智慧化企業整合

利用 NLP 分析句子情感

第 4 組

110034568 陳彥碩

指導教授：邱銘傳教授

目錄

一、 背景介紹.....2

1.1 背景介紹.....2
1.2 問題定義 5W1H.....2

二、 模型訓練與績效.....3

2.1 資料蒐集.....3
2.2 資料前處理.....3

2.2.1 資料清理與整理.....3
2.2.2 資料分類.....6

2.3 模型架構.....7
2.4 模型績效比較.....9

三、 結論及未來展望.....12

4.1 結論.....12
4.2 未來展望.....12

一、 背景介紹

1.1 背景介紹

當我們想要了解一些時事，或者是想要了解某張股票現在的行情、商品的評論時一定會參考上網先做工作，會在留言板上看到許多人對這件事情或是商品的看法，這些訊息往往會成為我們了解事情的一個關鍵訊息。但往往這些訊息的資訊量通常不一定是正確的，有些網友會用看似正常的言論去批評一件事情，往往這樣的留言會導致不確定網友的評價到底是正面的意思還是反面的意思。

為了避免吸收不正當的訊息導致錯誤的決策，因此便想利用最近火紅的方法，自然語言處理(Natural Language Processing，NLP)來分析知名網路平台推特(twitter)的留言情緒，判斷留言是正面還是反面的意思。

1.2 問題定義 5W1H

進行本研究前，先以 5W1H 針對我們的問題定義進行分析，以更了解問題的本質。

Table 1 5W1H 分析

Who	想了解事情並上網查看評論的人。
What	正確判別文字所要表達的意思。
Why	避免誤解文字意思，造成錯誤的決策。
When	任何時刻。
Where	任何地點。
How	利用循環神經網路(RNN)和卷積神經網路(CNN)。

二、 模型訓練與績效

2.1 資料蒐集

本研究的資料來源來自 Kaggle 網站上的 Sentiment140 dataset，共包含了 160 萬筆的推特留言。每筆資料包含了正確的情感(0 代表負面的意思，4 代表正面的意思)、使用者 id、發文日期、使用者名稱以及留言的文字，資料如下圖所示：

	target	ids	date	user	text
2	0	1467810369	Mon Apr 06 22:19:45	_TheSpecialOne_	@switchfoot http://twit
3	0	1467810672	Mon Apr 06 22:19:49	scotthamilton	is upset that he can't u
4	0	1467810917	Mon Apr 06 22:19:53	mattycus	@Kenichan I dived ma
5	0	1467811184	Mon Apr 06 22:19:57	ElleCTF	my whole body feels it
6	0	1467811193	Mon Apr 06 22:19:57	Karoli	@nationwideclass no,
7	0	1467811372	Mon Apr 06 22:20:00	joy_wolf	@Kwesidei not the wh
8	0	1467811592	Mon Apr 06 22:20:03	mybirch	Need a hug
9	0	1467811594	Mon Apr 06 22:20:03	coZZ	@LOLTrish hey long
10	0	1467811795	Mon Apr 06 22:20:05	2Hood4Hollywood	@Tatiana_K nope they
11	0	1467812025	Mon Apr 06 22:20:09	mimismo	@twittera que me mue
12	0	1467812416	Mon Apr 06 22:20:16	erinx3leannexo	spring break in plain c
13	0	1467812579	Mon Apr 06 22:20:17	pardonlauren	I just re-pierced my ea
14	0	1467812723	Mon Apr 06 22:20:19	TLeC	@caregiving I couldn't
15	0	1467812771	Mon Apr 06 22:20:19	robrobbierobert	@octolinz16 It it coun
16	0	1467812784	Mon Apr 06 22:20:20	bayofwolves	@smarrison i would've
17	0	1467812799	Mon Apr 06 22:20:20	HairByJess	@iamjazzyfizzle I wish
18	0	1467812964	Mon Apr 06 22:20:22	lovesongwriter	Hollis' death scene wil
19	0	1467813137	Mon Apr 06 22:20:25	armotley	about to file taxes
20	0	1467813579	Mon Apr 06 22:20:31	starkissed	@LettyA ahh ive alwa

