專題:情感分析(sentiment analysis)

使用技術:深度深度(Deep Learning)/自然語言處理(Natural Language Processing)

資料來源:https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

IMDB 數據集包含用於自然語言處理或文本分析的 50K 電影評論。 這是一個用於二元情感分類的數據集，使用深度學習算法預測正面和負面評論的數量。

資料集(DataSet):IMDB_Dataset.csv

程式共有兩種版本 (1)一般作法 (2)使用 Simple Transformers 框架

以下說明(1)一般作法

Seaborn 本質上是一個基於 matplotlib 庫的高級 API。它包含更適合處理圖表的默認設置。

```
!pip  install  seaborn
```

詞雲又叫文字雲，是對文字資料中出現頻率較高的“關鍵詞”在視覺上的突出呈現，形成關鍵詞的渲染形成類似雲一樣的彩色圖片，從而一眼就可以領略文字資料的主要表達意思。

```
!pip  install  wordcloud
```

Pandas 是一種常用的資料分析處理工具，主要應用於單維度(Series)與二維度(DataFrame)的資料處理。

Numpy 可以產生一維、二維陣列進行向量（vector）和矩陣（matrix）運算，其在大量運算時有非常優異的效能。

```
import  pandas  as  pd
import  numpy  as  np
```

```
import  seaborn  as  sns
```

```
import  string
```

NLTK 全名是 Natural Language Tool Kit， 是一套基於 Python 的自然語言處理工具箱。

```
import  nltk
```

Matplotlib 是一個用於創建二維圖和圖形的底層庫。

```
import  matplotlib.pyplot  as  plt
```

tensorflow 一個用於機器學習的開源軟體庫，可以支援深度學習的各種演算法。

```
import  tensorflow  as  tf
```

Keras是一款用Python編寫而成的開源神經網路庫，也可以說是開放的高階深度學習程式庫，能搭配TensorFlow使用

```
from  tensorflow.keras.preprocessing.text  import  Tokenizer
from  tensorflow.keras.preprocessing.sequence  import  pad_sequences

from  wordcloud  import  WordCloud,  STOPWORDS,  ImageColorGenerator
```

函式: 將資料集 IMDB_Dataset.csv 的 'positive' 改為'1', 'negative'改為 '0'

```
def  Convert(data_p):
          data_p  =  data_p[data_p['sentiment'].isin(['positive',  'negative'])]
          data_p['sentiment']  =  data_p['sentiment'].replace({'positive':1,  'negative':  0})
          return  data_p
```

使用 pandas 讀取 DataSet(CSV格式)

```
df  =  pd.read_csv('/content/drive/MyDrive/Colab  Notebooks/IMDB_Dataset.csv')
```

呼叫 Convert 函式 , 將資料集 IMDB_Dataset.csv 的 'positive' 改為'1', 'negative'改為 '0'

```
df  =  Convert(df)
```

```
df.info
```

```
<bound method DataFrame.info of                                                  review  se
0      One of the other reviewers has mentioned that ...          1
1      A wonderful little production. <br /><br />The...          1
2      I thought this was a wonderful way to spend ti...          1
3      Basically there's a family where a little boy ...          0
4      Petter Mattei's "Love in the Time of Money" is...          1
...                                                  ...        ...
49995  I thought this movie did a down right good job...          1
49996  Bad plot, bad dialogue, bad acting, idiotic di...          0
49997  I am a Catholic taught in parochial elementary...          0
49998  I'm going to have to disagree with the previou...          0
```

```
49999  No one expects the Star Trek movies to be high...          0

[50000 rows x 2 columns]>
```

df

預處理

在進行分析之前，需要進行一些文本清理和處理。預處理和數據清洗是數據分析的重要組成部分。首先，我將"review"列設為字符串格式。它看起來像字符串。但是，如果有一些不是字符串格式的數據，我會簡單地將整列轉換為字符串。

```
df['review']  =  df['review'].astype(str)
```

