智慧化企業整合

U.S. Airline. Twitter 評論分析

第三組

109034544 姜柏宏

109034545 蘇沛強

109034550 周宛昀

109034552 謝予欣

一、現況描述及問題分析

1.情境描述

出國時選擇航空公司很重要，選擇安全、服務良好的公司，是首要的目標。但是現代人生活繁忙，不可能一篇評價都有時間觀看，而且文字過長的評論也不適閱讀。

2. 問題描述-5W1H

What?	判斷評論是正/負面或中立評價
When?	想要選擇航空公司時
Who?	想出國的旅客
Where?	所有的網路平台及論壇
Why?	節省使用者自行判斷的時間
How?	Word embedding、LSTM、SSO

為了解決這些問題，我們期望藉由深度學習的模型工具，直接幫使用者判斷評論是偏向正/負面或是中立的，讓使用者判斷航空公司的優劣並做出最正確的選擇。

二、資料預處理

1.資料來源

為了建立合適的模型，我們採用 Kaggle 內的數據作為資料來源。Kaggle 是一個數據建模和數據分析競賽平台。企業和研究者可在此發布數據，提供統計學家和數據挖掘專家以此資料為基礎，進行競賽以產生最好的模型。

此次報告使用當中“U.S. airline. Twitter data”所蒐集之數據庫資料，共計 14605 筆資料，作為模型的資料來源。

2.資料概況

2.1 評價比例

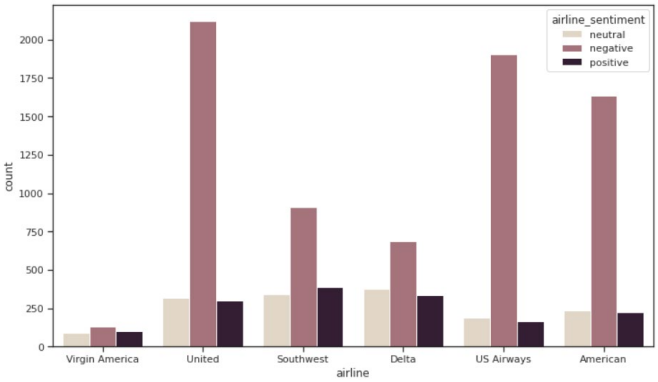
類型	占比
負面評論	62.78%
中立評論	21.10%
正面評論	16.12%

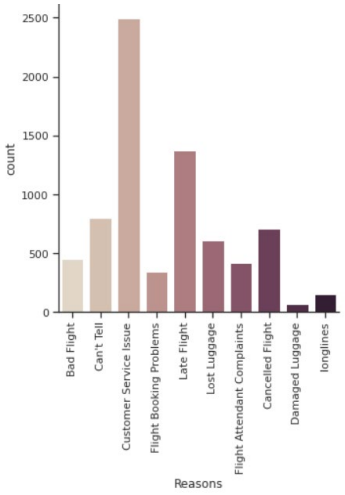
2.2 資料分割

類型	占比
Testing	47.8%
Training	27.2%
Validation	25%

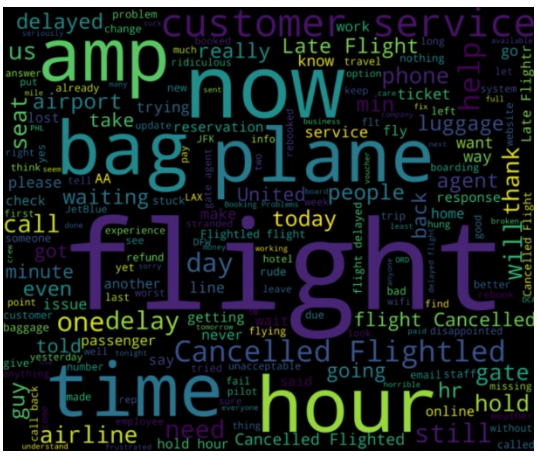
3.資料預處理過程

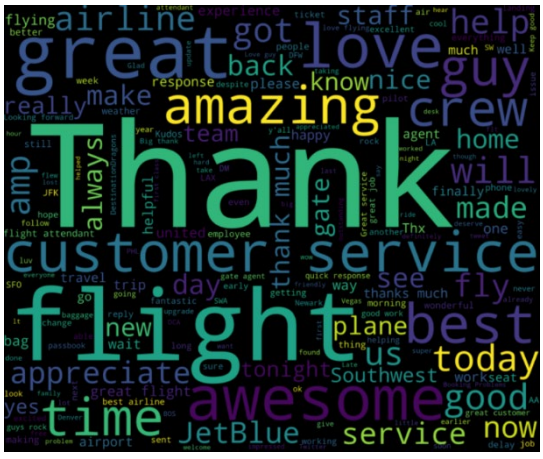
3.1 資料分析





- 資料中發現大部份推特內容都與時間管理或個人物品有關，或者向社群尋求幫助。
- 若”help”出現於推文中，則高機率為負面評論，表示航空公司某部分服務不周，造成顧客需要尋求幫助。
- 若出現”thanks”或” gratitude”則對航空公司形象有助益，表示航空公司有提供了讓顧客滿意的服務。