Figure 1 原始資料型態

2.2 資料前處理

2.2.1 資料清理與整理

由於我們的目標是要判斷文字的情感，因此首先先去除掉不需要的資料欄位，也就是使用者 id、發文日期、使用者名稱，只保留剩下的情感標籤以及文字，使用的程式如下圖所示：

```
df = pd.read_csv('C:/Users/Henry-Lab/Desktop/IIE_project 3/training.1600000.processed.noemoticon.csv',
                 encoding = 'latin', header=None)
# print(df.head())

df.columns = ['sentiment', 'id', 'date', 'user_id', 'text']
df = df.drop(['id', 'date', 'user_id'], axis=1)
df.to_csv("data.csv", encoding = "utf-8")
```

Figure 2 去除不必要的資料

資料清除完後，將原始資料欄位的情感標籤由 0、4 轉為 Negative 以及 Positive 方便判讀，處理的程式以及處理完後的資料如下圖所示：

```
lab_to_sentiment = {0:"Negative", 4:"Positive"}
def label_decoder(label):
    return lab_to_sentiment[label]
df.sentiment = df.sentiment.apply(lambda x: label_decoder(x))
#print(df.head())

df.to_csv("data1.csv", encoding = "utf-8")
```

Figure 2 更改情感標籤

1		sentiment	text
2	0	Negative	@switchfoot http://twit
3	1	Negative	is upset that he can't up
4	2	Negative	@Kenichan I dived ma
5	3	Negative	my whole body feels it
6	4	Negative	@nationwideclass no,
7	5	Negative	@Kwesidei not the wh
8	6	Negative	Need a hug
9	7	Negative	@LOLTrish hey long
10	8	Negative	@Tatiana_K nope they
11	9	Negative	@twittera que me mue
12	10	Negative	spring break in plain c
13	11	Negative	I just re-pierced my ea
14	12	Negative	@caregiving I couldn't
15	13	Negative	@octolinz16 It it coun
16	14	Negative	@smarrison i would've
17	15	Negative	@iamjazzyfizzle I wish
18	16	Negative	Hollis' death scene wil
19	17	Negative	about to file taxes
20	18	Negative	@LettyA ahh ive alwa

Figure 4 更改後的資料

為了要將文字餵入神經網路以供學習，需先將資料轉換成電腦能夠判讀的樣子，因此將每筆資料的停用詞清除掉，也就是清除掉每筆資料的 a、the、is 等等跟判斷情緒無關的單字。

因為英文單字會有時態問題，像是有 plays、played、playing 等等，雖然長相不一樣但所表達的意思是一樣的，因此會了簡化模型的訓練，只提取資料中的詞幹，也就是將上述三個單字全部轉換成 play。

再來，因為資料是來自推特上的留言，因此有許多留言會利用@來標記其他網友，或是會有超連結的問題，而這些資料是無法供給我們任何判斷情緒的資訊的，因此也需要將這些資料清除掉。

解決上述問題的方法本研究主要是利用 nltk 這個自然語言處理工具包，這個套件可以解決許多自然語言處理的資料前處理，下圖為前處理的程式：

```
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer

stop_words = stopwords.words('english')
stemmer = SnowballStemmer('english')
text_cleaning_re = "@\S+|https?:\S+|http?:\S|[^A-Za-z0-9]+"

def preprocess(text, stem=False):
    text = re.sub(text_cleaning_re, ' ', str(text).lower()).strip()
    tokens = []
    for token in text.split():
        if token not in stop_words:
            if stem:
                tokens.append(stemmer.stem(token))
            else:
                tokens.append(token)
    return " ".join(tokens)

df.text = df.text.apply(lambda x: preprocess(x))
```

