Homework #2 Report

Applied Deep Learning

資工碩一　張凱庭　R10922178

Q1: Data processing

1. Tokenizer

The algorithm used in this project is subword tokenization which relies on the principle that frequently used words should not be split into smaller subwords, but rare words should be decomposed into meaningful subwords.

2. Answer Span

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

For the training model, we made the model input a fixed length. Hence we split the data that is over the fixed length. During the tokenization and the split process we record each token's origin position, called it "offset_mapping". With the "offset_mapping" we can easily convert from the new answer span to the origin context's answer span.

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

We keep top 30 highest position of start and end logits and each possible pair of $(start\ position,\ end\ position)$ is considered as a potential answer. The probability of each pair is then calculated by:

$$Softmax(star\ position\ logits\ +\ end\ position\ logits\ )$$

We take the highest probability as our answer. After the model predicts the probability of answer span start/end position, we convert it back to the origin context start/end position by the recorded "offset_mapping".

Q2: Modeling with BERTs and their variants

1. Model detail and performance

Multiple Choice Model	
accuracy	0.95
pretrained model	bert-base-chinese
loss functoin	CrossEntropyLoss
optimizer	AdamW
learning rate	3e-5
batch size	4
num epochs	1

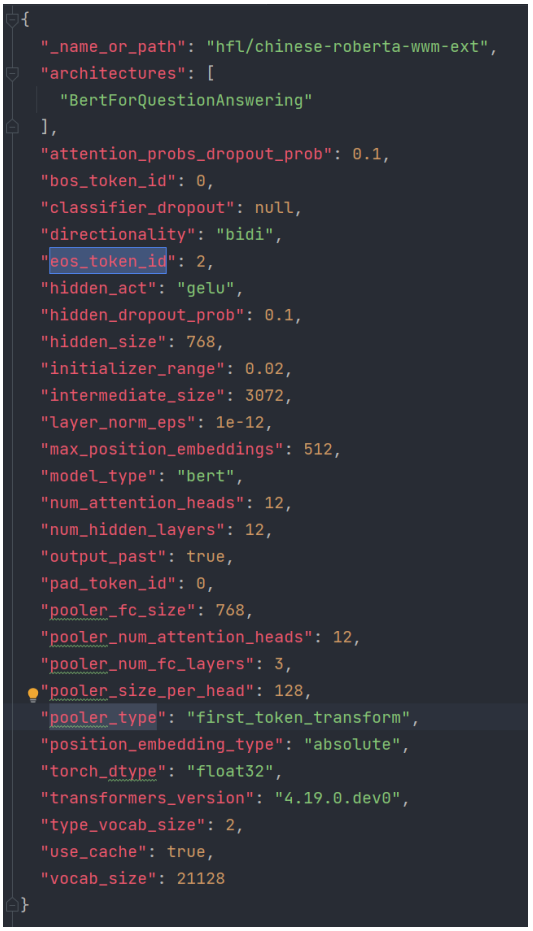
Question Answering Model	
exact match	0.81
pretrained model	chinese-roberta-wwm-ext
loss functoin	CrossEntropyLoss
optimizer	AdamW
learning rate	3e-5
batch size	4
num epochs	1

Multiple Choice Model	Question Answering Model


```
{
  "_name_or_path": "bert-base-chinese",
  "architectures": [
    "BertForMultipleChoice"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "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",
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.19.0.dev0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

```
{
  "_name_or_path": "hfl/chinese-roberta-wwm-ext",
  "architectures": [
    "BertForQuestionAnswering"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "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",
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.19.0.dev0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```


2. **Try another type of pretrained model and describe**

bert-base-chinese	chinese-roberta-wwm-ext
	


```
{
  "_name_or_path": "bert-base-chinese",
  "architectures": [
    "BertForQuestionAnswering"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "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",
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.19.0.dev0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

```
{
  "_name_or_path": "hfl/chinese-roberta-wwm-ext",
  "architectures": [
    "BertForQuestionAnswering"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "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",
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.19.0.dev0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

