b10507008 沈亮燁

Q1

1. Tokenizer

The tokenizer algorithm I use is WordPiece. WordPiece finds the longest subword in the word which is in the vocabulary and splits on it. For example, for the word “gets”, it will be splits into [“get”, “##s”]. When it is not able to find out a subword in the given word, the whole word will be tokenized as unknown.

2. Answer Span

- I convert the answer span start/end position on characters to position on tokens after BERT tokenization by setting the “return_offsets_mapping” option in tokenizer as true. It will return the characters’ offset and thus we can map to the position on tokens.
- The rule is in utils_qa.py. First, it run through all the examples and get rid of those impossible sets (eg. start_index > end_index, answer has a length > max length…...). Next, sort the potential predictions by their score, only keep a certain amount(in the code is 20) of predictions. Finally, compute the softmax of each prediction and pick the one with highest probability as the best prediction.

Q2

1. My Model

- Model: bert-base-chinese
- Performance
  - Public score: 0.76401
  - Private score: 0.76513

$$C(\theta) = -\sum \hat{y} \log(f(x;\theta))$$

- loss function: Cross entropy loss
- optimization Algorithm: AdamW
  learning rate: 3e-5
  batch size: 2 (per_gpu_train_batch_size 1 * gradient_accumulation_steps 2)

2. Another Model

- Model: hfl/chinese-roberta-wwm-ext
- Performance
  - Public score: 0.78933
  - Private score: 0.77687

$$C(\theta) = -\sum \hat{y} \log(f(x;\theta))$$

- loss function: Cross entropy loss
- optimization Algorithm: AdamW
  learning rate: 1e-5
  batch size: 2 (per_gpu_train_batch_size 1 * gradient_accumulation_steps 2)

The difference between two models above:

Below is the config of two models. The architecture difference between bert-base-chinese and hfl/chinese-roberta-wwm-ext is that roberta has more parameters such as bos_token_id, eos_token_id and output_past. Also, the performance of roberta is better than bert-base-chinese. Roberta is pretrained based on the original bert. It trains longer and uses dynamic masking to enhance the performance instead of the next sentence prediction.

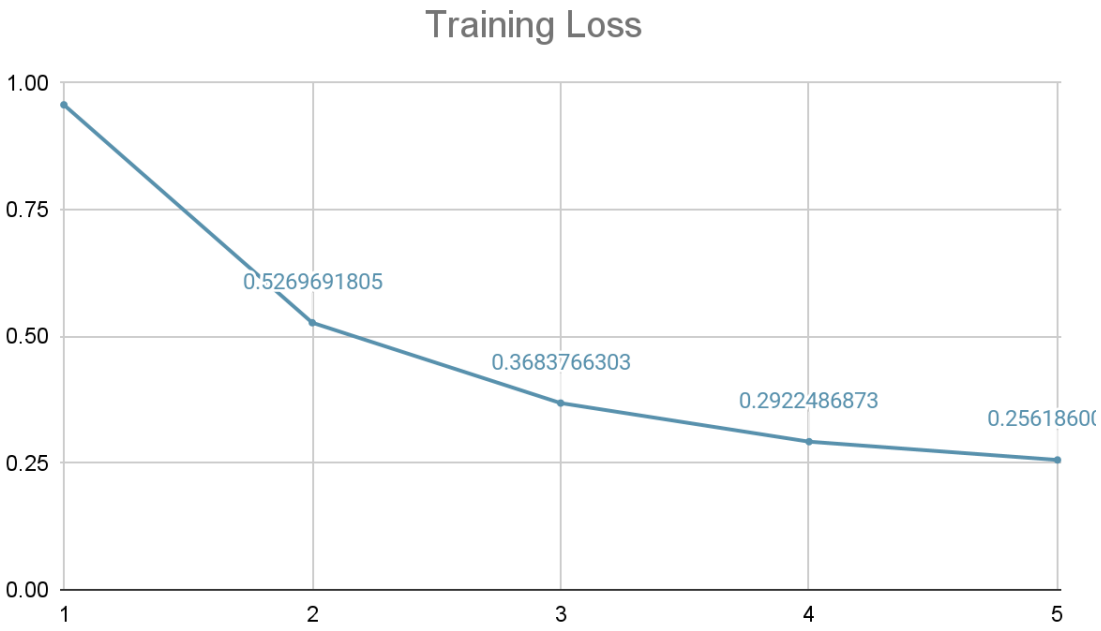
```
{
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "directionality": "bidi",
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "pooler_fc_size": 768,
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "type_vocab_size": 2,
  "vocab_size": 21128
}
```


