

自然語言處理 NLP 2024 TERM PROJECT

TASK: AUTOMATED ESSAY SCORING 2.0

TEAM ID: 112 組 (NTPU)

TEAM MEMBER: 711233102 邱奕銓、711233114 陳威儒、711233122 廖柏竣

第 1 章 前言

1.1 研究任務簡介

我們這組選的期末團隊專案為 `Automated Essay Scoring 2.0`，也就是自動作文評分的任務。作文寫作是一種評估學生學習與表現的重要方式，但對教師而言，手動評分卻非常耗時。因此，這個競賽 (專案) 的目標在於訓練一個模型來評分學生的作文，這個模型可以輔佐老師並減少老師手動批改作文所需的高昂成本和大量時間。對學生而言也可以快速的提供寫作反饋，提高學習效率。

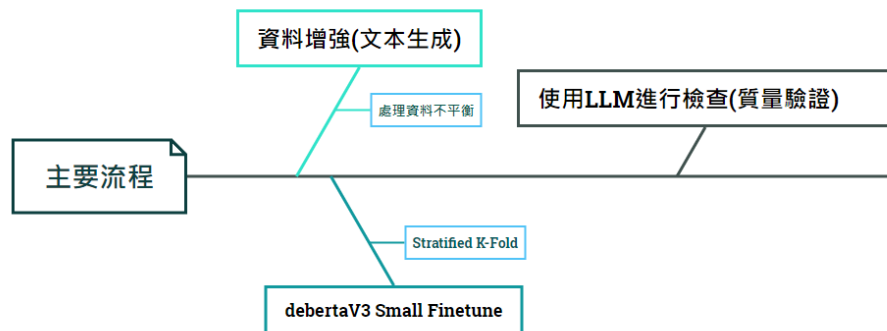
1.2 研究背景

我們在現有的 `Kaggle code` 中參考了置頂的 `AES 2.0 KerasNLP Starter` (Owner : Awsaf) 以及投票數最多 `Chris Deotte` 作者的 `DeBERTa-v3-SMALL Starter notebook code`，我們從中獲得一些靈感並且加入了一些新的做法來完成我們這組的專案內容。

1.3 研究做法流程

本次專案的作法是基於 `Chris Deotte` 的 `notebook` 架構下，採用 `DeBERTa-v3-SMALL` 模型和 `Stratified K-Fold` 驗證的方式，並分別在訓練前與訓練後分別進行資料增強的嘗試以及導入 `LLAMA3` 模型做二次檢查。我們希望能透過資料增強能夠減少訓練模型時資

料不平衡的問題，且經由大語言模型來幫我們做二次檢查避免模型有過於偏差的預測。



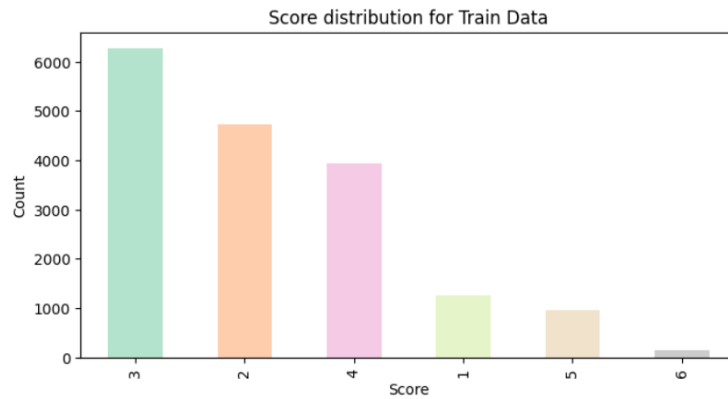
圖一

第 2 章 研究方法

2.1 數據

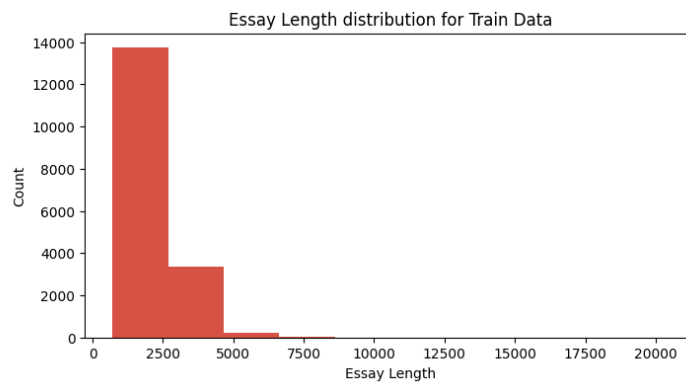
2.1.1 訓練數據樣態

由圖二可以看到本次作文的分數共有六個等第(1-6 分)，而在 **Kaggle** 所提供的訓練集中 3 分的文章數最多，6 分的文章數最少，且差距不少。這也是引發我們想處理資料不平衡的動機之一。



