Introduction to Artificial Intelligence HW3

Sentiment Analysis

112511188 甯宇誠

1. Introduction / Objective

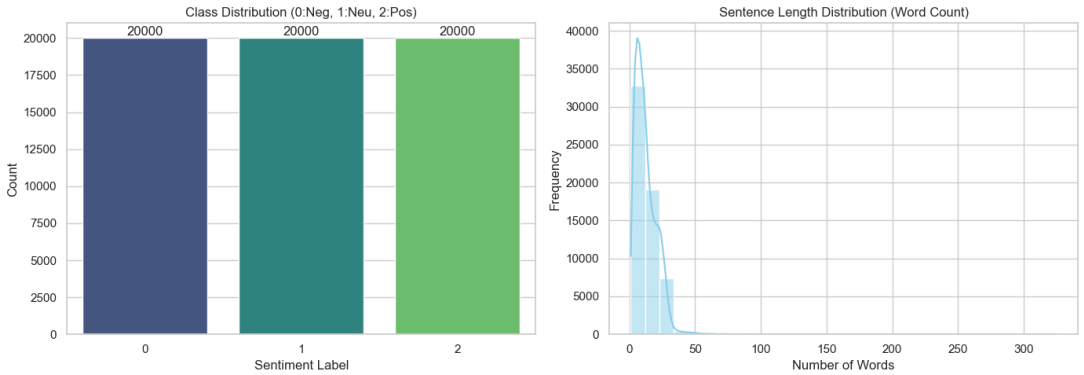
Objective:
本次作業的目標是利用深度學習模型，替社群平台上的貼文做三分類的情緒判斷（Positive/Neutral/Negative）。這類短句、語氣多變、寫法不固定的文本非常常見，也是日常應用中最容易遇到的 NLP 問題之一。

在模型選擇上，我使用 DeBERTa-v3-base 作為主要架構。這個模型在處理否定詞、口語語氣、句子細微差別時的表現不錯，特別適合應付社群語料中常見的語意反轉與模糊語氣。訓練流程的設計則是以穩定、簡潔為優先：批次資料採用固定長度 padding、計算使用 mixed precision、學習率使用 linear warmup、分類頭則維持精簡的兩層 MLP。這些設定都是希望在單 GPU 的環境下，也能讓模型保持穩定收斂，同時方便觀察訓練過程並做超參數調整。

後續章節會分別介紹資料處理方式、模型架構、訓練流程與最終結果，並說明每個步驟對模型效果的影響。

2. Implementation Details

Data Preprocessing :
為了確保模型能有效學習，必須先對數據進行資料分析，如類別是否平衡、句子的長度分佈等等，以下是我將資料視覺化後得到的結果：



可以得知資料集內類別完美平衡，所以不需要使用 Class Weights 或 Focal Loss

等處理不平衡的技巧，且模型不會傾向學習某個類別。

至於長度方面，絕大多數的句子長度落在 5 ~ 40 個單字（Words）之間，超過 50 個字的句子非常稀少，超過 100 字就幾乎沒有了。故句子的長 max_len 使用 128 就非常安全，幾乎沒有任何資訊會被 Truncated。確定關於資料集的基本資料後，就可以進行資料預處理：

1. 標籤標準化 (Label Standardization)
為了方便模型訓練，我先將標籤統一轉成整數編號 0, 1, 2，避免 dataset 的 label 採用字串型態的"0","1","2" 而導致有型別不一致的問題。

2. 靜態填充 (Fixed-Length Padding)
雖然動態填充在理論上較省算力，但為了讓最終單模型的行為更穩定、也避免動態 collator 可能帶來的 batch shape 差異，我在最終版本中改採固定長度 padding（128）。經過實測，這樣的設定對單一 seed 的收斂較穩，也比較容易重現同一個 checkpoint，缺點則是速度稍慢一些，但由於我使用的模型 DeBERTa-v3-base 本身不大，我覺得算是不錯的 trade-off。

3. 資料切分策略（Train / Validation Split）
在整理完標籤後，我以 90%作為訓練資料、10%作為驗證資料進行切分。由於最終評分會使用外部測試資料，我沒有另外保留本地的 test set，而是把可用資料集中在訓練與驗證上，以提升模型的穩定度。

而進行資料切分時則採用 stratified split，確保三個情緒類別在兩個子集中都維持與原始資料相同的比例。這樣的設定能讓模型在訓練階段看到足夠的樣本，同時保有一份分布一致的驗證集，用來監控收斂情況並選擇最佳 checkpoint。