3.2 資料清理

- 分析出高缺失欄位並刪除

```
tweet_id                      0.000000
airline_sentiment             0.000000
airline_sentiment_confidence  0.000000
negativereason                0.373087
negativereason_confidence     0.281284
airline                       0.000000
airline_sentiment_gold        0.997268
name                          0.000000
negativereason_gold           0.997814
retweet_count                 0.000000
text                          0.000000
tweet_coord                   0.930396
tweet_created                 0.000000
tweet_location                0.323292
user_timezone                 0.329235
```

- 將評價分類信心度不足 80%的資料刪除, 刪除了資料集中的 29%資料

```
Best_confidence = Tweet[Tweet['airline_sentiment_confidence'] > .80]
print('you have dropped :'+ str((round((1 - Best_confidence.shape[0]/Tweet.shape[0]) * 100 )))+ '% of your data which has less that 80% confidence')
Tweet = Best_confidence
```

```
you have dropped :29% of your data which has less that 80% confidence
```

3.3 資料標準化

1. 將推文轉換為一個一個單詞，並刪除 stopword
   *stopword example : “the”, “a”, “an”, “in”

```
def tweet_to_words(raw_tweet):
    letters_only = re.sub("[^a-zA-Z]", " ",raw_tweet)
    words = letters_only.lower().split()
    stops = set(stopwords.words("english"))
    meaningful_words = [w for w in words if not w in stops]
    return( " ".join( meaningful_words ))
```

2. 將推文內容轉換全英文小寫

```
def clean_tweet_length(raw_tweet):
    letters_only = re.sub("[^a-zA-Z]", " ",raw_tweet)
    words = letters_only.lower().split()
    stops = set(stopwords.words("english"))
    meaningful_words = [w for w in words if not w in stops]
    return(len(meaningful_words))
```

3. 將推文性質轉換為數字('negative’ = 0, ’ neutral’ = 1, 'positive’ = 2)

```
train_df = Tweet[['text', 'airline_sentiment']]
train_df.text = Tweet.text.apply(remove_mentions)
train_df.loc[:,'sentiment'] = train_df.airline_sentiment.map({'negative':0,'neutral':1,'positive':2})
train_df = train_df.drop(['airline_sentiment'], axis=1)
```

三、模型架構

1. One-hot Encoding

使用 N 位狀態暫存器來對 N 個狀態進行編碼，每個狀態都由他獨立的暫存器位，並且在任意時候，其中只有一位有效。

	1	2	3	4	5	6	7	8
I	1	0	0	0	0	0	0	0
ate	0	1	0	0	0	0	0	0
an	0	0	1	0	0	0	0	0
apple	0	0	0	1	0	0	0	0
and	0	0	0	0	1	0	0	0
played	0	0	0	0	0	1	0	0
the	0	0	0	0	0	0	1	0
piano	0	0	0	0	0	0	0	1

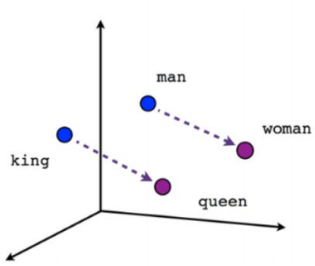
這樣做的好處主要有：

- 解決了分類器不好處理屬性資料的問題
- 在一定程度上也起到了擴充特徵的作用

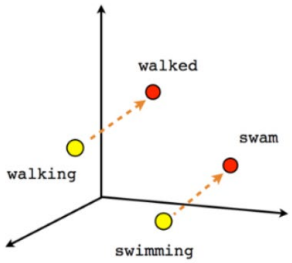
2. Word Embedding

2.1 概念

如果將 word 看作文本的最小單元，可以將 Word Embedding 理解為一種映射，其過程是：將文本空間中的某個 word，通過一定的方法，映射或者說嵌入到另一個數值向量空間，而通常這個過程維度會隨之降低。



