



# HW4 - Sentiment Analysis 2

## 資料前處理

讀取資料僅保留"text"、"stars"兩個欄位，利用分割符號切割字串、建立train&test之DataFrame

```
import pandas as pd

data_csv = pd.read_csv('/content/drive/MyDrive/碩一上課堂/1131_dataMining/dataMining/HW4/yelp.csv')
data_csv.columns = data_csv.columns.str.strip()
data_csv.head()
# 保留 "text" 和 "stars" 欄位
data = data_csv[["text", "stars"]].copy()

# 將 stars 欄位轉換為二元類別
# 大於等於 4 的轉為 1 (positive), 其餘轉為 0 (negative)
data['stars'] = data['stars'].apply(lambda x: 1 if x >= 4 else 0)

# 檢查處理後的資料
data.head()
```

	text	stars
0	My wife took me here on my birthday for breakf...	1
1	I have no idea why some people give bad review...	1
2	love the gyro plate. Rice is so good and I als...	1
3	Rosie, Dakota, and I LOVE Chaparral Dog Park!!...	1
4	General Manager Scott Petello is a good egg!!!...	1

將stars欄位內值大於等於4的轉成1，其餘轉成0

1: positive

0: negative

## 去除停頓詞stop words

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# 確保下載所需的資源
nltk.download('stopwords')
nltk.download('punkt')
# Download the 'punkt_tab' resource for sentence tokenization
nltk.download('punkt_tab') # This line is added to download the missing resource.

# 定義停頓詞
```

```
stop_words = set(stopwords.words('english'))

# 去除停頓詞的函數
def remove_stop_words(text):
    # 分詞
    words = word_tokenize(text)
    # 過濾掉停頓詞
    filtered_words = [word for word in words if word.lower() not in stop_words]
    # 將過濾後的詞組合回句子
    return ' '.join(filtered_words)

# 移除停頓詞
data['text'] = data['text'].apply(remove_stop_words)

# 檢查處理後的資料
print(data.head())
```

```

                                text  stars
0  wife took birthday breakfast excellent . weath...      1
1  idea people give bad reviews place . goes show...      1
2  love gyro plate . Rice good also dig candy sel...      1
3  Rosie , Dakota , LOVE Chaparral Dog Park ! ! !...      1
4  General Manager Scott Petello good egg ! ! ! g...      1

```



**stopwords**：下載包含多語言的停頓詞表。

**punkt**：一個用於分詞的標點符號模型，適用於多語言分詞。

去除了如

**This** , **is** , **a** , **the** , **over** 等常見的停頓詞。

## 文字轉向量、資料分割

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np
from tensorflow.keras.utils import to_categorical

# 使用 Tfidf 向量化文字（或可替換成 Word2Vec）
tfidf_vectorizer = TfidfVectorizer(max_features=500) # 5000 維
X = tfidf_vectorizer.fit_transform(data['text']).toarray()
y = data['stars']

# 分割資料集
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
# 將標籤轉為 one-hot encoding
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

	text	stars	tokenized_text	sentence_vector
0	wife took birthday breakfast excellent . weath...	1	[wife, took, birthday, breakfast, excellent, ...	[-0.23142909, 0.18114784, 0.40228108, 0.203167...
1	idea people give bad reviews place . goes show...	1	[idea, people, give, bad, reviews, place, ,, g...	[-0.14577849, 0.036988724, 0.24384479, 0.33297...
2	love gyro plate . Rice good also dlig candy sel...	1	[love, gyro, plate, ., Rice, good, also, dlig, ...	[-0.21582472, 0.4143621, 0.27463105, 0.1084079...
3	Rosie , Dakota , LOVE Chaparral Dog Park !! !...	1	[Rosie, ,, Dakota, ,, LOVE, Chaparral, Dog, Pa...	[-0.0936799, 0.05060749, 0.039755546, 0.220056...
4	General Manager Scott Petello good egg !!! g...	1	[General, Manager, Scott, Petello, good, egg, ...	[-0.23984121, 0.05407579, -0.044937532, -0.005...
...	...	...	...	...
9995	First visit ... lunch today - used Groupon . o...	0	[First, visit, ..., lunch, today, -, used, Gro...	[-0.1824445, 0.236286, 0.32286814, 0.19796889,...
9996	called house deliciousness ! could go item , l...	1	[called, house, deliciousness, !, could, go, l...	[-0.043673962, 0.016979076, 0.2815596, 0.20090...
9997	recently visited Olive Ivy business last week ...	1	[recently, visited, Olive, Ivy, business, last...	[-0.17075238, 0.069469444, 0.20196898, 0.22462...
9998	nephew moved Scottsdale recently bunch friends...	0	[nephew, moved, Scottsdale, recently, bunch, f...	[-0.14085771, -0.03388557, 0.026640192, 0.1187...
9999	4-5 locations .. 4.5 star average .. think Ari...	1	[4-5, locations, .., 4.5, star, average, .., t...	[-0.28668922, 0.28517798, 0.26714477, 0.123777...

10000 rows x 4 columns

- 1. 文字向量化：使用 Tfidf 方法將文字資料轉換為數值特徵矩陣，限制特徵數量為 500（max\_features=500）。
- 2. 特徵與標籤分離：將文字特徵存入 x，將標籤（假設為評分星級）存入 y。
- 3. 分割訓練集和測試集：按照 80% 訓練集、20% 測試集的比例隨機分割資料，並根據標籤分布進行分層抽樣（stratify=y）。
- 4. 標籤編碼：將目標標籤（y\_train 和 y\_test）轉換為 one-hot 編碼格式，以便用於深度學習模型。

## 建模（CNN、LSTM）

### CNN 建模

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Conv1D, MaxPooling1D, Flatten, Dropout, Dense

# 修正的 CNN 模型
cnn_model = Sequential([
    Input(shape=(X_train_cnn.shape[1], 1)), # 指定輸入形狀
    Conv1D(64, 3, activation='relu'), # 第一層卷積
    MaxPooling1D(2), # 使用最大池化層
    Conv1D(128, 3, activation='relu'), # 第二層卷積
    MaxPooling1D(2), # 再次最大池化
    Flatten(), # 扁平化輸出
    Dropout(0.5), # Dropout 防止過擬合
    Dense(128, activation='relu'), # 全連接層
    Dropout(0.5),
    Dense(2, activation='softmax') # 輸出層
])

# 編譯模型
cnn_model.compile(optimizer='adam',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])

# 顯示模型摘要
cnn_model.summary()
```

