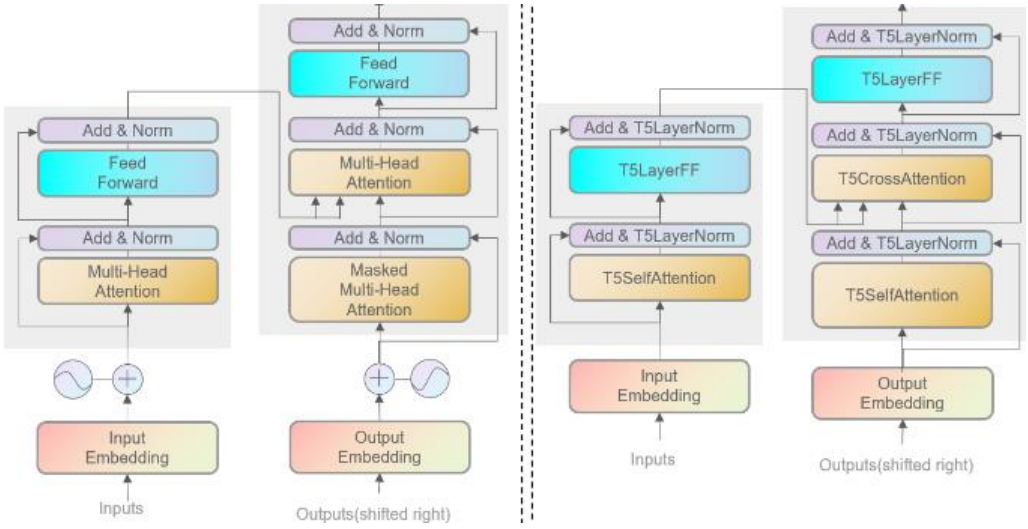
ADL HW2

R13922022 王凱右

Q1-1: Model architecture

We are using the pre-trained model google/mt5-small and fine-tuning it on the dataset provided by TA. Multilingual T5 (mT5) [5] is a natural language processing (NLP) model based on the Text-to-Text Transfer Transformer (T5) [2] and trained on the C4 dataset.



Transformer's general architecture (left) and T5's general architecture (right) [4]

The architecture of T5 itself is not far different from the original Transformer, and naturally, the architecture of mT5 is similar to that of T5. However, several details have been modified to improve the model's performance, such as replacing the original activation function with GeGLU, determining the sampling rate for different languages, and proposing solutions for illegal translations and unexpected predictions when they occur. [5]

Q1-1: Text Summarization

mT5 is a language model capable of converting all NLP tasks into text-to-text generation tasks, and the task of paragraph summarization can also be seen as a text-to-text generation task. During the training process, the model is fine-tuned based on the cross-entropy loss function, using pairs of source text and reference summaries for training. The model learns to generate summaries that are close to the reference summaries by minimizing the cross-entropy loss between the generated output and the target output.

During the inference process, the model uses the output from the encoder and the input from the decoder as cross-attention inputs, generating one token at a time. It predicts the probability distribution of vocabulary tokens based on the previously generated tokens and selects the next best token using various decoding strategies. This process (feeding the generated token sequence back into the model) is repeated until the model generates an End-of-Sentence (EOS) token. Finally, the model decodes the output tokens into words to obtain the predicted summary.

Q1-2: Preprocessing

We did not perform data cleaning, and tokenization was also done using the pre-trained tokenizer of mT5. It employs SentencePiece for text segmentation, splitting words into multiple characters, which are then converted into id according to the vocabulary, followed by the addition of special tokens as needed.

To prompt mT5 to perform the summarization task. It is necessary to add the prompt "summarize: " to the source_prefix.

Q2-1: Hyperparameter

Loss Function	Cross-Entropy
Optimization Algorithm	AdaFactor
Learning Rate	5e-4
Warmup step	300
Batch Size per Device	28
Gradient Accumulation Steps	2
Epochs	16
Max Source Length	512
Max Target Length	128

Optimization Algorithm:

Used AdaFactor to reduce memory usage.