Male-Female



Verb tense

2.2 Word Embedding 的輸入

Word Embedding 的輸入是將原始文本中的一組不重疊的詞彙，將這些詞彙放置到一個 dictionary 裡，這個 dictionary 就可以看作是 Word Embedding 的一個輸入

假設有句子：apple on a apple tree => [“apple”, “on”, “a”, “tree”]

2.3 Word Embedding 的輸出

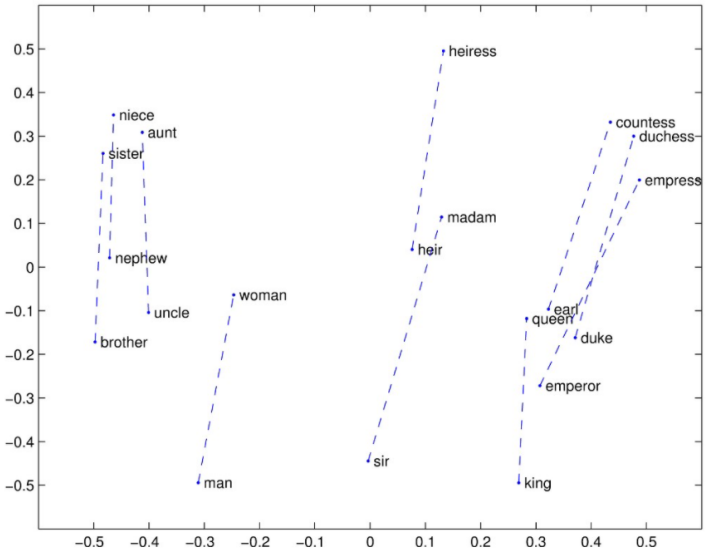
Word Embedding 的輸出就是每個 word 的向量表示。對於上文中的原始輸入，每個 word 都對應了一種數值表示。例如，apple 對應的 vector 就是[1, 0, 0, 0]，a 對應的 vector 就是[0, 0, 1, 0]，各種機器學習應用可以基於這種 word 的數值表示來構建各自的模型。

2.4 Frequency based embedding

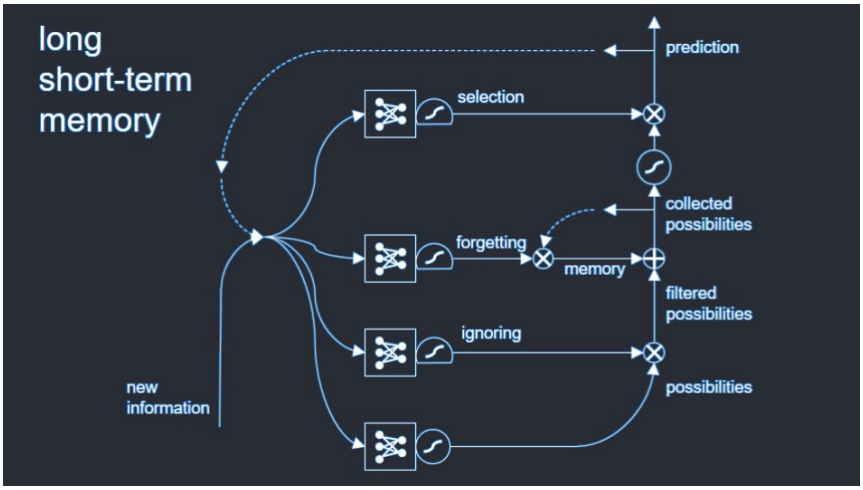
- Count Vector
- TF-IDF Vector

2.5 Predicted Based Embedding

- Word2vec
- Doc2Vec
- GloVe (Global Vectors for Word Representation)
  GloVe 的全稱叫 Global Vectors for Word Representation，它是一個基於全局詞頻統計（count-based & overall statistics）的詞表徵（word representation）工具，它可以把一個單詞表達成一個由實數組成的向量，這些向量捕捉到了單詞之間一些語義特性，比如相似性（similarity）、類比性（analogy）等。我們通過對向量的運算，比如歐幾里得距離或者 cosine 相似度，可以計算出兩個單詞之間的語義相似性。



2.6 LSTM (long short-term memory)



2.7 Simplified Swarm Optimization(SSO)

參數 Cg、Cp 和 Cw 是預先設定好的，這當中，Cg、Cp 和 Cw 的值分別由 cg、cg+cp 和 cg +cp+cw 累加而成，Xijt 代表在第 t 迭代中的粒子 i，ρijt 是介於 [0,1] 之間均勻分佈的隨機數，x 則是變數 j 的隨機值。

$$x_{ij}^{t} = \begin{cases} x_{ij}^{t} & \text{if } \rho_{ij}^{t} \in [0, C_w) \\ p_{ij} & \text{if } \rho_{ij}^{t} \in [C_w, C_p) \\ g_i & \text{if } \rho_{ij}^{t} \in [C_p, C_g) \\ x & \text{if } \rho_{ij}^{t} \in [C_g, 1) \end{cases}$$

