOS] (beginning of sequence) and [EOS] (end of sequence) tokens at the start and end of the sequence, respectively.

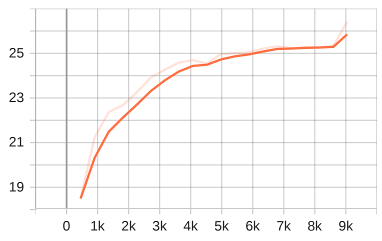
Q2. Training

Hyperparameters

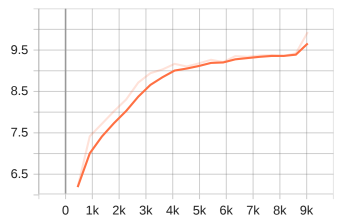
- **Epoch**: 20 → This seems to be a good stopping point based on trials. It shows good convergence results in evaluation. Too few epochs result in poor performance, while too many are redundant with little improvement.
- **Batch size**: 12 × 4 (accumulated gradient steps) → Fully utilizes computational power, accelerating model convergence.
- **Learning rate**: 5e-5 → 5e-5 is a commonly used value for fine-tuning.
- **Linear warm-up**: 300 steps → A reasonable number chosen at random.
- **Learning scheduler**: cosine → Provides better fine-tuning results.

Learning Curves

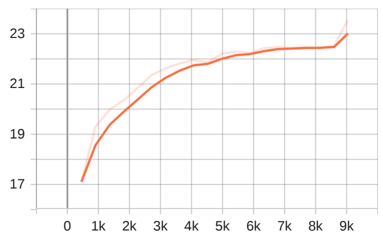
The presented value is 100 x ROUGE score.



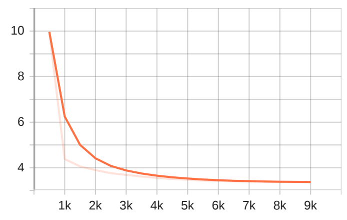
rouge-1 on validation set



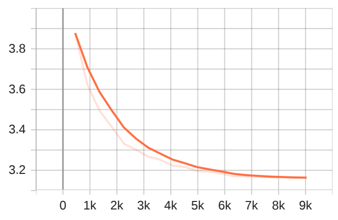
rouge-2 on validation set



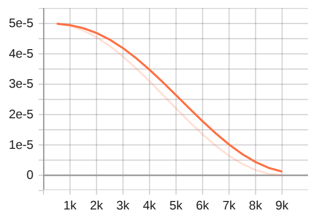
rouge-l on validation set



training loss



validation loss



Learning rate

Q3. Generation Strategies

Strategies

Greedy

Greedy selects the highest probability output for each character.

Beam Search

Beam search is an improved algorithm over greedy search. It expands the search space compared to greedy but not as extensively as exhaustive search. Typically, a beam size (k) is used to select the top k best candidate word combinations at each step. Each new character generation maintains k candidate word combinations.

Top-k Sampling

Top-k sampling selects characters similarly to greedy, but before selection, it preprocesses the predicted logits. It selects the top k words with the highest prediction probabilities and normalizes them into a new probability distribution.

Top-p Sampling

Top-p sampling aims to address issues with Top-k sampling in certain distributions. When Top-k encounters a flat distribution, the selected distribution may lose its original characteristics. Top-p uses a cumulative threshold to dynamically select the top k words for the new distribution calculation.

Temperature

Used to address issues with imbalanced class distributions in datasets. By adjusting with γ, if $\gamma = 1$, it means no adjustment. If $\gamma > 1$, the probability of low-frequency sample classes will be increased, making them more likely to be sampled. Conversely, if $\gamma < 1$, the probability will be decreased.

$$p_i = \frac{N_i^{1/\gamma}}{\sum_{j \in [1,M]} N_j^{1/\gamma}}$$

Hyperparameters

Evaluation	rouge-1	rouge-2	rouge-l	rouge_combined	gen_len
GreedySearch	26.375	9.905	23.508	3.511	21.872
BeamSearch=3	27.687	11.032	24.509	3.752	23.452
BeamSearch=5	27.567	11.168	24.441	3.759	23.866
BeamSearch=7	27.344	11.056	24.214	3.725	24.206
Top-k=25	22.228	7.171	19.355	2.798	22.284
Top-k=50	20.720	6.466	18.065	2.584	22.411
Top-k=100	20.051	6.215	17.452	2.494	22.347
Top-p=0.8	23.202	8.023	20.398	2.994	22.111
Top-p=0.95	21.837	7.037	19.057	2.750	22.240
Top-p=0.95,T=2.0	12.322	2.108	10.420	1.316	25.921
Top-p=0.95,T=0.5	25.625	9.417	22.642	3.377	21.832

- Text summarization is a subjective and deterministic task, so performance metrics often don't look good. Upon closer inspection of some sample predictions, it can be found that some predictions are not necessarily wrong but may just be rephrased, with meanings very close to the original. This makes it seem more like a task suited for dialogue systems.

- Using a single method, Beam search performs better than other methods and is generally a smarter approach.
- Additionally, experiments with Top-p + Temperature show that excessively increasing the temperature causes the prediction distribution to fail, as the new distribution starts to resemble a uniform distribution, providing no benefit to the predictions.

Final generation strategy

Finally, I chose Beam search as my generation strategy.

Evaluation	rouge-1	rouge-2	rouge-l	rouge_combined	gen_len
BeamSearch=5	27.567	11.168	24.441	3.759	23.866