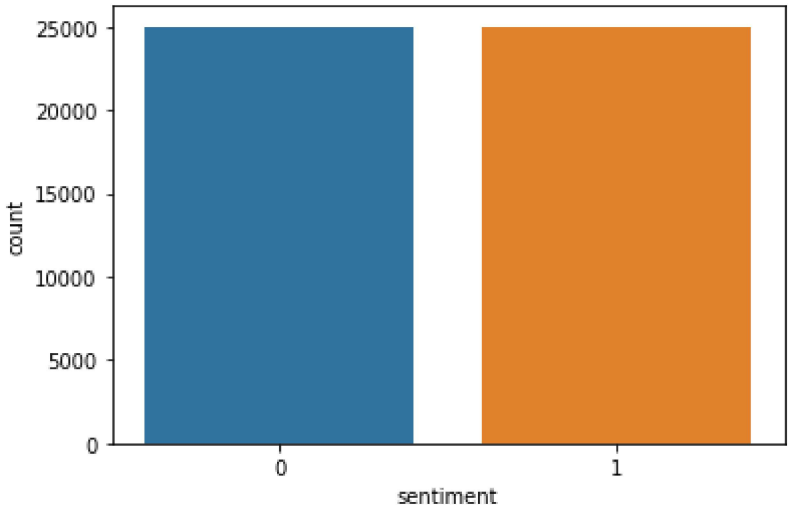
剔除該行是NULL值

```
df  =  df[~df["review"].isnull()]
```

看一下 正面1/負面0 資料分布

```
sns.countplot(df['sentiment'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f43c0d73d50>
```



英文字母 大寫 轉換成 小寫

```
df['review1']  =  df['review'].apply(lambda  x:  " ".join(x.lower()  for  x  in  x.split()))
```

```
df
```

	review	sentiment	review1
0	One of the other reviewers has mentioned that ...	1	one of the other reviewers has mentioned that ...
1	A wonderful little production. The...	1	a wonderful little production. the...
2	I thought this was a wonderful way to spend ti...	1	i thought this was a wonderful way to spend ti...
3	Basically there's a family where a little boy ...	0	basically there's a family where a little boy ...
4	Petter Mattei's "Love in the Time of Money" is...	1	petter mattei's "love in the time of money" is...
...	...	...	...
49995	I thought this movie did a down right good job...	1	i thought this movie did a down right good job...
49996	Bad plot, bad dialogue, bad acting, idiotic di...	0	bad plot, bad dialogue, bad acting, idiotic di...

```
string.punctuation
```

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

文本(review) 可能包含許多對任何分析都沒有幫助的特殊字符(PUNCTUATIONS)。清除之。

去除 PUNCTUATIONS 函式

```
def  remove_func(message):
       Test_punc_removed  =  [char  for  char  in  message  if  char  not  in  string  .punctuatic
       Test_punc_removed_join  =  ''.join(Test_punc_removed)

       return  Test_punc_removed_join


df['review2']  =  df['review1'].apply(remove_func)


df[['review','sentiment','review2']]
```

	review	sentiment	review2
0	One of the other reviewers has mentioned that ...	1	one of the other reviewers has mentioned that ...
1	A wonderful little production. The...	1	a wonderful little production br br the filmin...
2	I thought this was a wonderful way to spend ti...	1	i thought this was a wonderful way to spend ti...
3	Basically there's a family where a little boy ...	0	basically theres a family where a little boy j...
4	Petter Mattei's "Love in the Time of Money" is...	1	petter matteis love in the time of money is a ...

刪除停用詞

停用詞是一些語法或具有約束力的詞，如“is”、“the”、“and”、“so”、“my”等。這些詞經常出現。但可能不會為分析增加任何價值。

49999	Bad plot, bad dialogue, bad acting,	0	bad plot bad dialogue bad acting

```
nltk.download('stopwords')
from nltk.corpus import stopwords
stop = stopwords.words('english')
df['review3'] = df['review2'].apply(lambda x: " ".join(x for x in x.split() if x not i
df['review3'].head()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
0    one reviewers mentioned watching 1 oz episode ...
1    wonderful little production br br filming tech...
2    thought wonderful way spend time hot summer we...
3    basically theres family little boy jake thinks...
4    petter matteis love time money visually stunni...
Name: review3, dtype: object
```

```
df[['review2','sentiment','review3']]
```

	review2	sentiment	review3
0	one of the other reviewers has mentioned that ...	1	one reviewers mentioned watching 1 oz episode ...