Layer (type)	Output Shape	Param #
conv1d_2 (Conv1D)	(None, 498, 64)	256
max_pooling1d_2 (MaxPooling1D)	(None, 249, 64)	0
conv1d_3 (Conv1D)	(None, 247, 128)	24,704
max_pooling1d_3 (MaxPooling1D)	(None, 123, 128)	0
flatten_1 (Flatten)	(None, 15744)	0
dropout_2 (Dropout)	(None, 15744)	0
dense_2 (Dense)	(None, 128)	2,015,360
dropout_3 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 2)	258

Total params: 2,040,578 (7.78 MB)  
Trainable params: 2,040,578 (7.78 MB)  
Non-trainable params: 0 (0.00 B)

1. 卷積與池化層：

- 第1層卷積：64個3×1過濾器，輸出 (None, 498, 64)，參數256。
- 第1層池化：窗口大小2，輸出 (None, 249, 64)。
- 第2層卷積：128個3×64過濾器，輸出 (None, 247, 128)，參數24,704。
- 第2層池化：窗口大小2，輸出 (None, 123, 128)。

2. 展平層：

- 展平成一維，輸出 (None, 15744)。

3. 全連接層與Dropout：

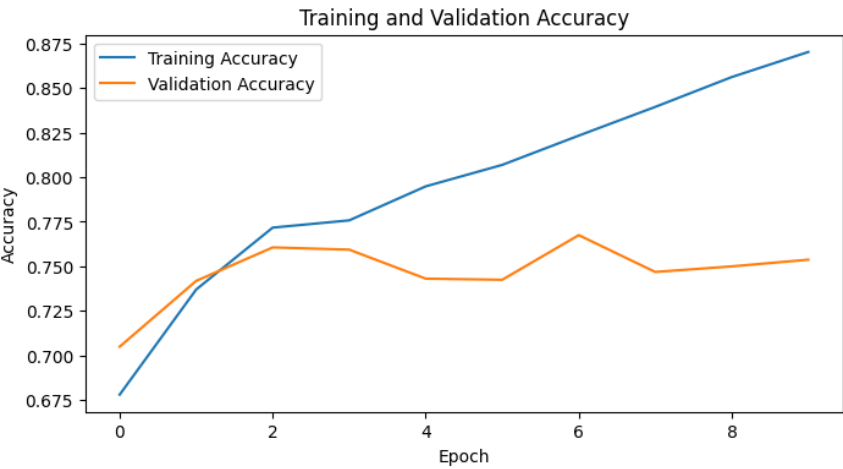
- Dense層：128個節點，參數2,015,360。
- Dropout層：丟棄部分神經元。


