ADL Hw2

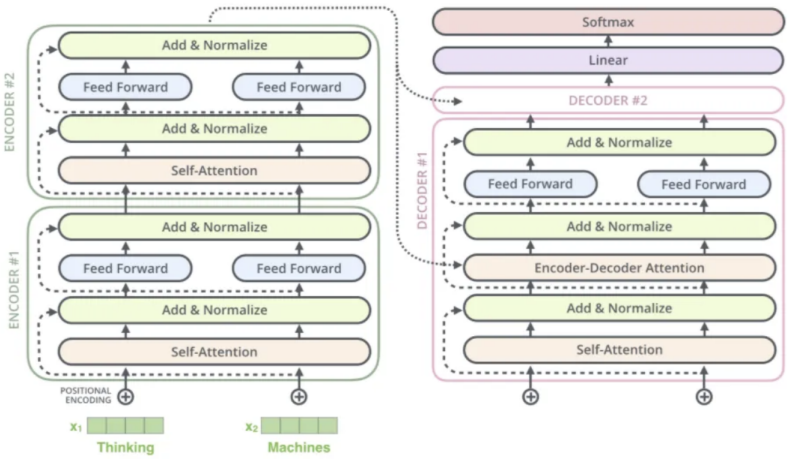
1.

Model (1%)

Describe the model architecture and how it works on text summarization.


```
{
  "_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": "4.34.0",
  "use_cache": true,
  "vocab_size": 250112
}
```


mt5 is a transformer-based model, which contains encoder-decoder architecture.



The architecture includes tokenizer, embedding layer, encoder, decoder and linear transform and softmax layer.

When working on summarization:

1. tokenizer
   The raw text data is being tokenized under the sentencepiece tokenization algorithm. (Will be explained in the Preprocessing section.)
2. embedding layer
   The tokens are sent into an embedding layer, which converts the tokens and the positions of them into vectors.
3. encoder
   Encoder contains self attention mechanism and feed-forward structure. The encoders receive inputs from previous encoders and the self attention mechanism will decide the weight of each vector and then generate the output. Next, the feed-forward network will process those output encodings. Finally, the output encoding will be passed into the next encoder layer.
4. decoder
   Decoder contains self attention mechanism, feed-forward structure and masked multihead attention. Decoder takes the position information and embedding of the output sequence of the previous decoder, and processes it in a similar way in the encoder when occurring self attention and feed-forward. Masked multihead attention mechanism is aiming to prevent the decoder from predicting the result using current or future output, hence making the output sequence partially masked.
5. linear transform and softmax
   Finally, after linear transform, the probability of output sequence is calculated in the softmax layer. After that, according to the conditional probability, we can decide which word is the best choice based on our generation strategy(which will be introduced in later questions).

picture source: https://jalammar.github.io/illustrated-transformer/

Preprocessing (1%)
Describe your preprocessing (e.g. tokenization, data cleaning and etc.)

I use sentencepiece tokenization algorithm to do tokenization. Sentencepiece is a language-independent subword tokenizer and detokenizer. SentencePiece performs tokenization by using subword unit algorithms, such as byte-pair-encoding (BPE) and unigram language models. After it segments input text into smaller subword tokens, using the BPE and unigram language models to construct the appropriate vocabulary.

reference:
https://huggingface.co/docs/transformers/model_doc/t5
https://huggingface.co/docs/transformers/tokenizer_summary#sentencepiece
https://github.com/google/sentencepiece

2.

Hyperparameter (1%)

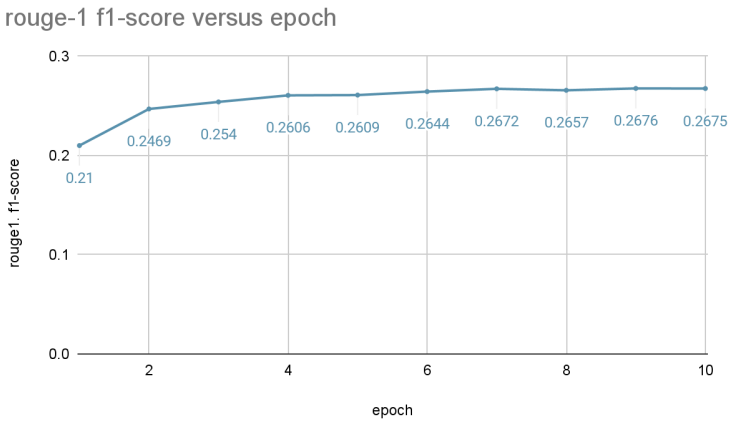
Describe your hyperparameter you use and how you decide it.