Figure 5 資料前處理

經過上圖程式的處理，資料已經可供電腦判讀了，再來就是要將資料再轉成可供神經網路判讀的樣子，也就是將資料向量化。

要將資料向量化首先必須先將每筆資料轉換成 token，token 為資料分解成的小單元，而

這一分解成 token 過程稱為分詞或斷詞(tokenization)。下圖為資料前處理後轉成 token 的例子。

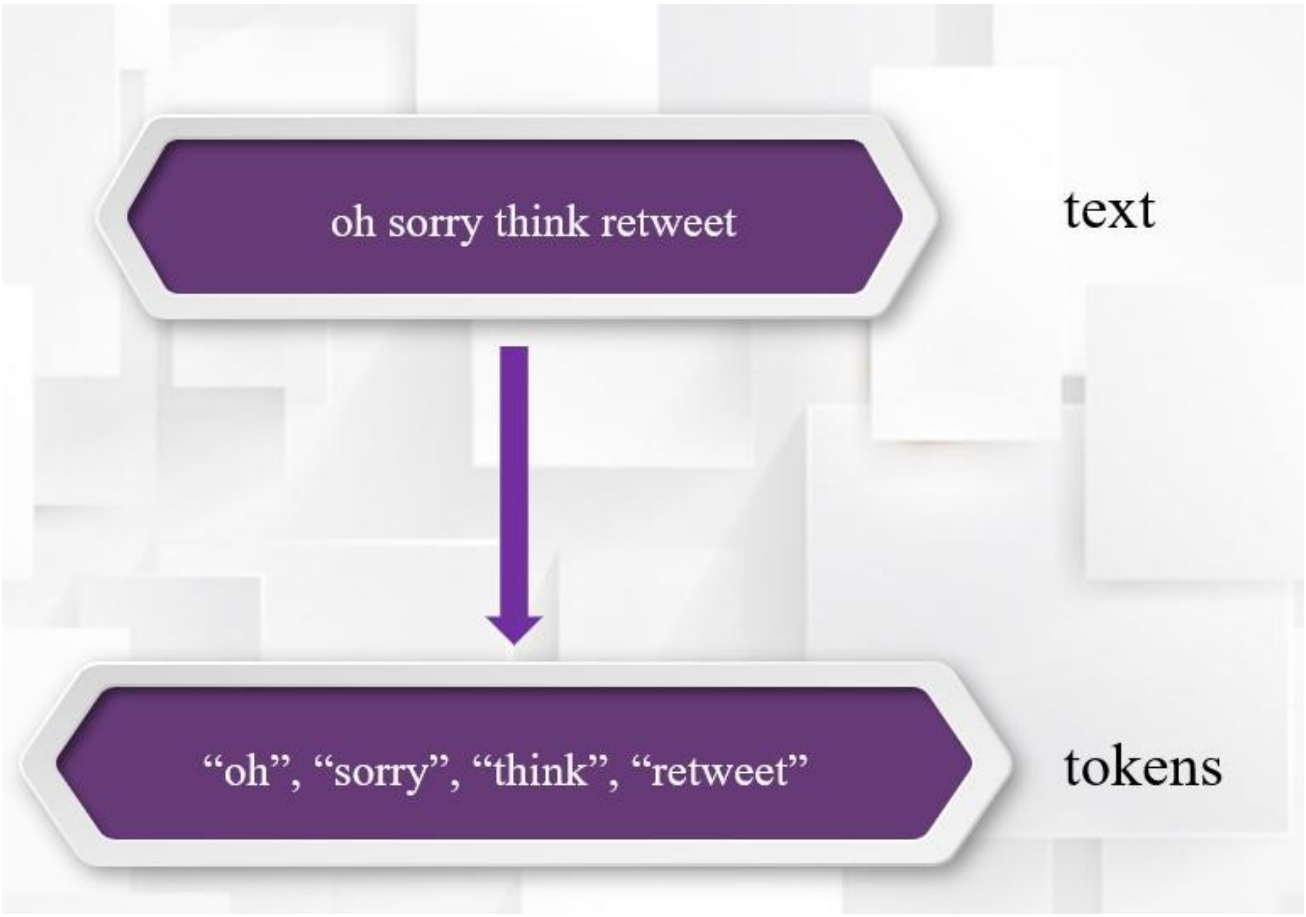

Figure 6 token 範例

將資料轉換成 token 後，須將每一 token 向量化，但本研究的資料筆數共有 160 萬筆，轉換成的 token 數共有 290575 個，要將全部轉成 one-hot-encoding 會非常吃資源。因此，本研究利用史丹佛 GloVe 團隊所開發的文字嵌入法，能夠直接將 token 轉成向量並且不吃資源，本研究主要是利用 glove.6B.300d 來進行向量化。

2.2.2 資料分類

經過一連串繁複的資料前處理，現在已經能夠將資料餵給神經網路學習了，因此將資料分成訓練集、驗證集以及測試集來進行訓練，所選擇的比例為訓練集 6 成，驗證集和測試集皆為 2 成。

