CSIE5431 Applied Deep Learning

Homework 1 Report

Name: 高榮浩 **ID:** R12922127

1. Data Processing

- **Tokenizer**

I utilized the `BertTokenizerFast`. To simplify the explanation of the tokenizer algorithm, I will illustrate it using the following example.

When tokenizing `"Hello 作業一"`, the resulting tokens are as `['hello', '作', '業', '一']` For more detailed information about this tokenization, please refer to the following table.

`'input_ids'`	`[[ 101, 8701, 868, 3511, 671, 102, 0, 0, 0, 0]]`
	List of token ids to be fed to a model
`'token_type_ids'`	`[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]`
	List of token type ids to be fed to a model
`'attention_mask'`	`[[1, 1, 1, 1, 1, 1, 0, 0, 0, 0]]`
	List of indices specifying which tokens should be attended to by the model
`'offset_mapping'`	`[[[0, 0], [0, 5], [6, 7], [7, 8], [8, 9], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]]]`
	`List of information about the start and end of each entity in the original sentence`

It's important to note that the trailing zeros in each row are a result of setting the maximum sequence length to 10, which necessitates padding the sequence to meet that requirement. Additionally, the values `101` and `102` in `'input_ids'` correspond to special tokens. Here's a table detailing these special tokens:

Token	*ID*	*Usage*
[PAD]	0	Use to make arrays of tokens the same size for batching purpose.
[UNK]	100	Represent an out-of-vocabulary token.
[CLS]	101	Represent the class of the input.
[SEP]	102	Separate two different sentences in the same input.
[MASK]	103	Represent a masked token.

- **Answer Span**
  - **Converting Answer Span Position from Characters to Tokens**

In the `prepare_train_features` function, we start with the `start_char` and `end_char` values, which represent the initial character indices for the answer within the text. The goal is to find the first and last indices in the `sequence_ids` where the value is 1, which we'll denote as `token_start_index` and `token_end_index`, respectively.

We then need to determine whether the answer is contained within this text span. If not, we set the `start_position` and `end_position` both to `cls_index`.

However, if the answer is within this text span, we proceed as follows:

We search for the first occurrence of `start_char` equal to the token in `offset_mapping`, starting from `token_start_index`. We increase `token_start_index` until we find this match. The resulting index is recorded as the `start_position`.

Similarly, we search for the first occurrence of `end_char` equal to the token in `offset_mapping`, starting from `token_end_index`. We decrease `token_end_index` until we find this match. The resulting index is recorded as the `end_position`.

This process helps identify the exact token positions that correspond to the answer within the text span.

  - **Determining Final Answer Span Position from Predicted Probabilities**

In the `postprocess_qa_predictions` function found in the `utils_qa.py` file, you have `start_logits` and `end_logits`, which represent the probabilities of a token being the start or end position of the answer. Here's a summary of the steps performed in this function:

Choose the top `n_best_size` highest probabilities from both `start_logits` and `end_logits`. The default value for `n_best_size` is set to 20. This results in selecting the most likely token positions for both the start and end of the answer.

Calculate scores for pairs of `start_logits` and `end_logits`. The score for each combination is calculated by summing their probabilities. This process assesses how well the start and end tokens go together in forming a coherent answer.

Choose the combination with the highest score as the final start and end tokens. This combination represents the most likely answer span within the given context.

Translate the selected token indices into character positions within the context using the `offset_mapping`. This step helps map the selected tokens back to their original positions in the text.

2. Modeling with BERTs and their Variants

- **Paragraph Selection**

Model	bert-base-chinese	hfl/chinese-roberta-wwm-ext
Accuracy (Validation)	0.95746	**0.95813**
Loss Function	CrossEntropyLoss	CrossEntropyLoss
Optimizer	AdamW	AdamW
Learning Rate	1e-5	1e-5
Batch Size	2	16
`per_device_train_batch_size`	1	2
`gradient_accumulation_steps`	2	8

