ADL2022-Fall Homework#3 report

電信碩二 R09942171 黃繼綸

Q1: Model (2%)

- **Model (1%)**
  - Describe the model architecture and how it works on text summarization.

This is the model config:

```
{
  "_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": "4.24.0",
●       "use_cache": true,
●       "vocab_size": 250112
●     }
```
Mt5 model is a transformer-like model, which has encoder and decoder part.
In the training phase, the encoder's input is the tokenized "maintext". It will pass through the transformer encoder to conduct multi-head attention and feed-forward network and then generate a representation for the input. The input of the decoder is the tokenized "title" and conducts cross-attention with the encoder's representation. Finally, the output of the decoder will pass through a linear layer to generate the probability distribution of the vocabulary.
In the inference phase, the decoder takes the "start" token as input and gradually generates the corresponding word. The generated word will be used as the input for the next word generation until the decoder outputs the "end of word" token.

- ### Preprocessing (1%)

  - ■ Describe your preprocessing (e.g. tokenization, data cleaning and etc.)

  T5 tokenizer is based on SentencePiece, which implements subword units (e.g. byte-pair encoding (BPE) and unigram language model). It will divide the word into subword, and find the path with the largest probability.

Q2: Training (2%)

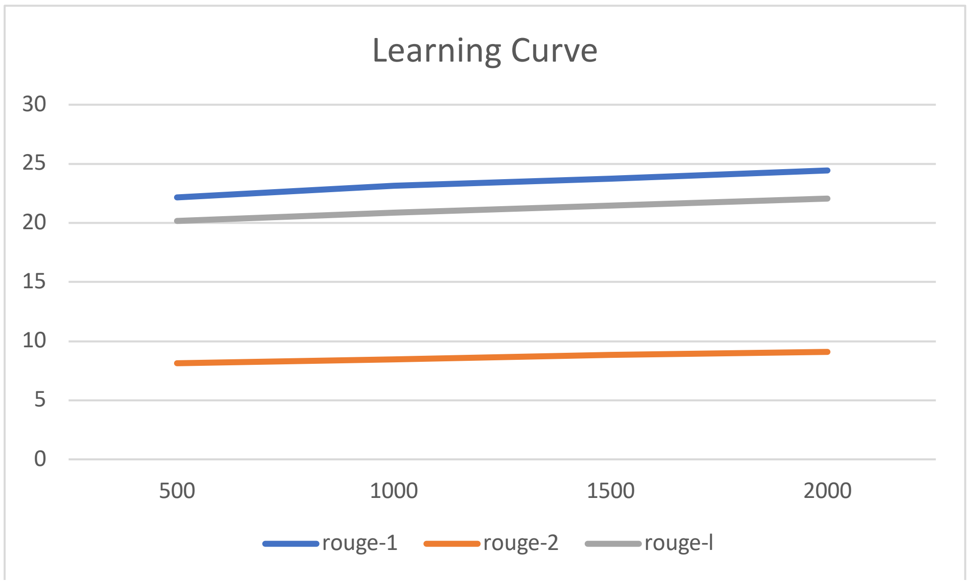
- ### Hyperparameter (1%)

  - ■ Describe your hyperparameter you use and how you decide it.

  I set Per_device_train_batch_size=4 and gradient_accumulation_steps=4 to reduce the usage of memory. In the optimizer setting, I followed the TA's suggestion to use Adafactor with a learning rate of 5e-4. Compared to Adam, Adafactor discards the momentum, which can further reduce the parameters.

- **Learning Curves (1%)**
  - Plot the learning curves (ROUGE versus training steps)

ROUGE versus training steps



Q3: Generation Strategies (6%)

- **Strategies (2%)**

(ref: https://huggingface.co/blog/how-to-generate)

- Describe the detail of the following generation strategies:

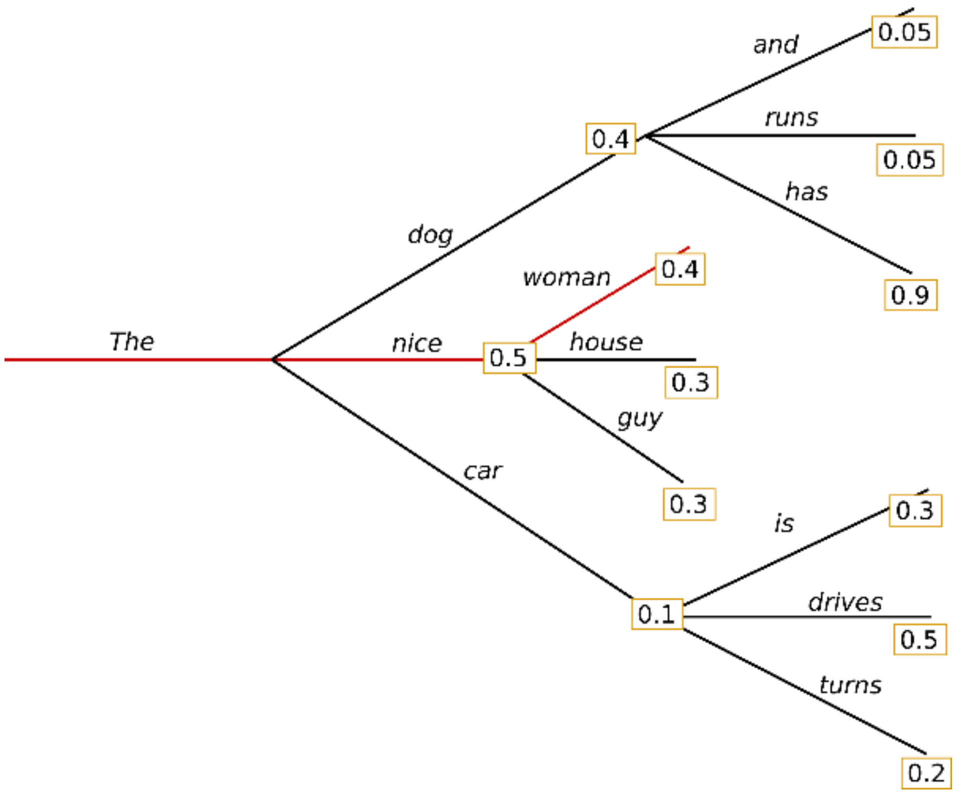


Fig. 1 Generation probability

- **Greedy**

Greedy search simply selects the word with the highest probability as its next word. The formulation is as follows:

$$w_t = argmax_w\ P(w|w_1, w_2, \ldots, w_{t-1})$$

As shown in Fig.1, starting from the word "The", greedy search will pick the next word with the highest probability "nice". After picking "nice", the following subwords are ["woman", "house" and "guy"]. The word "woman" has the highest probability in this phase, so pick "woman" as the word for this step. Therefore, the final decision of this strategy is "The nice woman".

- **Beam Search**

As introduced above, compared to greedy search, beam search chooses the word according to the "num_beams" parameter into a hypothesis set. For example, in Fig.1, I assume that the num_beams=2. Starting from the word "The", the hypothesis set is [“The”, “dog”] and [“The”, “nice”]. In the next time step, beam search will compare all probability of the hypothesis set to obtain the final decision. In this case, [“The”, “dog”, “has”] has a higher probability of 0.36 than the other hypothesis set [“The”, “nice”, “woman”]. Therefore, the final decision of the beam search is “The dog has” instead of “The nice woman” in the greedy search.

