ADL2022-Fall Homework #2 report

電信碩二 R09942171 黃繼綸

Q1: Data processing (2%)

1. Tokenizer (1%)

In this problem, I used the Bert-base model. The tokenizer of this model is Wordpiece, which is similar to the byte-pair encoding (BPE) method. Wordpiece tokenizer first splits the word into sub-word, avoiding similar words influencing each other. The difference between the BPE method and Wordpiece is that Wordpiece generates the new sub-word according to the one that maximizes the likelihood of the training data. In contrast, the BPE method generates the new sub-word according to the most frequent symbol pair. The step-by-step algorithm for Wordpiece tokenizer according to the paper is as follows:

(1) Initialize the word unit inventory with the base characters.
(2) Build a language model on the training data using the word inventory from 1.
(3) Generate a new word unit by combining two units out of the current word inventory. The word unit inventory will be incremented by 1 after adding this new word unit. The new word unit is chosen from all the possible ones so that it increases the likelihood of the training data the most when added to the model.
(4) Goto 2 until a pre-defined limit of word units is reached or the likelihood increase falls below a certain threshold.

2. Answer Span (1%)

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

BERT tokenizer will return an “offset_mapping”, which returns (char start, char end) for each corresponding token. In the start position, we can compare the answer span start position and char start; if they are the same, it means this is the start position. In the end position, we can compare the answer span end position and char end; if they are the same, it means this is the end position.

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?

First, remove the unreasonable cases for the prediction:

(1) Start position > End position
(2) Start position or End position > # of context,
(3) end – start position + 1 > max_answer_length

After removing the unreasonable cases, I took the 20-best prediction as the candidates. Add the logits of the start position and end position as score values, and we can obtain 20 score values in total. Apply softmax operator to these 20 score values to obtain the probability, and select the answer with the highest probability as the final decision.

Q2: Modeling with BERTs and their variants (4%)

1. Describe (2%)
   a. your model (configuration of the transformer model)

   b. performance of your model.

   Validation score:

   Context classification accuracy: 0.9651

   Question answering EM: 0.8023

   c. the loss function you used.

   Cross-entropy loss

   d. The optimization algorithm (e.g. Adam), learning rate and batch size.

   Both Context classification and QA models:

   Optimizer: AdamW with lr=3e-5

   Per device batch size= 8, gradient accumulation steps = 1

2. Try another type of pretrained model and describe (2%)
   a. your model


```
{
  "_name_or_path": "ckiplab/bert-base-chinese-qa",
  "architectures": [
    "BertForMultipleChoice"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "directionality": "bidi",
  "gradient_checkpointing": false,
  "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",
  "tokenizer_class": "BertTokenizerFast",
  "torch_dtype": "float32",
  "transformers_version": "4.22.2",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