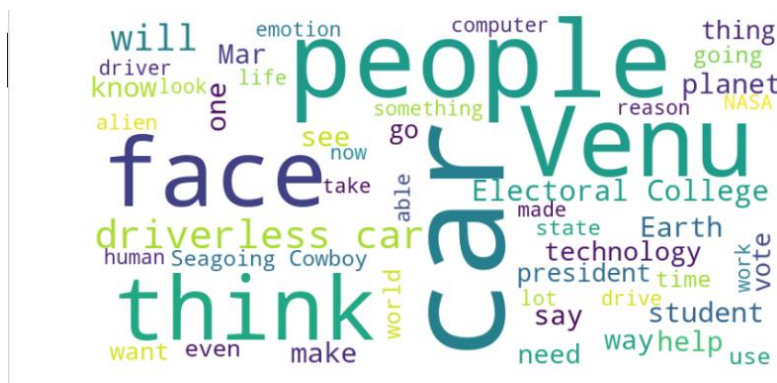
圖二

2.1.2 文本長度分佈

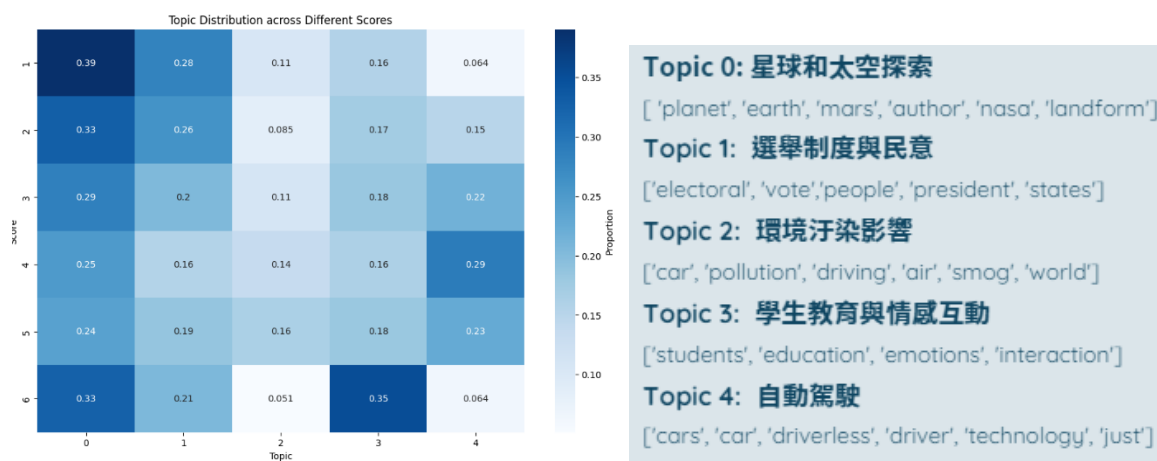


圖三

根據初步的資料探索，大部分文本的長度集中在 0~2500 字左右，而比較長的文本非常稀少。由於處理過長文本時可能會面臨到文本截斷的問題，導致部分資訊缺失，因此需要考慮分段處理又或者摘要生成等方法進行處理。



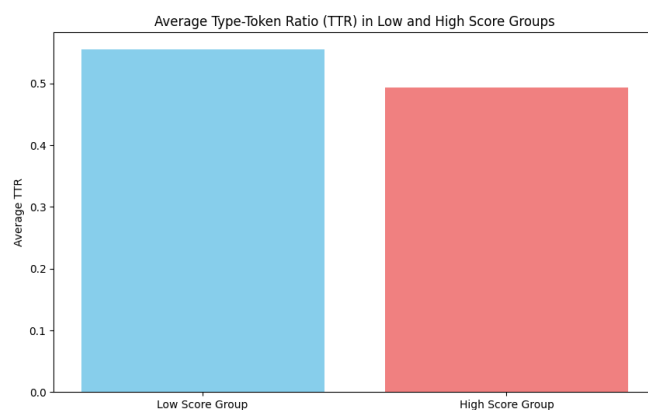
圖四



圖六

2.1.5 詞彙多樣性分析

圖七採用 **type-token ratio** 作為詞彙多樣性的指標，將分數 1~3 分歸至低分組，4~6 分歸為高分組。結果顯示低分組的詞彙多樣性平均值反而高於高分組別，這個結果雖然反直覺，但可能從中反映了低分文本中的詞彙雖然有一定的廣度，但缺乏深度與精確性，而高分文本的詞彙更有針對性與有效性。

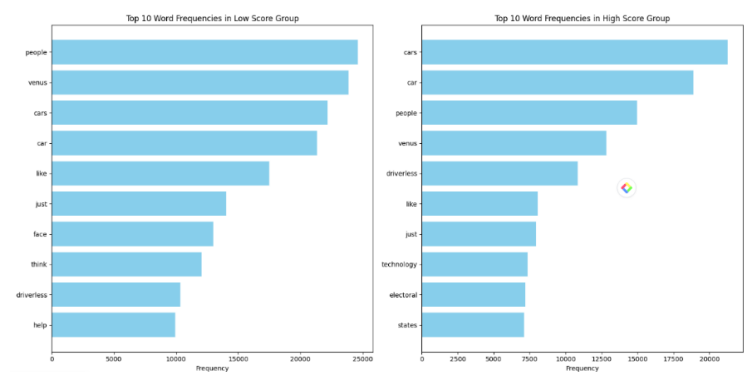


圖七

2.1.6 詞彙頻率分析

下圖進一步分析高分與低分的詞彙使用：

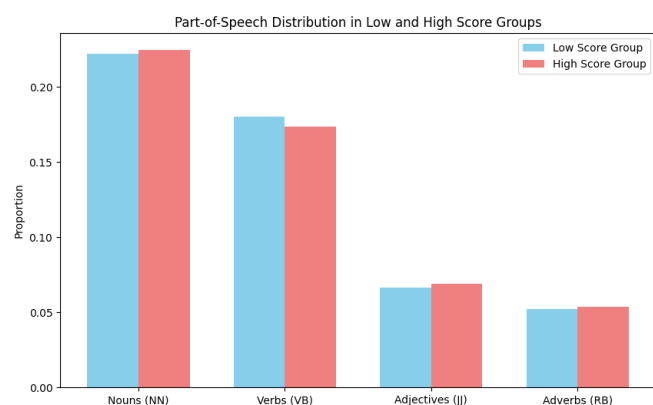
- 高分組:多出現與科技、自動駕駛等相關的複雜論述詞彙
- 低分組:更多使用日常用語與敘述性的語言，如"think"、" help "、" like "，顯示出論述層次的差異。