- **Span Selection (Extractive QA)**

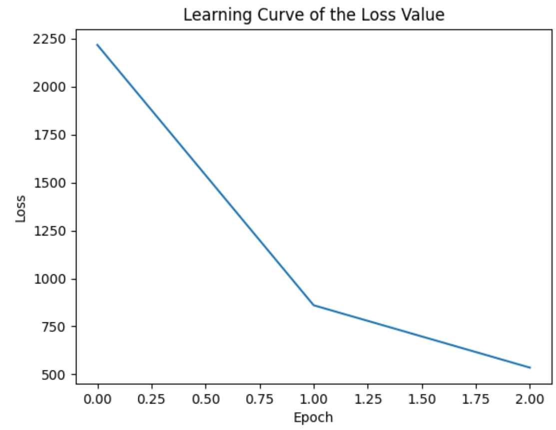
Model	bert-base-chinese	hfl/chinese-roberta-wwm-ext-large
Exact Match (Validation)	0.79860	**0.84580**
Loss Function	CrossEntropyLoss	CrossEntropyLoss
Optimizer	AdamW	AdamW
Learning Rate	1e-5	1e-5
Batch Size	2	16
`per_device_train_batch_size`	1	2
`gradient_accumulation_steps`	2	8

- **Difference**
  - **RoBERTa: A Robustly Optimized BERT Pretraining Approach**
    - Larger training dataset
    - Dynamic masking for better pattern understanding
    - Longer training
    - Larger batch size
    - Removing the Next Sentence Prediction task
    - Using SentencePiece tokenization for handling various languages and subword units effectively
  - **wwm: Whole Word Masking**

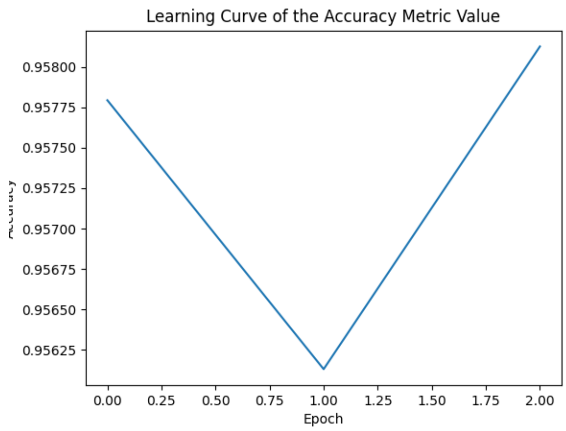
    Whole Word Masking improves traditional Masked Language Model (MLM) training by masking entire words instead of subword tokens. This approach preserves word-level semantics, reduces fragmentation, simplifies the model, and enhances interpretability.
  - **ext: Extended**

3. Curves

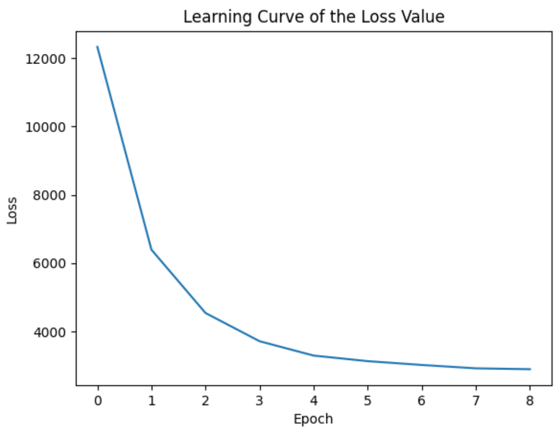
- **Paragraph Selection**
  - **Learning Curve of the Loss Value**



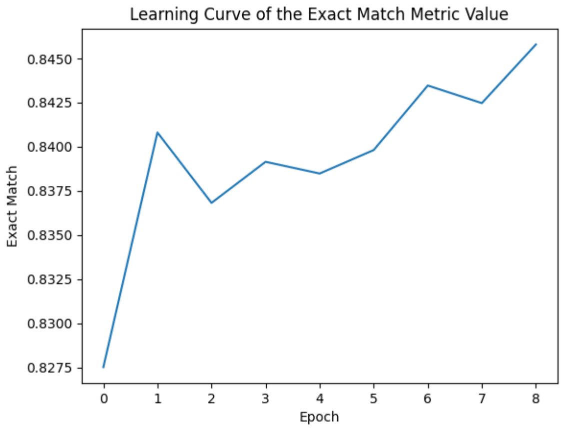
  - **Learning Curve of the Accuracy Metric Value**



