**Report：NLP\_HW3 成功大學\_N96134645\_許宸華**

|  |  |
| --- | --- |
| **Running environment：Colab** | **Python version：Colab** |
| **GPU(s) used：Tesla T4** | **CUDA 版本: 12.2** |

**以下的chatgpt皆為chatgpt-4o**

**● Which (pre-trained) model do you use? Why to choose the model?(5%)**

我最終使用的model是bert-base-cased，原先是使用bert-base-uncased的model來進行訓練，但Spearman Corr始終維持在0.79左右，無法突破至0.8以上。當看到slido上助教說可以使用bert-base-cased的模型後便使用它重新訓練看看。

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

在經過5個epoch之後，Spearman Corr為0.8455, Accuracy為 0.8760, F1 Score為0.8731，很明顯的三項數值都比使用bert-base-uncased的模型來的還要好。

bert-base-cased與bert-base-uncased的差異在於是否保留大小寫的資訊，由於此次任務是要判斷語句的之間的關係，因此在訓練上大小寫的保留有助於模型更加理解語句，再加上data本身就是有大小寫的文本，用bert-base-cased模型會更加適配。

**● Compared with models trained separately on each of the sub-task,  
does multi-output learning improve the performance? (8%)**

在這次的訓練中兩個label分別是relatedness\_score 與entailment\_judgment，前者是要分析相似性，後者是分析語意。兩者雖然是在做不同項目的評分，但其任務是有相關性的，可以互相補充來提高模型訓練的成效。[此外multi-output learning可以起到正則化的作用，由於同時訓練兩個label可以避免模型過度擬合一個任務，防止overfitting。(由chatgpt補充)]， **● Why does your model fail to correctly predict some data points?  
Please provide an error analysis. (8%)**

我使用chatgpt幫忙生成一個confusion matrix來幫助我判斷Entailment\_judgment哪些地方判斷錯誤的比較多。一張含有 文字, 螢幕擷取畫面, 圖表, 鮮豔 的圖片

自動產生的描述

根據生成的confusion matrix，其中錯誤最多的是Neutral被誤認為Entailment（45次），次多的是Entailment被誤認為Neutral（11次），可以判斷說模型在判斷這兩者上相對不擅長，但由於在test中這兩者的data量也是比較多的，所以錯誤較多也是合理的，所以我也計算了一下錯誤率為0.23，在比較後也是比其他項目來的高。

|  |  |  |  |
| --- | --- | --- | --- |
|  | **正確判斷** | **錯誤判斷** | **錯誤率** |
| Neutral | 230 | 52 | 0.23 |
| Entailment | 133 | 11 | 0.08 |
| Contradiction | 63 | 11 | 0.16 |

**● How do you improve your model performance? (9%)**

1. 首先是針對參數的調整**，**在learning rate上我覺得loss值的變化蠻穩定的下降，下降的比率也還不錯，經過一些測試後也覺得原先的學習率比較好。

一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

自動產生的描述

1. 再來測試了不同batchsize的影響，總共試了4種不同的batchsize：4、8、16、32，在epoch為5的情況下進行實驗，結果在各項數據上全部都還是batchsize = 8時表現的最好。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Batchsize = | 4 | 8 | 16 | 32 |
| Spearman Corr | 0.8064 | 0.8231 | 0.7968 | 0.8118 |
| Accuracy | 0.8380 | 0.8540 | 0.8200 | 0.8400 |
| F1 Score | 0.8391 | 0.8564 | 0.8246 | 0.8386 |

1. 在epoch的調整上增加到5個epoch，增加epoch來讓模型訓練得更徹底。

一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

自動產生的描述

1. 增加中間層及dropout防止過擬合

將模型結構改變，增加hidden layer來幫助模型訓練，並加上dropout層來防止overfitting。(chatgpt輔助生成)

**一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 字型, 行 的圖片

自動產生的描述**

表現相比只有layer層還要穩定。

**一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述**

**● ... Anything that can strengthen your report. (10%)**

在loss值的設定上我使用了動態的調整(chatgpt推薦使用)，在multi-output learning中若將兩個任務的loss值直接相加計算可能會導致模型專注在loss值比較大的任務上，而忽略了loss值較小的。因此在訓練時設定了不同的weight來影響loss值的加總。一張含有 文字, 字型, 行, 螢幕擷取畫面 的圖片

自動產生的描述

loss值針對任務類型分別使用MSELoss(線性任務)跟CrossEntropyLoss(分類任務) 在run每一個epoch時我的初始loss值的weight設定是不一樣的，設置initial weight = 0.8，並且兩個任務間的loss值權重會因為epoch數的增加而改變。前兩個epoch針對relatedness\_score進行訓練，第三個平等，最後兩個針對entailment\_judgment進行訓練。

一張含有 文字, 字型, 行, 螢幕擷取畫面 的圖片

自動產生的描述

(一開始relatedness\_weight = 0.7、entailment\_weight = 0.3；

第5個epoch時relatedness\_weight = 0.3、entailment\_weight = 0.7)

在這樣的設定下模型可以很好的學習到兩個任務，並且也不會過度偏頗哪一個。