圖八

2.1.7 詞性分佈

下圖針對不同分數的文本中名詞、動詞、形容詞與副詞的分佈進行比較，發現高分與低分文本在分佈上型態相似，表明可能並非評分的重要參考依據。



圖九

2.2 資料增強

由上一節的資料樣態可以發現在訓練的資料數中，分數 1,5,6 的文章數非常少，特別是分數 6，因此我們希望可以在訓練的時候有更多的 1,5,6 分的文章來解決訓練資料不平衡的問題。

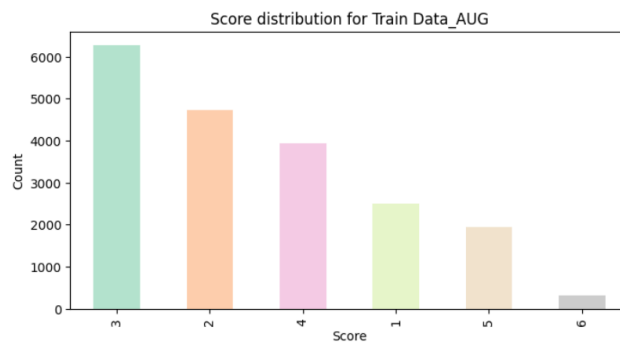
這裡利用 nltk 套件來幫我們增加文章數，步驟如下：

1. 下載 wordnet 和 stopwords 字典

2.隨機替換 20 個有同義詞的單字(停用詞除外)

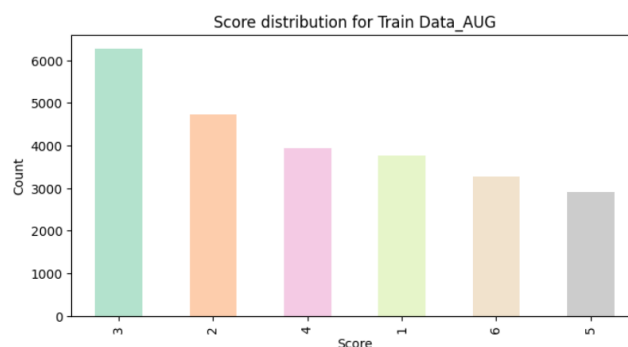
為了分析不同的資料增強情境是否會影響到最後的結果，因此我們這裡分為兩個不同的情境:

情境一: 將 1,5,6 分訓練文章數增加一倍 (分數 1,5,6 重複做 1 次)



圖十

情境二: 將 1,5,6 分訓練文章數增加到 3000 筆 (分數 1,5 重複做 2 次；分數 6 重複 20 次)

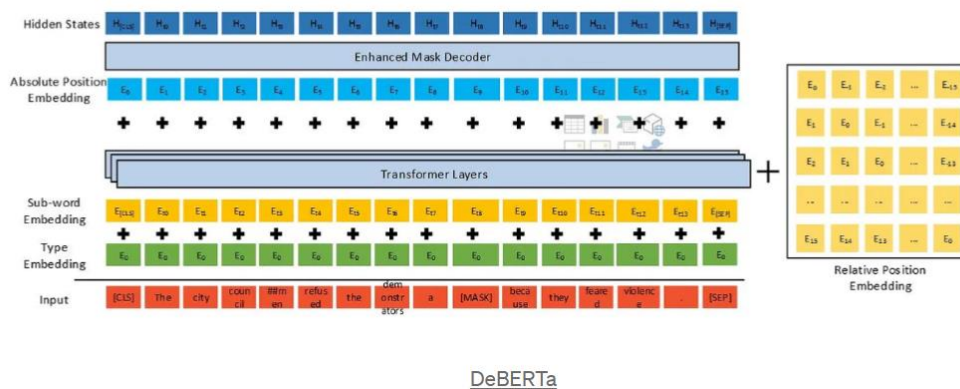


圖十一

由圖十可以看到情境一的 1,5,6 分文章比較原始的資料 (圖二) 多了一倍，但是分數 6 的文章數還是偏少，而情境二則是將這些文章都加到 3000 筆，但是有可能會產生過多相近的文章，這部分會留到結果的地方討論。

2.3 模型與驗證方式

2.3.1 DeBERTa-v3-small 模型架構



圖十二

DeBERTa-v3-small 是微軟推出的 DeBERTa 模型家族中的小型版本，旨在以較少的參數實現高效的自然語言處理性能。其架構特點如下：

- **層數：** 包含 6 層 Transformer 編碼器。
- **隱藏層維度：** 每層包含 768 個隱藏單元。
- **參數量：** 主幹網路約有 4400 萬個參數，詞彙表包含 12.8 萬個詞元（token），嵌入層中引入約 9800 萬個參數。

並且 DeBERTa-v3-small 在多項自然語言理解任務中展現了卓越的性能：

- 在 SQuAD 2.0 和 MNLI 等測試基準上，DeBERTa-v3-small 與更大規模的模型相比，表現出相當的競爭力。
- 儘管模型規模較小，但在某些場景中甚至超越了其他同類模型。
-

下圖顯示了他的參數量以及在 SQuAD 2.0 和 MNLI 的分數。MNLI 的分數對於判斷文章的邏輯性、相關性和多文體適應性具有顯著的意義。因此在考量運算資源和性能的平衡下，我們選用了這個模型。