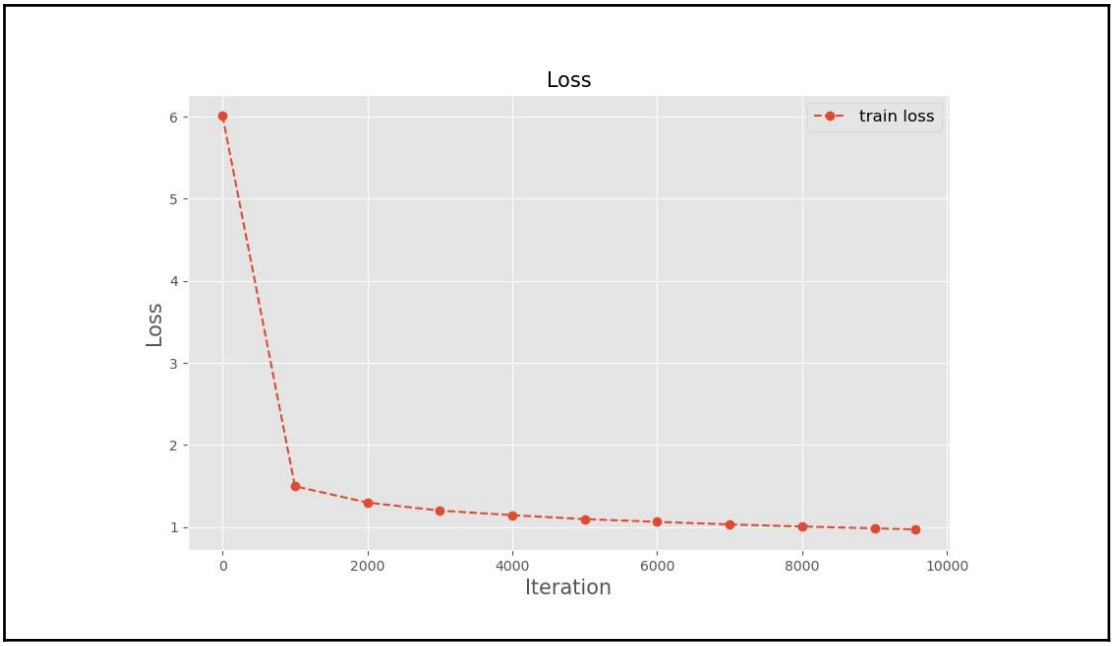

	bert-base-chinese	chinese-roberta-wwm-ext
EM	0.76	0.81

Between the pre-train method there are mainly 2 differences. First, BERT uses a static mask but RoBERTa uses a dynamic mask which randomly generates the mask every time a sample is fed into the model. Second, the BERT pre-train method is guided by the classification loss of asking “is sentence B the sentence which follows sentence A” but RoBERTa simply inputs 2 consecutive full sentences without asking if the sentences are consecutive or not.

