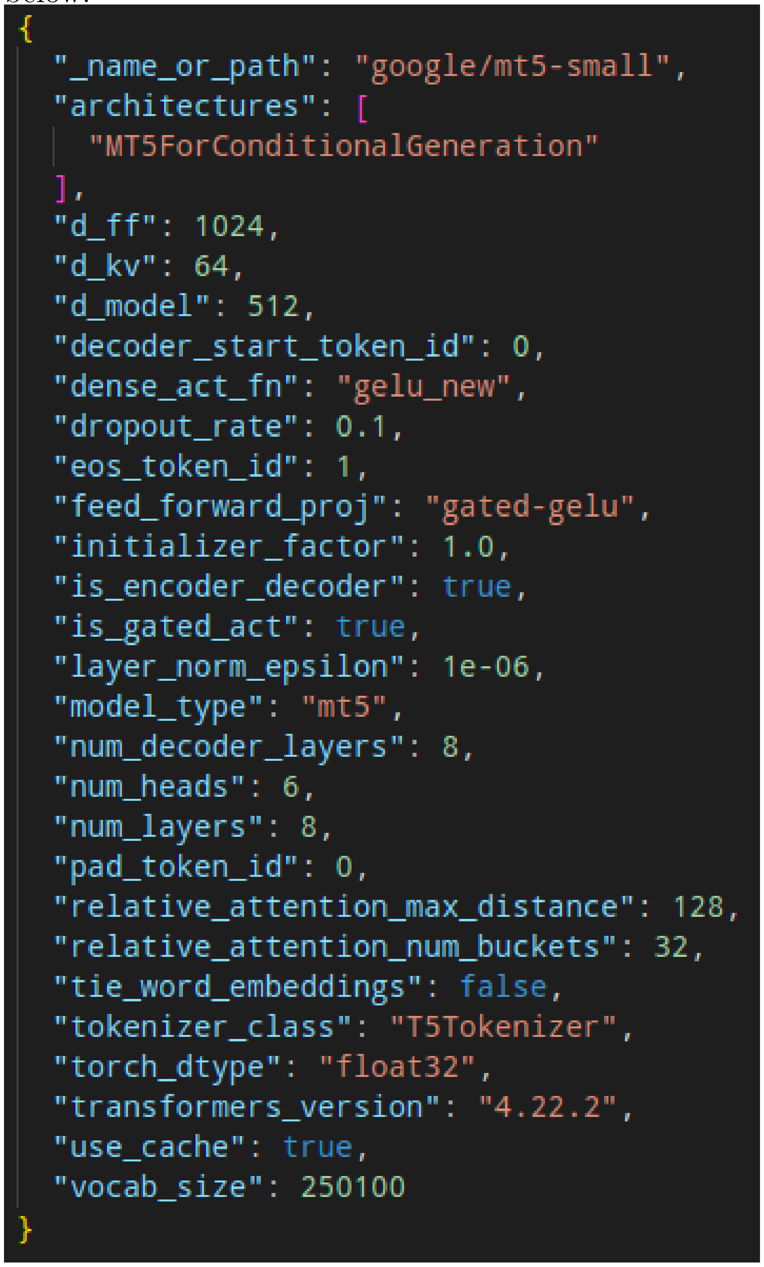
B09902053
何碩宸

ADL HW3

ADL
November 28, 2022

Q1. Model

1. Model
   Describe the model architecture and how it works on text summarization.
   The model I use in this homework is google/mt5-small, and its configuration is shown below.



```
{
  "_name_or_path": "google/mt5-small",
  "architectures": [
    "MT5ForConditionalGeneration"
  ],
  "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": "4.22.2",
  "use_cache": true,
  "vocab_size": 250100
}
```

   MT5Model overrides T5Model, and T5Model is an encoder-decoder model. Since we're using summarization, the prefix "summarize:" will first be added. The model will first take the tokenized text input as the encoder's input. Then the output of the last layer of the encoder will be the encoder hidden states of the decoder. The decoder will perform attention within inputs and encoder hidden states. Finally, the output of the decoder passes a linear layer, and the probabilities for the words are generated.