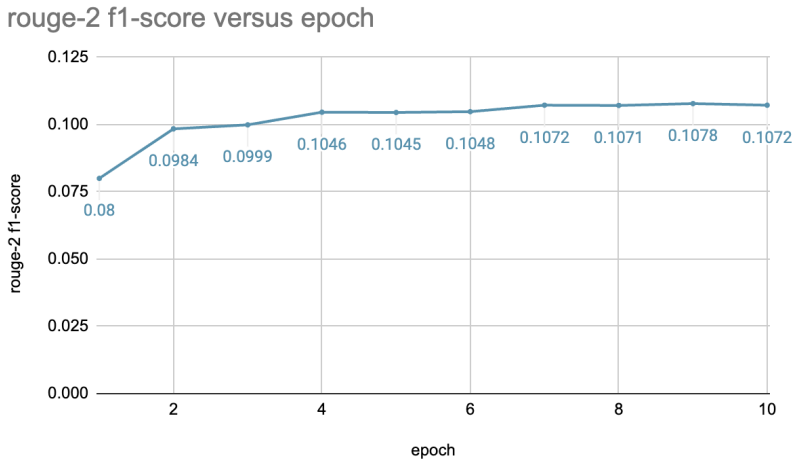
Training Batch size: 2
Gradient accumulation: 1
Epoch: 9
Number of beam: 3
Learning rate: 8e-5
Optimizer: AdamW

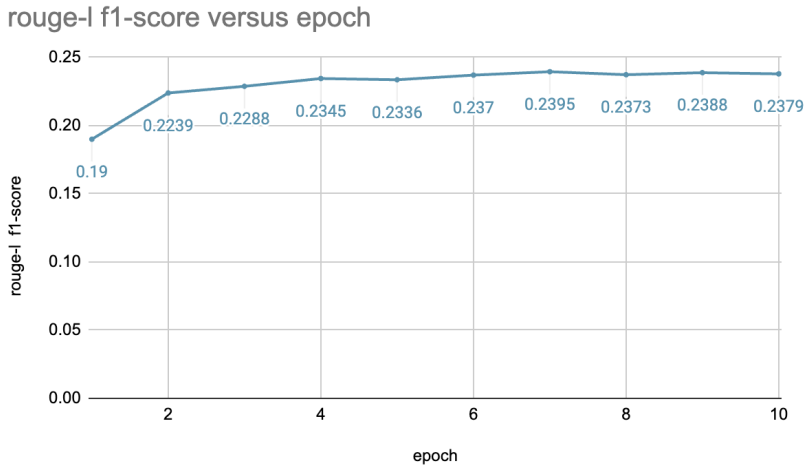
In the final version, I set the batch size to 2 since I've tried using 4 and got a bad result and a smaller batch size (for example, 1) will take a lot more time. Training epoch at about nine epoch will have the best score in my experiments. Learning rate is picked randomly and the result is not bad so I didn't change it. Optimizer remains the same as the sample code.

Learning Curves (1%)

Plot the learning curves (ROUGE versus training steps)







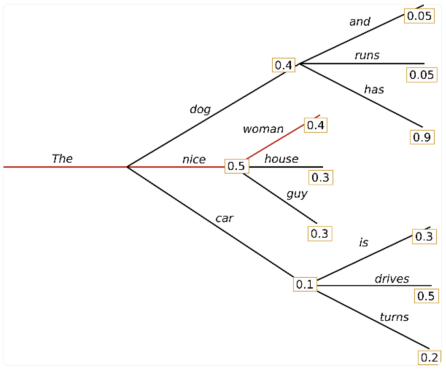
The score is evaluated on the validation data. We can observe that after 4 epochs, the score becomes gentle.

3.

