NLP_HW3

NTU b10705013 陳彥廷

Questions

Q1: Which (pre-trained) model do you use? Why to choose the model?

A1: 我最後選擇的模型是 **microsoft/deberta-v3-base**，我嘗試過了另外兩個模型 **google-bert/bert-base-uncased** 跟 **FacebookAI/roberta-base**，但是都沒有這個模型的效果好，這就是我選擇這個模型的原因。

Q2: Compared with models trained separately on each of the sub-task, does multi-output learning improve the performance?

A2:

我使用了兩種模型分別跑過一次，參數如下 ：

lr = 3e-5

epochs = 3

train_batch_size = 16

validation_batch_size = 16

pretrain_model = microsoft/deberta-v3-base

weight_decay=0.01

multi-output 的結果如下 ：

Spearman: 0.8823131322860718

Accuracy: 0.9163791537284851

F1 Score: 0.9058225154876709

而分開訓練的結果如下 ：

Spearman: 0.8902292251586914

Accuracy: 0.8920000195503235

F1 Score: 0.885209321975708

從結果可知，分開的訓練跟一起訓練的結果互有優勢，並沒有哪個比較有優勢

Q3: Why does your model fail to correctly predict some data points? Please provide an error analysis.

A3:

針對預測錯誤的部分，我嘗試在預測階段將 loss 相對較高的 batch 印出來並人工觀察這些預測錯誤的部分。我發現大部分的情況是有一些較為罕見的動詞沒有出現在訓練資料集中，如 shaved 等等，又或者是出現過多次導致語意可能出現混淆，造成模型的結果沒辦法達到 100% 的正確。

Q4: How do you improve your model performance?

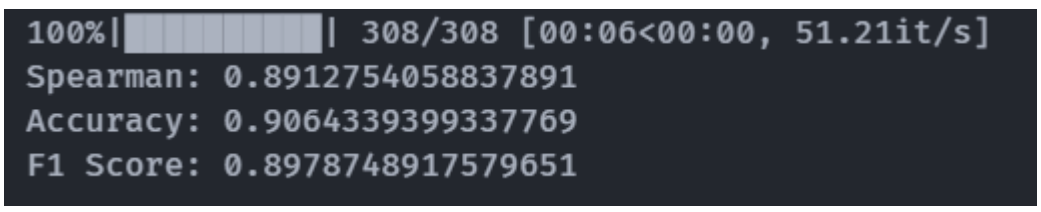
A4:

我的最後結果如下 ：

Spearman: 0.8912754058837891

Accuracy: 0.9064339399337769

F1 Score: 0.8978748917579651



我認為第一步我們應該選擇模型，經過我的測試 **google-bert/bert-base-uncased** 在不調參數的情況下約為 0.79 左右，**FacebookAI/roberta-base** 在不調參數的情況下約可以到 0.83 左右，而 **microsoft/deberta-v3-base** 在不調參數的情況下就能直接達到 0.86，分類任務也可以接近 0.9。

選擇完模型之後要做的是針對模型來調整參數，這裡面我認為影響比較大的是 batch size 跟 learning rate，可以多嘗試一些參數進行測試。

Q5: DeBERTa 簡介

A5:

microsoft/deberta-v3-base 是微軟推出的 DeBERTa (Decoding-enhanced BERT with disentangled attention) 系列中的一個預訓練語言模型。這個模型特別設計來解決 BERT 和 RoBERTa 在處理詞彙上下文關係中的局限，並且在許多基準測試中有很好的表現。

模型的主要特色大致是以下兩個 Disentangled Attention Mechanism 跟 Enhanced Mask Decoder。

Disentangled Attention Mechanism 可以將內容和位置的 embedding 分開處理使模型可以更好的捕捉 context，而 Enhanced Mask Decoder 則可以提供比傳統 BERT 的 MLM 更好的效能。

Q6: 針對模型細部優化做的嘗試

A6:

除了前面所提到的最重要選擇模型跟調整重要參數外，我還針對了模型做了一些嘗試，如下所示 ：

Optimizer : 測試 Adam 跟 AdamW，發現 AdamW 效能較佳

Gradient Clipping: 測試之後發現影響很小

Loss Weights: 一開始是直接將兩個 loss 值相加，後來發現 Spearman 較難

通過 baseline，因此相應的調整了權重

Learning Rate Scheduler: 我使用了 transformers 中的 get_linear_schedule_with_warmup 來動態調整學習率，對結果有略為的幫助

我最後的參數設定如下

lr = 3e-5

epochs = 3

train_batch_size = 16

validation_batch_size = 16

pretrain_model = microsoft/deberta-v3-base

alpha = 0.75

beta = 0.25

optimizer = AdamW, weight_decay=0.01

scheduler= get_linear_schedule_with_warmup (num_warmup_steps=0.1*step)

Q7: Bonus 的結果

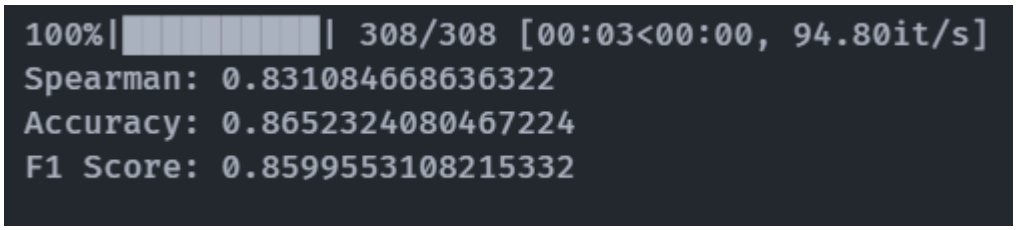
A7:

在 Bonus 中規定只能使用 bert-base 的模型，因此我採用了預設的 **google-bert/bert-base-uncased** 來進行參數優化，結果如下

Spearman: 0.831084668636322

Accuracy: 0.8652324080467224

F1 Score: 0.8599553108215332



與前面的參數不同的是我調整了 epoch 數量為 4，並略為修改了 alpha 跟 beta 的值為 0.7 跟 0.3 來完成。

Settings

All code run in win11、CPU i5-12400、GPU 4070s、Python 3.10.11

References

Copilot in all code

Chatgpt in report