詞幹/詞條提取`

Stemming and Lemmatization

```
from nltk.stem import PorterStemmer
st = PorterStemmer()
df['review4'] = df['review3'].apply(lambda x: " ".join([st.stem(word) for word in x.split
```

money is a ... stunni...

```
df[['review3','sentiment','review4']]
```

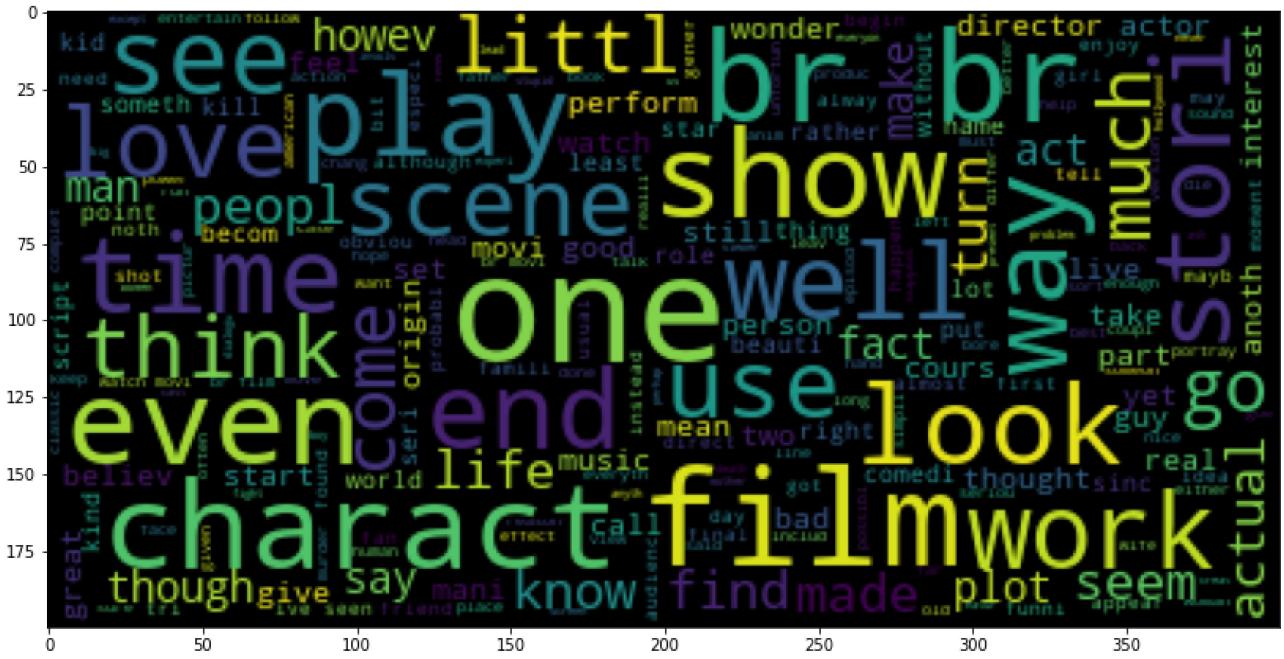
	review3	sentiment	review4
0	one reviewers mentioned watching 1 oz episode ...	1	one review mention watch 1 oz episod youll hoo...
1	wonderful little production br br filming tech...	1	wonder littl product br br film techniqu unass...
2	thought wonderful way spend time hot summer we...	1	thought wonder way spend time hot summer weeke...
3	basically theres family little boy jake thinks...	0	basic there famili littl boy jake think there ...
4	petter matteis love time money visually stunni...	1	petter mattei love time money visual stun film...
...	...	...	...
49995	thought movie right good job wasnt creative or...	1	thought movi right good job wasnt creativ orig...
49996	bad plot bad dialogue bad acting idiotic direc...	0	bad plot bad dialogu bad act idiot direct anno...