Model Architrecture:
本次模型以 DeBERTa-v3-base 為主體，搭配一個自己設計的輕量化 MLP 分類頭。選擇 DeBERTa 的原因是它的 Disentangled Attention 能分開處理 token 的內容與位置資訊，對否定詞、語氣反轉與細微語意特徵特別敏感，這些正是情緒分類中最容易影響判斷的關鍵詞。

在輸入表示上，我直接使用 CLS token 作為句子的向量表示。對於情緒分類這類任務，CLS 的效果比 Mean Pooling 更穩，因為模型在預訓練階段就已經被強化去把整句資訊集中到 CLS 上，因此較容易捕捉到情緒強度、否定詞位置與語氣轉折等訊息，尤其是對短句或語意集中在局部的文本，CLS 的表現更為明顯。

在取得 CLS 後，我加入 LayerNorm 與 Dropout 提升穩定性，並接上一個兩層式的 Classification Head：第一層 Linear (hidden→hidden) 結合 GELU 提供非線性表達能力，第二層 Linear (hidden→3) 輸出三分類結果。從實驗觀察，這樣的設計比單層線性分類器在 Neutral 這類模糊的情緒表現上更穩定，同時仍維持輕量，不會造成額外的訓練負擔。

在特色方面，我覺得模型主要有兩個吸收效果比較好的地方：第一，使用 CLS pooling 搭配 LayerNorm 與兩層 MLP 能有效集中情緒訊號，提升類別間的可分性。第二，整體分類頭雖輕量，但透過 GELU 與 Dropout 仍能保持良好的穩定度與泛化能力，讓模型在不同類型的句子上都有一致的表現。

整體架構雖然不複雜，但每個設定都是根據任務特性與訓練環境調整過的，希望在有限算力下仍維持可控、穩定且容易微調的模型設計。

Training Pipeline:
本次實驗的訓練流程可以大致分成四個階段：資料進模型前的準備、前向運算、反向更新、以及模型選擇與保存。整個 pipeline 的目標是在有限算力下，讓 DeBERTa-v3-base 能穩定地收斂，並且方便我持續調整超參數與觀察訓練表現。

最終的訓練流程主要分為三個部分：資料批次處理、模型的前向與反向運算、以及驗證與最佳模型的選擇。整體設計的重點一樣是讓 DeBERTa-v3-base 能在單 GPU 上穩定收斂，同時保持訓練過程簡潔、容易觀察。

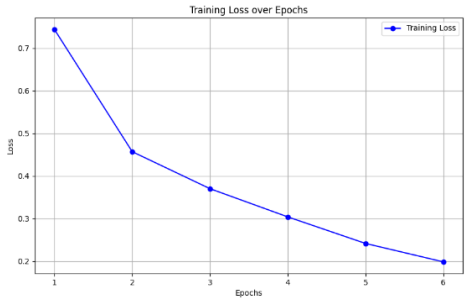
資料前處理部分已在前述章節說明，這裡不再贅述。進入訓練階段後，每個 batch 會先被移到 GPU，再進行前向傳遞取得三類別的 logits，Loss 採用 CrossEntropyLoss。為兼顧運算效率與訓練穩定度，本實驗採用 PyTorch 的 mixed precision，並搭配 GradScaler 自動縮放，不僅降低記憶體占用，也能在相同硬體下使用較大的 batch size。

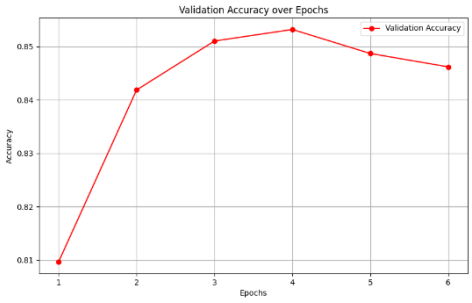
優化器使用 AdamW，整體採用統一的 LR（2e-5），並結合 linear warmup 與 linear decay 排程。前 10%的訓練步驟作為 Warmup，有助於模型在初期穩定啟動；後期則讓 Learning Rate 緩慢下降，以提升收斂的平滑度與最終效能。

每個 epoch 結束後，會使用 Validation Set 計算 Accuracy，只要成績提升就更新 checkpoint，同時輸出對應的混淆矩陣與分類報告，方便後續分析。最後也會把 Loss 與 Validation Accuracy 的變化畫成圖表，讓整體訓練過程更清楚。