- **Span Selection (Extractive QA)**
  - **Learning Curve of the Loss Value**



■ Learning Curve of the Exact Match Metric Value

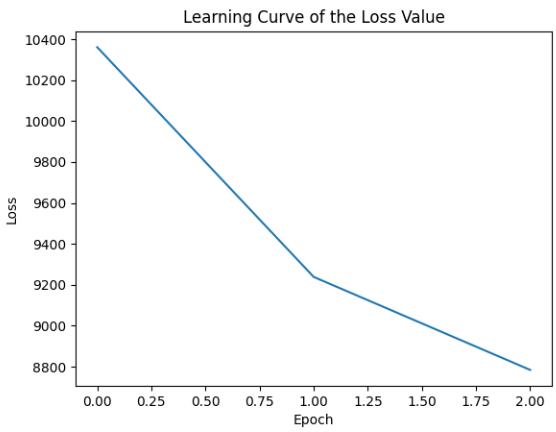


4. Pre-trained vs Not Pre-trained

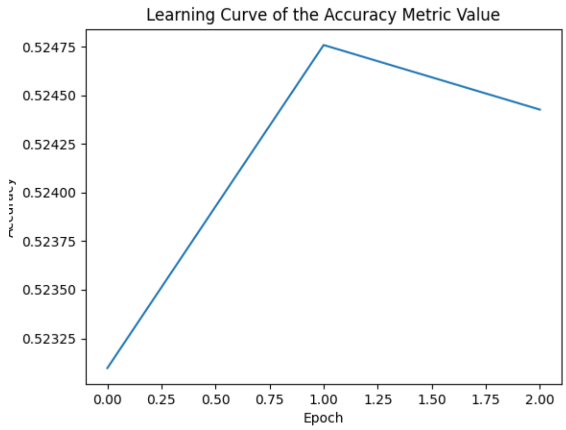
● Paragraph Selection

	Pre-trained	***Not Pre-trained***
Model	hfl/chinese-roberta-wwm-ext	
Accuracy (Validation)	**0.95813**	0.52443
Loss Function	CrossEntropyLoss	CrossEntropyLoss
Optimizer	AdamW	AdamW
Learning Rate	1e-5	1e-5
Batch Size	16	16
`per_device_train_batch_size`	2	2
`gradient_accumulation_steps`	8	8

■ Learning Curve of the Loss Value in Not Pre-trained Model



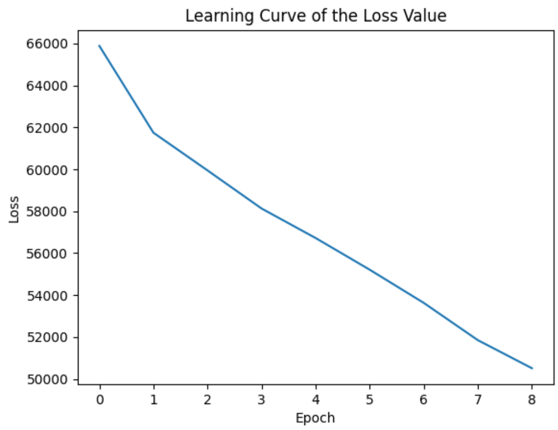
■ Learning Curve of the Accuracy Metric Value in Not Pre-trained Model



● Span Selection (Extractive QA)

	Pre-trained	*Not Pre-trained*
Model	hfl/chinese-roberta-wwm-ext-large	
Exact Match (Validation)	**0.84580**	0.05517
Loss Function	CrossEntropyLoss	CrossEntropyLoss
Optimizer	AdamW	AdamW
Learning Rate	1e-5	1e-5
Batch Size	16	16
`per_device_train_batch_size`	2	2
`gradient_accumulation_steps`	8	8

■ Learning Curve of the Loss Value in Not Pre-trained Model



- ## Learning Curve of the Exact Match Metric Value in Not Pre-trained Model