如果 0≤ρijt < Cg 成立，那麼該欄位的值將會被 gbest 相對應欄位的值所取代。

如果 Cg≤ρijt < Cp 成立，那麼該欄位的值將會被 pbest 相對應欄位的值所取代。

如果 Cp≤ρijt < Cw 成立，那麼該欄位的值將會被保留下來。

如果 Cw≤ρijt < 1 成立，那麼該欄位的值將會被隨機值 x 所取代，其中這一隨機值 x 可以幫助解在更新時，增加粒子的變異性，避免解困在區域解裡。

四、模型設定

1.定義 SSO 模型

初始化 SSO，利用 SSO 來選擇參數配置，初始化:

1. LSTM 層數
2. LSTM 層節點數
3. 全連接層層數
4. 全連接層節點數
5. dropout 比例
6. 激勵函數選擇
7. 訓練批量

```
def  SSO_Init():
        sol_list  =  []
        sol_list.append(np.random.randint(1,  4))
        sol_list.append(np.random.randint(64,  256))
        sol_list.append(np.random.randint(1,  4))
        sol_list.append(np.random.randint(64,  256))
        sol_list.append(np.random.uniform(0.4, 0.8))
        sol_list.append("softmax")
        sol_list.append(np.random.randint(16,  128))

        return  sol_list
init_best_dict  =  SSO_Init()
```

建立 best_dict 值，紀錄 gbest, pbest 與其 fitness 值

```
# 建立best_dict 紀錄gbest, pbest 與其fitness值
best_dict  =  {
        'gbest_sol':  init_best_dict,
        'gbest_fitness':  0,

        'pbest_LSTM_layers':  [0,  0,  0,  0,  0,  0],
        'pbest_LSTM_layers_fitness':  [0,  0,  0,  0,  0,  0],

        'pbest_LSTM_units':  [0,  0,  0,  0,  0,  0],
        'pbest_LSTM_units_fitness':  [0,  0,  0,  0,  0,  0],

        'pbest_Dense_layers':  [0,  0,  0,  0,  0,  0],
        'pbest_Dense_layers_fitness':  [0,  0,  0,  0,  0,  0],

        'pbest_Dense_nurons':  [0,  0,  0,  0,  0,  0],
        'pbest_Dense_nurons_fitness':  [0,  0,  0,  0,  0,  0],

        'pbest_Drop_rate':  [0,  0,  0,  0,  0,  0],
        'pbest_Drop_rate_fitness':  [0,  0,  0,  0,  0,  0],

        'pbest_batch':  [0,  0,  0,  0,  0,  0],
        'pbest_batch_fitness':  [0,  0,  0,  0,  0,  0],

}
best_dict
```

2.定義 SSO 更新機制

更新 SSO，透過亂數選擇下一個解打算用的參數，若之前使用過較好的參數(準確度較高)，有較高機率會選擇到該參數來使用。

```
def SSO_UPDATE(sol, sol_list, best_dict):

    # LSTM_layers
    rnd = np.random.rand(1)
    if rnd < 0.3:
        sol_list[0] = best_dict['gbest_sol'][0]
    elif rnd < 0.5:
        sol_list[0] = best_dict['pbest_LSTM_layers'][sol]
    else:
        sol_list[0] = np.random.randint(1, 4)

    # LSTM_units
    rnd = np.random.rand(1)
    if rnd < 0.3:
        sol_list[1] = best_dict['gbest_sol'][1]
    elif rnd < 0.5:
        sol_list[1] = best_dict['pbest_LSTM_units'][sol]
    else:
        sol_list[1] = np.random.randint(64, 256)

    # Dense_layers
    rnd = np.random.rand(1)
    if rnd < 0.3:
        sol_list[2] = best_dict['gbest_sol'][2]
    elif rnd < 0.5:
        sol_list[2] = best_dict['pbest_Dense_layers'][sol]
    else:
        sol_list[2] = np.random.randint(1, 4)
```

```
    # Dense_nurons
    rnd = np.random.rand(1)
    if rnd < 0.3:
        sol_list[3] = best_dict['gbest_sol'][3]
    elif rnd < 0.5:
        sol_list[3] = best_dict['pbest_Dense_nurons'][sol]
    else:
        sol_list[3] = np.random.randint(64, 256)

    # DropOut_rate
    rnd = np.random.rand(1)
    if rnd < 0.3:
        sol_list[4] = best_dict['gbest_sol'][4]
    elif rnd < 0.5:
        sol_list[4] = best_dict['pbest_Drop_rate'][sol]
    else:
        sol_list[4] = np.random.uniform(0.4,0.8)

    #   Batch_size
    rnd = np.random.rand(1)
    if rnd < 0.3:
        sol_list[6] = best_dict['gbest_sol'][6]
    elif rnd < 0.5:
        sol_list[6] = best_dict['pbest_batch'][sol]
    else:
        sol_list[6] = np.random.randint(16, 128)

    return sol_list
```

