ADL Hw2 Report

學號：B08502141　　姓名：石旻翰　　系級：電機四

1 Data Processing

1.1 Tokenizer Description

I use the `BertTokenizer` pretrained with its corresponding pretrained model(I use hfl/chinese-macbert-base) used for training, and it converts words into tokens which could be recognized by BERT-based pretrained models and tokenizers work as following:

1. Convert each word into token represented by their corresponding unique ID.
2. Add specific tokens such as `[CLS]`, `[SEP]` and `[UNK]` to represent the classification token, separation token and the unknown words during pretrained phase of the tokenizer, respectively.
3. Tokenizer does truncation, padding, or split the context into lists in a task-oriented manner.

1.1.1 Context Selection

Each data is a question and four relevant context, and they are tokenized by following manner:

`[CLS]` tokenized question... `[SEP]` tokenized context 1... `[SEP]` `[PAD]`...
`[CLS]` tokenized question... `[SEP]` tokenized context 2... `[SEP]` `[PAD]`...
`[CLS]` tokenized question... `[SEP]` tokenized context 3... `[SEP]` `[PAD]`...
`[CLS]` tokenized question... `[SEP]` tokenized context 4... `[SEP]` `[PAD]`...

1.1.2 Question Answering

Each data is a question and selected contexts, and they are tokenized and truncated in the following manner:

`[CLS]` tokenized question... `[SEP]` tokenized context part 1... `[SEP]`
`[CLS]` tokenized question... `[SEP]` tokenized context part 2... `[SEP]`
⋮
`[CLS]` tokenized question... `[SEP]` tokenized context part n... `[SEP]` `[PAD]`...

I use tokenizer with a `doc_stride = 32`, and only the context would be truncated, which be done with setting `truncation="only_second"`.

1.2 Answer Span

1.2.1 Find start/end position

As above, contexts are truncated while a stride of it would be preserved in each vector. To elaborate on the detail, denote the maximum length of question and context pair, the length of the question, the length of the context part in each vector, and the padding length by L, L_q, L_c, L_p, respectively, and assume that $L_q <$ `doc_stride` and the whole question context pair is divided into n vectors. Therefore, we have:

$$L = 1 + L_q + 1 + L_{c,i} + 1, 1 \leq i < n$$
$$L = 1 + L_q + 1 + L_{c,n} + 1 + L_p$$

For `doc_stride` = t, we have:

$$L_{c,i-1}[t\ :] = L_{c,i}[:\ t], \forall 1 < i \leq n.$$

Then I find out that each vector contains context of length $L'_c = L - 3 - L_q$, and hence the i-th vector will contain context$[i \cdot (L'_c - \texttt{doc_stride}) : i \cdot (L'_c - \texttt{doc_stride}) + L'_c]$. This information makes it more easy to find the original start/end positions in these vectors.

1.2.2 Determine the final start/end position

After predicting the start/end positions in each vector, I take the best 20 probabilties from the logits and exclude those positions that are

- Not in the context
- With negative length

After finding the valid start/end positions pairs, return the answer which has the highest probabilties at the start and the end. The predicted answer is then slightly modified by `post_process` in `utils.py`. In this function, I add (「,」 , 《,》) to complete the bracket if the complement present. In addition, the answers with half-width comma was changed to adding "<answer>", for TAs remind that it will be mistaken for csv compiler on kaggle.

2 Modeling with BERTs and their variants

2.1 Baseline Model

1. Pretrained Model: `bert-base-chinese`
   Configuration:
   - Hidden size: 768
   - Hidden Layers: 12
   - Attention Heads: 12
   - Intermediate Size: 3072
   - max_len: 512
2. Performance

	Train Acc. / EM	Train Loss	Val. Acc. / EM	Val. Loss
Context Selection	0.9967	0.0048	0.9579	0.1403
Question Answering	0.8369	0.0621	0.7688	0.9322

3. Loss Function: CrossEntropyLoss
4. Optimization:
   - Algorithm: AdamW with `lr` $= 2 \times 10^{-5}$ and `weight decay` $= 1 \times 10^{-6}$
   - Scheduler: `get_cosine_schedule_with_warmup` from transfromers

2.2 Improved model

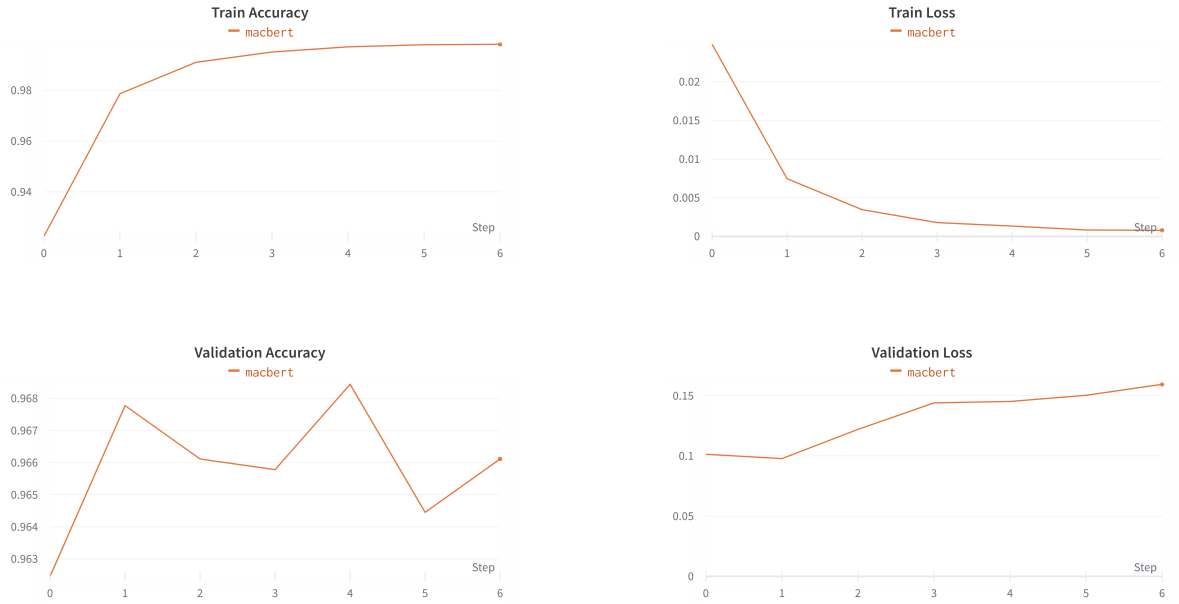
1. Pretrained Model: `hfl/chinese-macbert-base`
   Configuration:
   - Hidden size: 1024
   - Hidden Layers: 24
   - Attention Heads: 16
   - Intermediate Size: 4096
   - max_len: 512
2. Performance