Learning Rate & Warmup step:

After several tests, we determined parameters that allow the model to converge quickly during training without the loss decreasing too rapidly.

Batch Size:

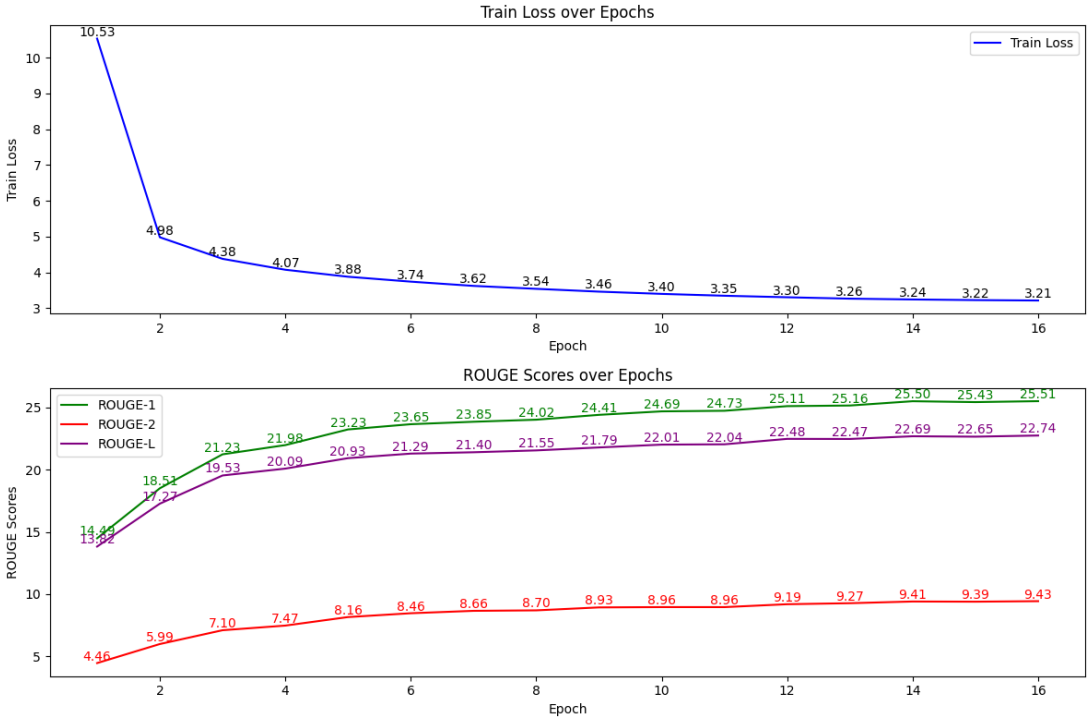
By using a larger batch size, training speed was increased, and the model's generalization ability to the data was enhanced.

Epoch:

Based on experiments, the training effectiveness began to converge after around 16 epochs.

Q2-2: Learning Curves

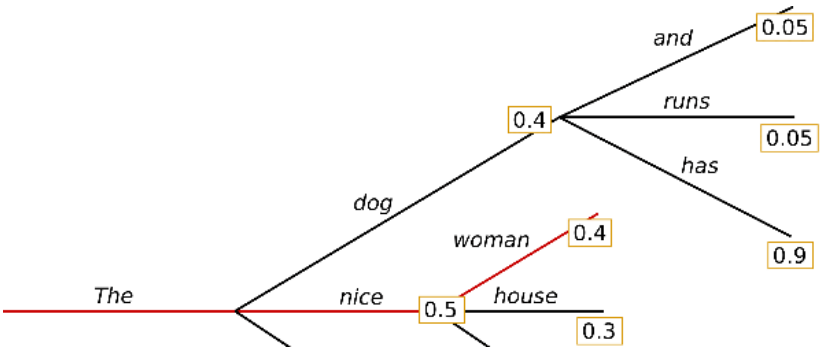
We split all the training data into a 9:1 ratio for training and validation, respectively. The ROUGE score was calculated using the get_rouge function provided by tw_rouge.



