CSIE5431 Applied Deep Learning – HW1

R12922129 賴翰霖

1. Data processing

Tokenizer:

Tokenizer最主要的工作就是將我們輸入的文字進行encode，變成model可以輸入的格式，如果要轉換為我們可以看得懂的文字就要進行decode。

在part 1 multiple choice的部分，在原本SWAG dataset是設定為上下文語意連接，first_sentence是上句，second_sentence是下句，而在我們的dataset中，我將tokenizer對應到的是將first_sentence設定為問題，second設定為4個候選文本，以便讓model可以從文本中擷取到問題有的關鍵字以進行對應。

而在part 2 Question Answering中，原本SQUAD dataset與我們的dataset所使用的格式非常相像，將第一個input對應到的是問題，第二個input對應到的是與問題有關的文本，也是方便model進行提取。

Answer Span:

在part 2給的sample code裡面，在model inference完之後會返回start_logits與end_logits，postprocess裡面並透過將Feature提取出來後，一個一個遍歷過後檢查合法性 (ex. start_logits 是否小於end_logits，start_logit與end_logits是否在offset_mapping的範圍內等)，並且擷取出正確的長度後使用使用offset的值對應到原來的文本後計算是該答案的機率後回傳機率最高的作為該問題的答案。

在post_process的部分其實很重要，我原本在predict answer時只使用

```
answer_start_index = outputs.start_logits.argmax()
answer_end_index = outputs.end_logits.argmax()
```

來預測答案，但是這樣就不會檢查index的合法性，導致我一開始的prediction表現都非常差。

2. Modeling with BERTs and their variants

我表現最好的Model參數如下:

	Multiple Choice	Question Answering
Pre-trained LM	chinese-macbert-base	chinese-macbert-large
max_seq_length	512	
per_device_train_batch_size	2	
gradient_accumulation_steps	4	
num_train_epochs	5	10
learning_rate	1e-5	
Loss function	CrossEntropy Loss	
Optimization algorithm	adamW	
Running Time on RTX4070	2.5hr	4.5hr
Eval Accuracy	0.97	0.85
Overall Performance (Public)	0.80470	

而我另一個嘗試助教給的參數所train的Model參數如下:

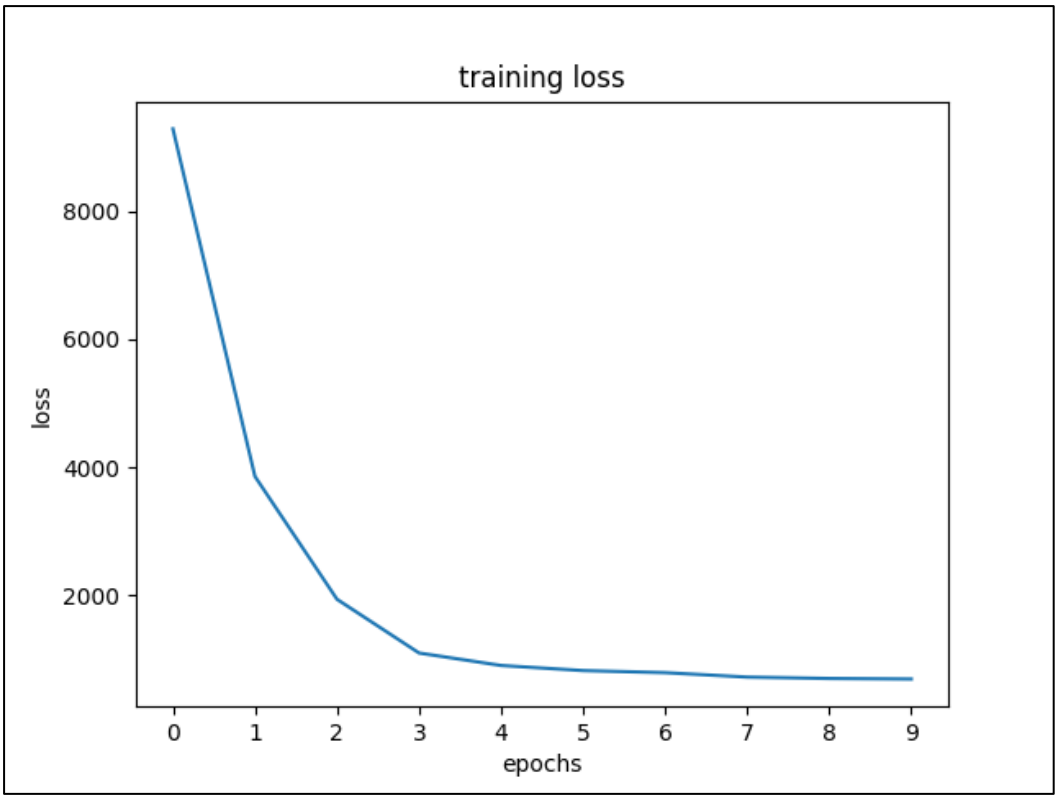
	Multiple Choice	Question Answering
Pre-trained LM	Bert-base-chinese	Bert-base-chinese
max_seq_length	512	
per_device_train_batch_size	1	
gradient_accumulation_steps	2	
num_train_epochs	3	3
learning_rate	3e-5	
Loss function	CrossEntropy Loss	
Optimization algorithm	adamW	
Running Time on RTX4070	2.5hr	2.5hr
Eval Accuracy	0.94	0.76
Overall Performance (Public)	0.69258	