	Train Acc. / EM	Train Loss	Val. Acc. / EM	Val. Loss
Context Selection	0.9980	0.0008	0.9661	0.1595
Question Answering	0.8788	0.0402	0.8060	1.2252

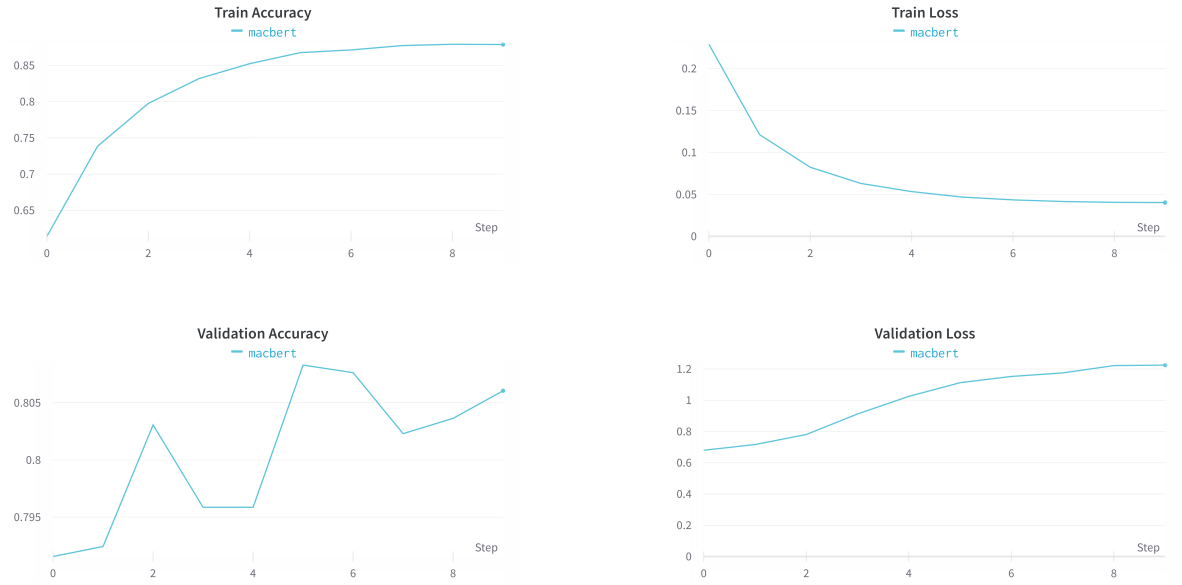
3. Loss Function: CrossEntropyLoss
4. Optimization:
   - Algorithm: AdamW with lr = 1×10^{-5} and weight decay = 1×10^{-5}
   - Scheduler: `get_cosine_schedule_with_warmup` from transfromers

3 Curves

3.1 Learning curves of context selection



3.2 Learning curves of question answering



4 Pretrained vs From Scratch

4.1 Configuration

The configuration on both tasks are the same. Configuration:

- Hidden size: 768
- Hidden Layers: 4
- Attention Heads: 4
- Intermediate Size: 512
- max_len: 512

4.2 Performance

The best accuracy and the lowest loss are reported in following table.

	Train Acc. / EM	Train Loss	Val. Acc. / EM	Val. Loss
Context Selection	0.9793	0.0434	0.374	3.112
Question Answering	0.8616	0.0327	0.449	0.927

Table 1: Performance of training model from scratch

4.3 Comparison

The model was trained from scrtach by simply removing the process of loading pretrained weight, who fits well on the training set but appears to be overfitting on context-selection task, for the validation loss has been increasing.

5 HW1 with BERTs

5.1 Model

I use `bert-base-uncased` from transformers.

5.1.1 Configuration

- Hidden size: 768
- Hidden Layers: 12
- Attention Heads: 12
- Intermediate Size: 3072
- max_len: 512

5.2 Performance

5.2.1 Intent Classification

	Train Acc.	Val. Acc.
HW1	1	0.85
BERT-base-uncased	0.9887	0.9701

Table 2: Use BERT for Intent Classification

5.2.2 Slog Tagging

	Train Acc.	Val. Acc.
HW1	0.998	0.785
BERT-base-uncased	0.812	0.831

Table 3: Use BERT for Slot Tagging

5.3 Loss Function

Still use `CrossEntropyLoss` for these two tasks (as hw1 report).

5.4 Optimizer Algorithm

Optimizer: AdamW with learning rate = 1×10^{-5} and weight decay = 1×10^{-6}
Scheduler: `get_cosine_schedule_with_warmup`

References

[1] https://huggingface.co/bert-base-chinese/tree/main

[2] https://huggingface.co/hfl/chinese-macbert-base/tree/main