1.
模型建構: 我使用 pytorch 的 nn.module 建立 RNN 模型。由於過去在實習時公司有將 GRU 應用於表格文字重建並有不錯的表現，因此決定以 GRU 作為基本模型，我首先用 raw data 訓練並同時也跟 torch 的 LSTM、RNN 比較效果，確認 GRU 在 raw data 上的表現是三者最好的。另外，再使用 pretrain embedding 前，模型採用單向的效果較好，但使用 pretrain embedding 後，使用 bidirectinoal model 又使效果進步了 0.01，因此最終採用 bidirectional。

資料預處理(包括 label 平衡): 由於資料集中 fear 與 disgust 的 label 較少，我最初嘗試直接去除專注於其他 label 的表現，卻得到很差的效果，因此之後決定改為進行 data augmentation 以平衡 label 數量。具體作法是將這兩個 label 的句子先經過 translator 翻譯為其他語言，再重新翻譯回英文句子，並將其加入資料集。為避免翻譯出來與原本的句子一樣，因此再進行了隨機交換與刪除單詞，實際測試後 f1 提升了 0.04，是很顯著的提升。之後再將整個資料集中字數多於 7 的句子再做一次隨機交換與隨機刪除，使增強後的資料集大約變為原來的兩被。確認所有參數、訓練方法後，對 valid set 做同樣的增強並與 trainset 一起拿來訓練模型。資料進入模型前會由 spacy 套件進行 tokenize。

2.
Word Embedding 部分我選擇使用 Glove，因為在 torchtext.vocab 中就包含了 pretrained 好的 Glove，使用起來比較方便，也讓我有更多時間調整參數訓練。使用 embedding 後，模型表現進一步有所提升，f1 從 0.331 進步到 0.364，是繼資料增強後又一次的大幅進步。

3.
12/1 上完課後了解了 attention 機制，並從 https://www.youtube.com/watch?v=lqCAfu6GI2c&list=PLE3Y6O9R81ly4UmavfbfaRKbih605E_uG&index=25 影片中了解實作細節，新增了 attention layer，然而加入這層並沒有使表現有所提升，已嘗試多種參數、並減少 batchsize 以增加學習時的變化，卻不見改善，因此最終決定放棄使用 attention。

4.
在完成 RNN 的訓練與預測後，我將預測結果變為資料集的其中一個 column，再與 speaker, episode 等資訊一起丟入 random forest 做預測，並且嘗試保留與去除各種 column 組合來 fit 模型，可惜最終都未能使預測效果有所提升。