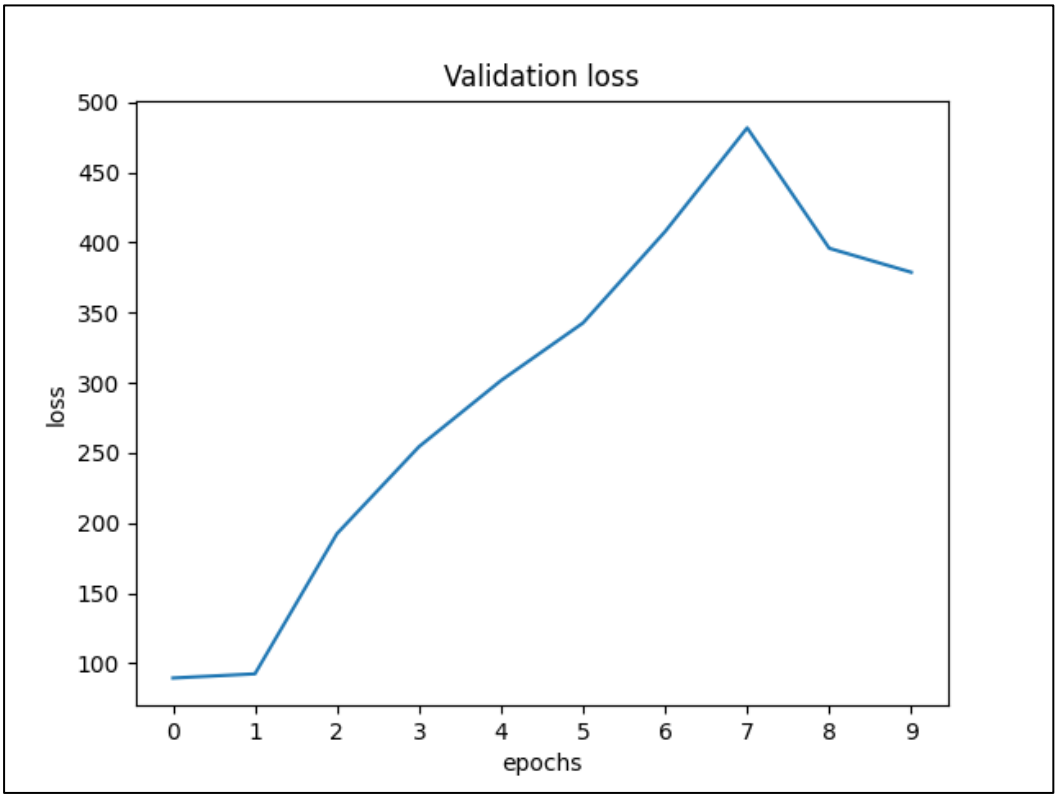
在這兩個不同Pre-trained LM中差異會在架構方面：

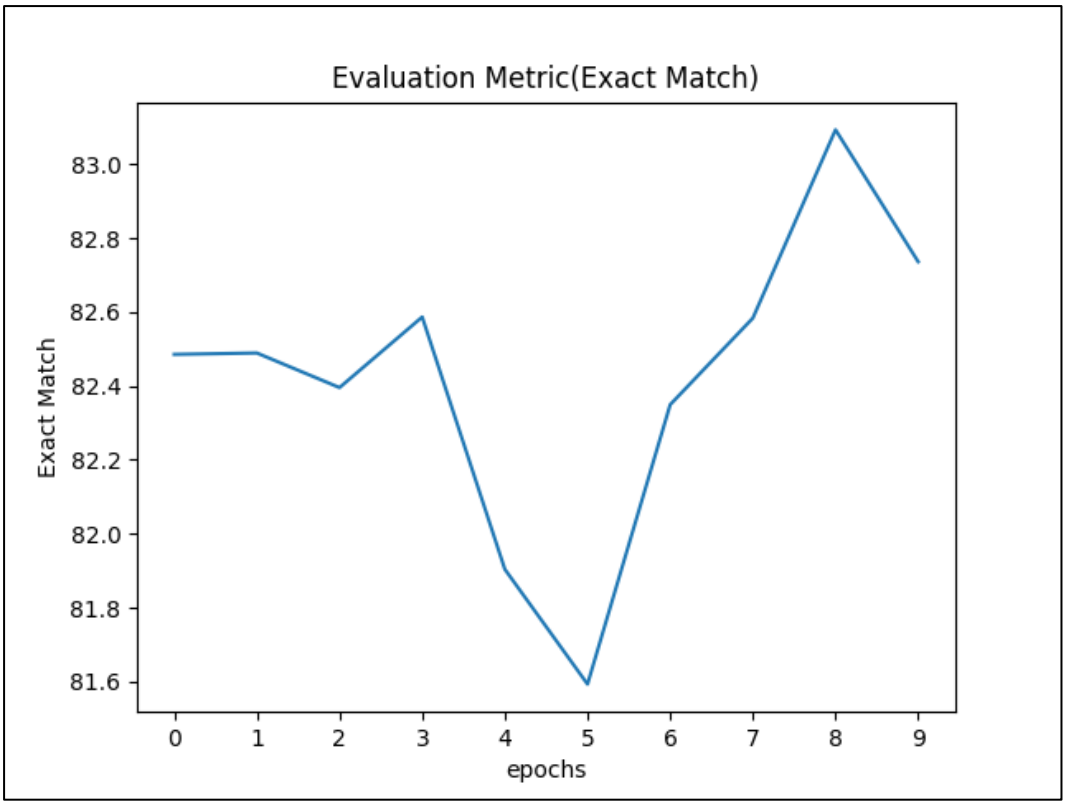
Bert-base-chinese中，中文是以字為單位進行切分，沒有考慮一般NLP裡面的chinese