詞雲

創建詞雲 文本組合

```
text = " ".join(review for review in df.review4)
```

```
stopwords = set(STOPWORDS)
wordcl = WordCloud(stopwords = stopwords, max_font_size = 50, max_words = 5000).generate
plt.figure(figsize=(14, 12))
plt.imshow(wordcl)
plt.show()
```



NLP的首要任務就是將文本內容做Tokenization（標識化）處理，也就是說我們將文本分割成一個小塊一個小塊的例如以一個英文單詞為單位或者一個漢字為單位，這樣子的操作主要是方便我們可以更集中的去分析文本資訊的內容和文本想表達的含義。當然分割是一個大範圍，不僅僅是文本分成不同的詞，也可以將整個文本分成段落，進而分成句子，句子在細分到詞。當然，我們一般所說的標識化就是將整句分割為單個識別字（tokens）。資料來源:https://www.cnblogs.com/jielongAI/p/10178585.html

---

tokenization就是通常所說的分詞，分出的每一個詞語我們把它稱為token。

在使用NLTK執行分詞之前，我們需要先安裝「punkt」部件。「punkt」包含了許多預訓練好的分詞模型。

```
nltk.download('punkt')
```

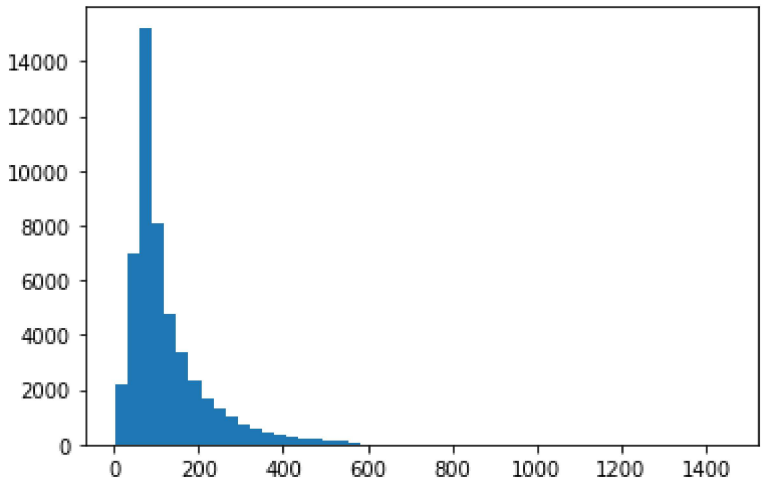
```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

每一文本字數

```
tweets_length = [ len(nltk.word_tokenize(x)) for x in df['review4'] ]
tweets_length
```

每一文本字數區分圖 -- 觀察此曲線圖,可以大約估計每一文本以150個英文字來計算

```
plt.figure()
plt.hist(tweets_length,  bins  =50)
plt.show()
```



拆分資料集(訓練集80% 測試集20%)

```
X  =  df['review4']
y  =  df['sentiment']

from  sklearn.model_selection  import  train_test_split
X_train,  X_test,  y_train,  y_test  =  train_test_split(X,  y,  test_size  =  0.2)
```

```
X_train.shape
```

```
(40000,)
```

```
X_train
```

再次確定每條文本轉換成一個字符串

```
training_sentences  =  []
training_labels  =  []
testing_sentences  =  []
testing_labels  =  []
for  row  in  X_train:
        training_sentences.append(str(row))
for  row  in  y_train:
        training_labels.append(row)
for  row  in  X_test:
        testing_sentences.append(str(row))
for  row  in  y_test:
        testing_labels.append(row)
```

```
training_sentences
```

```
training_labels
```

文本進行標記

Tokenizer 函數 在默認情況下，它會刪除所有標點符號並將文本設置為以空格分隔的組織形式。每個單詞通過分詞器功能變成一個整數。

---

oov_token 的值被設置為“OOV”。這意味著任何未知的單詞都將被 oov_token 替換。這是一個更好的選擇，而不是扔掉未知的單詞。