3.定義 SSO gbest, pbest 更新機制

```
def best_UPDATE(sol, sol_fitness, sol_list, best_dict):

    if sol_fitness >= best_dict['gbest_fitness']:
        best_dict['gbest_sol'] = sol_list
        best_dict['gbest_fitness'] = sol_fitness

    if sol_fitness >= best_dict['pbest_LSTM_layers_fitness'][sol]:
        best_dict['pbest_LSTM_layers_fitness'][sol] = sol_fitness
        best_dict['pbest_LSTM_layers'][sol] = sol_list[0]

    if sol_fitness >= best_dict['pbest_LSTM_units_fitness'][sol]:
        best_dict['pbest_LSTM_units_fitness'][sol] = sol_fitness
        best_dict['pbest_LSTM_units'][sol] = sol_list[1]

    if sol_fitness >= best_dict['pbest_Dense_layers_fitness'][sol]:
        best_dict['pbest_Dense_layers_fitness'][sol] = sol_fitness
        best_dict['pbest_Dense_layers'][sol] = sol_list[2]
```

```
    if sol_fitness >= best_dict['pbest_Dense_nurons_fitness'][sol]:
        best_dict['pbest_Dense_nurons_fitness'][sol] = sol_fitness
        best_dict['pbest_Dense_nurons'][sol] = sol_list[3]

    if sol_fitness >= best_dict['pbest_Drop_rate_fitness'][sol]:
        best_dict['pbest_Drop_rate_fitness'][sol] = sol_fitness
        best_dict['pbest_Drop_rate'][sol] = sol_list[4]

    if sol_fitness >= best_dict['pbest_batch_fitness'][sol]:
        best_dict['pbest_batch_fitness'][sol] = sol_fitness
        best_dict['pbest_batch'][sol] = sol_list[6]

    return best_dict
```

4.定義 LSTM 模型

利用前面 SSO 選擇的參數建立 LSTM 模型，使用到 sol[0](LSTM 層數)、sol[4](斷開比例)、sol[2](全連接層層數)、sol[3]全連接層節點數，並將其組合及準確度記錄下來，選出一組最好的組合進行正式訓練。

```
def ltsm_model(input_shape, word_to_vec_map, word_to_index, sol):

    sentence_indices = Input(shape=input_shape, dtype='int32')
    embedding_layer = pretrained_embedding_layer(word_to_vec_map,
                                                 word_to_index)
    embeddings = embedding_layer(sentence_indices)

    for layer in range(sol[0]):
        X = LSTM(sol[1], return_sequences=True)(embeddings)

    X = LSTM(sol[1], return_sequences=False)(X)
    X = Dropout(sol[4])(X)

    for layer in range(sol[2]):
        X = Dense(sol[3], activation=None)(X)

    X = Dropout(sol[4])(X)
    X = Dense(3, activation=None)(X)
    X = Activation('softmax')(X)

    model = Model(inputs=[sentence_indices], outputs=X)

return model
```

5.定義訓練評估模型

```
from keras.optimizers import Adam,Adagrad,RMSprop,Adadelta
adadelta = Adadelta(lr=0.1, rho=0.95 ,epsilon=None, decay=0.0)

def train_evaluate(model, sol_list):
        model.compile(loss='categorical_crossentropy', optimizer=adadelta,
                                metrics=['accuracy'])
        model.fit(X_train_indices, y=Y_oh_train, batch_size=sol_list[6],
                        epochs=100, verbose=1,
                        validation_data=(X_vaild_indices, Y_oh_vaild))

        # Plot train/test loss and accuracy
        acc = hist_1.history['accuracy']
        val_acc = hist_1.history['val_accuracy']
        loss = hist_1.history['loss']
        val_loss = hist_1.history['val_loss']
```

