**2024 NTHU Natural Language Processing Assignment 3 : Multi-output learning NTU** r12631055

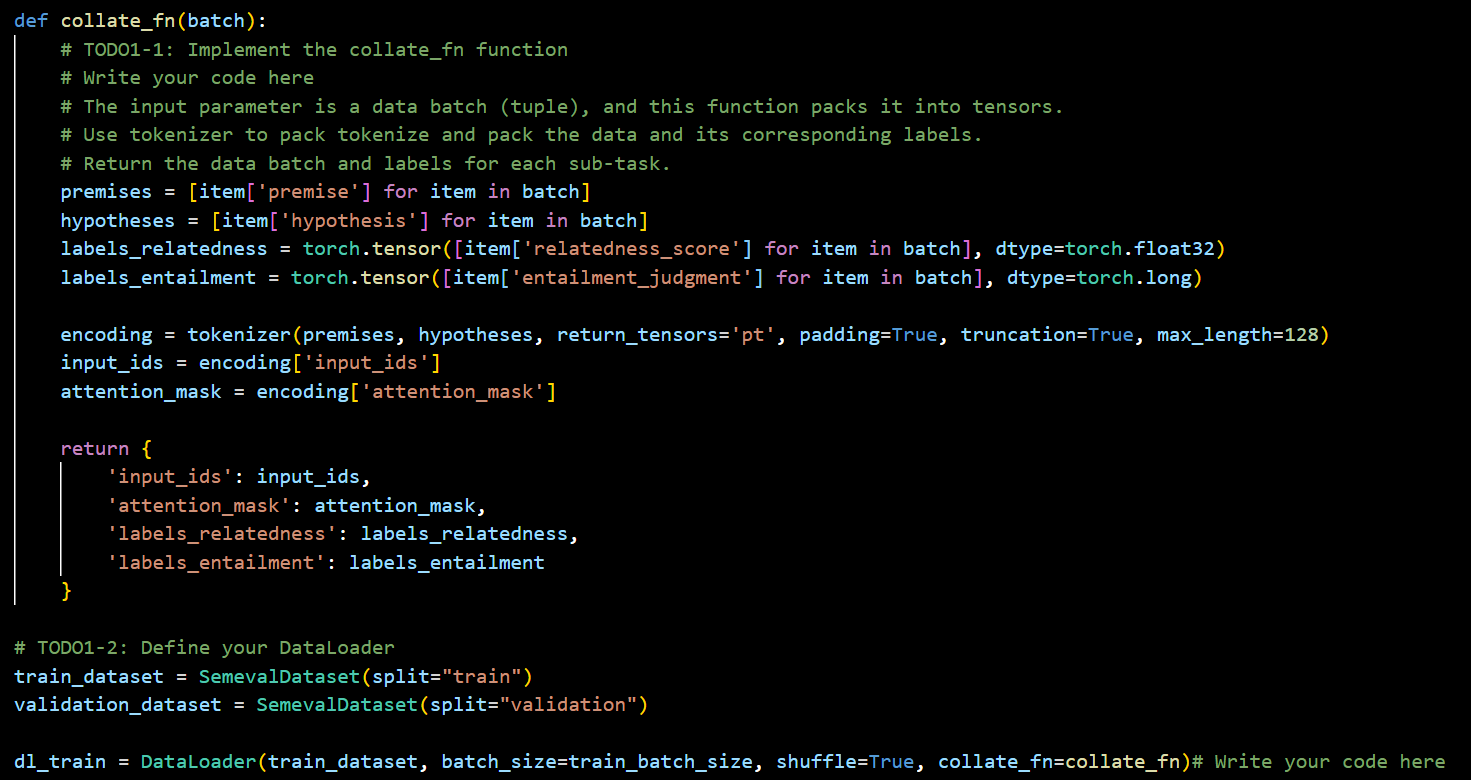
Running environment : CPU：AMD Ryzen 5955WX

