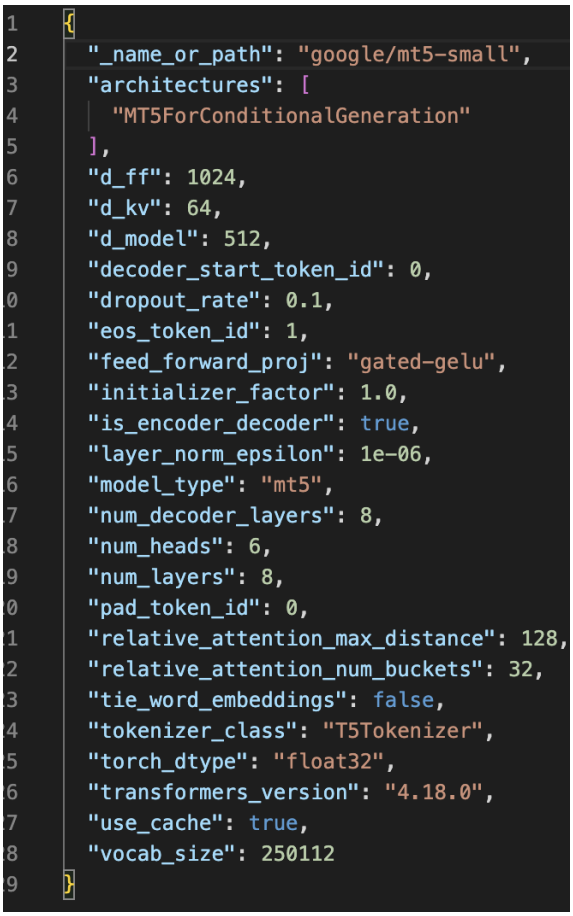
Q1: Model

Model architecture:



```
{
  "_name_or_path": "google/mt5-small",
  "architectures": [
    "MT5ForConditionalGeneration"
  ],
  "d_ff": 1024,
  "d_kv": 64,
  "d_model": 512,
  "decoder_start_token_id": 0,
  "dropout_rate": 0.1,
  "eos_token_id": 1,
  "feed_forward_proj": "gated-gelu",
  "initializer_factor": 1.0,
  "is_encoder_decoder": 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": "4.18.0",
  "use_cache": true,
  "vocab_size": 250112
}
```

- feed_forward_proj: gated-gelu
- number of layers: 8
- number of heads: 6

Preprocessing:

使用t5的built-in tokenizer，eos token, unknown token, padding token分別tag為”</s>”, “<unk>”, “-100”，自動將長度小於1024 or 128的sequence補齊到最大長度，超過最大長度的部分則對其進行truncation，且不採用前綴提示字串來進行任務標記(no prefix)。

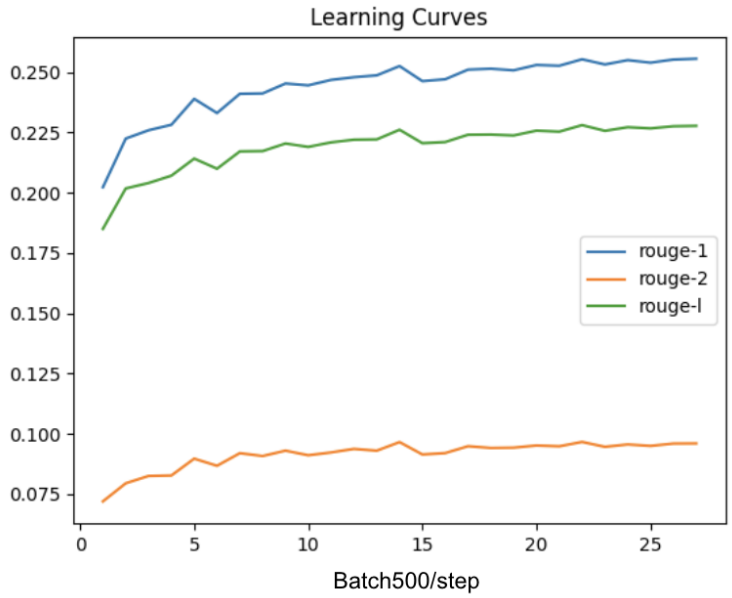
- Tokenizer_class: T5 Tokenizer
- max sequence length for input: 1024
- max sequence length for target text: 128
- pad to max sequence length: True
- source prefix: None