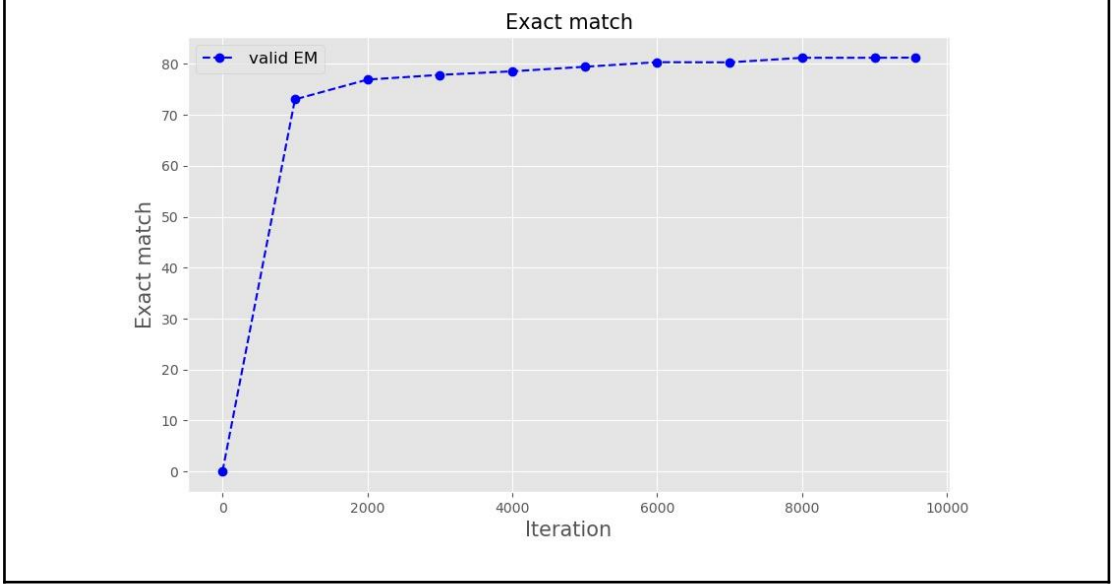
Q3: Curves

1. **Plot the learning curve of your QA model**

Loss curve



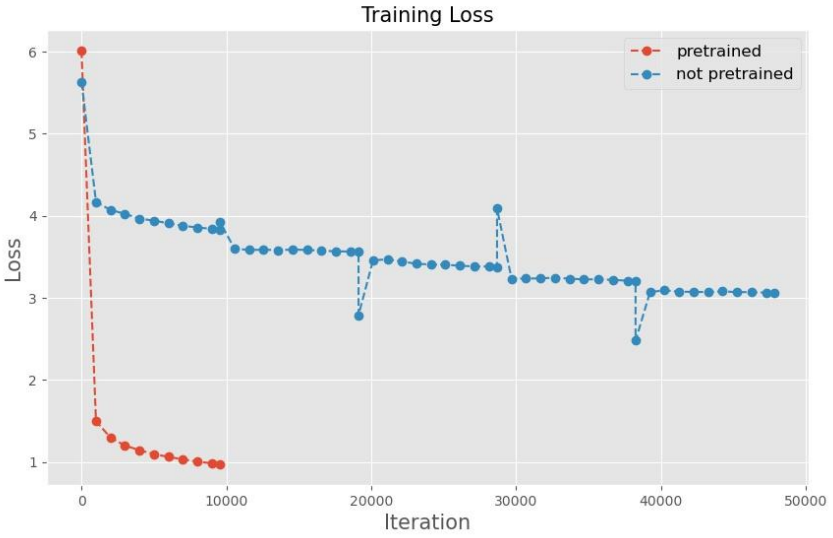
EM curve



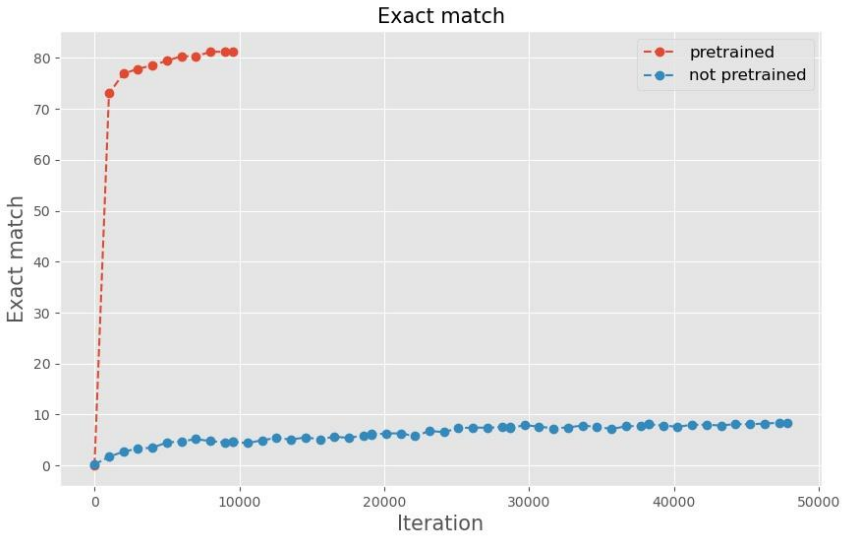
Q4: Pre-trained vs Not Pretrained

The experiment is run on the QA model to compare the difference of loading pretrained weights or not. Both pretrained and not pretrained model is the same as the Question Answering model in **Q2**, except the not pretrained model is trained with 5 epochs and the other only trained 1 epochs.

Loss curve comparison



EM curve comparison



As the learning curve and EM curve, using the pretrained weights can have a huge speed up for the training process. Also, the performance has a big difference. Although the model w/o loading pretrained weights seems to have poor performance, the loss and EM were improved stably. I think if trained with enough epoch, the model would reach a close performance as the model with pretrained weights loaded.

Q5: Bonus: HW1 with BERTs

Intent Classification	Slot tagging


```
{
  "_name_or_path": "bert-base-cased",
  "architectures": [
    "BertForSequenceClassification"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "id2label": {"0": "travel_notification"...},
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "label2id": {"accept_reservations": 57...},
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.19.0.dev0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 28996
}
```



```
{
  "_name_or_path": "bert-base-cased",
  "architectures": [
    "BertForTokenClassification"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "id2label": {"0": "B-people"...},
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "label2id": {"B-date": 4...},
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.19.0.dev0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 28996
}
```


Intent Classification	
pretrained model	bert-base-uncased
loss functoin	CrossEntropyLoss
optimizer	AdamW
learning rate	3e-5
batch size	8
num epochs	3

Intent Classification	
pretrained model	bert-base-uncased
loss functoin	CrossEntropyLoss
optimizer	AdamW
learning rate	3e-5

batch size	8
num epochs	3

Comparison with HW1:

Intent Classification Validation Accuracy	
BERT-based model	0.958
RNN-based model (HW1)	0.892

Slot Tagging Validation Accuracy	
BERT-based model	0.969
RNN-based model (HW1)	0.816