GPU：NVIDIA GeForce RTX 4090

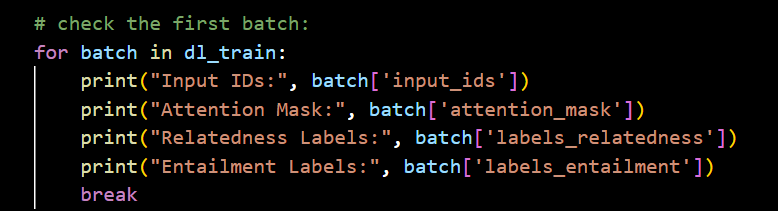
System：Ubuntu 22.04.3 LTS

Python version : Python 3.10.12

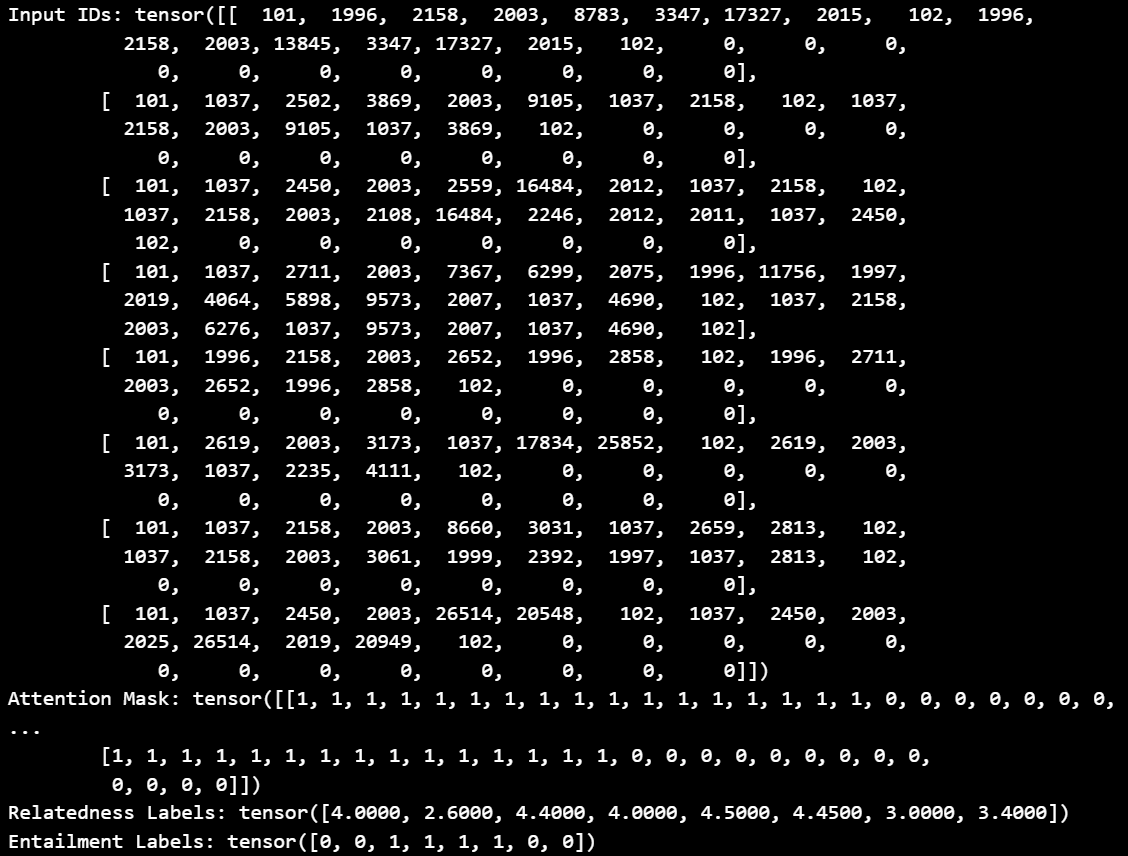
**TODO1**



分成兩步進行：首先根據raw data格式以及要求的輸出tensors格式來設計collate\_fn：主要是將’premise’和’hypothesis’逐項讀取，接著是兩個label的部分，分別將其也轉成torch.tensor的格式，然後用之前已經寫好的tokenizer把’premise’和’hypothesis’分別包裝起來。第二步就是透過上面定義好的SemevalDataset放到Dataloader中。



最後測試一下是否成功；最終結果如下，看起來都有tensor格式。



**TODO2**

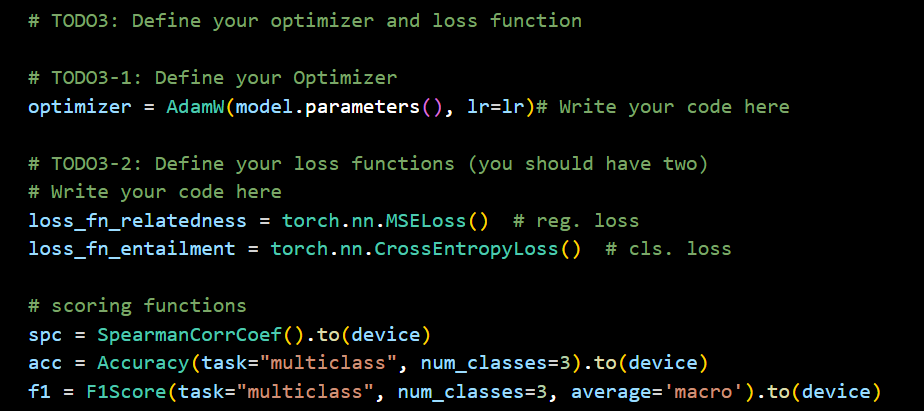
接下來要進行Multi-output Learning模型的建構，本次作業規定必須是BERT-base的模型才行，考量到題目設計的意涵，在這裡只用了簡單的google-bert pretrained model，接下來後面分別接兩個layer，一個是Regression，另一個則是3類的classification：