五、訓練及測試

1.開始執行 SSO

```
# Initial solution list with random solution
print("Initialization")
for i in range(Nsol):
        sol_list = SSO_Init()
        sso_info.append(sol_list)
        print(f"Initialize sol_list{i}:{sol_list}")
        model = ltsm_model((maxLen,), word_to_vec_map, word_to_index, sol_list)
        sol_fitness = train_evaluate(model, sol_list)
        best_dict = best_UPDATE(i, sol_fitness, sol_list, best_dict)
        print(f'best_dict of initialization: {best_dict}')

# Start SSO
for gen in range(Ngen):
        print(f'============== Generation_{gen} ==============')
        for j in range(Nsol):
                sol_list = SSO_Init()
                sol_list = SSO_UPDATE(j, sol_list, best_dict)
                sso_info.append(sol_list)
                print(f"sol_{j} sol_list: {sol_list}")
                model = ltsm_model((maxLen,), word_to_vec_map, word_to_index, sol_list)
                sol_fitness = train_evaluate(model, sol_list)
                best_dict = best_UPDATE(j, sol_fitness, sol_list, best_dict)
                fitness_info = [sol_fitness]
                plot_info.append(fitness_info)
                print(plot_info)
        print('Global best fitness: {}'.format(best_dict['gbest_fitness']))
        print('Global best solution: {}'.format(best_dict['gbest_sol']))
```

2.選擇超參數

LSTM層數	2
LSTM層節點數	107
全連接層層數	1
全連接層節點數	204
dropout比例	0.4554
激勵函數選擇	softmax
訓練批量	117
accuracy	80.30%

LSTM層數	2
LSTM層節點數	129
全連接層層數	3
全連接層節點數	233
dropout比例	0.7961
激勵函數選擇	softmax
訓練批量	101
accuracy	80.15%

LSTM層數	2
LSTM層節點數	171
全連接層層數	2
全連接層節點數	150
dropout比例	0.5772
激勵函數選擇	softmax
訓練批量	97
accuracy	80.38%

LSTM層數	2
LSTM層節點數	129
全連接層層數	1
全連接層節點數	187
dropout比例	0.5455
激勵函數選擇	softmax
訓練批量	83
accuracy	80.92%

LSTM層數	1
LSTM層節點數	89
全連接層層數	2
全連接層節點數	118
dropout比例	0.4696
激勵函數選擇	softmax
訓練批量	86
accuracy	80.80%

LSTM層數	2
LSTM層節點數	129
全連接層層數	3
全連接層節點數	250
dropout比例	0.4184
激勵函數選擇	softmax
訓練批量	83
accuracy	80.19%

LSTM層數	2
LSTM層節點數	226
全連接層層數	2
全連接層節點數	204
dropout比例	0.6982
激勵函數選擇	softmax
訓練批量	83
accuracy	79.85%

LSTM層數	2
LSTM層節點數	191
全連接層層數	3
全連接層節點數	198
dropout比例	0.7053
激勵函數選擇	softmax
訓練批量	34
accuracy	79.96%

LSTM層數	***2***
LSTM層節點數	***129***
全連接層層數	***1***
全連接層節點數	***68***
dropout比例	***0.4184***
激勵函數選擇	***softmax***
訓練批量	***71***
accuracy	***82.83%***

LSTM層數	3
LSTM層節點數	238
全連接層層數	1
全連接層節點數	80
dropout比例	0.4184
激勵函數選擇	softmax
訓練批量	48
accuracy	82.72%

LSTM層數	2
LSTM層節點數	129
全連接層層數	1
全連接層節點數	127
dropout比例	0.4353
激勵函數選擇	softmax
訓練批量	71
accuracy	80.76%

LSTM層數	1
LSTM層節點數	76
全連接層層數	1
全連接層節點數	80
dropout比例	0.4184
激勵函數選擇	softmax
訓練批量	48
accuracy	81.11%

LSTM層數	1
LSTM層節點數	129
全連接層層數	2
全連接層節點數	189
dropout比例	0.4184
激勵函數選擇	softmax
訓練批量	34
accuracy	80.27%

LSTM層數	2
LSTM層節點數	94
全連接層層數	1
全連接層節點數	80
dropout比例	0.6025
激勵函數選擇	softmax
訓練批量	32
accuracy	81.41%

LSTM層數	1
LSTM層節點數	190
全連接層層數	1
全連接層節點數	68
dropout比例	0.4184
激勵函數選擇	softmax
訓練批量	71
accuracy	80.23%

LSTM層數	3
LSTM層節點數	216
全連接層層數	1
全連接層節點數	80
dropout比例	0.7510
激勵函數選擇	softmax
訓練批量	48
accuracy	81.41%