```
{
  "_name_or_path": "ckiplab/bert-base-chinese-qa",
  "architectures": [
    "BertForQuestionAnswering"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "directionality": "bidi",
  "gradient_checkpointing": false,
  "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",
  "tokenizer_class": "BertTokenizerFast",
  "torch_dtype": "float32",
  "transformers_version": "4.22.2",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 21128
}
```

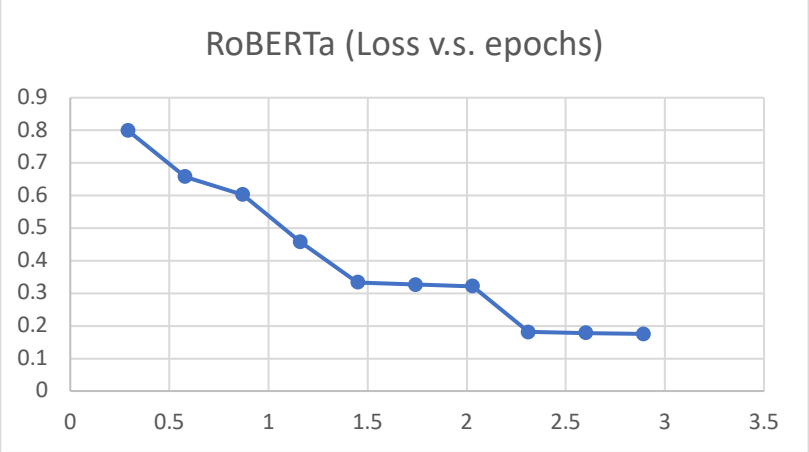

   b. performance of your model

   c. the difference between pretrained model (architecture, pretraining loss, etc.)

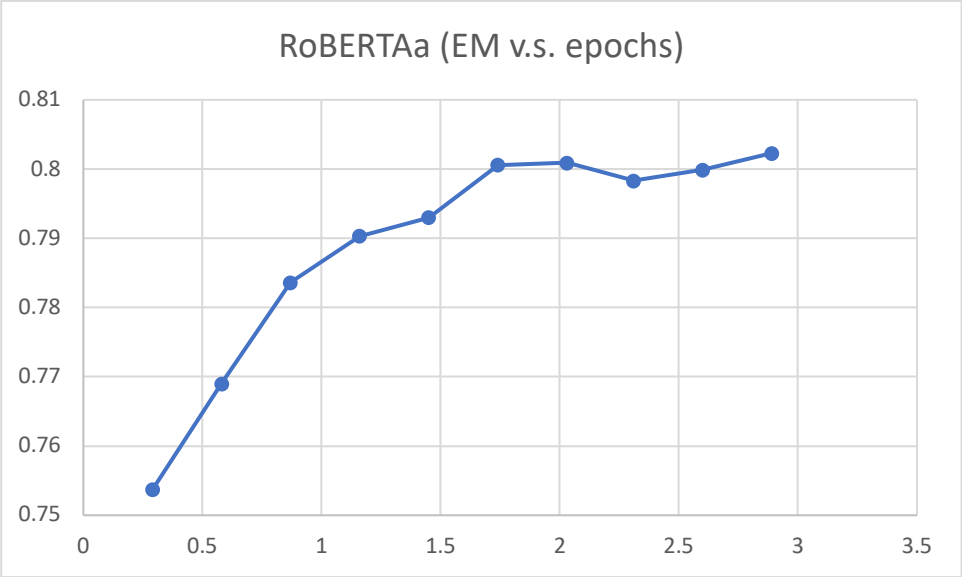
The difference between BERT and RoBERTa is that RoBERTa adopted the dynamic masking mechanism during training, while BERT did not change the mask location during training. In the pretext task, RoBERTa used more data to pretrain the BERT model, which can perform better than the original BERT.

Q3: Curves (1%)

1. Plot the learning curve of your QA model
   a. Learning curve of loss (0.5%)



b. Learning curve of EM (0.5%)



Q4: Pretrained vs Not Pretrained (2%)

- The configuration of the model and how do you train this model

- The performance of this model v.s. BERT

	Context accuracy	EM
This model	0.5274	0.0522
Pretrained BERT	0.9651	0.8023

In this experiment, I found that pretrained weight is very important in language modeling. If not, the loss of the model cannot almost decrease.

Q5: Bonus: HW1 with BERTs (2%)

- Train a BERT-based model on HW1 dataset and describe

1. your model

Both Intent classification and slot tagging:

Bert-base

The training script also provides in the .zip file!

2. performance of your model.

1. Intent classification (1%)

	Validation accuracy
BERT	0.94630
HW1 model	0.91200

2. Slot tagging (1%)

	Token accuracy	Precision	Recall	F1
BERT	0.9733	0.839	0.8539	0.8464
HW1 model	0.9672	0.81	0.80	0.81

.

3. the loss function you used.

Cross entropy loss

4. The optimization algorithm (e.g. Adam), learning rate and batch size.

Both Intent classification and Slot tagging:

Optimizer: AdamW with learning rate 2e-5

Per_device_train_batch_size = 8, gradient accumulation = 1