Forward的部分如註解所示使用BERT，然後接到pooling層，最後再分別到各自的output層計算結果。

**TODO3**

再來要進行optimizer和loss function的定義：



Optimizer的部分選用AdamW，它不但有Adam的自適應學習率和 Weight Decay正則化，因此非常適合用於訓練BERT-base的模型。

而在Loss計算的部分，按照課堂上所教的，在regression的任務就直接計算MSE；而在classificaion的任務則直接計算cross entropy。

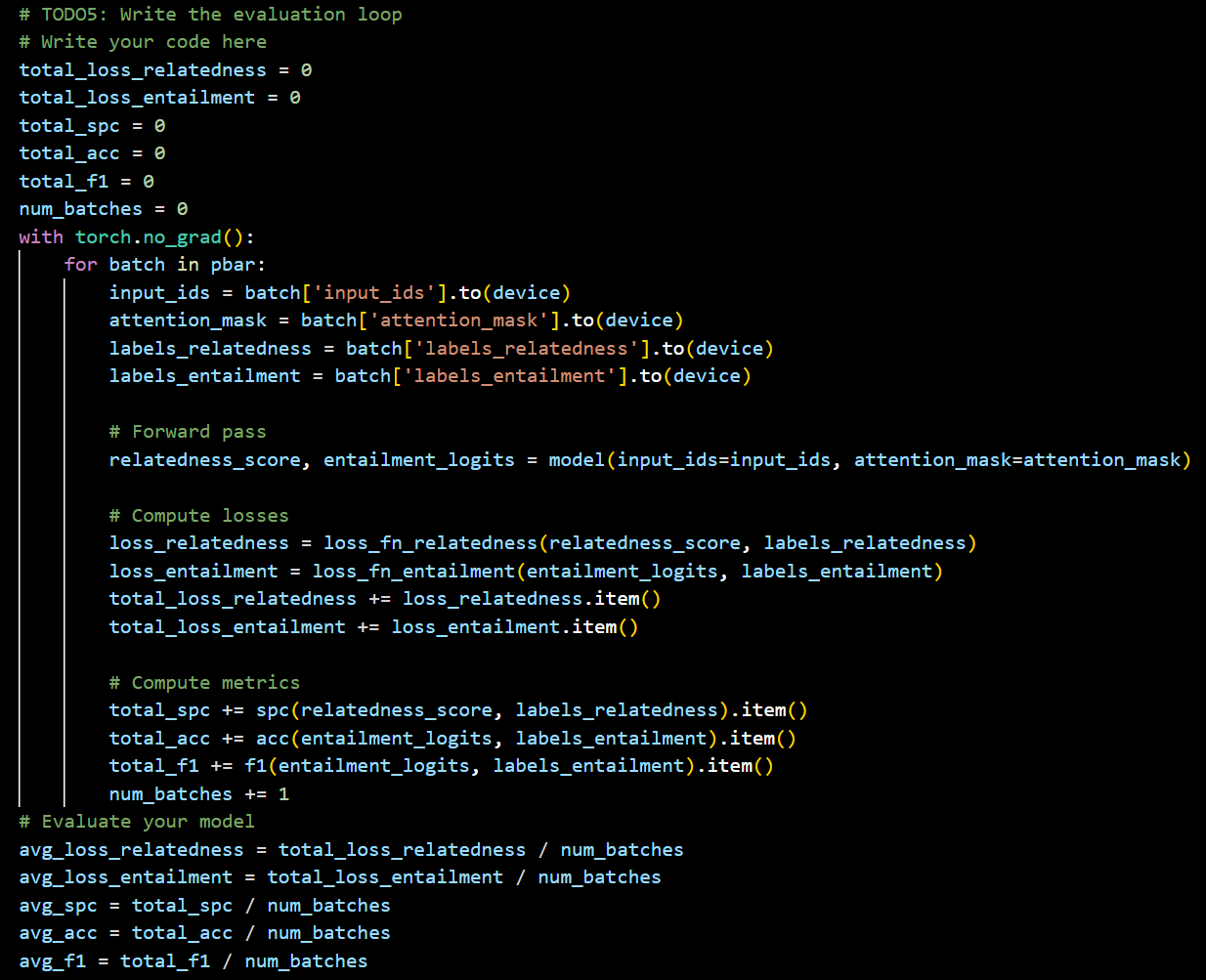
**TODO4**

接下來要準備好模型的訓練部分，首先註解提示先清除梯度，接著批次的倒入資料，觀察TODO1的return銜接到output，因為有使用到GPU來做運算，所以都得補上.to(device)。