[ADL 2023 Fall]
HW1 Report

Student ID: R12945039 Name: 楊瀚博

Q1: Data Processing

■ Tokenizer

Based on the Google research team documents, for BERT Chinese tokenizer, we use **"character-based"** tokenization for **"Chinese"**, and **"WordPiece"** tokenization for **"all other languages"** to construct the pre-trained vocabulary.

Character-based tokenization is a relatively simple text processing method where each individual character in a piece of text is treated as a separate token.

WordPiece is similar to Byte-Pair Encoding (BPE). It also represents text at a sub-word level. But unlike BPE, which merges the most frequent pairs of units, WordPiece employs a greedy approach. It tokenizes the text character by character while trying to maximize the likelihood of words and sub-word units present in the model's vocabulary.

The input text is split into sub-word tokens using the pre-trained vocabulary mentioned above, where each token is mapped to a unique identifier from the vocabulary. Additionally, BERT model requires the addition of **special tokens** to the input, like [CLS], [SEP], [UNK], etc.

The tokenizer looks up the token IDs in its vocabulary, which can be found in the **"vocab.txt"** file in the tokenizer model directory (For the case of BERT Chinese, we have **21128** distinct tokens in the vocabulary). Now, the **input text** becomes a **sequence** of **token IDs**, which is the actual input of a BERT model.

■ Answer Span

Question 1

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

Answer 1

Hugging Face's tokenizer provides a way to map tokens back to their original character-level positions. We can use this mapping (offset mapping) to find the character positions of the tokens that correspond to the answer span's start and end positions.

By looking up the "offset mapping" table, we adjust the **"token start index"** and **"token end index"** to the two ends of the answer span by **iteratively** incrementing "token start index" until it **exceeds "char start index"** and decrementing "token end index" until it's **less than "char end index"**.

These positions ("token start index - 1", "token end index + 1") represent the token-level start and end positions of the answer span within the tokenized text.

Question 2

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

Answer 2

Because the highest probability of answer span start/end position may give us some invalid answer, in these cases, we would need to look for the other best start/end position combination.

To find our final answers, we will use the **score** obtained by **adding** the **start** and **end logits** to determine which is the better answer. We won't try to order all the possible start/end pairs, but limit ourselves with a hyper-parameter called **"n_best_size"**. (In this homework, we set **n_best_size** to **20**)

We'll go through all the start and end position with top **"n_best_size"** logits and find all the pair **combinations** of them. If a combination provides a **valid** answer, we **append** it to our possible **answer list**.

Finally, we will **sort** them by their score and choose the start/end pair with the highest score as our final answer start/end position.

Q2: Modeling with BERTs and their variants

■ Chinese-MacBERT-base Description

Model Introduction

MacBERT is an improved version of BERT for Chinese Natural Language Processing. It introduces a correction-type masked language model (**MLM as correction, Mac**) pre-training task to alleviate the inconsistency between pre-training and downstream tasks.

In the traditional Masked Language Model (MLM), a [MASK] token is introduced for masking, but the [MASK] token does not appear in downstream tasks. In **MacBERT**, **similar words** are used to **replace** the **[MASK] token**. Similar words are obtained through the Synonyms toolkit, and the algorithm is based on **word2vec similarity** calculation.

MacBERT also introduces **N-gram masking** techniques. N-gram is a **contiguous sequence** of N items from a given sample of text. N-gram masking hides N-grams (**a group of N words**) instead of individual words, then the model is tasked with predicting a sequence of words. In MacBERT, when an N-gram is chosen for masking, similar words are looked up for each word in the N-gram. If there are no similar words available for replacement, random words are used.

Model Configuration

The main framework of MacBERT is **completely consistent** with the original BERT. We use Hugging Face's **"BertForMultipleChoice"** object with pre-trained weights to fine-tune the "Paragraph Selection" model and **"BertForQuestionAnswering"** object with pre-trained weights to fine-tune the "Span Selection" model. The pre-trained model configurations for both tasks are **identical**.