4. 輸出層：

- Dense層：2個節點（softmax），參數258。

總參數量：2,040,578，全為可訓練參數。

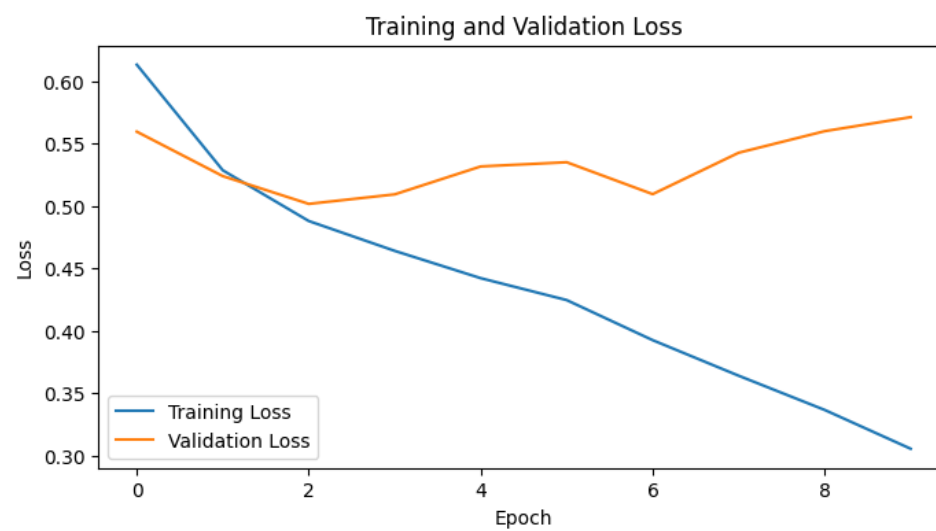
CNN 模型評估





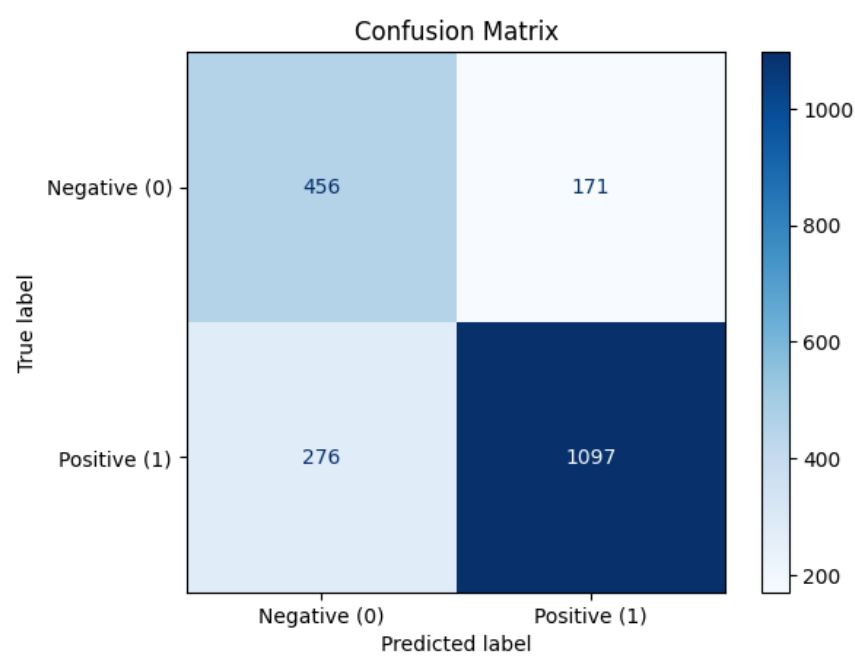
**Training and Validation Accuracy**

- 訓練準確率持續上升，從 0.67 提升至約 0.87，顯示模型在訓練集上表現良好。
- 驗證準確率在初期快速上升，但在約第 3 個 epoch 開始趨於平緩，徘徊在 0.75 左右。



## Training and Validation Loss

- 訓練損失持續下降，從 0.6 減少到 0.3，表明模型在訓練集上的學習效果不錯。
- 驗證損失在前幾個 epoch 中下降，但從第 3 個 epoch 開始呈現波動並略有上升。



## Confusion Matrix

- **True Negative (0)** 被正確分類：456
- **False Positive (1)**（實際為 0，預測為 1）：171
- **False Negative (0)**（實際為 1，預測為 0）：276
- **True Positive (1)** 被正確分類：1097
- 觀察：
  - 模型對於 Positive (1) 的分類表現較好，正確率明顯高於 Negative (0)。
  - False Negative (276) 和 False Positive (171) 的數量偏高，顯示模型在平衡兩類的分類上有改進空間。