接下來就是繼續按照註解的提示繼續完成forward pass和計算loss，最後再完成反向傳播和最佳化，過程如下：

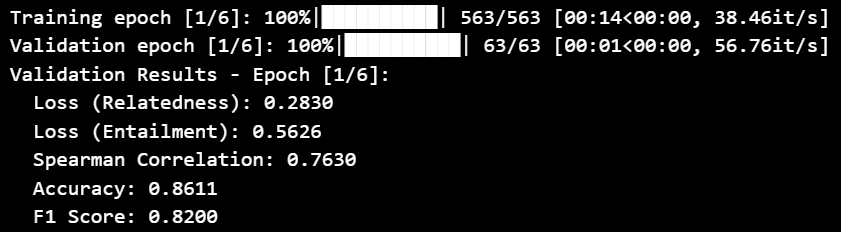


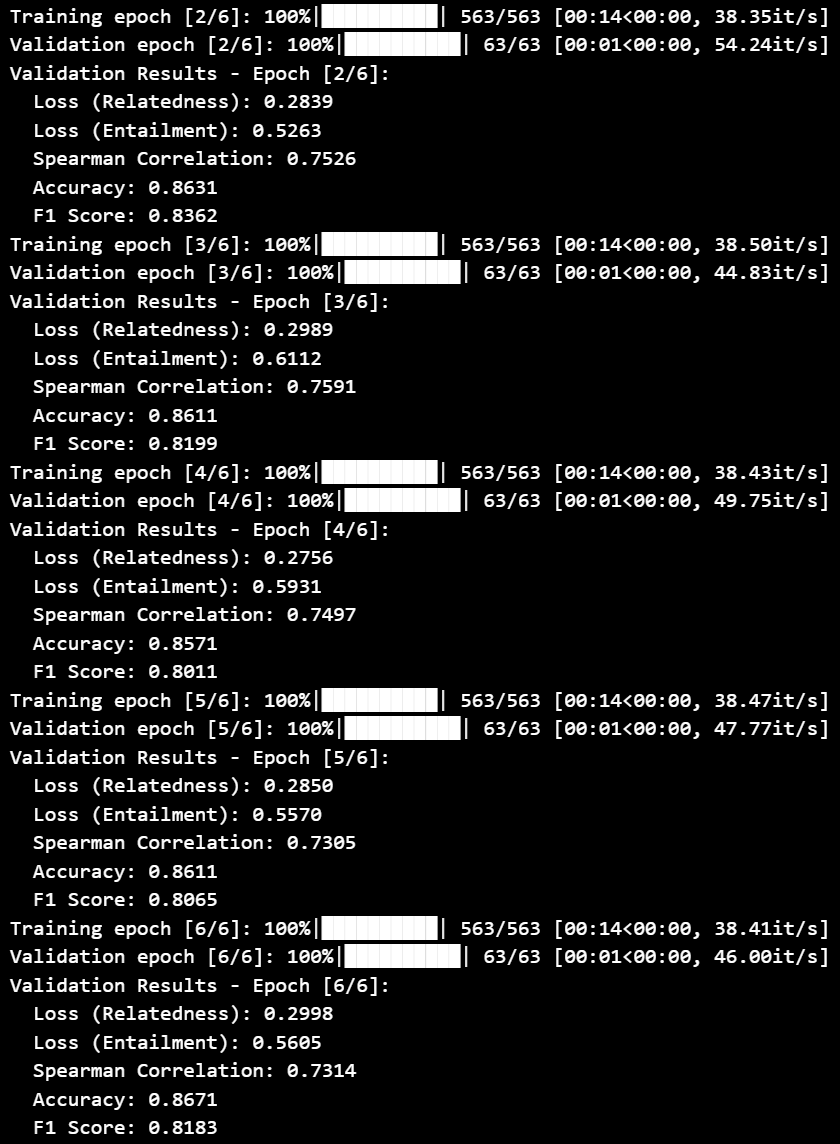
**TODO5**

最後來到評估的階段，剛才已經先完成訓練的部分，評估的主要內容其實和訓練差不多，基本上就是將評估資料集的答案和模型訓練的結果進行比較，針對relatedness\_score的部份我們使用Spearman correlation coefficients；至於entailment\_judgement則計算accuracy 和F1 score，最終程式碼如下：