bert-base-chinese

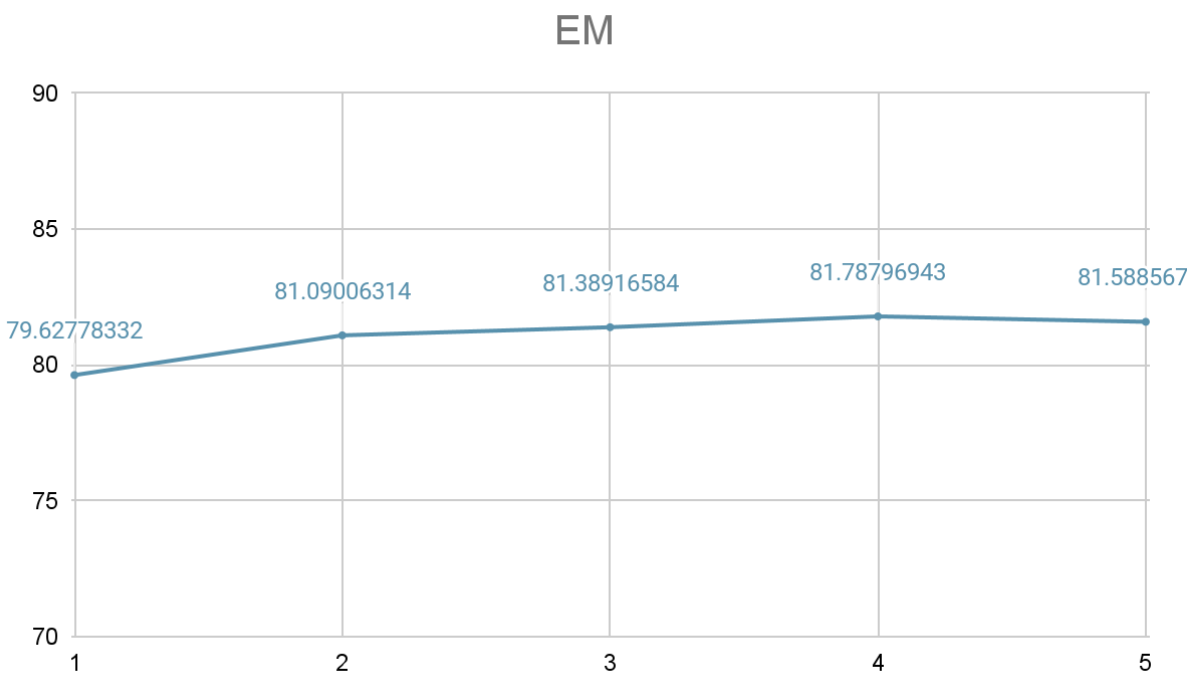

```
{
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "directionality": "bidi",
  "eos_token_id": 2,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "output_past": true,
  "pad_token_id": 0,
  "pooler_fc_size": 768,
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "type_vocab_size": 2,
  "vocab_size": 21128
}
```


hfl/chinese-roberta-wwm-ext

Q3



x-axis is the number of epochs, and the y axis is the training loss.
Training loss is decreasing when the epoch increases, which means the model is performing better and better after training. By this figure, we can also guess that it hasn't reached the minimum of training loss.



x-axis is the number of epochs, and the y axis is the Exact Match metric value on validation. EM value doesn't change significantly when the epoch increases. It looks like maybe it has reached the upper bound of EM.

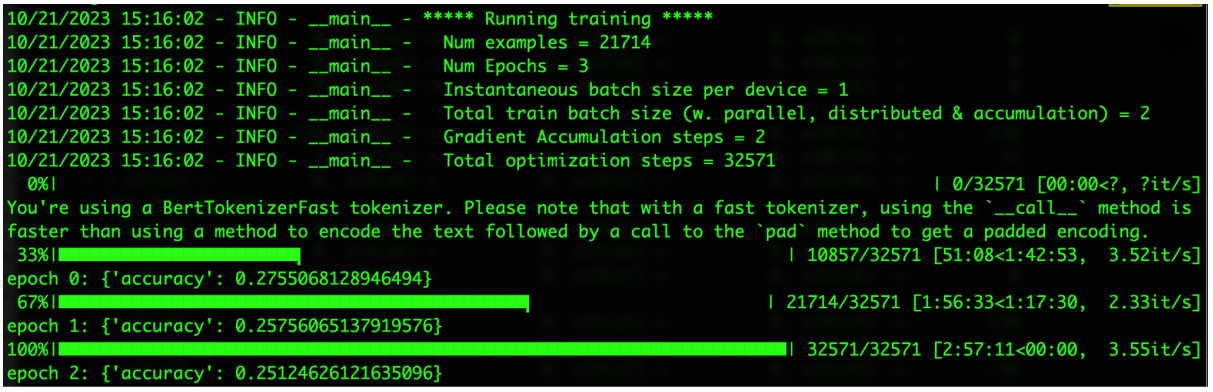
Q4

Config:


```
{
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "directionality": "bidi",
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "pooler_fc_size": 768,
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "type_vocab_size": 2,
  "vocab_size": 21128
}
```

Task: paragraph selection
Hyperparameters:

- batch size: 2 (per_gpu_train_batch_size 1 * gradient_accumulation_steps 2)
- learning rate: 3e-5
- epoch: 3
- max_seq_length: 512



The accuracy is 0.25124626121635096, which means its performance is not good compared with pretrained bert (accuracy:0.9518112329677634).

Q5
I didn't have time to do the bonus, and below is some of the information I found which can build an end-to-end transformer-based model which may solve problems in this homework.

Model: distilbert-base-uncased

It has a smaller size compared with original bert but is faster than original bert. It is trained on raw text without being labeled by humans thus it can use a wide range of data. It is able to be finetuned into a wide range of tasks.

reference:
Q1-1:
https://huggingface.co/learn/nlp-course/chapter6/6?fw=pt&fbclid=IwAR0lGGrmYBc5IaPy_-DyLEto8w0lO2OtLbqBVnnBYnXPIx0KY1CWqIXA99Q
Q2:
https://arxiv.org/pdf/2004.13922.pdf
Q5:
https://huggingface.co/docs/transformers/model_doc/distilbert
https://huggingface.co/distilbert-base-uncased