# LSTM 建模

```
vector_length = X_train.shape[1]
X_train = X_train.reshape(-1, vector_length, 1) # LSTM 需要 3D 輸入
X_test = X_test.reshape(-1, vector_length, 1)

lstm_model = Sequential([
    Input(shape=(vector_length, 1)),
    Bidirectional(LSTM(256, return_sequences=True, activation='tanh')), # 第一層 LSTM
    BatchNormalization(),
    Dropout(0.5),
    Bidirectional(LSTM(128, return_sequences=True, activation='tanh')), # 第二層 LSTM
    BatchNormalization(),
    Dropout(0.5),
    LSTM(64, activation='tanh'), # 第三層 LSTM
    BatchNormalization(),
    Dense(128, activation='relu'), # 全連接層
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dense(2, activation='softmax') # 輸出層
])

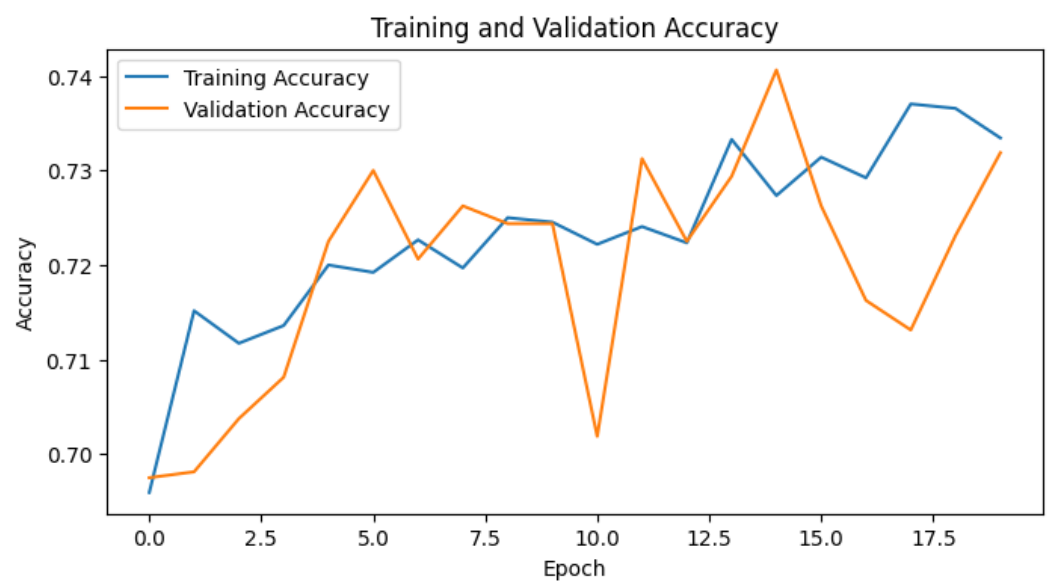
# 顯示模型摘要
lstm_model.summary()
```

Layer (type)	Output Shape	Param #
bidirectional_18 (Bidirectional)	(None, 100, 512)	528,384
batch_normalization_11 (BatchNormalization)	(None, 100, 512)	2,048
dropout_20 (Dropout)	(None, 100, 512)	0
bidirectional_19 (Bidirectional)	(None, 100, 256)	656,384
batch_normalization_12 (BatchNormalization)	(None, 100, 256)	1,024
dropout_21 (Dropout)	(None, 100, 256)	0
lstm_42 (LSTM)	(None, 64)	82,176
batch_normalization_13 (BatchNormalization)	(None, 64)	256
dense_40 (Dense)	(None, 128)	8,320
dropout_22 (Dropout)	(None, 128)	0
dense_41 (Dense)	(None, 64)	8,256
dense_42 (Dense)	(None, 2)	130
Total params: 1,286,978 (4.91 MB)		
Trainable params: 1,285,314 (4.90 MB)		
Non-trainable params: 1,664 (6.50 KB)		

1. 第一層雙向 LSTM ( **bidirectional\_18** ) :
  - 輸出形狀： (None, 100, 512) ， 參數量：528,384。
2. 批次正規化與 Dropout ( **batch\_normalization\_11** 和 **dropout\_20** ) :
  - 批次正規化參數量：2,048。
3. 第二層雙向 LSTM ( **bidirectional\_19** ) :
  - 輸出形狀： (None, 100, 256) ， 參數量：656,384。
4. 第三層單向 LSTM ( **lstm\_42** ) :
  - 輸出形狀： (None, 64) ， 參數量：82,176。
5. 全連接層與 Dropout :
  - **dense\_40** ：128 節點， 參數量：8,320。
  - **dense\_41** ：64 節點， 參數量：8,256。
  - 輸出層 **dense\_42** ：2 節點， 參數量：130。