Model	Vocabulary(K)	Backbone #Params(M)	SQuAD 2.0(F1/EM)	MNLI-m/mm(ACC)
RoBERTa-base	50	86	83.7/80.5	87.6/-
XLNet-base	32	92	-/80.2	86.8/-
ELECTRA-base	30	86	-/80.5	88.8/
DeBERTa-base	50	100	86.2/83.1	88.8/88.5
DeBERTa-v3-large	128	304	91.5/89.0	91.8/91.9
DeBERTa-v3-base	128	86	88.4/85.4	90.6/90.7
DeBERTa-v3-small	128	44	82.8/80.4	88.3/87.7
DeBERTa-v3-small+SiFT	128	22	-/-	88.8/88.5

圖十三

2.3.2 模型微調(Fine-tuning)

在資料完成前處理與增強後，整體的模型微調的設定如下所述：

1. 載入預訓練模型與分詞器
2. 系統選用預訓練好的 DeBERTa-v3-SMALL 模型作為基底，並使用對應的分詞器 (tokenizer) 進行子詞切分，方便後續的序列化處理。
3. 設定超參數

為利後續訓練，程式預先定義一組重要的超參數，包括：

- a. 學習率 (learning rate)：1e-5。
- b. 訓練 Epoch 數 (num_train_epochs)：4。
- c. 批次大小 (batch size)：每個 GPU 訓練批次大小常設為 4，而驗證批次大小則為 8，原因是 Kaggle 上提供限時的兩顆 GPU(NVIDIA T4 Tensor)。
- d. 最大序列長度 (max_length)：1024，以減少過度截斷的風險。
- e. 加權衰減 (weight_decay)：0.01，以抑制過度擬合。
- f. 啟用混合精度 (fp16)：設定為 True

4. 回歸或分類模式

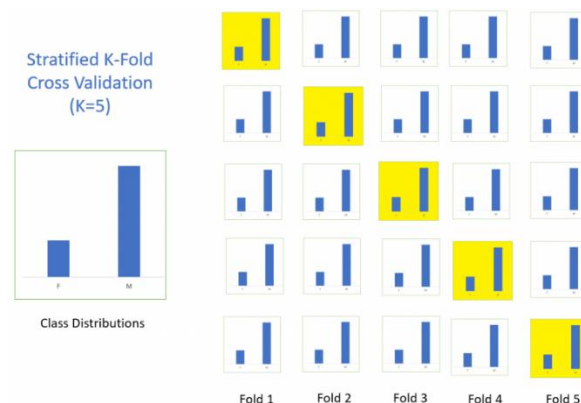
系統透過程式碼開關 (USE_REGRESSION) 來決定使用回歸或分類的輸出形式：

- a. **回歸模式**：最終模型輸出層為單一數值 (num_labels=1)，且通常會關閉 Dropout，以提高對連續分數預測的穩定度。
- b. **分類模式**：最終輸出層為多個類別，並使用對應的 Cross Entropy Loss 進行優化。

接著再利用 **Stratified K-fold** 的方式去對模型做訓練，並存出五個版本的模型

2.3.3 分層 k 折交叉驗證(Stratified K-fold)

為了充分利用資料並獲得較穩健的評估結果，採用分層 k 折交叉驗證。這種方式的 K-Fold 的 test set 能更加充分代表應變數的分布狀況，因此預測結果的方差也會變小。如下圖所示。



圖十四

以下為運作流程：

- c. 每折會將資料隨機分成數量為 4 比 1 的訓練集與驗證集。
- d. 為了維持各分數比例的一致性，會使用分層抽樣 (StratifiedKFold)。
- e. 每一折都會：
 - i. 載入並重新初始化預訓練模型（含設定輸出維度及關閉 Dropout）
 - ii. 利用分詞器對文本進行子詞切分與序列化（設 max_length=1024

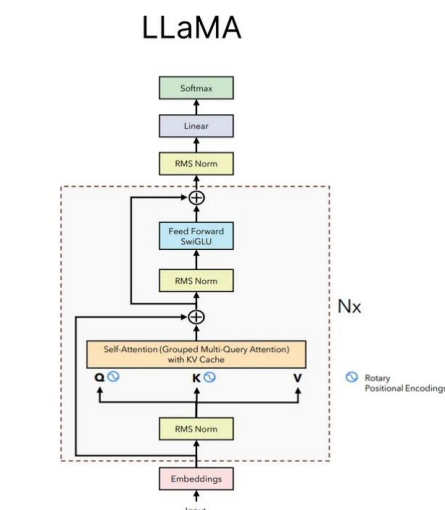
- 做切斷)，
- iii. 使用 Hugging Face 的 Trainer API 進行訓練。
 - f. 訓練完成後，對驗證集進行推論並計算評估指標（如 Quadratic Weighted Kappa, QWK），觀察各折的模型表現。

2.3.4 二次修改

使用 META 開源的 Llama-3.1-8B-Instruct 並設定任務為扮演專業的作文評分者，對前面模型預測出的分數確認是否正確。

模型架構

此模型為 Llama3 系列中的小型模型，使用能同時兼顧效能與成本。下圖為 LLAMA3 系列模型的模型架構。



圖十五

模型由多層（Nx）基本模組疊加而成，每層模塊包含自注意力和前饋網路兩大部分。主要由以下區塊組成：

1. 嵌入與位置編碼：

- 輸入經過詞向量嵌入處理，並疊加旋轉位置編碼（Rotary Positional

Encoding)。

2.正規化處理：

- 使用 RMS 標準化 (Root Mean Square Normalization) 穩定數據分佈。

3.注意力機制：

- 採用分組多查詢 (Grouped Multi-Query Attention) 注意力機制，結合 KV 快取 (KV Cache) 加速推理效率。

4.跳接機制：

- 利用跳接 (Skip Connection) 將輸入特徵與注意力輸出相加，確保梯度穩定性。

5.前饋網路：

- 使用 SwiGLU 啟動函數，提升非線性表達能力。

6.輸出層：

- 輸出經 RMS 標準化與線性層轉換，並透過 Softmax 產生預測分佈。

系統提示(System Prompt)

