ADL-HW2-2024 report

R11946012 王奕方

Q1: Model

- Model

I use mT5-small as my pretrained model, which is a Text-to-Text Transfer Transformer model, designed for multilingual tasks. mT5-small features an encoder-decoder architecture with about 300M parameters.

The input text is tokenized and embedded with positional embeddings.

The encoder consists of 12 layers and encodes the embeddings into a sequence of hidden representations that capture contextual relationships.

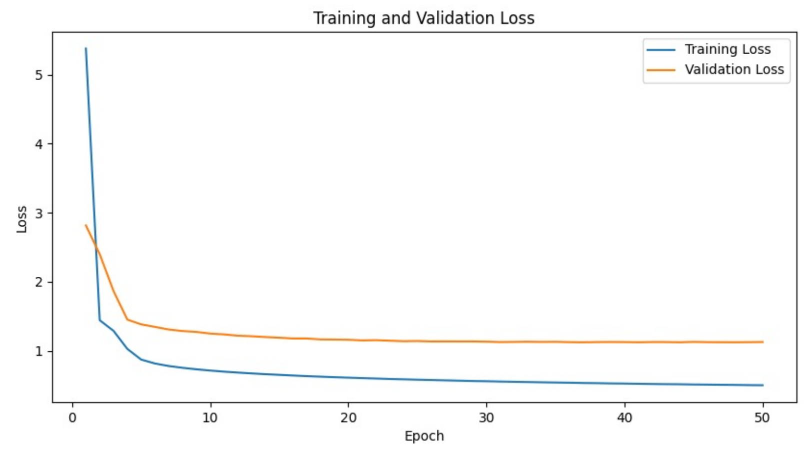
The decoder has 12 layers and generates the output sequence of the summary in an autoregressive manner. On top of decoder is a language modeling head, which takes the hidden states from the decoder and converts them into probabilities for each token in the vocabulary. It starts with start token <pad>. At each step, it applies self-attention on the previously generated tokens and cross-attention to attend to the encoder's output at every layer. The process continues until the model predicts the end-of-sequence token.

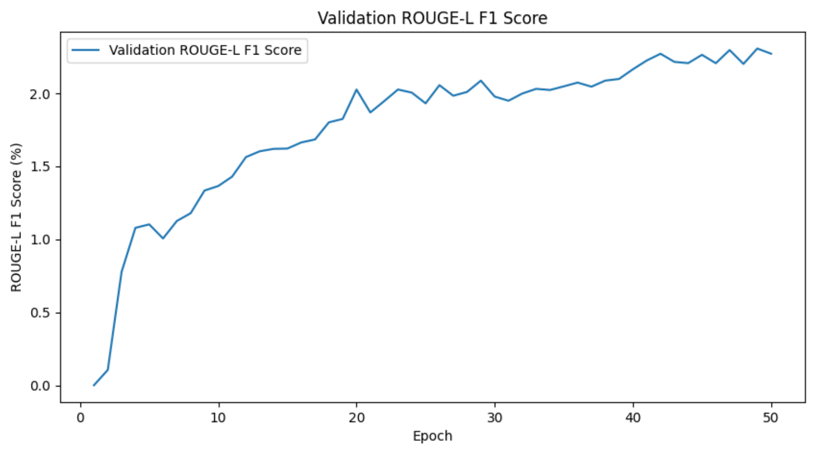
- Preprocessing

I use T5Tokenizer to convert both the input text and output summary into sequences of tokens. It uses SentencePiece algorithm to splits text into subword units to handle unknown words. The input text is truncated and padded to a maximum length of 256 tokens, and the output summary is to a maximum length of 64 tokens. Also for summarization, the input text is prefixed with "summarize: " to instruct the model that it needs to generate a summary.

Q2: Training

- Hyperparameter
  - Batch size: 16 & Gradient Accumulation Steps: 2 (fit in available GPU memory)
  - Training epoch: 50 (training and validation comes to a stable performance)
  - Learning rate: 1e-4 with linear scheduler decay lr in 0.95 every two epochs (low for finetuning a pretrained model)
  - Max Input Length: 256 & Max Output Length: 64 (capture enough context for inputs without excessive memory usage)
  - Number of Beams in beam search: 6 (explore multiple possible sequences during generation)
  - Repetition Penalty: 1.2 & No Repeat N-gram Size: 2 (more diverse summaries)
- Learning Curves