word segment (CWS)。

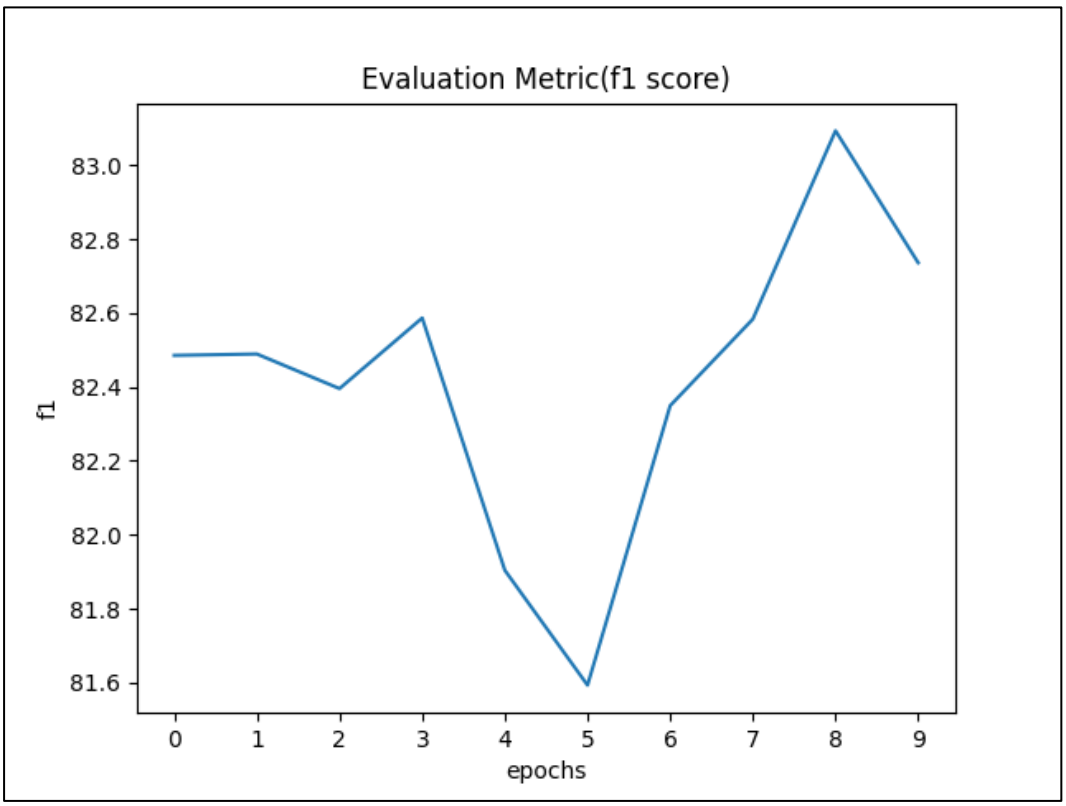
其他所使用的 Loss function 等都相同的情況，bert-base-chinese 的表現比 macbert 差非常多，並且答案常常出現[CLS], 空白, 或是一大段話的情況。

