ADL HW2 Report

Q1: Model

Model

- Model Config


```
{
  "architectures": [
    "MT5ForConditionalGeneration"
  ],
  "d_ff": 1024,
  "d_kv": 64,
  "d_model": 512,
  "decoder_start_token_id": 0,
  "dropout_rate": 0.1,
  "eos_token_id": 1,
  "feed_forward_proj": "gated-gelu",
  "initializer_factor": 1.0,
  "is_encoder_decoder": 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_num_buckets": 32,
  "tie_word_embeddings": false,
  "tokenizer_class": "T5Tokenizer",
  "vocab_size": 250112
}
```


- **Architecture:**
  The model architecture I used is mT5-small. I used the configuration as default. It follows a transformer-based design, consisting of 8 encoder layers and 8 decoder layers with a hidden size of 512. Notably, mT5 uses the gated-gelu activation function and doesn't employ dropout during pretraining. This architecture is well-suited for various natural language processing tasks, including text summarization.

- **Text Summarization:**
  mT5 is adept at text summarization because it was pretrained on mC4 dataset, a large scale, web-crawled, multilingual dataset with a text-to-text pretraining objective. The task of summarizing paragraphs can be approached as text-to-text generation, where mT5 creates concise summaries based on the input text. With further fine-tuned on news data and optimized using teacher-forced maximum likelihood training, mT5 becomes suitable for summarizing paragraphs.

Preprocessing

- **Tokenization:** I used the `AutoTokenizer` to load the mT5 tokenizer, which is designed for the multilingual T5 (mT5) model by Google. It primarily uses SentencePiece, a popular subword tokenization method that combines Byte Pair Encoding (BPE) and unigram language modeling. SentencePiece is capable of subword tokenization, allowing it to handle various languages and character sets effectively.

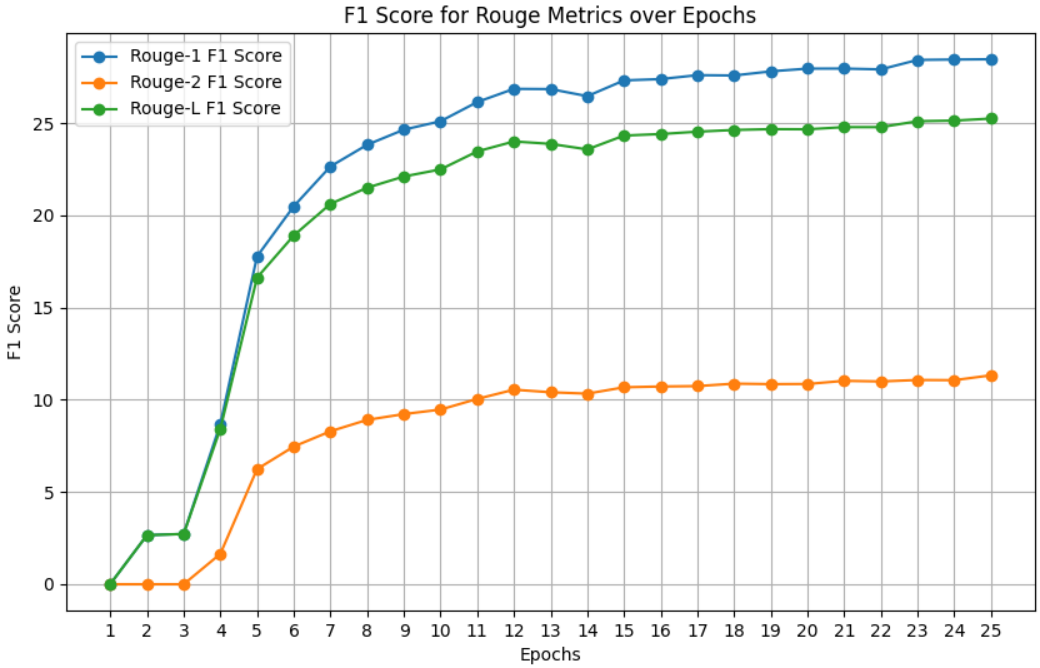
Q2: Training

Hyperparameter

Hyperparameter	Value
batch_size	16
max_input	256
max_output	64
num_epoch	25
num_beams	2
optimizer	torch.optim.AdamW
learning rate	1e-4
weight-decay	5e-5

I choose the hyperparameters according to strike a balance between model performance, training speed, and memory usage.

