ADL 2023 HW1 Report

Q1

Tokenizer

The tokenizer algorithm I use is WordPiece.

1. Split the original text into single characters and add the unique ones to the vocabulary list.
2. Compute the score of each pair, using the following formula.

`score=(freq_of_pair)/(freq_of_first_element×freq_of_second_element)`

3. Merge the highest score pair and add it to the vocabulary list.
4. Repeat 2. and 3. until the vocabulary list size reach a threshold or no score is higher than the threshold.

Answer Span

How did you convert the answer span start/end position on characters to position on tokens after BERT tokenization?

When tokenizing, we will record a map called "offset_mapping" which give us a corresponding relationships between token and character position in the original context, this will help us to compute the start_positions and end_positions.

After your model predicts the probability of answer span start/end position, what rules did you apply to determine the final start/end position?

After we know the probability of answer span start/end position, we can use the map "offset_mapping" to convert from the token position to the original context position, and get the final start/end position.

Q2

First Model

Model: bert-base-chinese

Performance: Exact Match Metric Value: 77.93

Loss function: Cross-Entropy Loss

Optimization algorithm: AdamW

Learning rate: 1e-5

Batch size: 32 (8(per device) * 4(gradient_accumulation))

Second Model

Model: hfl/chinese-roberta-wwm-ext

Performance: Exact Match Metric Value: 80.69

Loss function: Cross-Entropy Loss

Optimization algorithm: AdamW

Learning rate: 1e-5

Batch size: 32 (8(per device) * 4(gradient_accumulation))

Difference

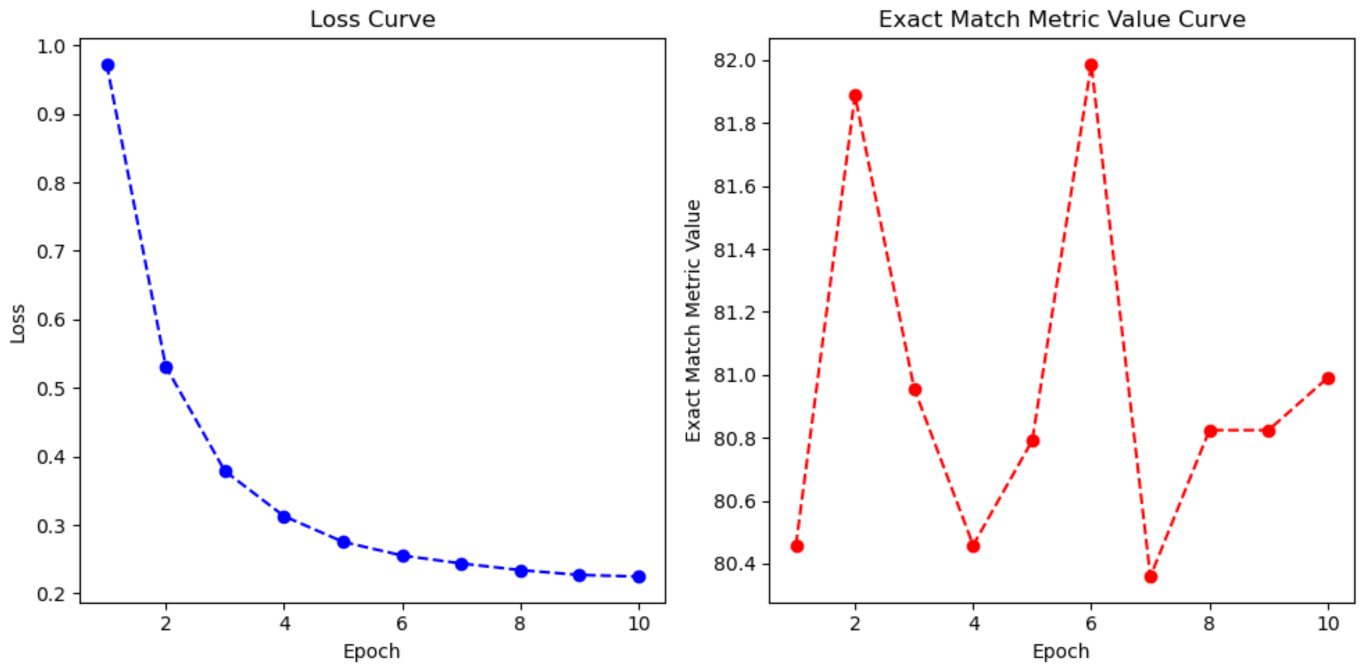
Masking algorithm in tokenization: When masking, bert-base-chinese would mask a single chinese character, while chinese-roberta-wwm-ext mask the whole word.

Example:

When masking 模型, the output of bert-base-chinese may be [MASK] 型, while the output of

chinese-roberta-wwm-ext would be [MASK] [MASK]

Q3



Q4

Non-pretrained model Configuration:

```
export CUDA_VISIBLE_DEVICES=1
file=nonPretrained
python3 question-answering.py \
  --tokenizer_name hfl/chinese-roberta-wwm-ext \
  --model_type bert \
  --train_file ./dataset/preprocessed/train_QA.json \
  --validation_file ./dataset/preprocessed/valid_QA.json \
```

```
  --max_seq_length 512 \
  --per_device_train_batch_size 8 \
  --per_device_eval_batch_size 8 \
  --gradient_accumulation_steps 2 \
  --learning_rate 3e-5 \
  --num_train_epochs 5 \
  --seed 821 \
  --pad_to_max_length \
  --with_tracking \
  --output_dir ./model/QA/"$file"
```

Pretrained model Configuration:

```
export CUDA_VISIBLE_DEVICES=2
file=pretrained
python3 question-answering.py \
  --model_name_or_path hfl/chinese-roberta-wwm-ext \
  --train_file ./dataset/preprocessed/train_QA.json \
  --validation_file ./dataset/preprocessed/valid_QA.json \
  --max_seq_length 512 \
  --per_device_train_batch_size 8 \
  --per_device_eval_batch_size 8 \
  --gradient_accumulation_steps 2 \
  --learning_rate 3e-5 \
  --num_train_epochs 5 \
  --seed 821 \
  --pad_to_max_length \
  --with_tracking \
  --output_dir ./model/QA/"$file"
```

Performance:

Exact Match Metric Value: 7.87(non-pretrained) v.s. 81.38(pertrained)

Q5

Model: hfl/chinese-roberta-wwm-ext

Performance: Exact Match Metric Value: 79.52

Loss function: Cross-Entropy Loss

Optimization algorithm: AdamW

Learning rate: 3e-5

Batch Size: 16 (8(per device) * 2(gradient_accumulation))