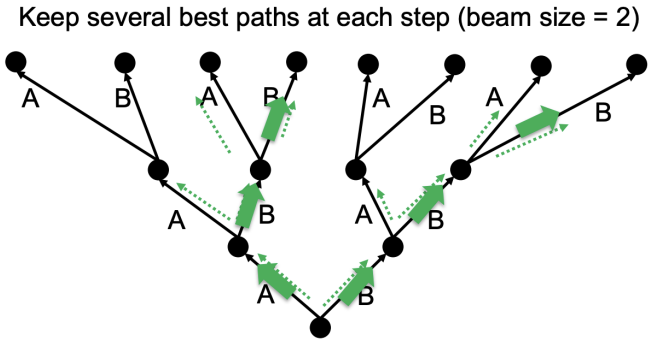
Stratgies (2%)

Describe the detail of the following generation strategies:

1. Greedy:
   Choosing the word which has the highest probability (by argmax) as the current word's next word.

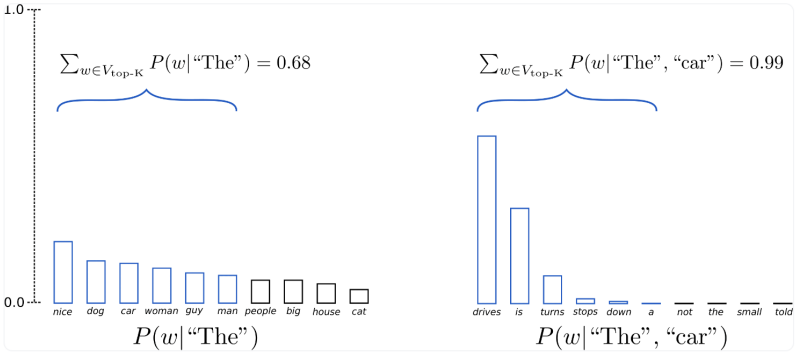


2. Beam Search:
   Keeping the num_beams-highest-probability word at each step. Overall, find the better one among all candidates.



3. Top-k Sampling:
   Sampling the words from the distribution but is limited to top-k probable words, which can avoid weird rare words. (Sample means to randomly pick the next words by its probability distribution.

4. Top-p Sampling:
   Sampling the words from a subset whose sum of the words have bigger probability mass than p. This helps to dynamically shrink and expand the top-k.



5. Temperature:
   Making the distribution sharper(increasing the probability of probable words and lowering the probability of less probable words) or smoother.

information and picture source:

1. https://huggingface.co/blog/how-to-generate
2. https://www.csie.ntu.edu.tw/~miulab/f112-adl/doc/231026_NLGDecoding.pdf

Hyperparameters (4%)
Try at least 2 settings of each strategy and compare the result.

	greedy	beam=3	beam=5	top-p=0.9	top-p=0.5	top-k=50	top-k=30	temperature = 0.9	temperature = 0.6
rouge-1 r	23.36	25.70	26.08	20.56	23.01	19.80	20.28	20.44	22.56
rouge-1 p	28.36	28.96	28.72	22.80	26.93	21.41	22.07	22.40	26.05
rouge-1 f	24.77	26.37	26.50	20.99	24.07	19.97	20.52	20.75	23.46
rouge-1 r	8.89	10.48	10.71	7.19	8.65	6.42	6.70	6.90	8.26
rouge-1 p	10.35	11.56	11.60	7.72	9.71	6.83	7.07	7.30	9.18
rouge-1 f	9.24	10.63	10.78	7.20	8.86	6.40	6.66	6.87	8.42
rouge-1 r	20.91	23.00	23.38	18.35	20.47	17.46	17.91	18.04	20.05
rouge-1 p	25.43	25.98	25.82	20.35	24.01	18.93	19.52	19.78	23.17
rouge-1 f	22.18	23.62	23.77	18.73	21.42	17.62	18.12	18.30	20.85

In my observation, greedy and beam search performs better than sampling. Beam search with a higher number of beams has a greater score than those with a low number of beams. That is quite self-explanatory since having more candidates at any step means we have more choice to choose. Also, setting a lower value of temperature can make the distribution sharper. It makes the result better compared to a higher temperature. Comparing top-k and top-p, guessing that since top-p provides a dynamically changing size of word set, it is more likely to have a greater result. Both top-k and top-p with a lower value have a better consequence by avoiding rare words from the word set.

What is your final generation strategy?
Final strategy: beam search with num of beam = 3. It has a good score and does not take much time.