Lin, Kun-Hsiang
D13922021
jacklin@cmlab.csie.ntu.edu.tw

Q1. Data processing (2%)

(1) Tokenizer

(a) Describe in detail about the tokenization algorithm you use. You need to explain what it does in your own ways.

In Bert, the method used is WordPiece. WordPiece literally means breaking a word into pieces. The BPE (Byte-Pair Encoding) mentioned in class is one implementation method. The basic steps are as follows:

1. Break down all the words in the training data into the smallest character units and create a vocabulary.
2. Select the two most adjacent words in the vocabulary, merge them, and add them to the vocabulary.
3. Repeat step two until the vocabulary reaches the required size.

(2) Answer Span

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

After obtaining the instantiated tokenizer, we can set return_offsets_mapping = True to map the start and end positions of the answer to the original context.

(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?

Using postprocessing, first eliminate impossible answers. After elimination, calculate all probabilities and select the sentence start and end points with the highest probability to form the final answer.

Q2. Modeling with BERTs and their variants

(1) bert-base-chinese (baseline)

(a) Configuration

Key	Paragraph Selection	Question Answering
_name_or_path	bert-base-chinese	bert-base-chinese
architectures	BertForMultipleChoice	BertForQuestionAnswering
attention_probs_dropout_prob	0.1	0.1
classifier_dropout	null	null
directionality	bidi	bidi
hidden_act	gelu	gelu
hidden_dropout_prob	0.1	0.1
hidden_size	768	768
initializer_range	0.02	0.02
intermediate_size	3072	3072
layer_norm_eps	1e-12	1e-12
max_position_embeddings	512	512
model_type	bert	bert
num_attention_heads	12	12
num_hidden_layers	12	12
pad_token_id	0	0
pooler_fc_size	768	768
pooler_num_attention_heads	12	12
pooler_num_fc_layers	3	3
pooler_size_per_head	128	128
pooler_type	first_token_transform	first_token_transform
position_embedding_type	absolute	absolute
torch_type	float32	float32
transformers_version	4.44.2	4.44.2
type_vocab_size	2	2
use_cache_size	true	true
vocab_size	21128	21128

(b) Performance of my model

Paragraph Selection' ACC	0.959
Question Answering's EM	0.802
Question Answering's F1	0.802
Public Score	0.746
Private Score	0.750

(c) Loss function

Cross Entropy Loss.

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

	Paragraph Selection	Question Answering
Optimizer	AdamW	AdamW
Learning rate	3.e-5	3.e-5
Batch size	8	8
Weight decay	0	0
Gradient accumulation	6	6

(2) Variant Bert (hfl/chinese-macbert-base)

(a) Configuration

Key	Paragraph Selection	Question Answering
_name_or_path	hfl/chinese-macbert-base	hfl/chinese-macbert-base
architectures	BertForMultipleChoice	BertForQuestionAnswering
attention_probs_dropout_prob	0.1	0.1
classifier_dropout	null	null
directionality	bidi	bidi
hidden_act	gelu	gelu
hidden_dropout_prob	0.1	0.1
hidden_size	768	768
initializer_range	0.02	0.02
intermediate_size	3072	3072
layer_norm_eps	1e-12	1e-12
max_position_embeddings	512	512
model_type	bert	bert
num_attention_heads	12	12
num_hidden_layers	12	12
pad_token_id	0	0
pooler_fc_size	768	768
pooler_num_attention_heads	12	12
pooler_num_fc_layers	3	3
pooler_size_per_head	128	128
pooler_type	first_token_transform	first_token_transform
position_embedding_type	absolute	absolute
torch_type	float32	float32
transformers_version	4.44.2	4.44.2
type_vocab_size	2	2
use_cache_size	true	true
vocab_size	21128	21128

(b) Performance of my model

Paragraph Selection' ACC	0.969
Question Answering's EM	0.819
Question Answering's F1	0.819
Public Score	0.784
Private Score	0.811

(c) The difference between pre-trained LMs (architecture, pretraining loss, etc.)

MacBERT (Yiming et al., 2020) is an improved BERT with novel MLM as correction pre-training task, which mitigates the discrepancy of pre-training and fine-tuning.
Instead of masking with [MASK] token, which never appears in the fine-tuning stage, the authors propose to use similar words for the masking purpose. A similar word is obtained by using Synonyms toolkit (Wang and Hu, 2017), which is based on word2vec (Mikolov et al., 2013) similarity calculations. If an N-gram is selected to mask, they will find similar words individually. In rare cases, when there is no similar word, they will degrade to use random word replacement.