Q2:Training

Hyperparameter:

- number of epoch: 20
- learning rate: 0.0004
- dropout rate: 0.1
- gradient accumulation steps: 8
- batch size: 32
- real batch size: 2
- loss function: CrossEntropyLoss
- optimizer: AdamW

Learning Curve:



Q3: Generation Strategies

Strategies:

- Greedy: 在每一個step中，挑選出highest conditional probability的token來當作predicting result，greedy最大的缺點為其忽略了posterior tokens中的highest probability sequence，而可能因此錯過了best possible sentence
- Beam search: 為greedy的改良版，在每個step中挑選條件機率前N高的tokens作為possible predicting result，最後再從組合中選出整體擁有highest probability的tokens作為predicting result，缺點為隨

著N的增加，decoder speed也會跟著遞減，需要更多的運算成本，且beam search一樣不保證獲得best possible sentence

- Top-k sampling: 每次只從條件機率前k高的tokens取樣，而不考慮其他tokens，即”not probable tokens”的取樣機率為0，使用限制為k的值在一開始就要決定，不能在過程中依照model動態調整
- Top-p sampling: 和top-k的邏輯類似，只是換成計算累積機率分佈要>= p，一旦超過p值就切斷，將不在集合內的probability設為0，並將集合內的進行re-scale以確保他們的sum為1
- Temperature: 實際上就是在原本softmax function各項分母加上一個t，達到”increase probable tokens and reducing not probable tokens”的效果，公式如下

$$P(x_i|x_{1:i-1}) = \frac{\exp(u_i/t)}{\sum_j \exp(u_j/t)}$$

Hyperparameter:
主要試了以下幾種generation，最基本的greedy加上sampling後分數明顯下降，而top-k的部分在結合temperature獲得機率放大的效果後，分數其實沒有太大的變化，然而top-p結合temperature後rouge score卻獲得明顯的提升，最後則是使用現在主流的beam search，並將group設置為2，在分數上僅獲得些微的提升，效果並不顯著，於是最後決定採用一開始的beam_search這個strategy，且每個time step都挑選conditional probability最高的8個作為prediction。

strategy	rouge1(r,p,f)	rouge2(r,p,f)	rougeL(r,p,f)
greedy	(0.244, 0.287, 0.255)	(0.093, 0.105, 0.095)	(0.217, 0.255, 0.227)
greedy with sampling	(0.169, 0.173, 0.167)	(0.054, 0.054, 0.052)	(0.151, 0.155, 0.149)
beam search=8	(0.253, 0.289, 0.262)	(0.103, 0.116, 0.106)	(0.225, 0.257, 0.234)
beam search=8, num_beam_group=2	(0.253, 0.291, 0.263)	(0.103, 0.116, 0.106)	(0.225, 0.259, 0.234)
top k=10 with sampling	(0.222, 0.247, 0.227)	(0.077, 0.083, 0.078)	(0.196, 0.218, 0.200)
top k=10 with	(0.222,	(0.077,	(0.196,

sampling,temperature=1.2	0.244, 0.225)	0.081, 0.076)	0.215, 0.199)
top p=0.9 with sampling	(0.188, 0.195, 0.186)	(0.062, 0.064, 0.061)	(0.167, 0.174, 0.166)
top p=0.9 with sampling,temperature=0.5	(0.237, 0.273, 0.246)	(0.089, 0.098, 0.091)	(0.211, 0.243, 0.219)