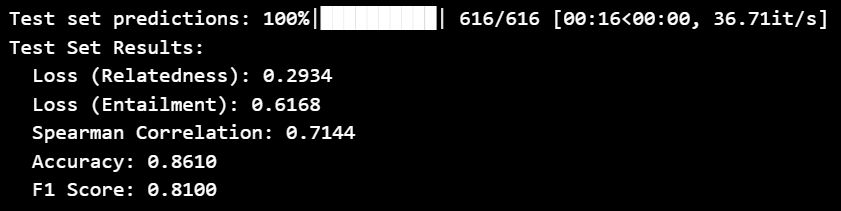


而evaluation成果如下：



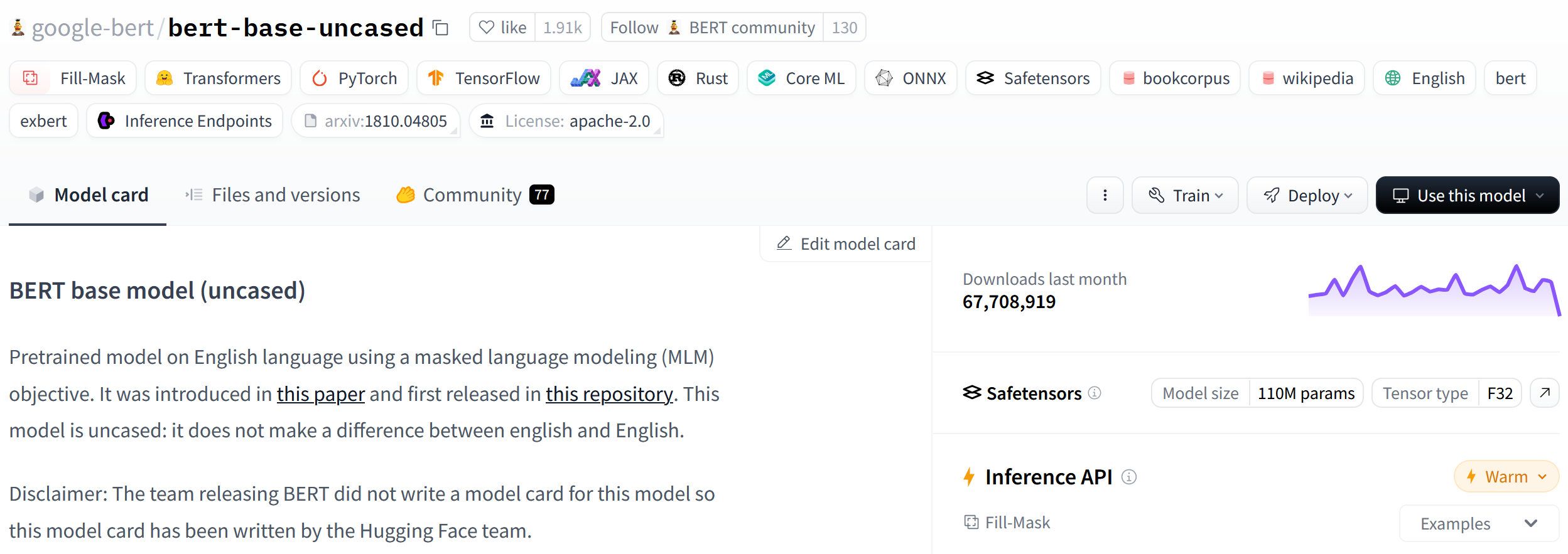


接續來到testing的部分，基本上和evaluation差不多，成果如下：



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

因為根據自己對本次作業目標的認知，主要是在於如何針對一個多任務目標來建立一個預訓練模型來進行訓練，因此模型本身並不需要有多先進或多複雜，所以在這份作業訓練中主要只使用了最基礎的google-bert/bert-base-uncased：



