ADL HW1 Report

Q1: Data processing (2%)

Tokenizer (1%):

I used the tokenizer from *bert-base-chinese*. It uses WordPiece tokenization algorithm. It first splits text into basic tokens (Chinese characters or words) and then further breaks unknown tokens into subword units from a fixed vocabulary.

Answer Span (1%):

During preprocessing, I stored the offset mapping between characters in the original context and the corresponding tokens after tokenization. The start and end indices were mapped to the closest token indices using this offset mapping.

At inference, the model outputs probability distributions for start and end positions. It selects the most likely (start, end) pair that satisfies start $\leq$ end, and restrict the span length to be within a maximum answer length.

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

Describe (2%)

Task	Paragraph selection	Span selection
pre-trained model	bert-base-chinese	bert-base-chinese
training time (h:mm:ss)	1:59:28	2:55:02
validation accuracy	95.21%	79.23%
testing accuracy	73.80%	
loss function	cross-entropy loss	cross-entropy loss (mean of start and end)
optimization algorithm	AdamW	AdamW
max len	512	512
batch size	2	2
epoch	1	2
learning rate	3e-5	3e-5

Try another type of pre-trained LMs and describe (2%)

Task	Paragraph selection	Span selection
pre-trained model	chinese-macbert-base	chinese-macbert-base
training time (h:mm:ss)	2:03:55	3:08:26
validation accuracy	95.71%	81.99%
testing accuracy	76.84%	
loss function	cross-entropy loss	cross-entropy loss (mean of start and end)
optimization algorithm	AdamW	AdamW
max len	512	512
batch size	2	2
epoch	1	2
learning rate	3e-5	3e-5

I only change the pre-trained model from *bert-base-chinese* to *chinese-macbert-base*, other settings are remained the same. The accuracy raised, but the training time only slightly increased.

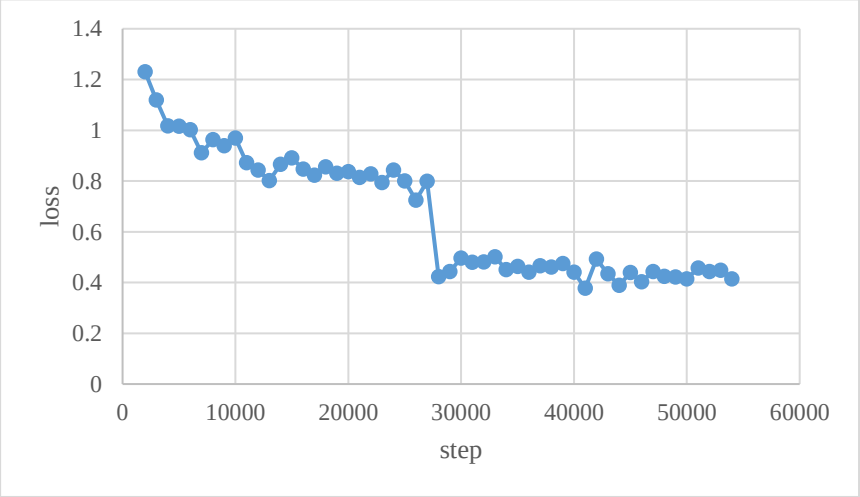
chinese-macbert-base is a variant of BERT designed to reduce the discrepancy between pretrain and finetune. It uses WWM and MLM to achieves better downstream task performance.

WWM: masking entire word instead of part.

MLM: replaces tokens with similar words instead of using [MASK] which never appears in real tasks.

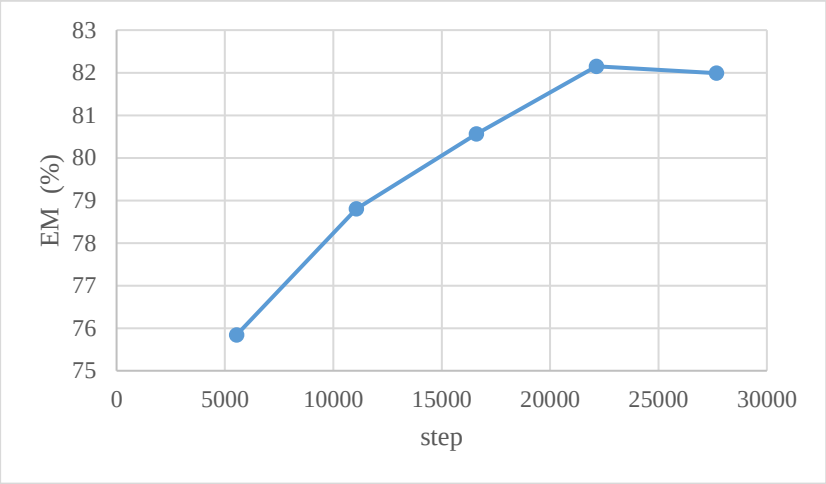
Q3: Curves (1%)

Learning curve of the loss value for training set (0.5%)



Each epoch has 27,676 steps, and I trained for a total of two epochs. You can see a significant drop in loss around step 28,000, which happens because the model starts seeing the data for the second time.

Learning curve of the Exact Match metric value for validation set (0.5%)



During training, I inserted five validation checkpoints on average, and at the fifth one the EM shows a slight decline, which indicates that the model has been sufficiently trained.

Q4: Pre-trained vs Not Pre-trained (2%)

I implemented a small Transformer model for the paragraph selection task.