3. Curves


training loss
loss
8000
6000
4000
2000
0
1
2
3
4
5
6
7
8
9
epochs


Validation loss
loss
500
450
400
350
300
250
200
150
100
0
1
2
3
4
5
6
7
8
9
epochs


Evaluation Metric(Exact Match)
Exact Match
83.0
82.8
82.6
82.4
82.2
82.0
81.8
81.6
0
1
2
3
4
5
6
7
8
9
epochs


Evaluation Metric(f1 score)
f1
83.0
82.8
82.6
82.4
82.2
82.0
81.8
81.6
0
1
2
3
4
5
6
7
8
9
epochs

4. Pre-trained vs Not Pre-trained

以Pre-trained與Not Pretrained的模型比較，我使用的是我表現最好的模型進行對比，並且更改的是第二部分去進行not pre-trained

My model (Pre-trained) :

Pre-trained	Multiple Choice	Question Answering
Pre-trained LM	chinese-macbert-base	chinese-macbert-large
max_seq_length	512	
per_device_train_batch_size	2	
gradient_accumulation_steps	4	
num_train_epochs	5	10
learning_rate	1e-5	
Loss function	CrossEntropy Loss	
Optimization algorithm	adamW	
Running Time on RTX4070	2.5hr	4.5hr
Eval Accuracy	0.97	0.85
Overall Performance (Public)	0.80470	

Not Pre-trained：

(model-type: bert, 所使用的tokenizer為我表現最好的model的tokenizer)

Not Pre-trained	Multiple Choice	Question Answering
Pre-trained LM	chinese-macbert-base	-
max_seq_length	512	
per_device_train_batch_size	2	
gradient_accumulation_steps	4	
num_train_epochs	5	10
learning_rate	1e-5	
Loss function	CrossEntropy Loss	
Optimization algorithm	adamW	
Running Time on RTX4070	2.5hr	2.5hr
Eval Accuracy	0.97	0.07
Overall Performance (Public)	0.06781	

在訓練時，我發現並沒有助教於簡報中提及的參數過多跑不動的情況，並且可能因為並沒有加載pre-trained LM的關係讓他的訓練速度比我有載pre-trained還要快，進去查看他所預測的答案，都是無法擷取出正確的片段，而非可能多前後幾個字導致答案錯誤，看來pre-trained LM由於看過的文本數量夠多可以非常高的提升performance

5. References

- https://chat.openai.com/
- ymcui/MacBERT: Revisiting Pre-trained Models for Chinese Natural Language Processing
- ymcui/Chinese-BERT-wwm: Pre-Training with Whole Word Masking for Chinese BERT
- Multiple choice (huggingface.co)
- Question answering (huggingface.co)
- matplotlib 关于使用 MultipleLocator 自定义刻度间隔_CSDN