It can be observed that the ROUGE score continuously improved, but the loss decreased at a slower pace, so we set 16 epochs as the standard for training.

Q3-1:Describe the detail of the following generation strategies

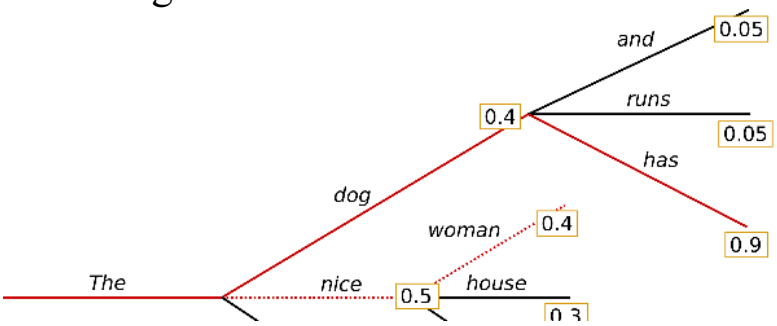
Greedy: The main idea is to select the word with the highest probability at each step as the next word in the sentence.



The red line illustrates the Greedy selection strategy [3].

The issue with Greedy strategy is that it might encounter local optima, missing some high-probability words that appear later. For example, the word "has" in the figure has a low chance of being selected because the probability of the preceding word is relatively low.

Beam Search: The main idea is to retain the top *n* most likely candidate paths at each step, and finally select the path with the highest overall probability. This ensures that even if a path has a low probability at the beginning, it can still be chosen if the words following it have high probabilities, compensating for earlier shortcomings.



The red/hashed line illustrates the Beam Search selection strategy [3].

Top-k Sampling: The main idea of this method is to select the *k* most likely words from the model's predictions, and then choose the next word based on their probability distribution. By introducing randomness, it addresses the local optima problem encountered by the Greedy strategy. However, the randomness may occasionally lead to rare words being

selected or cause the sentence to lack coherence.

Top-p Sampling: Instead of choosing the top *k* words, Top-p determines a probability threshold *p* and selects the smallest set of words whose cumulative probabilities exceed *p*. A word is then selected from this set. The advantage is that the size of the set can vary dynamically based on the distribution, unlike Top-k, which always selects from a fixed number of words.

Temperature: The idea is to balance the probabilities between words by adjusting the softmax function to alter the likelihood of each word being selected. When the temperature is low, the softmax function approaches the argmax function, meaning the model tends to select the most likely option. When the temperature is high, the probabilities of all words become more equal. This method allows a trade-off between the model's certainty and diversity, with the appropriate temperature value chosen based on the task.

Q3-2:Hyperparameters

Stratgies	Rouge-1	Rouge-2	Rouge-L
Greedy	23.48	8.42	21.05
Beam Search (num_beams = 3)	24.65	9.58	22.21
Beam Search (num_beams = 5)	**24.87**	**9.88**	**22.38**
Top_k = 10	20.42	6.51	18.03
Top_k = 100	16.74	4.82	14.85
Top_p = 0.8	16.67	5.21	14.87
Top_p = 0.9	15.74	4.79	14.06
Temperature=0.6	21.31	7.31	19.01
Temperature=1.3	6.97	1.31	6.31

We also experimented with different combinations of parameters, but none performed as well as the simple Beam Search (with num_beams = 5). Therefore, we choose Beam Search with the number of beams equal 5 as our final generation strategies

Reference:

[1] https://huggingface.co/google/mt5-small

[2] https://arxiv.org/abs/1910.10683

[3] https://huggingface.co/blog/how-to-generate

[4] https://jalammar.github.io/illustrated-transformer/

[5] https://arxiv.org/abs/2010.11934

Claude

ChatGPT