3. Experiment Result

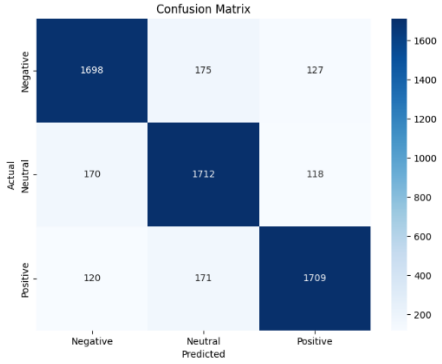
Loss and Accuracy:





Validation Acc (epoch 4) = **0.8532**
Private case Acc (codabench) = **0.8452**

Confusion Matrix:



4. Discussion

Analysis of Results:

本次共訓練 6 個 epoch。從 Loss 曲線可以觀察到模型在前期收斂速度相當快，進入中後期後下降幅度逐漸趨緩，呈現穩定收斂的型態。Validation Accuracy 則在第 4 個 epoch 達到最佳表現，而第 5 與第 6 個 epoch 的成績略有下滑，推測可能開始出現輕微的 Overfitting，因此第 4 個 epoch 被選為最終使用的模型。

就混淆矩陣的結果而言，三個情緒類別的辨識表現大致均衡，沒有明顯的偏向或極端誤判。較常見的錯誤跟預期一樣發生在 Neutral 與其他兩類之間，這與 Neutral 本身語意界線較模糊有關，容易被判成正向或負向。整體來看，模型的分類行為還算合理，錯誤類型也符合多類情緒分類任務的典型情況。

Additional Observations:

在訓練過程中感受到比起模型架構選擇，有時超參數的調整才更影響準確率
以下分三點討論幾個我認為在訓練時顯著影響準確率的超參數分析：

1. Learning Rate 的選擇

在 Fine-tuning 的過程中我發現若 Learning Rate 設得過大，Loss 雖然初期下降非常迅速，但下降過程中上下抖動也十分明顯，跑出來 Accuracy 的數值反而不太理想；若設得過小，則會導致收斂緩慢，並且在實際執行上在中後段 epoch 也出現了 Overfitting 的狀況，所以最後使用合適且安全的 Learning Rate 2e-5 進行訓練。另外本實驗使用的 Warmup 機制能在訓練初期提供較穩定的更新方向，使 Loss 曲線保持平滑，是訓練穩定的關鍵因素之一。

2. Batch Size 的影響

本實驗將 Batch Size 設為 32。從 Loss 曲線的表現可見，這個大小能提供穩定的梯度估計。如果 Batch Size 過小，Loss 會呈現明顯的鋸齒波動；若過大，雖然 Loss 下降的品質較為穩定，但會遇到記憶體的限制且可能卡在 Local Minima。目前的設定 Batch Size=32 在硬體資源和收斂品質之間取得良好平衡。

3. Dropout 的正則化效果

模型中使用 0.1 的 Dropout，有效延緩了 Overfitting 的發生，使模型的 Accuracy 能維持到第 5 個 Epoch 才開始略微下降。經過測試若刪除 Dropout 機制，過擬合會在第 3 ~ 4 個 Epoch 就提早出現，使 Accuracy 下滑。

5. Extra

Additional Experimentation:

在主要模型的 Fine-tuning 之外，我也嘗試做了一組集成知識蒸餾（Ensemble Knowledge Distillation）實驗，希望看看能不能把多個模型的能力合併起來，以達到互補的效果而提升最終表現。我採用的方式是從三個不同架構的老師模型（BERT-Base、RoBERTa-Base、DeBERTa-v3-base）抽取中間層特徵，然後要求一個 DeBERTa-v3-Large 學生去對齊它們的平均隱藏層。為了讓對齊有足夠影響力，我在 Loss 中加入了 BETA（約 0.4）作為中間層蒸餾的權重。

結果則不太理想，在加入中間層蒸餾後，模型在第一個 epoch 的準確率就直接掉到大約 78%，甚至比單純訓練的 DeBERTa-v3-Large 效果還差。我後來推測原因是不同模型家族的隱藏層空間並不相容，直接把它們平均只會得到一個沒有語意結構的「雜訊向量」。強迫學生去模仿這種向量，等於破壞了 DeBERTa 自己的預訓練語意空間，造成嚴重的負面知識轉移。

基於這些結果，我最後完全放棄 Distillation 的路線，改回穩定的 DeBERTa-v3-base 單模型微調流程，超參數設定也保持簡潔。實測下來這樣的設定反而比較可靠，而且能穩定把單模型的準確率維持在 84%左右。