```
#  from  tensorflow.keras.preprocessing.text  import  Tokenizer
#  from  tensorflow.keras.preprocessing.sequence  import  pad_sequences
tokenizer  =  Tokenizer(oov_token="<OOV>")
```

全部字數預設為10000

```
tokenizer  =  Tokenizer(num_words=10000,  oov_token='<OOV>')
tokenizer.fit_on_texts(training_sentences)
word_index  =  tokenizer.word_index
```

每個單詞如何有一個整數值

```
print(word_index)
```

```
{'<OOV>': 1, 'br': 2, 'movi': 3, 'film': 4, 'one': 5, 'like': 6, 'time': 7, 'good': 8, 'make
```

文本句子可以表示為一個單詞序列

句子轉換為單詞序列，然後在必要時進行padding

每一文本以150個英文字來計算

```
sequences  =  tokenizer.texts_to_sequences(training_sentences)
padded  =  pad_sequences(sequences,  maxlen=150,  truncating='post')
testing_sentences  =  tokenizer.texts_to_sequences(testing_sentences)
testing_padded  =  pad_sequences(testing_sentences,  maxlen=150)
```

```
print(len(sequences))
```

```
40000
```

```
training_sentences
```

```
sequences
```

建立模型(Sequential)

使用 10,000 個唯一詞來訓練網絡

Embedding dimension 16表示每個單詞將由一個16維向量表示

每一文本以150個英文字來計算

激勵函數 relu sigmoid ...

```
model  =  tf.keras.Sequential([
        tf.keras.layers.Embedding(10000,  16,  input_length=150),
        tf.keras.layers.GlobalAveragePooling1D(),
        tf.keras.layers.Dense(6,  activation='relu'),
        tf.keras.layers.Dense(1,  activation='sigmoid')
])
```

設定模型訓練方式

metrics 評估準確率方法

```
model.compile(loss='binary_crossentropy',  optimizer='adam',  metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 150, 16)           160000

 global_average_pooling1d (G  (None, 16)               0
 lobalAveragePooling1D)

 dense (Dense)               (None, 6)                 102

 dense_1 (Dense)             (None, 1)                 7

=================================================================
Total params: 160,109
Trainable params: 160,109
Non-trainable params: 0
_________________________________________________________________
```

標籤label轉換為數組array

```
training_labels_final  =  np.array(training_labels)
```

```
testing_labels_final  =  np.array(testing_labels)

training_labels


training_labels_final
```

```
array([0, 1, 0, ..., 1, 0, 0])
```

fit 方法,可以進行模組訓練

---

padded 訓練集/ training_labels_final 訓練標籤/ testing_padded 測試集/ testing_labels_final 測試標籤

---

epochs 訓練次數

```
history  =  model.fit(padded,  training_labels_final,  epochs=10,  validation_data=(testing_padded,
```

