DM2024 ISA5810 Lab2 Homework

NTHU 110062328 劉田元

I. Preprocess - Clean text

在整理完 data，將它們合併進同一個 data frame 並分拆成 df_train and df_test 後，首先我做的便是觀察原始文字，並發現以下幾點：

- Hashtags 其實包含在文字中
- 時常出現<LH>這個代號
- 有用戶符號與 url，例如：@ShinsukeN
- 刻意的拼寫錯誤、狀聲詞：RUNNNNN, erghhhhh
- 大量表情符號
- 重複的標點符號：!!!!!!!
- 簡語：wtf, lol

考慮到之後還需要做 Tokenize，我也進一步去把原始文字丟到 tokenizer，發現了一個問題：因為該分詞器採用的是 BPE (Byte Pair Encoding) 技術，會透過 @@ 表示子詞，比如 I've 會被分拆成'I@@', ''@@', 've'，這樣的技術能讓模型有效處理不常見詞彙以及縮寫等特殊語言現象。但是用在連續的表情符號上，比如 😁😂 就會被分詞成 '😁@@', '😂'，但一般來說這兩著沒有這樣的關係（雖然也確實有多個表情符號連用而產生新意思的例子，但是比例不高）。面對這個問題，由於 re 並不支援 emoji 的判斷，我採用 regex 來進一步提取出他們。

綜上所述，我最後的清理文字流程如下：

1. 替換 <LH> 為 "light-hearted"
2. 刪除 URL 和 @user
3. 移除標籤的 `#` 符號，但保留關鍵字
4. 在表情符號前後添加空格
5. 簡化重複標點（如 "!!!" -> "!")
6. 移除多餘空格

結果展示如下：

text	clean_text
o m g Shut Up And Dance though #BlackMirror <LH>	o m g Shut Up And Dance though BlackMirror lig...
On #twitch <LH> on the #Destinybeta #Destiny #...	On twitch light-hearted on the Destinybeta Des...
A nice sunny wak this morning not many <LH> ar...	A nice sunny wak this morning not many light-h...
I'm one of those people who love candy corn.....	I'm one of those people who love candy corn. a...
@metmuseum What are these? They look like some...	What are these? They look like something toddl...
Postive thinking is the only way to go followe...	Postive thinking is the only way to go followe...
Best Check I've had this year <LH> 👀🙏🙏	Best Check I've had this year light-hearted 👀 🙏 🙏

II. Preprocess – Tokenize and Embed

關於我使用的 Tokenizer 與 pre-train embedding model，我找專門用於 tweet emotion analysis 的開源模型 Bertweet，並且拿它的 Tokenizer 以及 model 的 last hidden state 作為嵌入向量。註：該模型是基於 BERT，我採用的是它的 base version，所以對於每串文字的固定輸出向量為 768 維。

不過，在生成 embedding 的過程不太順利，時常會遇到 CUDA 的一些不明錯誤，具體來說在 process 到 920000 多筆資料時會因為不明原因中斷，又因為整個 embedding 時間需要 4 小時以上，所以每次發生錯誤，如果要重新來過會需要很多時間。因此，我將生成 embedding 的函數改寫為一次處理固定數量，例如 10000 或 100000 筆資料，處理完成會先儲存一次，如果遇到任何 exception，會通知出問題並跳過該 batch，以避免需要重來的情況。

III. Training and Result

我有嘗試使用 SVM、XGBoost 等模型，並進行超參數調整，但是結果都不是很理想：

Classification Report:

	precision	recall	f1-score	support
anger	0.99	0.05	0.09	8,062
anticipation	0.70	0.28	0.40	50,129
disgust	0.42	0.05	0.09	27,789
fear	1.00	0.00	0.01	12,679
joy	0.38	0.97	0.55	102,986
sadness	0.59	0.07	0.12	38,793
surprise	1.00	0.04	0.07	9,790
trust	0.91	0.05	0.09	40,885
accuracy			0.41	291,113
macro avg	0.75	0.19	0.18	291,113
weighted avg	0.61	0.41	0.30	291,113

上圖是 XGBoost 在調整參數過後的結果，從分類報告來看，模型的整體準確率為 41%，表明在多分類任務中的表現並不理想。加權平均的 F1-score 僅為 0.30，顯示模型對支持數較多的類別有明顯的偏向，而對其他類別的辨識能力較差。

針對各情緒類別的表現，模型對 "joy" 類別的召回率高達 97%，但準確率僅為 38%，說明儘管模型能大多數情況下成功預測出 "joy" 類別，卻經常誤將其他類別的數據也標記為 "joy"。與此同時，模型在 "fear" 和 "anger" 類別的表現極差，幾乎無法正確識別這些情緒，顯示模型對小類別數據的特徵學習不足。"surprise" 類別的 precision 雖高達 100%，但召回率僅為 4%，反映出模型的判斷極度保守，錯失了絕大多數該類別的樣本。

此外，"sadness" 和 "anticipation" 的表現介於中間，模型在正確性和覆蓋率上均有一定的改善，但仍然未能達到令人滿意的水準。整體而言，模型在小類別上的表現顯著低於大類別，這與數據集中各類別間的支持數不平衡密切相關。

數據不平衡是模型表現不佳的主要原因之一。支持數較大的類別（如 "joy"）占據優勢，而小類別（如 "fear" 和 "anger"）在訓練過程中可能被忽略。從 Macro 平均值和 Weighted 平均值的對比可以進一步看出，Macro 平均值（每類別的簡單平均）顯示模型在小類別上的平均表現極低，F1-score 僅為 0.18，而 Weighted 平均值則反映出模型更偏向支持數大的類別。

但是因為時間關係，我未能繼續嘗試其他數據平衡技術，例如**過採樣、下採樣或調整類別權重的損失函數 (e.g. Focal loss)**，也無法深入優化模型的架構，例如引入更適合文本分類的預訓練模型如 BERT 或 RoBERTa。此外，多任務學習框架的應用或其他針對情緒分類特化的模型調整也未能嘗試。

在未來的工作中，可以首先解決數據不平衡問題，確保小類別的特徵能被充分學習。其次，探索如何在模型中更有效地融合情緒類別之間的關聯性，例如利用共通特徵或將情緒層級化處理，以提高整體分類的泛化能力。最後，對於小類別（如 "fear" 和 "anger"）的表現，可以專注於特徵工程，加入更多能幫助模型識別的上下文特徵，或採用專為小類別設計的優化目標，如提升 Macro F1-score。

總結來說，雖然本次模型的表現仍有較大改進空間，但通過未來對數據和模型架構的優化，模型應能在多分類情緒分析任務中取得更為均衡且精準的表現。