2.3 模型架構

本研究為了避免出現 overfitting 的問題，因此利用一些進階方法來訓練模型，讓模型的準確率可以更高、訓練時間更少、不會 overfitting。

首先，利用循環丟棄(recurrent dropout)的技術，利用此方法可以解決循環層中 overfitting 的問題。再來利用堆疊循環層(stacking recurrent layers)的技術來增加神經網路的學習效率，缺點為會花較多的運算成本。然後利用雙向循環層(Bidirectional recurrent layers)的技術以不同方式向循環神經網路提供網路資訊，進而提高準確率並減少神經網路的遺忘問題。程式如下圖所示：

```
from keras.layers import Bidirectional, LSTM
x = Bidirectional(LSTM(64,
                       dropout = 0.2,
                       recurrent_dropout = 0.2,
                       return_sequences = True))(x)
```

Figure 7 模型技巧

除了上述方法，本研究還利用了遞減學習率的概念，也就是若經過一段次數後驗證損失並沒有降低，則降低學習率讓模型可以更有效的去訓練，使用的程式如下圖所示：

```
ReduceLROnPlateau = ReduceLROnPlateau(factor=0.1,
                                      min_lr = 0.01,
                                      patience = 3,
                                      monitor = 'val_loss',
                                      verbose = 1)
```

Figure 8 遞減學習率

本研究使用的 RNN 模型如下圖所示：

```
sequence_input = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32')
embedding_sequences = embedding_layer(sequence_input)
x = SpatialDropout1D(0.2)(embedding_sequences)
x = Bidirectional(LSTM(64, dropout=0.2, recurrent_dropout=0.2, return_sequences = True))(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(512, activation='relu')(x)
x = Dense(256, activation = 'relu')(x)
outputs = Dense(1, activation='sigmoid')(x)
model = tf.keras.Model(sequence_input, outputs)
```

Figure 9 RNN 模型架構

除了利用 RNN 來訓練模型外，本研究還利用卷積神經網路來訓練模型，使用 CNN 中的 Conv1D。Conv1D 與 RNN 相比，雖然 Conv1D 的驗證準確率較低，但可擁有較快的運算時間，當訓練所需資源很大時可以 Conv1D。Conv1D 的程式如下圖所示：

```
sequence_input = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32')
embedding_sequences = embedding_layer(sequence_input)
x = SpatialDropout1D(0.2)(embedding_sequences)
x = Conv1D(64, 5, activation='relu')(x)
x = MaxPool1D()(x)
x = Conv1D(64, 5, activation = 'relu')(x)
x = GlobalMaxPool1D()(x)
outputs = Dense(1, activation='sigmoid')(x)
model = tf.keras.Model(sequence_input, outputs)
```

Figure 10 CNN 模型架構

本研究還嘗試了結合上述兩者的模型，也就是結合 RNN 與 CNN 來訓練模型，本研究所採用的模型程式如下圖所示：

```
sequence_input = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32')
embedding_sequences = embedding_layer(sequence_input)
x = SpatialDropout1D(0.2)(embedding_sequences)
x = Conv1D(64, 5, activation='relu')(x)
x = MaxPool1D()(x)
x = Conv1D(64, 5, activation = 'relu')(x)
x = MaxPool1D()(x)
x = Bidirectional(LSTM(64, dropout=0.2, recurrent_dropout=0.2, return_sequences=True))(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(512, activation='relu')(x)
x = Dense(256, activation = 'relu')(x)
outputs = Dense(1, activation='sigmoid')(x)
model = tf.keras.Model(sequence_input, outputs)
```

Figure 11 結合 RNN 和 CNN 模型架構

2.4 模型績效比較

本研究嘗試了 SimpleRNN、LSTM 以及 GRU 三種方法，並且結合 CNN 在訓練，一共嘗試六種組合。因為本研究探討的是留言是正面還是反面的意思，為一個二元問題，因此結果可以用混淆矩陣來表示，下圖為混淆矩陣的示例圖：