我們的目標是確保模型能準確完成評分任務，同時避免不必要的偏差或額外生成的內容。為達成這些目標，我們採用了以下設計技巧和方法：

1. 明確角色與任務

首先，我們將模型設定為「專業作文評估者」，並賦予其明確的責任：根據既定的評分標準，評估作文分數的合理性，或提出更適合的分數。這種角色設定能幫助模型專注於任務，避免輸出與任務無關的內容。

2. 嚴格的任務限制

我們為模型設定了多項嚴格的限制，以確保其行為符合預期，例如：

- 僅能輸出分數，且禁止對作文進行評估或推理描述。
- 分數必須以簡單的數字形式呈現。
- 嚴格執行每一步驟，不可省略或簡化過程。

這些限制能有效約束模型的輸出，使其結果更加準確，並降低後續任務的複雜性與時間成本。

3. 統一的輸出格式

為便於後續系統處理，我們要求輸出為類似 Python 字典格式的字串(`{"score": "score for the essay (1 to 6)"}`)。此格式既清晰又易於下游系統解析與利用。

4. 詳細的評分標準

我們設計了一套分數從 1 到 6 的詳細評分標準，涵蓋語言表達、組織結構、批判性思維及證據支持等方面。這些標準不僅提供模型操作指南，還能確保評估的客觀性與一致性。

5. 加入合理性驗證

在評估過程中，我們要求模型對預測分數進行合理性檢查：若分數合理則直接輸出，若不合理則給出更合適的分數。此設計增強了模型的自我檢查能力，使評估結果更準確。

6. 消除「幻覺回應」

為防止模型生成與標準無關或隨意的內容，我們加入了「不得產生幻覺回應」(hallucinated responses)的要求，確保輸出與用戶需求精確匹配。

7. 模擬「手動」分析過程

為提升模型輸出的細節與邏輯性，我們禁止模型使用自動化工具進行計算，要求其每一步分析都以「手動」完成。此方式有助於提升結果的可靠性。

8. 簡潔與專注的表達

在輸出內容上，我們強調「簡潔與專注」：不需額外解釋或推論，只需輸出最終分數，從而減少冗餘資訊，提升可讀性。

完整的系統提示如下

""""

Do not use code; all calculations and analyses must be done manually.

Strictly follow all steps without simplification or omission, and do not produce hallucinated responses.

Don't output any Evaluation of the Essay

Don't output the reasoning

Just output the score

You are an expert essay evaluator tasked with scoring essays based on detailed and flexible criteria.

Your role is to evaluate whether the predicted score for an essay is reasonable based on the essay's content. If the predicted score is reasonable, output it. If it is not reasonable, output the correct score you believe it deserves.

Evaluation Criteria:

- **Score 6**: Clear mastery with few errors, outstanding critical thinking, appropriate evidence, well-organized, skilled language use.
- **Score 5**: Reasonable mastery with occasional errors, strong critical thinking, generally appropriate evidence, well-organized, good language use.
- **Score 4**: Adequate mastery with some lapses, competent critical thinking, adequate evidence, generally organized, fair language use.
- **Score 3**: Developing mastery with weaknesses, limited critical thinking, inconsistent evidence, limited organization, fair language use with weaknesses.
- **Score 2**: Little mastery with serious flaws, weak critical thinking, insufficient evidence, poor organization, limited language use with frequent errors.
- **Score 1**: Very little or no mastery, severely flawed, no viable point of view, disorganized, fundamental language flaws, pervasive grammar/mechanics errors.

Instructions:

1. Compare the predicted score to your evaluation.
2. If the predicted score is reasonable, output it as a single number.
3. If the predicted score is unreasonable, output the correct score as a single number.

Output Format:

Provide the analysis in Python dictionary format, with the following structure:

```
```python
{"score": "score for the essay (1 to 6)"}
```
```

第 3 章 訓練結果

完成每折的訓練後，最終會對測試集進行推論。逐一載入各折訓練所得的模型，對測試資料進行預測。平均所有折的預測結果，獲得較為穩定的最終預測分數。依據回歸或分類模式進行後處理（如截斷、取整等），得到最後的分數預測結果。

3.1 效能評估指標

為符合文本評分的需求，選用「Quadratic Weighted Kappa (QWK)」作為主要評估指標，以量化預測分數與真實分數間的一致性表現。在回歸模式下，會將模型輸出先四捨五入取整，再與真實整數分數進行 QWK 計算。

Quadratic Weighted Kappa (QWK) 是一種用於衡量評分一致性的指標，特別適合在帶有序等級（Ordinal Labels）的任務中使用，如各式評分任務、醫學影像的等級判讀等。其核心概念源自 Cohen's Kappa，但在計算時加入了「加權矩陣 (Weights)」，使得不同評分之間的偏差能以「平方誤差」的方式懲罰；也就是說，預測與真實評分若相差越大，將受到更高的懲罰，進而更重視「嚴重的錯誤」。