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

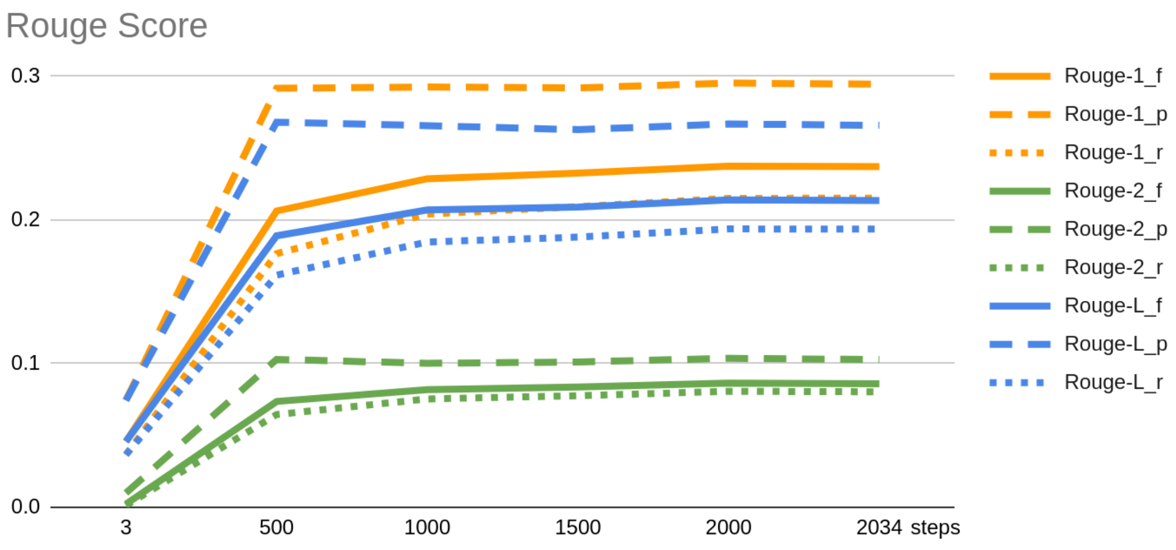
   - Tokenization:
     MT5 applies T5 tokenizer. T5 tokenizer is based on SentencePiece, which implements subword units(e.g. byte pair encoding and unigram language model).
   - Data cleaning:
     The raw data needs to be preprocessed to make it accessible for the mt5 model. Text_column and summary_column will be set to ”maintext” and ”title” respectively. If the title is not given, then the target will be set to be [””] * len.

Q2. Training

1. Hyperparameter
   Describe your hyperparameter you use and how you decide it.

   - fp16
     The memory requirement will be lessened during training.
   - optim adafactor
     The optimizer will be adafactor instead of Adam to reduce the memory requirement of storing momentum.
   - source_prefix "summarize: "
     It is set to identify that it is doing summarization task.
   - learning_rate 1e-4
     A moderate learning rate that is not too low or too high.
   - gradient_accumulation_steps 4
   - eval_accumulation_steps 4
   - per_device_eval_bath_size 4
   - per_device_train_batch_size 4
     Gradient accumulation is applied to reduce the memory requirements for larger batch size.

2. Learning Curves
   Plot the learning curves (ROUGE versus training steps)



Q3. Generation Strategies

1. Strategies
   Describe the detail of the following generation strategies:

   - Greedy
     In this strategy, each choice is greedily determined by the current most probable word. It does not consider previous word probability distribution into consideration.
   - Beam Search
     In this strategy, k is a hyper-parameter and k most probable sequences are being tracked. It will choose the best one among the k sequences at the end.
   - Top-k Sampling
     Instead of choosing the word with the most probable word directly, this strategy performs sampling from top k most probable words.
   - Top-p Sampling
     In this strategy, it adds most probable words into the sample set sequentially until the sum of their probabilities exceeds p. Then it samples the answer from the sample set.
   - Temperature
     Soft-max:

     $$P(w_t) = \frac{e^{s_w}}{\sum_{w' \in V} e^{s_{w'}}}$$

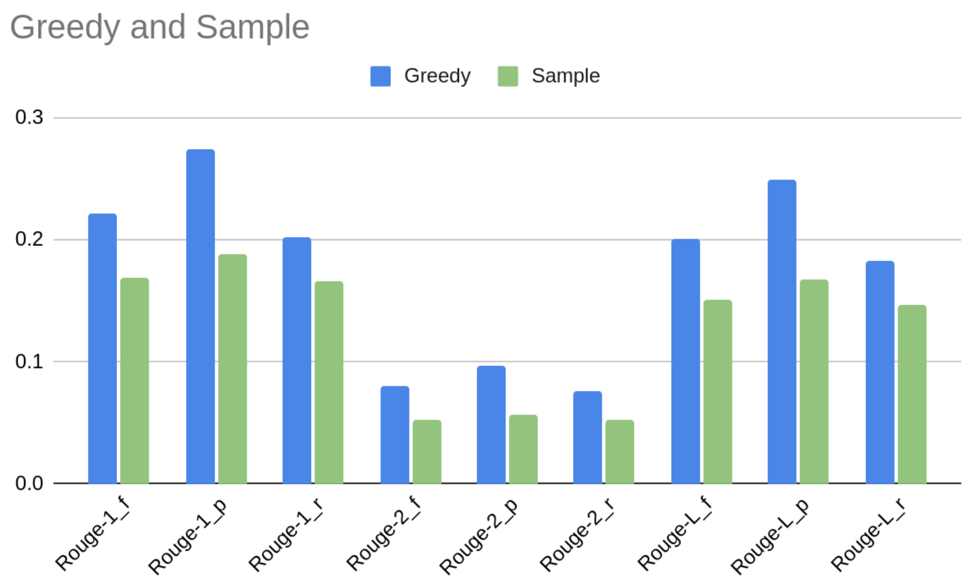
     Soft-max with temperature τ:

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

     τ is high (high temperature): The distribution will be more unified.
     τ is low (low temperature): The distribution will be spikier.

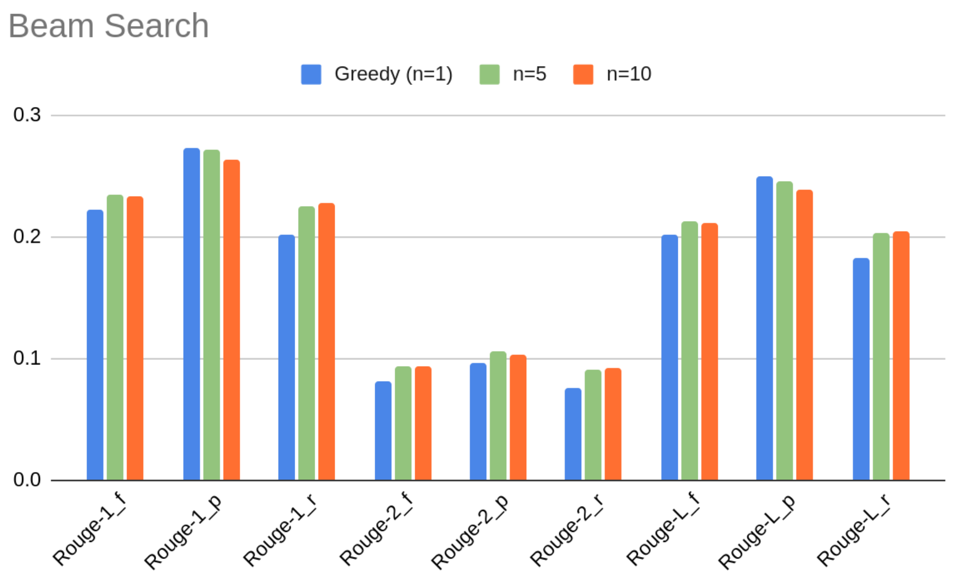
2. Hyperparameters
   Try at least 2 settings of each strategies and compare the result.

   (a) Greedy



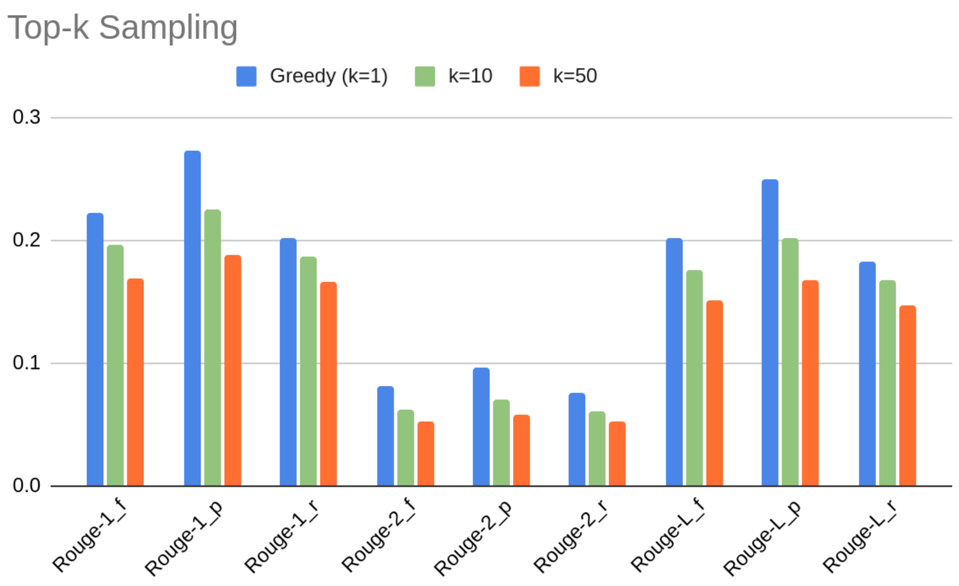
Obviously, greedy performs significantly better than sampling.