總參數量：1,286,978， 全為可訓練參數。

# LSTM 模型評估



## Training and Validation Accuracy：

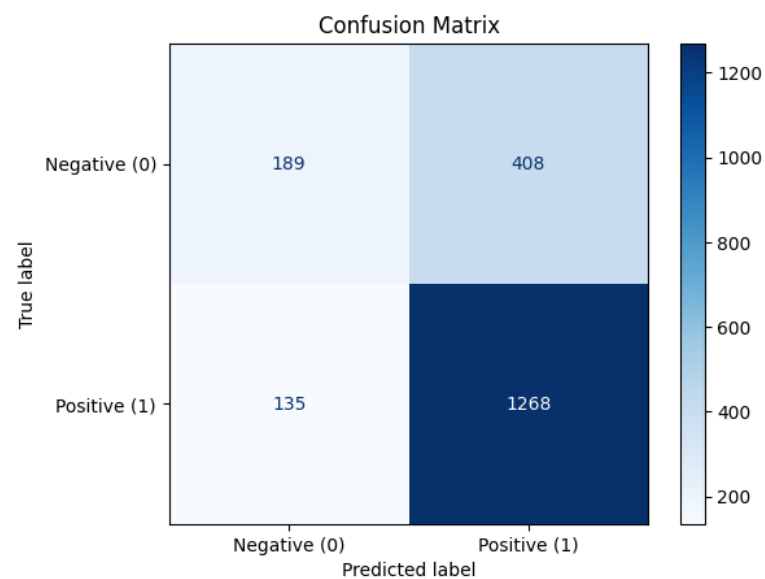
- **訓練準確率**：穩步提升，最高達到 0.74 左右。
- **驗證準確率**：起伏較大，最高接近 0.74，說明模型對驗證集的穩定性不足。
- **觀察**：訓練與驗證準確率接近，表明模型沒有明顯過擬合，但驗證準確率波動可能是數據質量或樣本不足導致。



## Training and Validation Loss：

- **訓練損失**：穩定下降，顯示模型持續學習。
- **驗證損失**：整體下降，但與準確率相符，出現波動，顯示模型對驗證集表現不穩定。
- **觀察**：損失下降與準確率的趨勢一致，但驗證損失的波動可能與驗證數據分布或模型對某些特徵的學習不足有關。





## 混淆矩陣 (Confusion Matrix)：

- **真實 Negative (0):**
  - 被正確分類：189。
  - 被錯誤分類為 Positive：408。
- **真實 Positive (1):**
  - 被正確分類：1,268。
  - 被錯誤分類為 Negative：135。
- **觀察：**
  - 模型對 Positive (1) 的分類能力較好，Recall 較高。
  - Negative (0) 的分類表現明顯較差，False Positive (408) 偏多，導致模型對類別不平衡的數據敏感。

## CNN、LSTM模型比較



### Accuracy

#### CNN：

- 訓練準確率穩定提升至 **0.87**。
- 驗證準確率停滯在 **0.75** 左右，且與訓練準確率差距明顯（約 12%）。
- **過擬合**現象較明顯，驗證集的泛化能力不足。

#### LSTM：

- 訓練準確率最高為 **0.74**，低於 CNN。
- 驗證準確率波動較大，但最高接近 **0.74**，與訓練準確率相近，表明 **過擬合問題較輕**。
- **整體穩定性不佳**，可能與數據集特性或模型結構有關。





# Loss

## CNN：

- 訓練損失穩定下降至 **0.3**，驗證損失在初期下降後出現明顯波動，後期趨於增長。
- 驗證損失與準確率不匹配，表現出模型過擬合。

## LSTM：

- 訓練損失穩定下降至 **0.54**，驗證損失整體下降但波動較大。
- 雖然損失下降，但 LSTM 模型對驗證數據的穩定性明顯不足。

指標	CNN 模型	LSTM 模型
訓練準確率	較高（0.87），學習能力強。	較低（0.74），學習能力稍弱。
驗證準確率	停滯在 0.75，且波動不大，但過擬合明顯。	接近 0.74，與訓練準確率相近，過擬合較輕。
穩定性	驗證準確率和損失波動小，但表現受限。	準確率和損失波動大，穩定性不足。
True Positive	表現良好（Recall 高，但 Precision 偏低）。	表現最佳（Recall 高，Precision 較平衡）。
True Negative	表現較差，但好於 LSTM。	表現最差，False Positive 高（408）。