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

	Paragraph Selection	Question Answering
Optimizer	AdamW	AdamW
Learning rate	3.e-5	3.e-5
Batch size	8	8
Weight decay	0	0
Gradient accumulation	6	6

Q3. Curve

Model: hfl/chinese-macbert-base

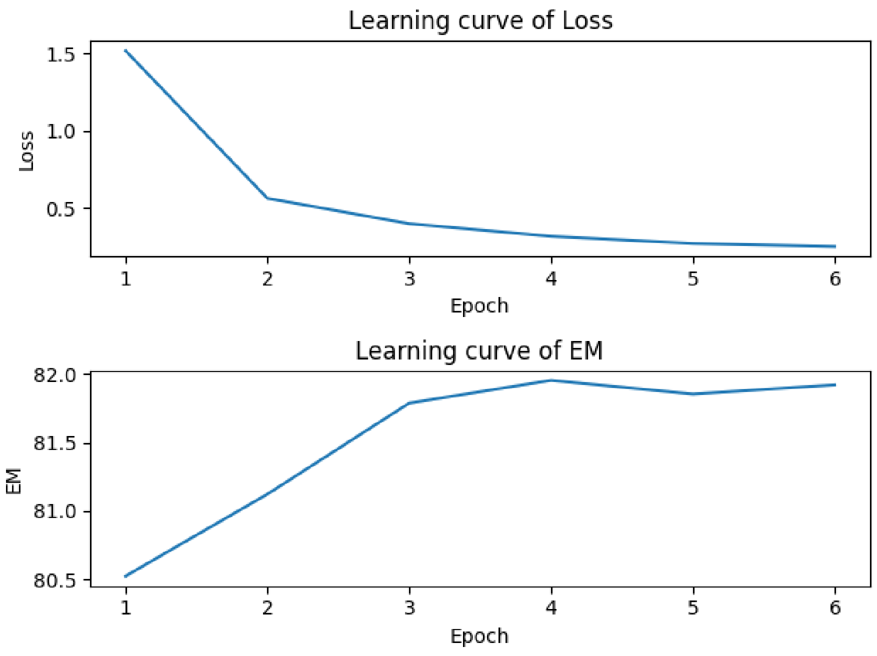

Figure 1: The learning curve of hfl/chinese-macbert-base

From the figure 1, it can be seen that epoch 4 has the best performance. Additionally, it was found that fine-tuning for general language model tasks typically uses a weight decay of 0.1 or 0.01, while the default for Hugging Face is 0. Moreover, the effect of wd=0 appears to be better.

Q4. Pretrained vs Not Pretrained

For this part, I only conducted experiments on QA tasks using the bert-base-chinese model. To remove the pre-trained weights, it is necessary to change **.from_pretrained** to **.form_config** to prevent the model from accessing the pre-trained data.

(a) Configuration

Key	Question Answering
_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_type	float32
transformers_version	4.44.2
type_vocab_size	2
use_cache_size	true
vocab_size	21128

(b) Performance of my model

Method	Metric
Paragraph Selection's ACC	0.959
Question Answering's EM	0.058
Question Answering's F1	0.058
Public Score	0.061
Private Score	0.079

Based on the above results, the QA model is completely unusable, achieving only score=0.06198 on the public dataset. It might be necessary to follow the TA's suggestion to either reduce the model size or train it for a longer period. This also highlights the importance of pre-training language models.

Q5. Bonus: End to end QA

(a) Model

Here, **bert-base-chinese** is applied and converted to an end-to-end model. I concatenated each question's paragraph to form a comprehensive context for end-to-end QA training. Consequently, the context length increased, necessitating an expansion of the model's maximum sequence length from 512 to 2048 tokens. This adjustment significantly increased the training complexity. To date, we have conducted only two experimental runs and have not yet achieved performance comparable to the previously mentioned models.

(b) Performance of my model

Question Answering's EM	32.13
Question Answering's F1	32.13

(c) Loss function

Cross Entropy Loss

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

	Question Answering
Optimizer	AdamW
Learning rate	3.e-5
Batch size	8
Weight decay	0.01
Gradient accumulation	6

(e) Curve and Conclusion

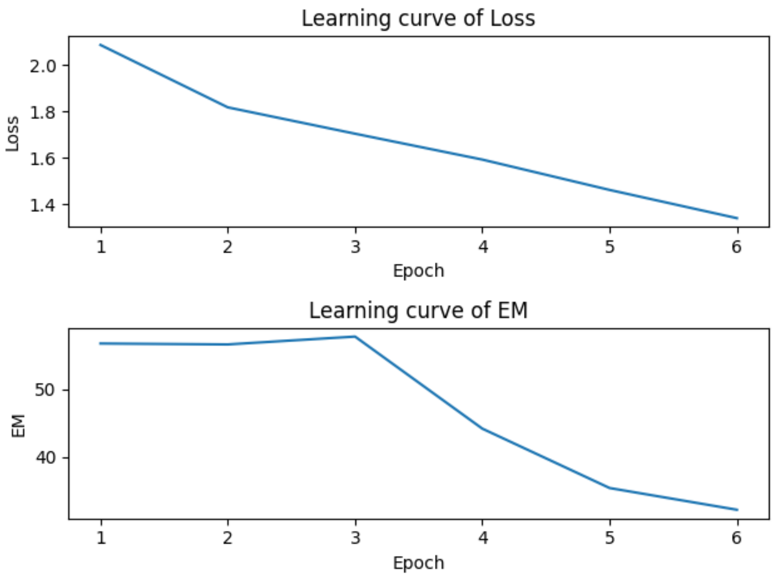

Figure 2: The learning curve of bert-base-chinese on end-to-end QA

Actually, the performance is poor, and the training results are showing a downward trend. This is partly because vanilla BERT generally performs worse on long texts. However, due to limited computational resources and time, I haven't conducted more experiments.