(b) Beam Search



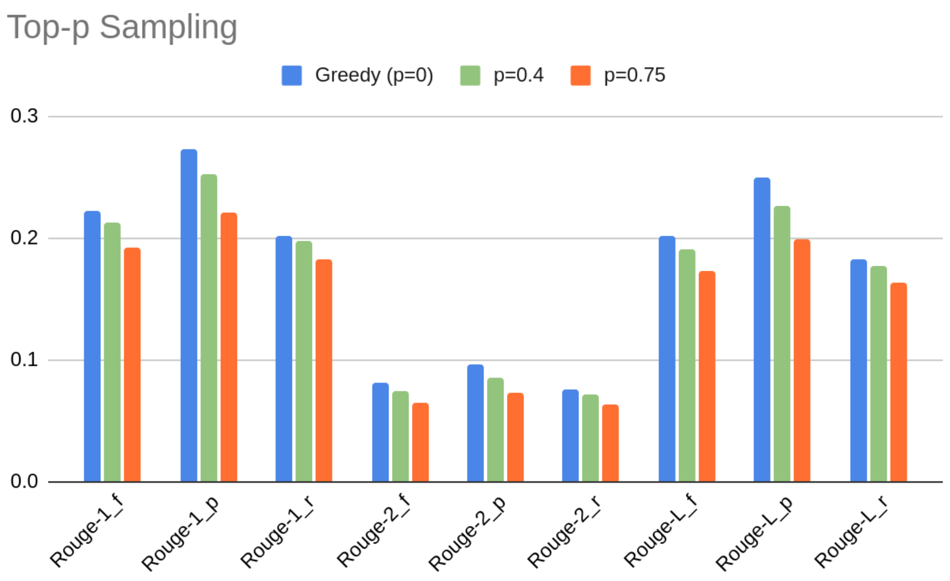
n=5 performs better than greedy in most cases. n=5 and n=10 have similar performance, but n=5 runs faster than n=10 does.

(c) Top-k Sampling



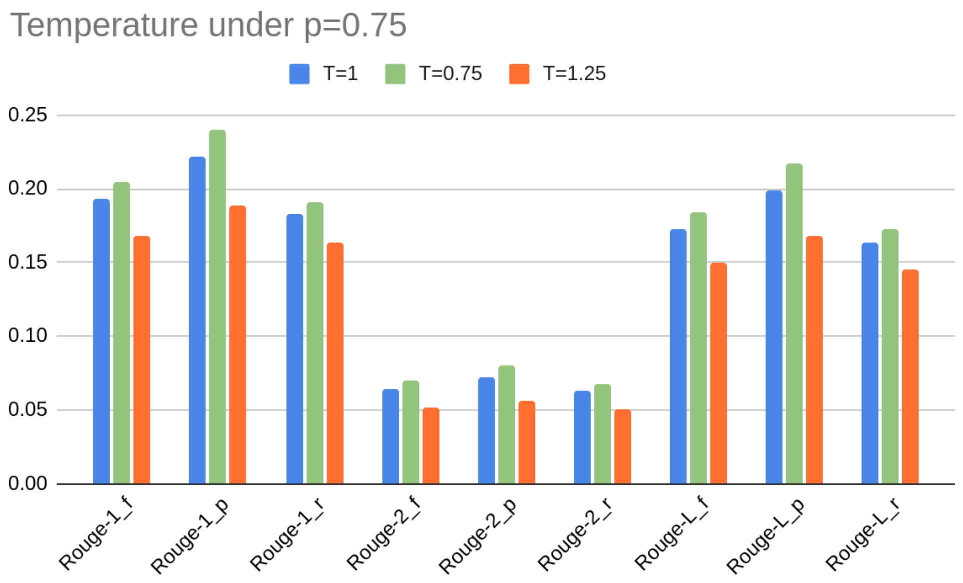
As k becomes higher, it will worsen the performance from the experiment. Probably the selected k is too large.

(d) Top-p Sampling



Similar to top_k, As p becomes higher, it will worsen the performance from the experiment. Probably the selected p is too large.

(e) Temperature



Under the settings of p=0.75 and different temperatures, T=0.75 has the best result out of the three settings.

What is your final generation strategy? (you can combine any of them)
My final generation strategy is numbeams=5.