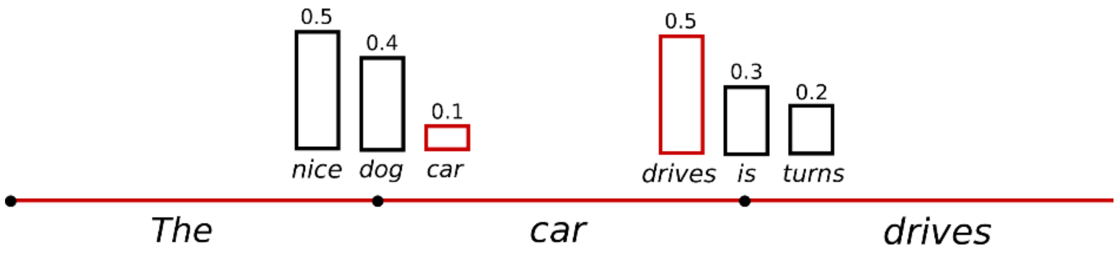


Fig.2 Example for sampling method

Top-k Sampling

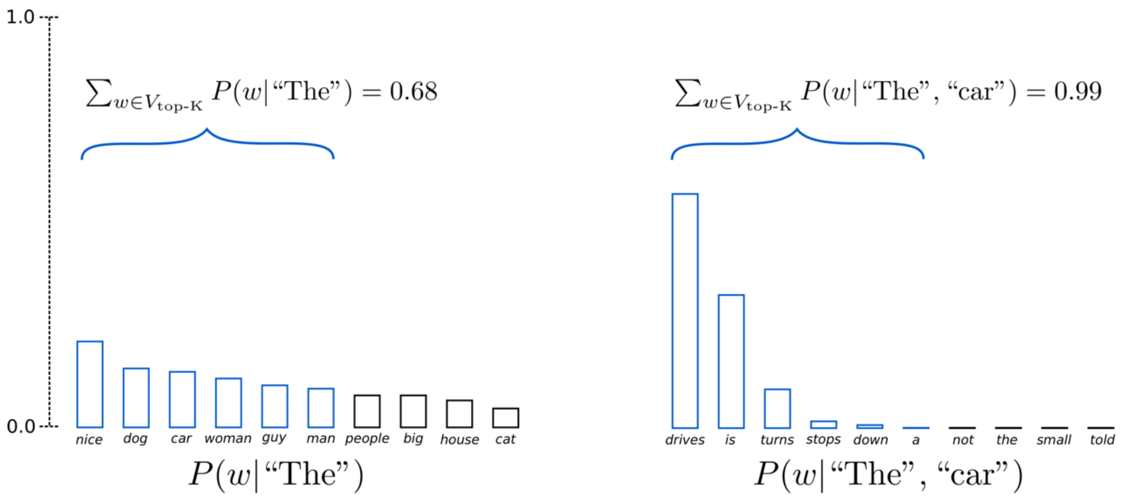


Fig.3 Example for Top-k sampling

As shown in Fig.2, sampling method randomly picks the next word w_t according to the conditional probability distribution, as follows:

$$w_t \sim P(w|w_1, w_2, \dots, w_{t-1})$$

Top-k sampling means we take the top k from the largest probability distribution, and then sample a word from this distribution. The example is demonstrated in Fig.3, the left figure shows that starting from the word “The”, the conditional distribution is determined. We randomly sample a word from this distribution to the next time step. In the next step, as shown in the right figure of Fig.3, given the word [“The” and “car], the distribution is also determined. We then sample the next word according to the probability distribution again.