Configs \ Tasks	**Paragraph Selection**	**Span Selection**
hidden_size	*768*	*768*
max_position_embeddings	*512*	*512*
num_attention_heads	*12*	*12*
num_hidden_layers	*12*	*12*

Functions & Hyperparameters

Configs \ Tasks	**Paragraph Selection**	**Span Selection**
Loss Function	*Cross-Entropy*	*Cross-Entropy*
Optimization Algorithm	*AdamW*	*AdamW*
Learning Rate	*3e-5*	*3e-5*
Batch Size per Device	*1*	*1*
Gradient Accumulation Steps	*64*	*64*
Effective Batch Size	*64*	*64*
Epochs	*1*	*3*
Max Sequence Length	*512*	*512*

Performance on Validation Set

Paragraph Selection Accuracy	*96.6102 %*
Span Selection Exact Match Score	*82.2200 %*

■ Chinese-RoBERTa-wwm-ext Description

Model Introduction

RoBERTa, which stands for Robustly Optimized BERT Pretraining Approach, is a variant of BERT. RoBERTa- wwm-ext is pre-trained with Whole Word Masking (wwm) technique for Chinese, where if a part of a word (in the case of Chinese, a character) is masked, then the other parts of the same word are also masked.

RoBERTa also modifies key hyperparameters in BERT, including removing BERT's next-sentence pretraining objective, and was trained with much larger mini-batch size and learning rate. It also introduces dynamic masking where tokens are masked differently at each epoch.

In training phase, RoBERTa was trained on an order of magnitude more data than BERT, for a longer amount of time.

Model Configuration

The main framework of RoBERTa is **completely consistent** with the original BERT. We use Hugging Face's **"BertForMultipleChoice"** object with pre-trained weights to fine-tune the “Paragraph Selection” model and **“BertForQuestionAnswering”** object with pre-trained weights to fine-tune the “Span Selection” model. The pre-trained model configurations for both tasks are **identical**.

Configs \ Tasks	**Paragraph Selection**	**Span Selection**
hidden_size	*768*	*768*
max_position_embeddings	*512*	*512*
num_attention_heads	*12*	*12*
num_hidden_layers	*12*	*12*

Functions & Hyperparameters

Configs \ Tasks	**Paragraph Selection**	**Span Selection**
Loss Function	*Cross-Entropy*	*Cross-Entropy*
Optimization Algorithm	*AdamW*	*AdamW*
Learning Rate	*3e-5*	*3e-5*
Batch Size per Device	*1*	*1*
Gradient Accumulation Steps	*64*	*64*
Effective Batch Size	*64*	*64*
Epochs	*1*	*3*
Max Sequence Length	*512*	*512*

Performance on Validation Set

Paragraph Selection Accuracy	*95.9787 %*
Span Selection Exact Match Score	*82.1535 %*

■ Model Variants Comparison

MacBERT-base and RoBERTa-wwm-ext both have the identical architecture to the original BERT model. Below shows some key differences between them:

Masking Strategy

- ***MacBERT***
  Introduces a correction-type masked language model (MLM as correction, Mac) pre-training task to alleviate the inconsistency between “pre-training” and “downstream tasks”. It also uses Whole Word Masking (wwm) and N-gram masking techniques.

- ***RoBERTa-wwm-ext***
  Uses Whole Word Masking (wwm) and dynamic masking during pre-training.

Pre-training Tasks

- ***MacBERT***
  Uses “Sentence Order Prediction” objective instead of “Next Sentence Prediction” during pretraining.

- ***RoBERTa-wwm-ext***
  Removes BERT’s “Next Sentence Prediction” pretraining objective.

Performance on validation Set

- ***MacBERT***
  Reaches *96.6102 %* accuracy in paragraph selection task and *82.2200 %* exact match score in span selection task.

- ***RoBERTa-wwm-ext***
  Reaches *95.9787 %* accuracy in paragraph selection task and *82.1535 %* exact match score in span selection task.

Conclusion

In summary, while both models are built based on the same architecture of the original BERT model, they differ in their pre-training tasks and masking strategies.

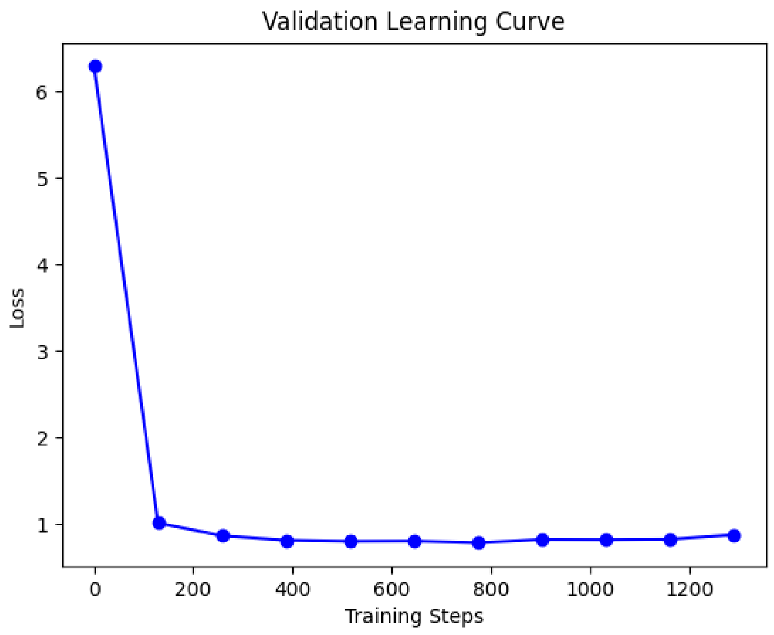
Based on the same fine-tuning strategy and settings, the model which uses **Chinese-MacBERT-base** pre-trained weights reaches a **better validation performance** in both of our downstream tasks: paragraph selection and span selection.