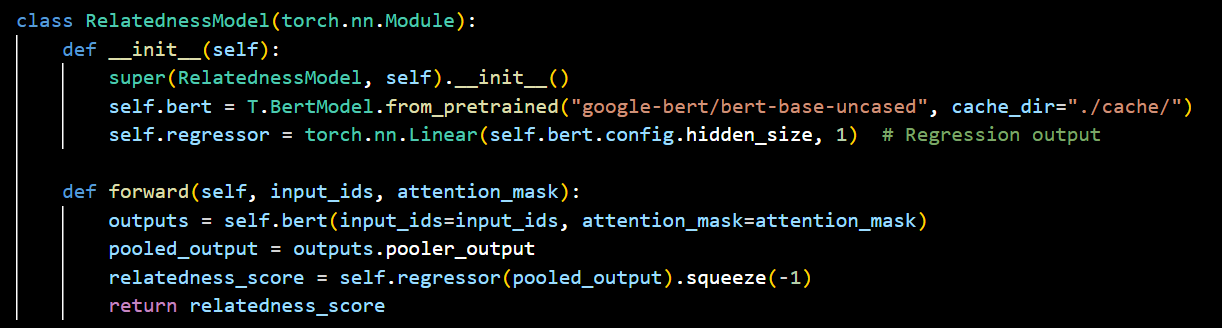
而光是這種基礎的預訓練模型，單純就結果而論，就已經有不俗的表現，因此我判斷以此作為標準之後再進行finetune即可。

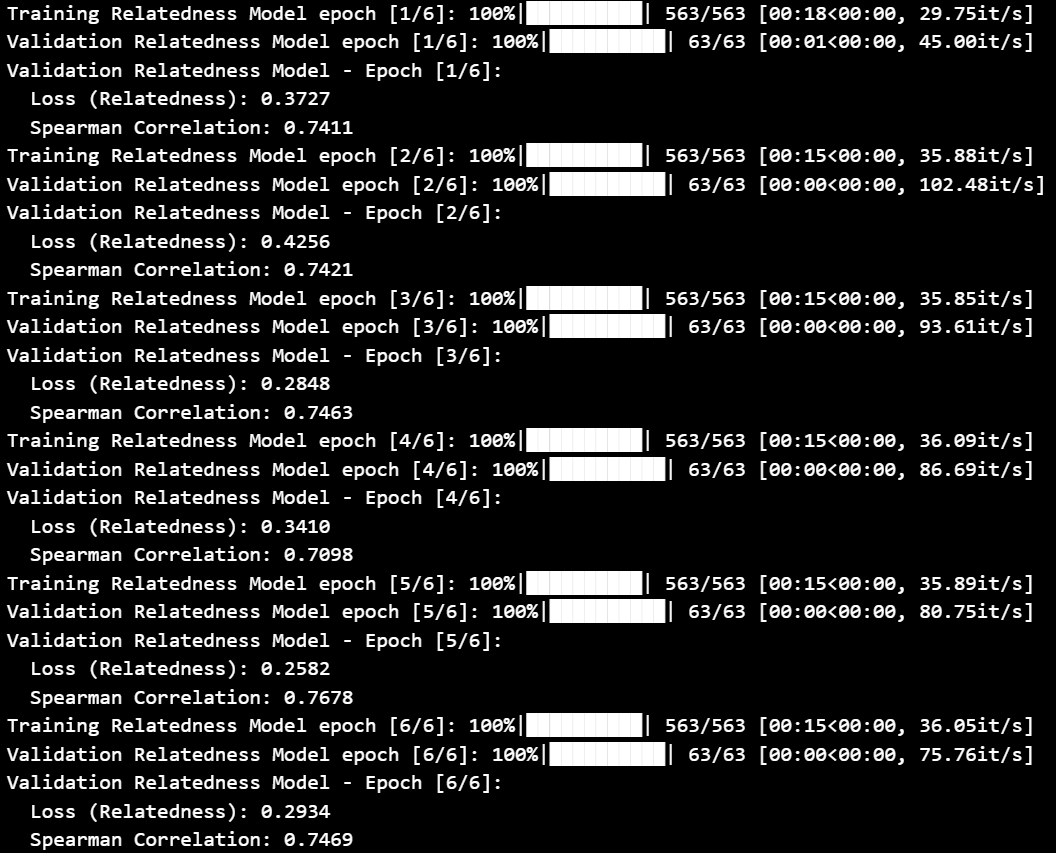
● Compared with models trained separately on each of the sub-task,

does multi-output learning improve the performance?