Configurations:

Vocabulary size: 30,000 (constructed using BERT's Chinese tokenizer, but embeddings initialized randomly)
Embedding dimension: 128
Transformer encoder layers: 2
Attention heads: 4
Feedforward dimension: 512
Maximum sequence length: 128 tokens
Dropout rate: 0.1
Classifier: Linear

Hyper-parameters:

Optimizer: AdamW

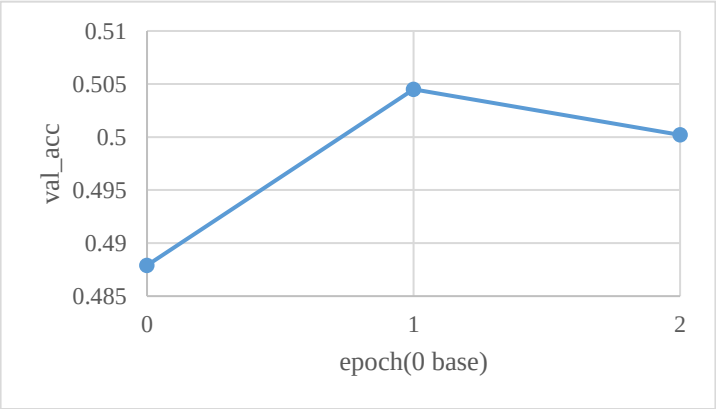
Learning rate: 3e-4

Batch size: 16

Number of epochs: 3 (val_acc falls in 3 epoches as the following figure shows)

Loss function: CrossEntropyLoss (over the 4 candidate paragraphs)

Evaluation metric: Accuracy on the validation set



The performance of this model v.s. BERT.

	this model	BERT
Training time (h:mm:ss)	0:02:17	2:03:55
Validation Accuracy	50.02%	95.21%

Q5: Bonus (2%)

Implementation:

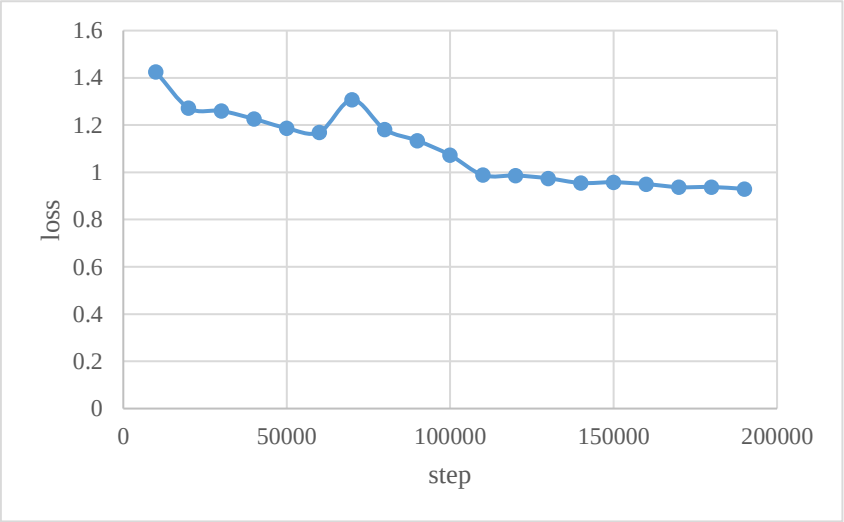
Modified from span selection code with following key changes:

1. Concatenation 4 candidate paragraphs with " " separators.
2. Calculate the starting index of each paragraph.
3. Add the correct paragraph's offset to the original answer_start to compute the new answer_start of the concatenated text.

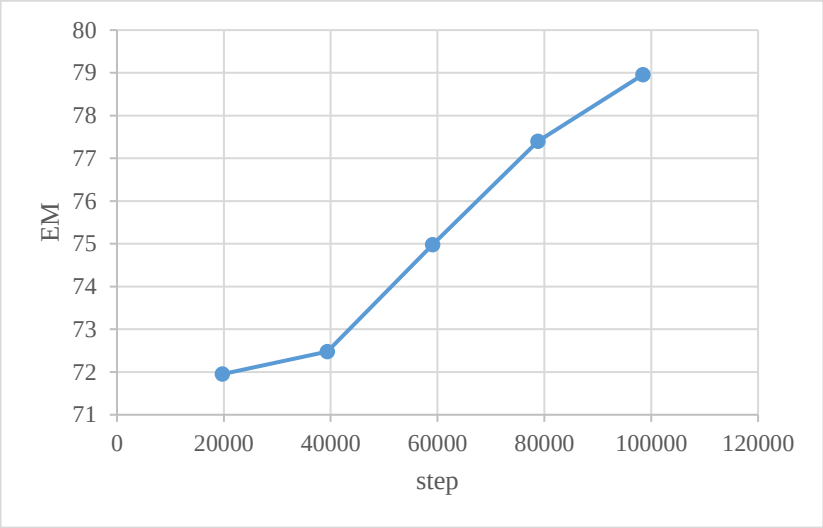
Task	Paragraph selection + Span selection
pre-trained model	bert-base-chinese
training time (h:mm:ss)	6:28:31
validation accuracy	78.96%
testing accuracy	76.37%
loss function	cross-entropy loss (mean of start and end)
optimization algorithm	AdamW
max len	512
batch size	2
epoch	2
learning rate	3e-5

The performance is surprisingly good.

Learning curve of the loss value for training set



Learning curve of the Exact Match metric value for validation set



The result may be improved by training more epochs, but I don't have that much GPU resource.

Note

All trainings are on Kaggle with 1 × P100 GPU.

References

1. ChatGPT-5

2. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT 2019.

3. Cui, Y., Che, W., Liu, T., Qin, B., Wang, S., & Hu, G. (2020). Revisiting Pre-trained Models for Chinese Natural Language Processing. Findings of ACL 2020.