Q3: Curves on Validation Set

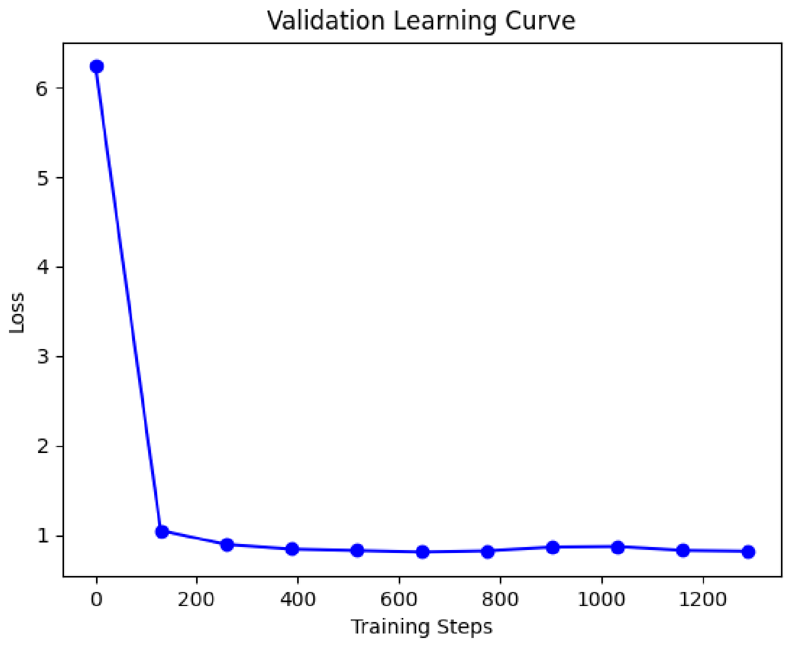
■ Learning Curve of the Loss Value

Below shows the loss curve on span selection **"Validation Set"**. The loss value is the average loss per example.