針對這個問題我分別作了單獨只訓練regression任務的模型和單獨只訓練classification的模型來看結果如何，看看一起訓練是否真的能夠對成果表現有所提升，我們就evaluation的部分來進行觀察：

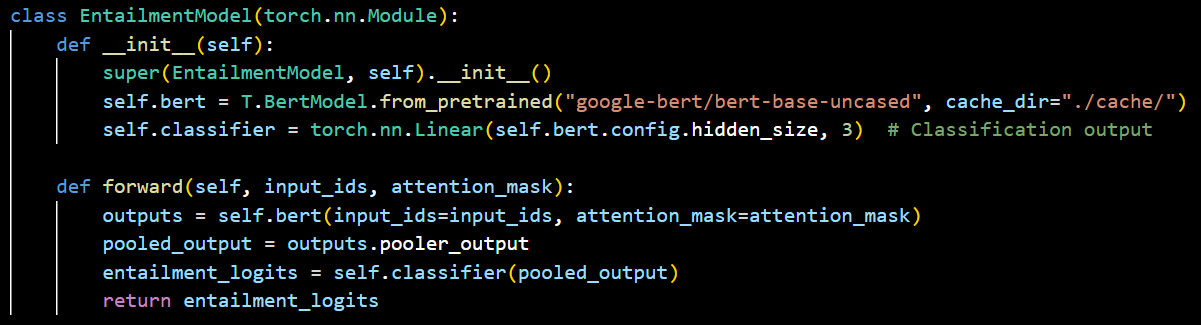
首先是針對Regression的部分：



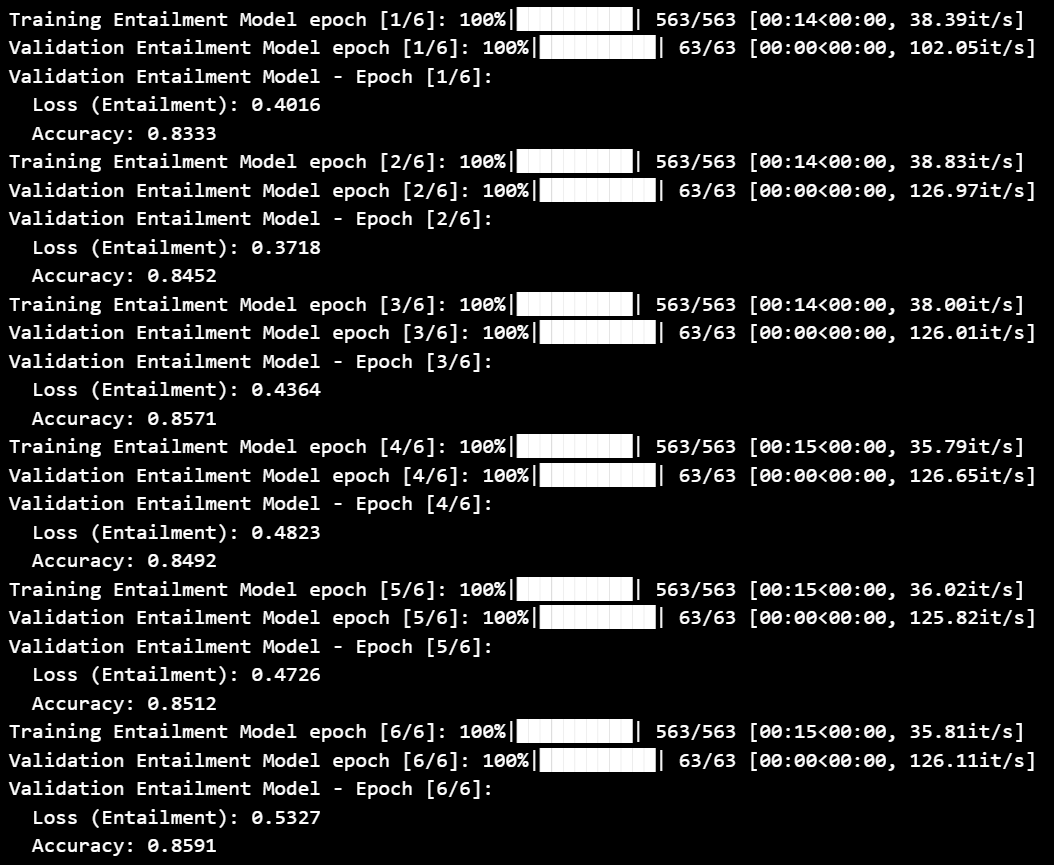


在六個epoch中，表現最好的是Loss:0.2582、Spearman:0.7678；而表現最差的是Loss:0.3410、Spearman:0.7098。

再來是Classification的部分：



以下則是訓練成果：



在六個epoch中，表現最好的是Loss:0.5327、Spearman:0.8591；而表現最差的是Loss:0.4016、Spearman:0.8333。

總結其結果：

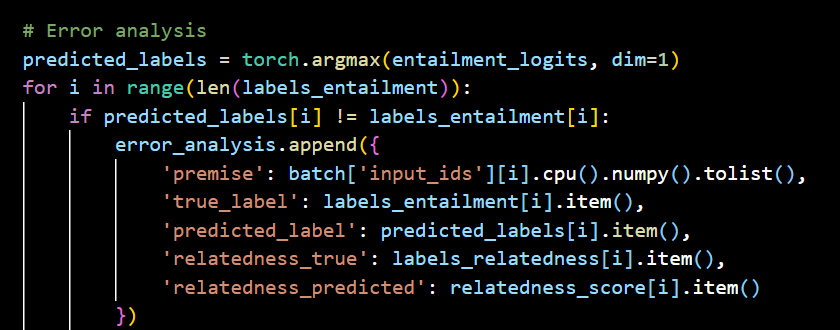
Multi-output learning在Entailment任務中的準確率和F1 Score都稍有提升，但在Relatedness預測的Spearman Correlation上略有下降。

就理論上而言Multi-output learning的優勢在於共享表示，因為它可以同時學習兩個任務，從而在某些方面提高模型對語義的捕捉，但也可能因為需要同時優化兩個loss，導致每個子任務的表現不一定能達到單獨訓練的最佳效果。而仔細觀察在過程當中，單獨分開來的訓練時的loss的起伏相當大，而在Multi-output時的loss則相對平穩，這可能表明Multi-output可以在不顯著降低性能的情況下，同時處理多個任務，甚至有機會在某些子任務上有更多提升，但它在每個子任務上的表現可能不如專門針對單一任務訓練的模型。

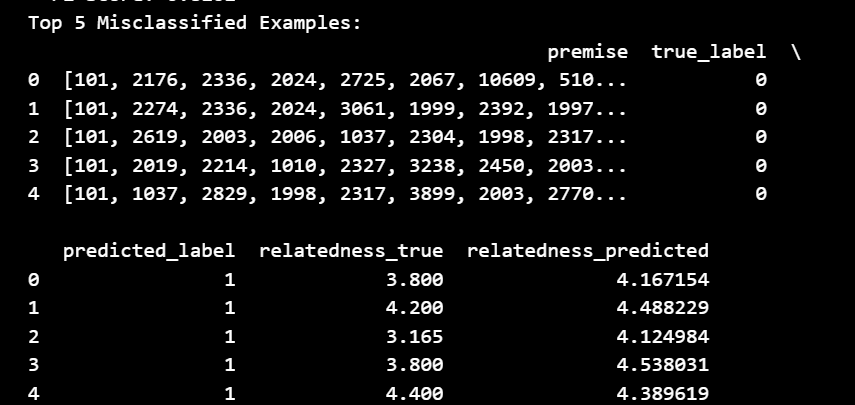