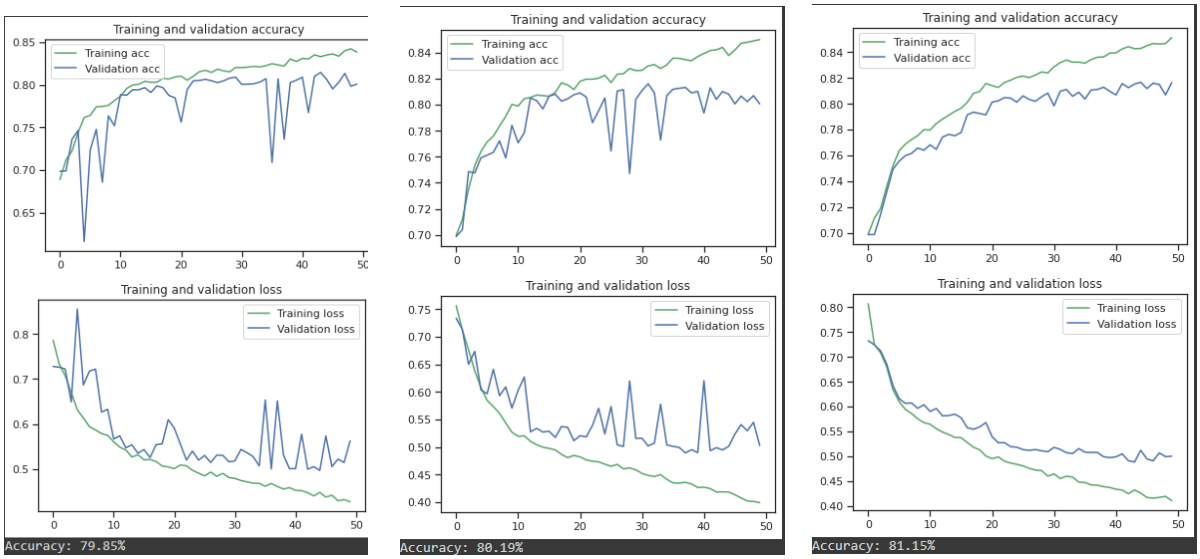
LSTM層數	1
LSTM層節點數	129
全連接層層數	3
全連接層節點數	200
dropout比例	0.4184
激勵函數選擇	softmax
訓練批量	71
accuracy	81.76%

LSTM層數	2
LSTM層節點數	221
全連接層層數	1
全連接層節點數	213
dropout比例	0.4184
激勵函數選擇	softmax
訓練批量	71
accuracy	80.19%

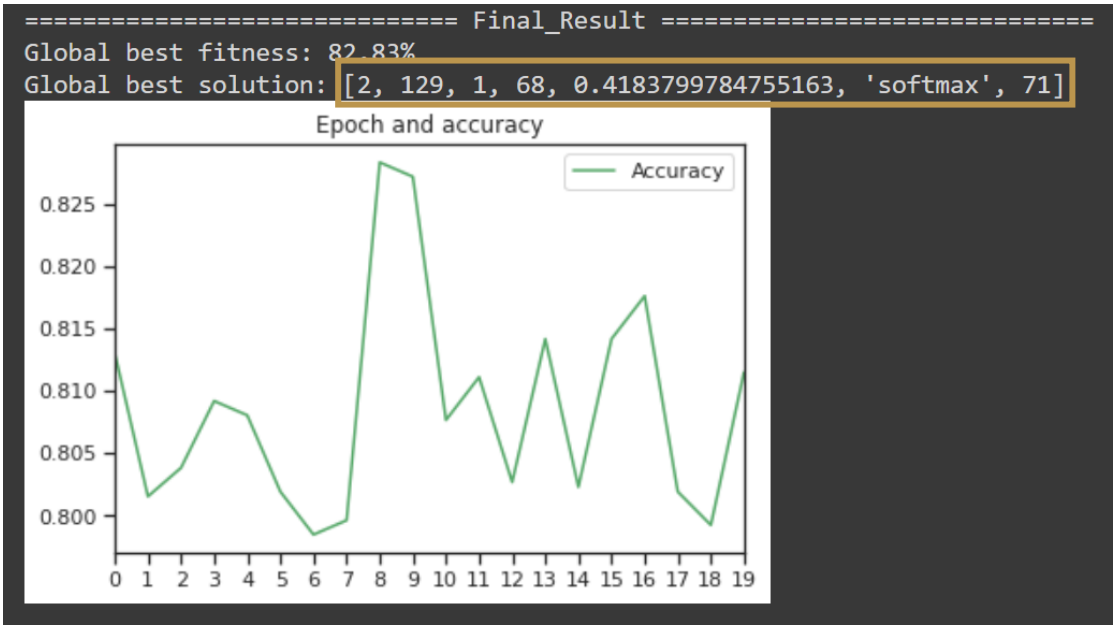
LSTM層數	2
LSTM層節點數	129
全連接層層數	1
全連接層節點數	241
dropout比例	0.7973
激勵函數選擇	softmax
訓練批量	71
accuracy	79.92%