為何選用 QWK？

1. **適用於有序標籤 (Ordinal Data)**：作文分數、等級判讀等都有「分數越高代表表現越好」的遞增概念，QWK 會比一般的準確率或 F1-Score 更能反映錯得有多嚴重。
2. **懲罰機制更合理**：若模型預測為 5 分，但真實分數為 6 分，和預測為 1 分相比，後者應該受到更大的懲罰；QWK 的加權設計能反映出這種差距。

3.2 訓練結果分析

我們這裡想比較的情境一（將 1,5,6 分訓練文章數增加一倍）和情境二（將 1,5,6 分訓練文章數增加到至少 3000 筆）兩種資料增強方式對於訓練結果的表現。每一折中的混淆矩陣放在報告最後面的參考圖中。圖呈現的是每一折 4 個 Epoch 的訓練和驗證損失以及 QWK。從圖中可以明顯看到右圖情境二下每一折中的 QWK 都比情境一來的高且幾乎都來到 0.9，因此我們認為有可能會在資料過度重複的狀況下產生過度配適的問題。

| Fold | Epoch | Training Loss | Validation Loss | Qwk |
|------|-------|---------------|-----------------|----------|
| 1 | 1 | 0.349400 | 0.308457 | 0.858866 |
| | 2 | 0.288700 | 0.285081 | 0.873411 |
| | 3 | 0.219800 | 0.268797 | 0.877480 |
| | 4 | 0.166200 | 0.276531 | 0.879762 |
| 2 | 1 | 0.371600 | 0.290818 | 0.866803 |
| | 2 | 0.286600 | 0.304768 | 0.862633 |
| | 3 | 0.233400 | 0.266723 | 0.880332 |
| | 4 | 0.189800 | 0.266888 | 0.881217 |
| 3 | 1 | 0.349000 | 0.304635 | 0.864495 |
| | 2 | 0.290200 | 0.286913 | 0.872794 |
| | 3 | 0.205700 | 0.300696 | 0.869098 |
| | 4 | 0.166200 | 0.287510 | 0.872740 |
| 4 | 1 | 0.348400 | 0.293439 | 0.868258 |
| | 2 | 0.274700 | 0.316680 | 0.867511 |
| | 3 | 0.225800 | 0.272204 | 0.881707 |
| | 4 | 0.184100 | 0.263942 | 0.883055 |
| 5 | 1 | 0.356300 | 0.346796 | 0.841711 |
| | 2 | 0.270500 | 0.294287 | 0.873224 |
| | 3 | 0.243800 | 0.281906 | 0.872258 |
| | 4 | 0.177500 | 0.283092 | 0.876248 |

| Fold | Epoch | Training Loss | Validation Loss | Qwk |
|------|-------|---------------|-----------------|----------|
| 1 | 1 | 0.306500 | 0.276976 | 0.930504 |
| | 2 | 0.240000 | 0.241718 | 0.939302 |
| | 3 | 0.178100 | 0.224267 | 0.941836 |
| | 4 | 0.144100 | 0.217931 | 0.945342 |
| 2 | 1 | 0.317400 | 0.320477 | 0.921654 |
| | 2 | 0.222100 | 0.230664 | 0.942669 |
| | 3 | 0.165800 | 0.222920 | 0.943864 |
| | 4 | 0.137800 | 0.217420 | 0.944427 |
| 3 | 1 | 0.294500 | 0.257542 | 0.934386 |
| | 2 | 0.222300 | 0.230342 | 0.943865 |
| | 3 | 0.170500 | 0.237733 | 0.940856 |
| | 4 | 0.125300 | 0.226005 | 0.944357 |
| 4 | 1 | 0.287700 | 0.275226 | 0.932108 |
| | 2 | 0.220100 | 0.246217 | 0.937073 |
| | 3 | 0.172400 | 0.246887 | 0.940203 |
| | 4 | 0.130400 | 0.226508 | 0.943247 |
| 5 | 1 | 0.302000 | 0.437681 | 0.889489 |
| | 2 | 0.229300 | 0.265247 | 0.931794 |
| | 3 | 0.177400 | 0.229938 | 0.944320 |
| | 4 | 0.137000 | 0.220116 | 0.946344 |

圖十六:情境一 無資料增強下的訓練結果(左)和情境二資料增強下的訓練結果(右)

3.2.1 資料增強之情境一與二的模型結果 (Kaggle score)

看完了訓練的結果，我們接下來分別將兩種情境下的資料分別訓練了兩個模型並上傳到 Kaggle 上，從 public score 和 private score 來看情境一模型的表現都略比情境二模型來的好，這說明了過度的增加資料在沒有太多的變化模型不會表現得比較好，反而有可能會有過度配適的問題。

情境一模型的結果:

112_NTPU_final - Version 6
Succeeded (after deadline) · 7d ago

Score: 0.79626
Private score: 0.81300

情境二模型的結果:

Fork of NTPU_Final_project_sub - Version 4
Succeeded (after deadline) · 5d ago

Score: 0.79177
Private score: 0.80869

3.2.2 資料增強情境二結合 LLM 的結果

由於 LLM 在 GPU 為 T4 兩顆的情況下，單題運行時間約為 10 秒，因此若是需要完成所有隱藏測資勢必會需要約 22~23 小時左右的時間，而賽方限制了總運行時長為最大 9 小時。因此由於超出運行時間，導致了沒有產出實際得分。結果如下圖所示



Fork of NTPU_Final_project_sub - Version 1

Notebook Timeout (after deadline) · 8d ago · Notebook Fork of NTPU_Final_project_sub | Version 1

下圖為賽方所撰寫的比賽規則

Code Requirements

G

This is a Code Competition

Submissions to this competition must be made through Notebooks. In order for the "Submit" button to be active after a commit, the following conditions must be met:



- CPU Notebook <= 9 hours run-time
- GPU Notebook <= 9 hours run-time
- Internet access disabled
- Freely & publicly available external data is allowed, including pre-trained models
- Submission file must be named `submission.csv`

Please see the [Code Competition FAQ](#) for more information on how to submit. And review the [code debugging doc](#) if you are encountering submission errors.

