

# NLP HW4

## Model design and concept

- 會把上下以及目前句子的 `ids`、`attention mask` 送到同一個BERT，取出第一個node值，之後送到LSTM，此外，會把處理後的 `speaker` 和 `listener` 送到 `layers.Dense` 處理成 `people`，之後 `people` 和LSTM來的結果串接，送到數層 `layers.Dense`，最後一層softmax 7個node對應7種label，model中會穿插 `layers.Dropout`，最後一層之外的 `layers.Dense` activation function都是ReLU

## Error Analysis and Experiment Discussion

- 實驗在訓練完成後，會對拆成validation的資料集再跑一次，根據預測以及真實的情緒，可以知道那些句子的預測情緒出錯
- 以我看到的錯誤，有一部分是發生在沒有上一句或下一句的情形（開頭或結尾），因為這時比平常少了一部份的資訊，也有另一部份來自過長的句子，在tokenizer截斷之後，損失了部分的資訊

## 描述嘗試過的方法，並且討論曾經遇到的問題以及解決的方法

- 本次作業的文本是繁體中文，huggingface的中文模型幾乎都是只支援簡體，加上又要支援tensorflow，我查詢之後沒有這東西，所以我只好用另一個套件把繁體字轉換成簡體字
- 如果直接用tokenizer的預設參數，會直接把所有句子padding到最長的那句，經過實驗除了降低模型表現，而且速度也降低，所以後來改設定長度，超過就截除
- 這次作業的文本相對於Bert來說很小，所以絕對要避免overfitting的問題，除了用 `tf.keras.callbacks.ModelCheckpoint` 儲存 validation表現最好的模型，還在裡面加了 `layers.Dropout`

根據實驗結果分析哪一類的情緒較容易預測？哪一類的情緒較難預測？較難預測的情緒容易被誤認為何種情緒？並探討可能的原因且提出解決辦法

	precision	recall	f1-score	support
angry	0.61	0.65	0.63	2073
disgust	0.57	0.19	0.29	1036
fear	0.67	0.20	0.31	1523
happiness	0.63	0.77	0.69	3934
neutral	0.70	0.71	0.71	6876
sadness	0.53	0.50	0.52	1008
surprise	0.67	0.78	0.72	4164
accuracy			0.66	20614
macro avg	0.63	0.54	0.55	20614
weighted avg	0.65	0.66	0.64	20614

- 好預測的情緒：`neutral`、`surprise`
- 難預測的情緒：`fear`、`disgust`
- 看了訓練資料的標籤分布，分數比較高的標籤在訓練資料中較常出現；分數比較低的標籤在訓練資料中較不常出現
- 錯誤的標籤常常會是`neutral`
- 實際例子
  - 「是」被標為`fear`，但怎麼看都是中立
  - 「怎麼樣了」被標為`fear`，但怎麼看都是中立
- 解決方法
  - 提高分數低的標籤資料量

## 分析不同上下文的資訊量對於預測該句話情緒的影響


上下文	準確度
只有目前的句子	0.56262
目前+上一句	0.57478
目前+下一句	0.59019
目前+上下一句	0.59262

- 實際例子
  - 「你聽我解釋」被標為`fear`，但怎麼看都是如果不包含上下句含容易被標記為中立

## Q1: Data processing

- Chinese Preprocessing and Tokenizer
  - Describe in detail about the tokenization algorithm you use. You need to explain what it does in your own ways.
    - 他會把句子切割成很多個字並且在前後都加上[CLS][SEP]，然後把每個字轉換成一個對應的整數，之後會pad 0或截斷達到設定的長度
  - Describe your preprocessing for Chinese term.
    - 句子中有不具意義的特殊符號：破折號、刪節號等等，這些可以用regex移除
    - 我會把上一句和下一句加入目前的句子，邊界時句子是空的
- Answer Prediction
  - After your model predicts the probability of answer, what rules did you apply to determine the final prediction?
    - 每個model最後一層softmax後，我用 `np.argmax()` 取出最大機率的node index
    - 會得到三份的結果，最後用voting的方式產生最後的結果，之後用label encoder反向轉換

## Q2: Modeling with LMs and their variants

- Describe
  - your model (configuration of the model) and details
    - 會把上下以及目前句子的 `ids`、`attention mask` 送到同一個BERT，取出第一個node值，之後送到LSTM，此外，會把處理後的 `speaker` 和 `listener` 送到 `layers.Dense` 處理成 `people`，之後 `people` 和LSTM來的結果串接，送到數層 `layers.Dense`，最後一層softmax 7個node對應7種label，model中會穿插 `layers.Dropout`，最後一層之外的 `layers.Dense` activation function都是ReLU
    - 我用KFold把資料拆3份，訓練三個模型，最後用三個模型的結果去voting產生最後結果
    - 參數說明
      - `len_speaker`: onehot之後的speaker長度
      - `len_listeners`: listener轉換後的長度
      - `len_ids`: 被tokenizer轉換後句子的長度
  - performance of your model.
    - 2308125010.59262279h
  - the loss function you used.
    - `categorical_crossentropy`
  - The optimization algorithm (e.g. Adam), learning rate and batch size.
    - Adam, lr=1e-4, batch\_size=16
- Plot