Learning Curves



Best model: Epoch 25

Evaluation criteria: The evaluation criteria involve calculating the maximum of the total F1-score relative to the baseline F1-score.

Q3: Generation Strategies

Greedy	num_beam=1. At each time step, select the word with the highest probability given the preceding words, i.e., $P(w\|w1...wt-1)$
Beam Search	num_beam>1. At each time step, consider a set of 'num beams' candidates. In the next time step, choose the top 'num beams' candidates with the highest probabilities from the candidates generated in the previous step.
Top-k Sampling	From the probability distribution $P(w\|w1...wt-1)$, sample 'wt' from the top 'K' most probable words. This strategy helps to control the diversity of the generated sequences.
Top-p Sampling	From the probability distribution $P(w\|w1...wt-1)$, sample 'wt' by accumulating probabilities of the most probable words until the cumulative probability 'p' is reached. This strategy allows for dynamic control of the number of words considered during sampling.
Temperature Strategies	Temperature strategies involve adjusting the softmax function to either sharpen or smooth the probability distribution. When the temperature is less than 1, it sharpens the distribution, emphasizing the most likely words. When the temperature is greater than 1, it smooths the distribution, reducing the emphasis on the most likely words and increasing the influence of less likely candidates.

Q4: Hyperparameters

default:

num_beams	top_k	top_p	temperature
5	50	0.9	1.0

Generation Strategies	**Parameter**	**rouge-1**	**rouge-2**	**rouge-l**
Greedy	num_beams = 1	0.24950460512607028	0.09584808215788704	0.22354562186526986
Beam Search	num_beams = 3	0.261472568239022	0.10425170860979847	0.2340411342447209
Beam Search	num_beams = 5	0.2622965024856237	0.10578067963865942	0.23525843266368365
Beam Search	num_beams = 10	0.2639914245883621	0.10750434635932082	0.23620277354156316
Beam Search	num_beams = 12	0.2641982906552653	0.10800071165579542	0.23641129565701355
Beam Search	num_beams = 16	0.2636973221612067	0.10802662780893427	0.23622376732572642
Top-k Sampling	top_k = 10	0.2601837624118818	0.10332843352680098	0.23296909246015612
Top-k Sampling	top_k = 50	0.26117850694802963	0.10424908764574069	0.234084322429481
Top-k Sampling	top_k = 100	0.25896569118480417	0.10229997581661186	0.23199748144994195
Top-p Sampling	top_p = 0.8	0.25972684434480664	0.10383463850114362	0.23245117744280722
Top-p Sampling	top_p = 0.9	0.26147572930718643	0.10397474165653757	0.23374143077398482
Top-p Sampling	top_p = 1.0	0.26134843568545296	0.1034932751633401	0.23282892268875294
Temperature	temperature = 0.7	0.2622965024856237	0.10578067963865942	0.23525843266368365
Temperature	temperature = 1	0.2622965024856237	0.10578067963865942	0.23525843266368365
Temperature	temperature= 1.5	0.2622965024856237	0.10578067963865942	0.23525843266368365
Temperature+ top p	temperature= 0.7 top_p = 0.9	0.2591934589731801	0.10188393144474801	0.23265944391685442
Temperature+ top p	temperature= 2 top_p = 0.9	0.24598848266557807	0.09364532948838458	0.21872692736797664

- **Beam Search vs. Greedy Decoding**: Beam search is better than greedy decoding as it considers multiple candidates, while greedy picks the most likely word at each step, which can be suboptimal if an early choice is incorrect. `num_beams=12` yielded the best results.
- **Effect of Temperature on Beam Search**: Temperature is a hyperparameter that can influence the distribution of probabilities in the decoding process. In the case of beam search, it does not have a significant impact on the results. This is because beam search already relies on selecting candidates based on their probabilities, so temperature does not alter this fundamental process.

Beam search aims to find the most likely sequences without introducing diversity through temperature adjustments.

- **Effect of Temperature on Top-p Sampling**: For top-p sampling, tetemperature matters. Top-p sampling aims to control the number of candidate tokens at each step based on the cumulative probabilities. Temperature values less than 1 sharpen the probabilities. This results in fewer candidates being considered, making the decoding process more focused on the most probable choices.. Conversely, higher temperature values smooth the probabilities, evening out the distribution. This encourages a more random sampling approach. As we can see, top-p sampling with high temperature performs bad and produces unreasonable sentences since the distribution almost degrades to a uniform distribution.

Final generation strategy: beam search (n beams = 12)