Chinese-MacBERT-base



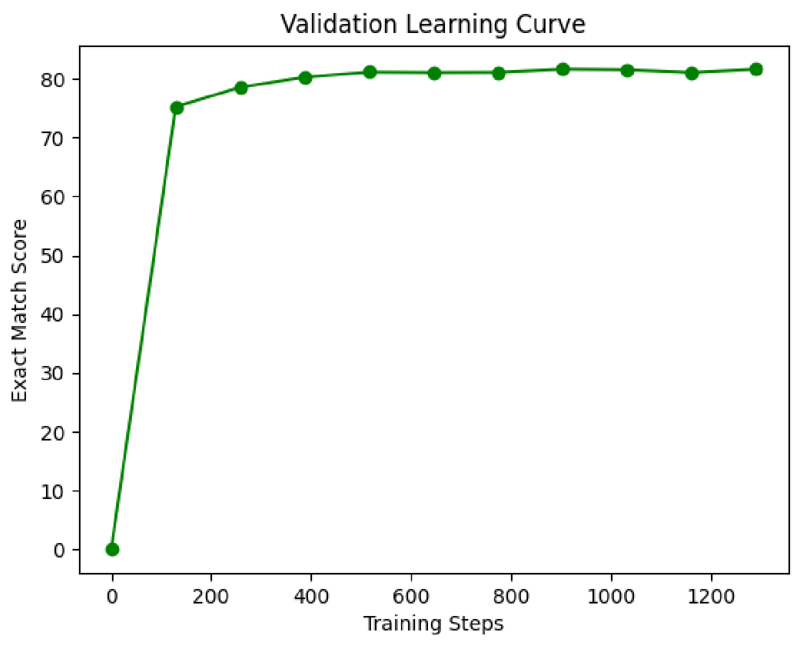
Chinese-RoBERTa-wwm-ext



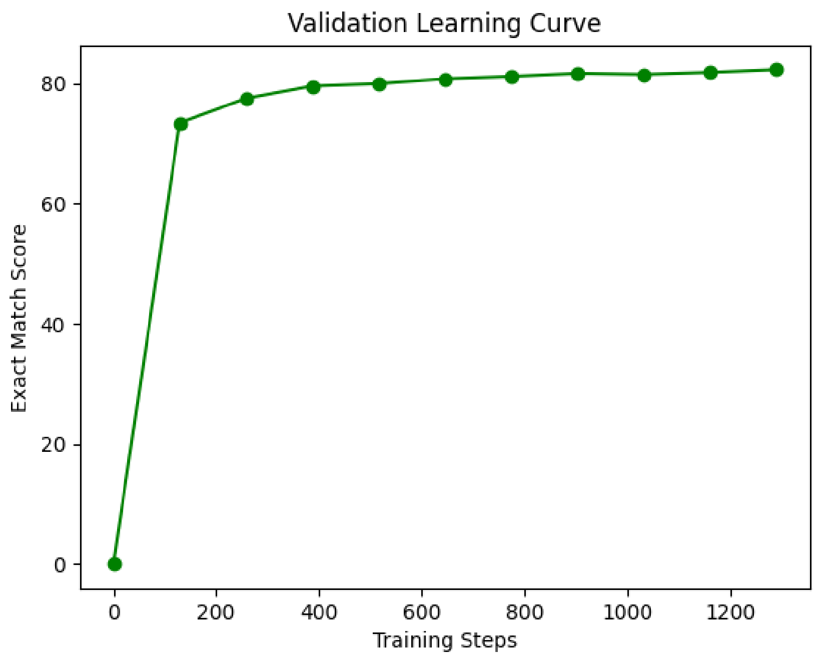
■ Learning Curve of the Exact Match Metric Value

Below shows the exact match metric learning curve on span selection **"Validation Set"**.

Chinese-MacBERT-base



Chinese-RoBERTa-wwm-ext



Q4: Pre-trained vs Not Pre-trained

We choose to train the model from scratch on **“paragraph selection”** task.

■ Model Configuration & Training Method

Model Configuration (BERT based architecture)

*(This table shows the adjusted model configuration, including reduced **“intermediate_size”**, **“num_attention_heads”**, **“num_hidden_layers”**, and set the **“dropout_prob”** to **0**)*

hidden_size	*768 (unchanged)*
intermediate_size	*1024 (original: 3072)*
max_position_embeddings	*512 (unchanged)*
num_attention_heads	*4 (original: 12)*
num_hidden_layers	*4 (original: 12)*
attention_probs_dropout_prob	*0 (original: 0.1)*
hidden_dropout_prob	*0 (original: 0.1)*

Functions & Hyperparameters

Loss Function	*Cross-Entropy*
Optimization Algorithm	*AdamW*
Learning Rate	*3e-5*
Batch Size per Device	*1*
Gradient Accumulation Steps	*4*
Effective Batch Size	*4*
Epochs	*1*
Max Sequence Length	*512*

■ Performance Compared to Pre-trained BERT Variant

Without pre-trained weights, we find it hard for the model to reach the same performance of those models fine-tuned with either MacBERT-base or RoBERTa-wwm-ext pre-trained weights.

The table below shows the **"paragraph selection"** performance comparison between models "with or without" pre-trained weights and also the difference between "original model configuration" and the "adjusted model configuration", which is mentioned above.

Model Types	***Training Hyperparameters***	***Accuracy (Validation)***
With Pre-trained Weights (MacBERT-base)	Epoch Num = 1 Effect. Batch Size = 64 Learning Rate = 3e-5	*96.6102 %*
Without Pre-trained Weights (Original Model Configs)	Epoch Num = 1 Effect. Batch Size = 4 Learning Rate = 3e-5	*53.6723 %*
Without Pre-trained Weights (Adjusted Model Configs)	Epoch Num = 1 Effect. Batch Size = 4 Learning Rate = 3e-5	*55.6663 %*

Based on those different experiment results, we could find that the performance dropped significantly from the model with pre-trained weights to without pre-trained weights. Additionally, for the model without pre-trained weights, we could **reduce** the **model size** and **shut off** the **dropout** mechanism to achieve a **slightly better performance**.