Q3: Generation Strategies

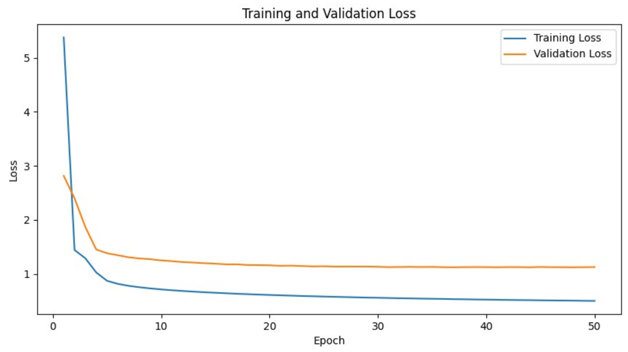
- Strategies
  - Greedy:
    - Choose the most probable word
    - Cons: it may miss better sequences and lack of diversity
  - Beam Search (beam=k):
    - Keep track of the k most probable sequences and and discard the rest
    - Pros: a better overall probability than greedy but requires more computation
    - Cons: it may still generate repetitive text since it relies on probability scores
    - k=1 is greedy
  - Top-k Sampling
    - Randomly sample tokens via the probability distribution and select top-k highest-probability tokens
    - Pros: Prevent choosing highest-probability and have more diverse outputs compared to greedy and beam search
    - k=1 is greedy
  - Top-p Sampling
    - Sample the smallest possible set of tokens whose cumulative probability is greater than or equal to p
    - Pros: it adjusts the number of tokens based on the probability distribution, while top-k sampling fixes the number of tokens
  - Temperature
    - Apply a temperature hyperparameter τ to the SoftMax to controls the randomness of the predictions

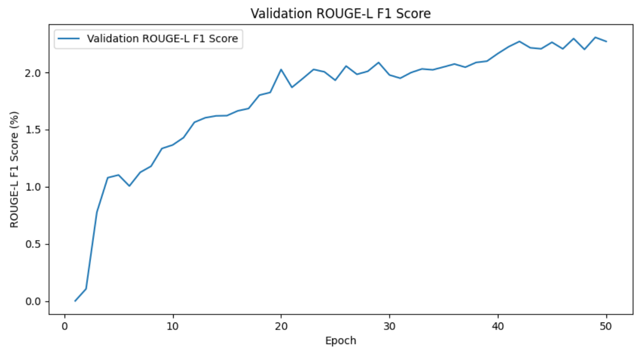
$$P(w) = \frac{\exp(s_w/\tau)}{\sum_{w' \in V} \exp(s_{w'}/\tau)}$$

    - Higher temperature becomes more uniform/diversity, while lower temperature becomes more spiky → less diversity

Q3: Generation Strategies

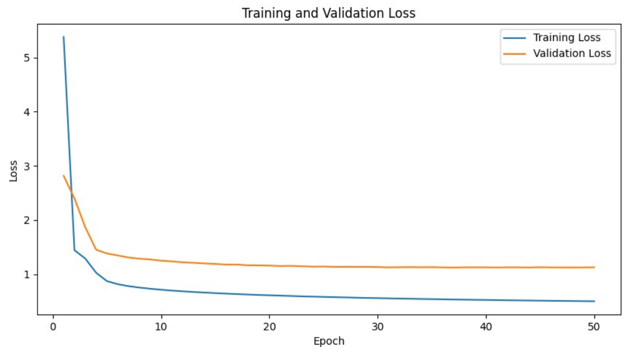
1. **Deterministic (Final Strategy)**
   - Beam Search: 6
   - Frequency Penalty: 1.2
   - No repeat Ngam size: 2
   - Early Stopping

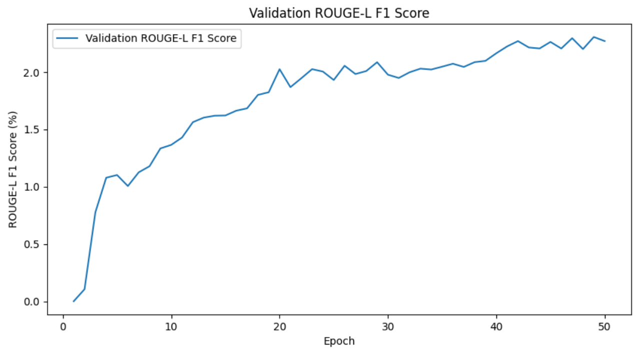




Evaluation on public: rouge-1: 25.9%, rouge-2: 10.3%, rouge-l: 22.8%

2. Sampling
   - Top-k: 50
   - Top-p: 0.95
   - Temperature: 0.7
   - Frequency Penalty: 1.2
   - No repeat Ngam size: 2
   - Early Stopping





Evaluation on public: rouge-1: 22.5%, rouge-2: 7.7%, rouge-l: 19.5%