```
Epoch 1/10
1250/1250 [==============================] - 8s 5ms/step - loss: 0.4603 - accuracy: 0.8118 -
Epoch 2/10
1250/1250 [==============================] - 6s 5ms/step - loss: 0.2648 - accuracy: 0.8954 -
Epoch 3/10
1250/1250 [==============================] - 6s 5ms/step - loss: 0.2268 - accuracy: 0.9110 -
Epoch 4/10
1250/1250 [==============================] - 6s 5ms/step - loss: 0.2033 - accuracy: 0.9219 -
Epoch 5/10
1250/1250 [==============================] - 6s 5ms/step - loss: 0.1867 - accuracy: 0.9291 -
Epoch 6/10
1250/1250 [==============================] - 6s 5ms/step - loss: 0.1727 - accuracy: 0.9358 -
Epoch 7/10
1250/1250 [==============================] - 6s 5ms/step - loss: 0.1615 - accuracy: 0.9409 -
Epoch 8/10
1250/1250 [==============================] - 6s 5ms/step - loss: 0.1523 - accuracy: 0.9453 -
Epoch 9/10
1250/1250 [==============================] - 6s 5ms/step - loss: 0.1426 - accuracy: 0.9498 -
Epoch 10/10
1250/1250 [==============================] - 6s 5ms/step - loss: 0.1356 - accuracy: 0.9534 -
```

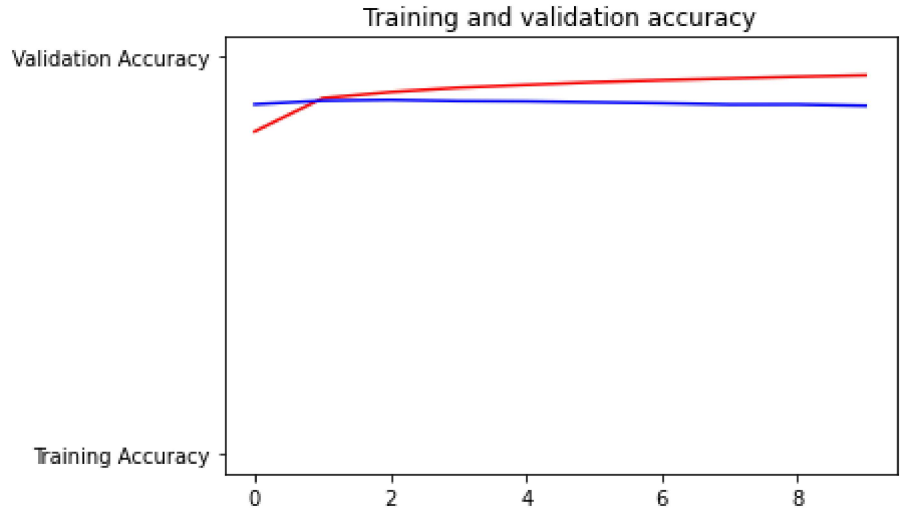
訓練準確率為 95.34%，驗證準確率為 87.65 看起來有點過擬合。

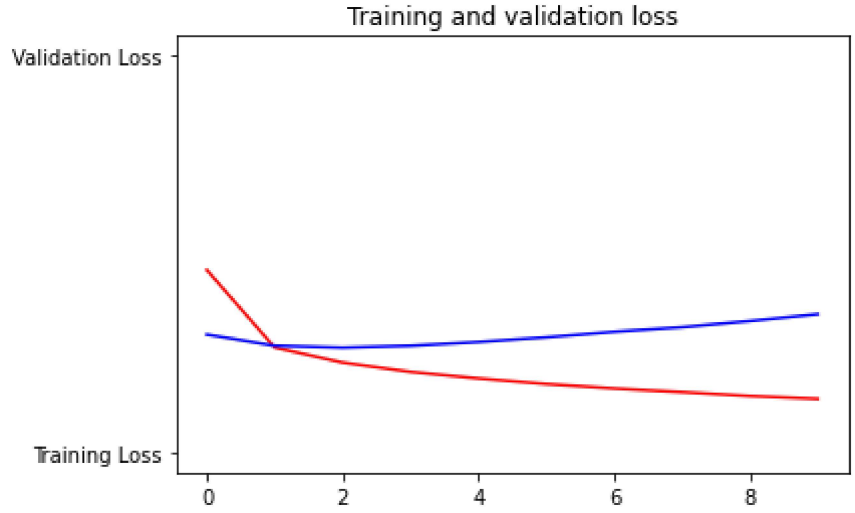
繪製訓練和驗證準確度，以及訓練和驗證損失

```
%matplotlib  inline
import  matplotlib.pyplot  as  plt
import  matplotlib.image  as  mpimg
acc  =  history.history['accuracy']
val_acc  =  history.history['val_accuracy']
loss  =  history.history['loss']
val_loss  =  history.history['val_loss']
epochs=range(len(acc))
```

```
plt.plot(epochs, acc, 'r', 'Training Accuracy')
plt.plot(epochs, val_acc, 'b', 'Validation Accuracy')
plt.title('Training and validation accuracy')
plt.figure()
plt.plot(epochs, loss, 'r', 'Training Loss')
plt.plot(epochs, val_loss, 'b', 'Validation Loss')
plt.title('Training and validation loss')
plt.figure()
```

```
<Figure size 432x288 with 0 Axes>
```






```
<Figure size 432x288 with 0 Axes>
```