◆ Top-p Sampling

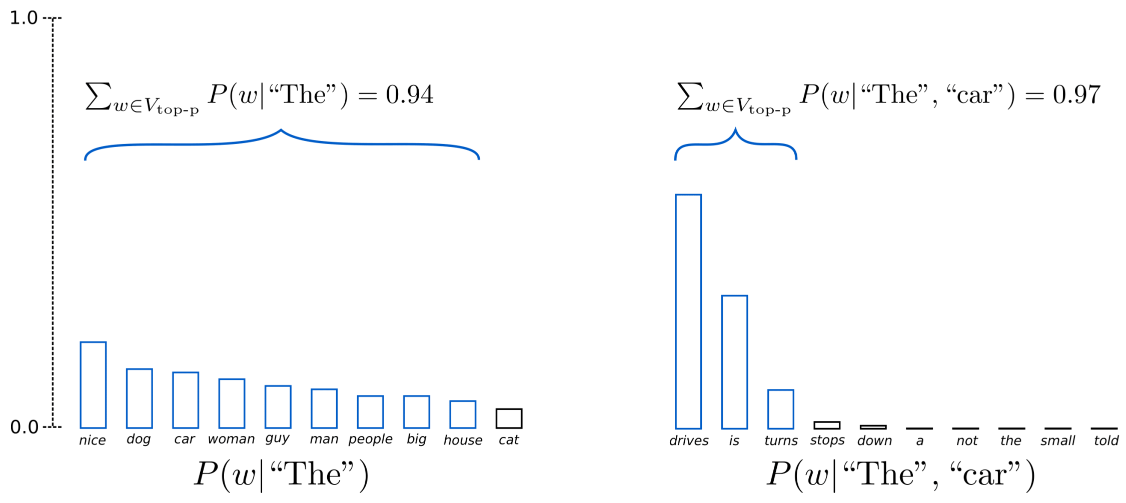


Fig.4 Example for Top-p sampling method

Compared to Top-k sampling, Top-p sampling determines the distribution according to the cumulative probability (CDF). Sort probability distribution from high to low, and calculate whether the CDF exceeds the given threshold. An example is shown in Fig.4, in the left figure, starting from the word "The", the number of subwords in the hypothesis set is 9 while the right figure is 3. Therefore, instead of sampling only from the top-k words, Top-p sampling method can dynamically adjust the next word's probability distribution according to the CDF.

◆ Temperature

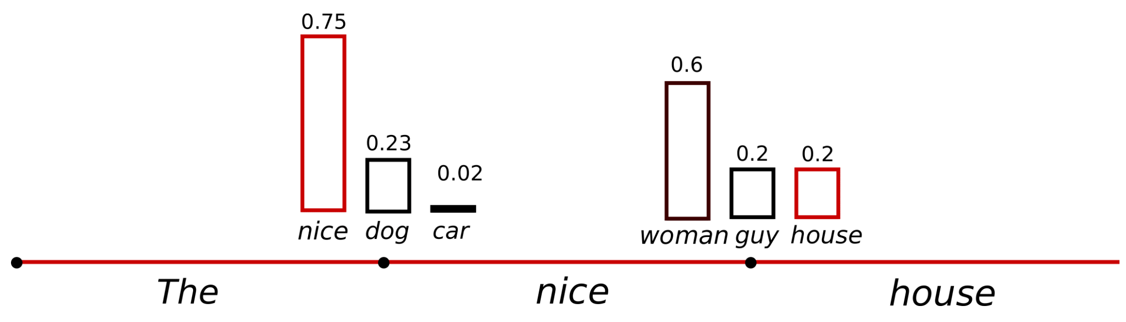


Fig.5 Example of temperature trick

To avoid the sampling method picking the strange text during random sampling, this trick can adjust the distribution (either sharpen or smooth). As demonstrated in Fig.5, after applying temperature to the distribution, the highest probability word will be highlighted.

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

Beam Search:

	Rouge-1	Rouge-2	Rouge-l
Greedy (num_beam=1)	24.44	9.09	22.06
Num_beam=3	25.99	10.39	23.31
Num_beam=5	**26.02**	**10.48**	**23.31**
Num_beam=7	25.95	10.48	23.19

Top-k sampling:

	Rouge-1	Rouge-2	Rouge-l
K=5	22.79	7.88	20.26
K=10	21.6	7.76	19.15
K=20	20.5	7.31	18.17
K=30	19.92	6.48	17.73

Top-p sampling:

	Rouge-1	Rouge-2	Rouge-l
P=0.92	16.02	5.00	14.36
P=0.60	20.12	6.92	17.97
P=0.30	22.92	8.40	20.55
P=0.15	23.95	8.80	21.5

Temperature method:

	Rouge-1	Rouge-2	Rouge-l
Beam5_temp0.7	26.02	10.48	23.31
Beam5_temp1.3	26.02	10.48	23.31
Top_p0.6_temp0.7	23.79	8.69	21.39
Top_p0.6_temp1.3	21.12	7.11	18.84

From the experiments of different decoding strategies, I found that in this task, beam search with k=5 gets the highest performance. Greedy search (k=1) cannot see the global, so the beam search has the better performance compared to the greedy search. However, sampling method in this task will get the bad performance since it may sample a strange word which may in turn affect subsequent results.

In the temperature method experiments, we can see that it has little effect on beam search method. But it can improve the performance of sampling method since the temperature method sharpens the distribution to reduce the probability of sampling bad result.

- What is your final generation strategy? (you can combine any of them)

My final generation strategy is beam search with k=5.