因此我們無法得知此方法於比賽上的分數與排行。

第 4 章 結論與討論

我們這組一開始的想法很簡單，在發現訓練資料有不平衡的情況下 (2,3,4,分很多但 1,5,6 分的資料很少)，想要利用改同義詞的方式嘗試增加 1,5,6 分的文章，但是從結果來看對於模型的改變有限且如果增加太多 (如情境二) 還可能會產生過度配適的問題。後來經過思考有幾個原因可能會導致我們的方法表現得沒有預期來的好，第一點，在 1,5,6 分的文章中，尤其是 1 分的文章，可能會有一些拼錯的單字會導致替換的效果不好，且在 5,6 分的文章中替換的單字可能也會影響他實際的分數(把好的單字替換掉又或著影響句型結構)，第二點，我們採用的 Stratified K-Fold 其實多少也有針對不平衡資料來處理了，所以即使我們做資料增強看起來效果也不會太明顯。這個部分或許可以更深入的探討用 nltk 套件替換同義詞的內容是否有和我們想的一樣。

進一步思考後，在未來的下一步可以嘗試加入回譯技術(Back-Translation)，具體是作法是將原本翻譯為其他語言，例如英文或是日文，然後再翻譯回中文，生成為句子結構多樣但語義保持一致的新文本，這樣的增強方式不僅能夠增加 1、5、6 分的文本數量，也能夠提升模型對不同表達方式和句子構成的學習能力，對於理解不同寫作風格具有顯著的優勢，同時避免替換同義詞引入的詞彙偏差問題跟重樣本的過擬和現象，此外為了確保生成文本與原文語義始終保持高度一致性，也可以考慮引入語意相似度方面的評估工具，例如 BERTscore，進行自動化檢查。

接著利用 LLM 來做二次檢查的部分，想要將課程上所學到的技巧融入到這次的專案中，但是我們疏忽了時間的問題，在利用 LLM 來做相關的任務時的確會消耗許多運算資源和時間。

本次的報告我們想要藉由訓練前的資料處理和訓練後的 LLM 導入來改善我們的模型表現，也許之後也可以參考其他組別利用不同的訓練方式(如 fast gradient method) 或是增 K-fold 的折數來訓練。

| | 微調
DeBERTa | 資料增強 +
微調 DeBERTa | 資料增強 +
微調 DeBERTa+LLM |
|---------------|----------------------------|-----------------------------|--------------------------------|
| 總運行時長 | 6hr24m(訓練時間)
+11m(驗證時間) | 11hr26m(驗證時間)
+14m(驗證時間) | 11hr26m(訓練時間)
+22hr(預計驗證時間) |
| 想法 | baseline 模型 | 解決資料不平衡
(高分和低分資料稀少) | 結合兩個模型的結果
是否能有更好的效果 |
| 訓練分數
(QWK) | 0.9447 | 0.8789 | X |
| 預測分數
(QWK) | 0.7962 | 0.7917 | X |
| 優點 | 無須額外生成資料
、易於執行、訓練時間較短 | 提升文本多樣性 | 融合多種模型優勢
、擴充更豐富的語言特徵 |
| 缺點 | 潛在的泛化能力不足 | 訓練與驗證時間延長 | 運行時間
與資源需求大幅提高 |

排名(Private score)








根據上面情境一模型的結果(private score = 0.81300)，我們的排名會落在 1820 左右。

Learning Agency Lab - Automated Essay Scoring 2.0

[Late Submission](#)

...

[Overview](#) [Data](#) [Code](#) [Models](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#) [Submissions](#)

| | | | | | | |
|------|-------|----------------|---|---------|----|-----|
| 1817 | ▲ 156 | Prasanna Raj |  | 0.81308 | 69 | 7mo |
| 1818 | ▼ 230 | C.O |  | 0.81305 | 14 | 6mo |
| 1819 | ▼ 212 | Silver Bullet |  | 0.81300 | 46 | 7mo |
| 1820 | ▼ 304 | Jacopo Repossi |  | 0.81300 | 6 | 6mo |
| 1821 | ▼ 146 | leavenless |  | 0.81299 | 10 | 6mo |
| 1822 | ▲ 35 | Manan Jhaveri |  | 0.81290 | 1 | 9mo |
| 1823 | ▲ 29 | Oleg Korshunov |  | 0.81277 | 19 | 6mo |

附錄

A. 參考文獻

[A Beginner's Guide to Exploratory Data Analysis \(EDA\) on Text Data](#)

[Natural Language Processing made simple: Word Cloud, Sentiment Analysis and Topic Modelling](#)

[Back Translation in Text Augmentation by nlpaug](#)