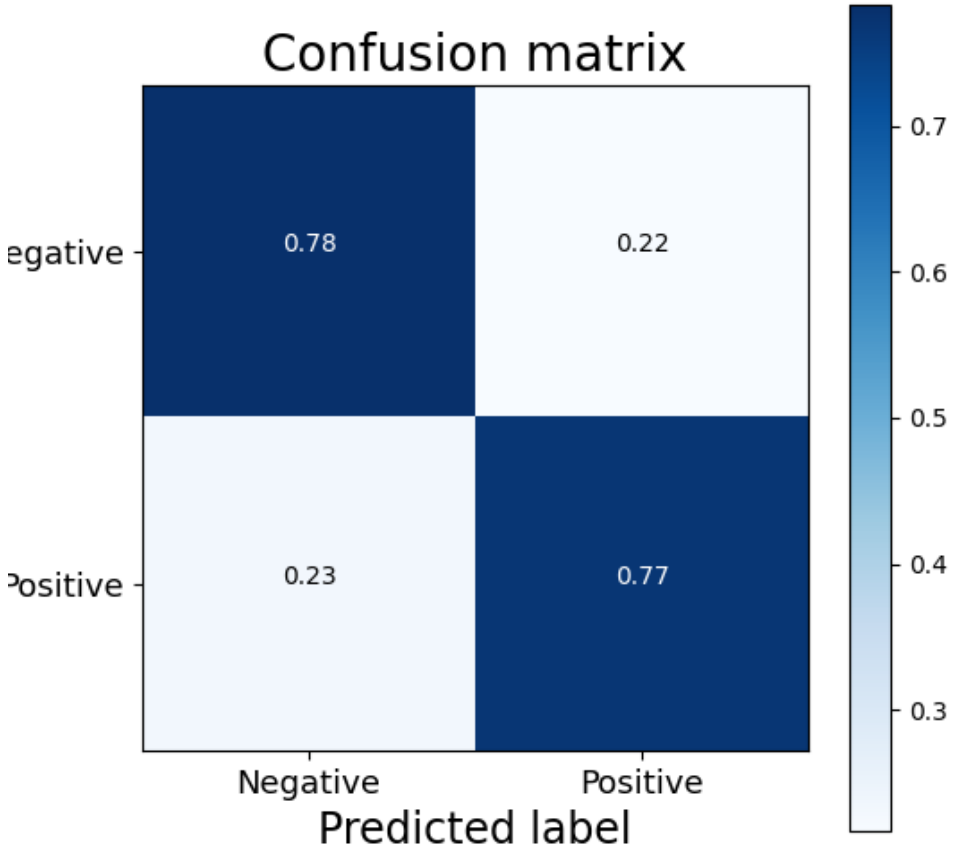

Figure 12 混淆矩陣

衡量績效指標的方法有很多，像是準確率、損失率、精確率、召回率等等的。本研究每次的 epochs = 20 次，利用每次訓練結果的混淆矩陣去計算每次訓練的準確率、精確率、召回率以及 F1 分數並將資料整理在下表：

Table 2 績效衡量表格

	Accuracy	Precision	Recall	F1-score
SimpleRNN	0.765	0.752	0.790	0.771
LSTM	0.790	0.802	0.770	0.786
GRU	0.785	0.788	0.780	0.784
1D CNN	0.775	0.780	0.770	0.784
SimpleRNN+CNN	0.700	0.679	0.760	0.717
LSTM+CNN	0.695	0.676	0.750	0.711
GRU+CNN	0.695	0.673	0.760	0.714

由上表可知，若只單看準確率的話，LSTM 的準確率是最高的，高達了 79%。而若看整體綜合表現，也就是 F1 分數的話也是 LSTM 的準確率最高，高達了 78.6%。並且 LSTM 的訓練情況是沒有發生 overfitting 的。

三、 結論及未來展望

4.1 結論

從模型結果可以得出，嘗試不同模型組合後，本研究在 LSTM 的模型表現最佳，測試準確率高達了 79%，F1 分數也高達了 78.6%。

由實驗結果可以發現，結合 CNN 的結果其實是沒有比較好的，可能的原因為此問題可能真的不適合結合 CNN 來進行訓練，也有可能是因為常識的組合太少，導致沒有發掘到績效更好的模型。

4.2 未來展望

之後若有機會，可再嘗試不同的組合、不同的權重、參數等等，看能不能找到準確率更高的模型，提高模型的泛化性，讓模型可以更正確的判斷文字情感。

在應用方面，可以應用在股票分析上，觀察人們對股價的反應去判斷這支股票之後的趨勢為何；或者是用在商品分析上，可以查看想要購買的商品評價，看大家對商品的評價為何來決定要不要購買。