LSTM層數	2
LSTM層節點數	64
全連接層層數	1
全連接層節點數	213
dropout比例	0.4184
激勵函數選擇	softmax
訓練批量	71
accuracy	81.15%

選擇的過程。



我們使用 SSO 隨機生成了 20 組超參數，最終編號第九組的超參數之 accuracy 最高，故我們選擇該組超參數放入 LSTM 模型訓練。



2. LSTM 模型訓練

```
# model = LSTM_model(best_dict['gbest_sol'])
model = ltsm_model((maxLen,), word_to_vec_map, word_to_index, best_dict['gbest_sol'])
model.compile(loss='categorical_crossentropy', optimizer=adadelta , metrics=['accuracy'])
model.fit(X_train_indices, y=Y_oh_train, batch_size=best_dict['gbest_sol'][6],
                  epochs=100, verbose=1, validation_data=(X_vaild_indices, Y_oh_vaild))
# Final evaluation of the model
scores = model.evaluate(X_test_indices, Y_oh_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Epoch 97/100
71/71 [==============================] - 1s 9ms/step - loss: 0.2797 - accuracy: 0.9002 - val_loss: 0.6347 - val_accuracy: 0.7887
Epoch 98/100
71/71 [==============================] - 1s 9ms/step - loss: 0.2816 - accuracy: 0.8984 - val_loss: 0.5315 - val_accuracy: 0.8143
Epoch 99/100
71/71 [==============================] - 1s 9ms/step - loss: 0.2748 - accuracy: 0.8962 - val_loss: 0.5056 - val_accuracy: 0.8235
Epoch 100/100
71/71 [==============================] - 1s 10ms/step - loss: 0.2730 - accuracy: 0.9056 - val_loss: 0.5341 - val_accuracy: 0.8231
Accuracy: 82.98%
```

放入 100 筆資料訓練後，最終 Accuracy 為 82.98%。

六、結論與未來展望

1.定義推論模型

```
def TestModel(stringList):
        classess = ['negative','neutral','positive']
        results=[]
        if len(stringList) == 1:
                x_test = np.array(stringList)
                X_test_indices = sentences_to_indices(x_test, word_to_index, maxLen)
                index = np.argmax(model.predict(X_test_indices))
                return x_test[0] +' is a '+ classess[index]+' word'
        for i,x in enumerate(stringList):
                x = [x]
                x_test = np.array(x)
                X_test_indices = sentences_to_indices(x_test, word_to_index, maxLen)
                index = np.argmax(model.predict(X_test_indices))
                results.append(x_test[0] +' is a '+ classess[index]+' word')
        return results
```

隨機輸入一句話，讓模型判斷是正面\負面還是中立的評論。

2.推論結果

```
TestModel(['Nice AirPlance Service I love Them ', # Positive
           'bad AirPlance', # negative
           'What The Fuckk Plane' , # negative
           'my Flight is so late and its about 1 hour why ?', # negative
           'okay its fine flight , thank you ',# Postive
           'I lost my Bag and and They Back it to me So Thanks ', # Negative
           'okay' # neutral
          ])
```

```
['Nice AirPlance Service I love Them  is a positive word',
 'bad AirPlance is a negative word',
 'What The Fuckk Plane is a neutral word',
 'my Flight is so late and its about 1 hour why ? is a negative word',
 'okay its fine flight , thank you  is a positive word',
 'I lost my Bag and and They Back it to me So Thanks  is a negative word',
 'okay is a negative word']
```

3.未來方向

若未來仍要朝這個方向繼續深入探討，我們提供了三個方向可做為研究方向。首先是新增資料，添加更多複雜度較高的文句，以提升模型彈性。也可藉由持續新增資料來訓練模型，進而提升其準確度。第二個是判斷不同語言，目前此模型只能判斷英文，期望未來能辨識不同語言之評論，讓模型的適用性更廣泛。第三個是增加模型的應用層面，例如可以加入至機器人，使機器人在辨識完正/負面評價後可以做出相對的反應，使機器人做出更加合適的反應，更加智能化。