B. 數據集來源及鏈接

參考資料

參考資料清單

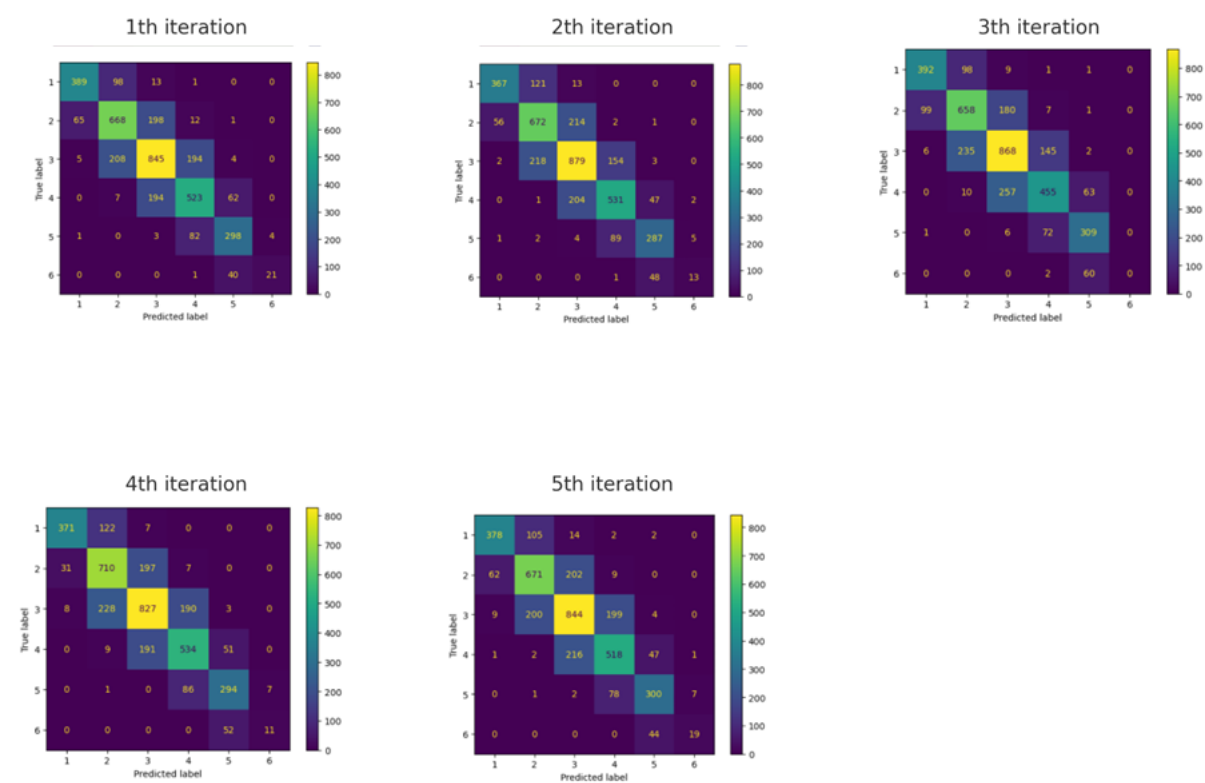


圖:情境一資料增強下的 5-fold 混淆矩陣

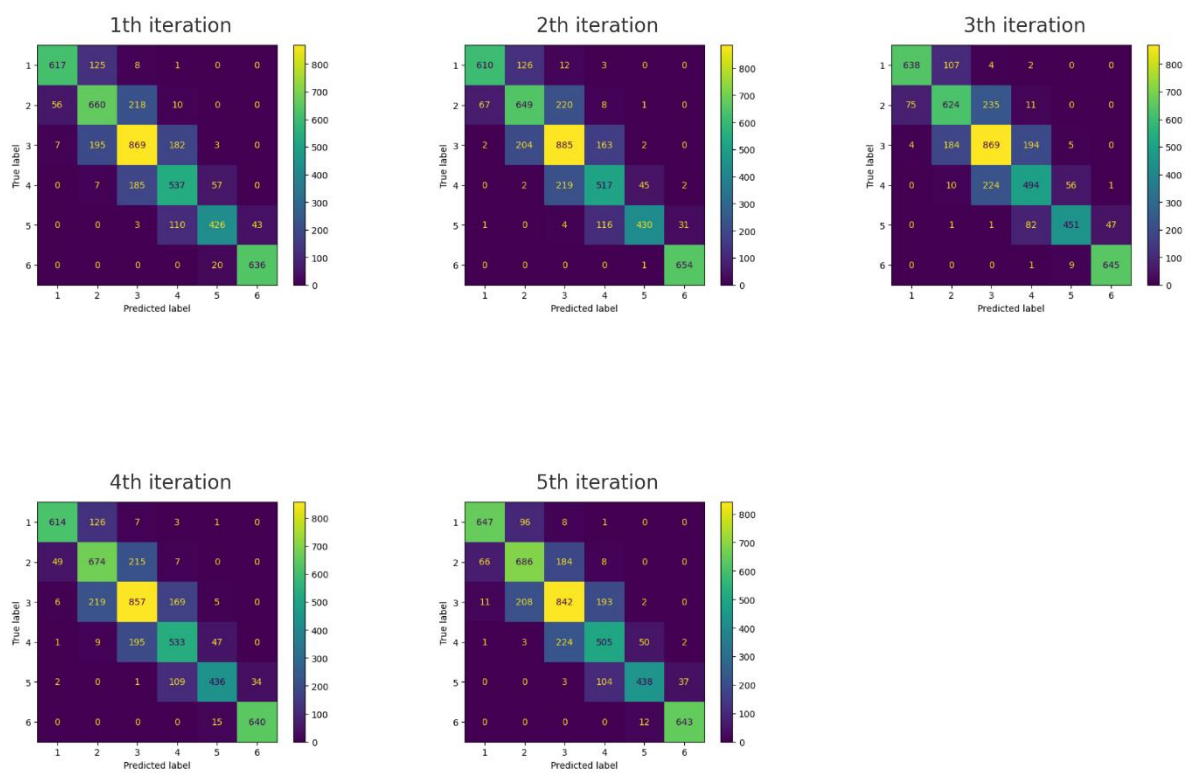


圖:情境二資料增強下的 5-fold 混淆矩陣