● Why does your model fail to correctly predict some data points?

Please provide an error analysis.

同樣針對這個問題，由於regression的表現較好，即使是fail也都算很接近ground truth，因此在這邊主要針對classification來探討，在其中加入這段程式碼進而將fail的部分給印出來：



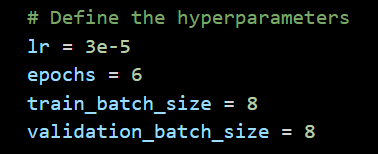
我們擷取片段成果來看：



結果並未按照順序排列，觀察發現model經常將label為0(neutral)的判斷為1(entailment)，也許有些statement確實有些模糊導致某些句子對之間的關係可能讓模型難以做出準確的判斷。

● How do you improve your model performance?

在這次任務中由於pretrained model本身的強大，我並沒有進行太過於複雜的調整，只有簡單經過訓練準確率的試誤來調整訓練參數的部分：例如將epochs設定為6，或者上下調整learing rate的部分。



如果要進行更進一步的調整，可以針對像是選擇更強大的預訓練模型，引入一些避免overfitting的方法例如：Earlystop、Dropout或L2等等讓模型具有更好的泛化能力。

● etc

本次在載入datasets的時候一直預到error，正考慮要手動下載data的時候突然想起作業指示中建議要將datasets的版本設定為2.21.0，因此想到有可能是不同版本的datasets的不同檔案路徑導致錯誤，因此將datasets版本改為2.21.0後才總算修正這個問題，在程式碼的部分，借鑒了很多上一次作業的內容，尤其是在collate\_fn的部分和定義模型框架內容以及訓練流程的部分，相較於上次作業的棘手有倒吃甘蔗的感覺。對於本次作業整體感覺循序漸進，獲益良多。