HW2 Sentiment Analysis

0813458 資財 簡辰穎

一. Import Library + Load Data

Import library

In [1]:
```
import numpy as np
import pandas as pd
#import logging
#Logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
#去除停頓詞
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords
import nltk
import spacy  #要先下載好model
from nltk import word_tokenize
#將文字轉換成向量
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from gensim.utils import simple_preprocess
from gensim import models
from gensim.models.word2vec import Word2Vec
import gensim
#Create model
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import random
#評估模型好壞
from sklearn.metrics import accuracy_score
import sklearn.metrics as metrics
```

Load Data

In [3]:
```
df = pd.read_csv('./archive/yelp.csv')
df.head()
```

Out[3]:

	business_id	date	review_id	stars	text	type	user_id	cool	useful	funny
0	9yKzy9PApeiPPOUJEtnvkg	2011-01-26	fWKvX83p0-ka4JS3dc6E5A	5	My wife took me here on my birthday for breakf...	review	rLtl8ZkDX5vH5nAx9C3q5Q	2	5	0
1	ZRJwVLyzEJq1VAihDhYiow	2011-07-27	IjZ33sJrzXqU-0X6U8NwyA	5	I have no idea why some people give bad review...	review	0a2KyEL0d3Yb1V6aivbIuQ	0	0	0
2	6oRAC4uyJCsJl1X0WZpVSA	2012-06-14	IESLBzqUCLdSzSqm0eCSxQ	4	love the gyro plate. Rice is so good and I als...	review	0hT2KtfLiobPvh6cDC8JQg	0	1	0
3	_1QQZuf4zZOyFCvXc0o6Vg	2010-05-27	G-WvGaISbqqaMHlNnByodA	5	Rosie, Dakota, and I LOVE Chaparral Dog Park!!...	review	uZetl9T0NcROGOyFfughhg	1	2	0
4	6ozycU1RpktNG2-1BroVtw	2012-01-05	1uJFq2r5QfJG_6ExMRCaGw	5	General Manager Scott Petello is a good egg!!!...	review	vYmM4KTsC8ZfQBg-j5MWkw	0	0	0

二. Data Prepocessing

a. 讀取 csv 檔並僅保留 stars, text 兩個欄位

Data Preprocessing

a. 讀取csv檔僅保留"text"、"stars"兩個欄位

In [5]:
```
df1 = df[['stars','text']]
df1.head(20)
```

Out[5]:

	stars	text
0	5	My wife took me here on my birthday for breakf...
1	5	I have no idea why some people give bad review...
2	4	love the gyro plate. Rice is so good and I als...
3	5	Rosie, Dakota, and I LOVE Chaparral Dog Park!!...
4	5	General Manager Scott Petello is a good egg!!!...
5	4	Quiessence is, simply put, beautiful. Full wi...
6	5	Drop what you're doing and drive here. After l...
7	4	Luckily, I didn't have to travel far to make m...
8	4	Definitely come for Happy hour! Prices are ama...
9	5	Nobuo shows his unique talents with everything...
10	5	The oldish man who owns the store is as sweet ...
11	5	Wonderful Vietnamese sandwich shoppe. Their ba...
12	5	They have a limited time thing going on right ...
13	4	Good tattoo shop. Clean space, multiple artist...
14	4	I'm 2 weeks new to Phoenix. I looked up Irish ...
15	2	Was it worth the 21$ for a salad and small piz...

b. 將 stars 內大於等於 4 的轉成 1, 其餘轉成 0

(1 代表 positive 0 代表 negative)

c. 將 text 中的文字全部轉成小寫 (對於 stop words 處理有差別)

將stars欄位內值大於等於4的轉成1，其餘轉成0

1: positive

0: negative

In [6]:
```
df1.stars[df1.stars < 4] = 0
df1.stars[df1.stars >= 4] = 1
#一定要先小於4的轉成0, 再將大於等於4的轉成1 不然會全部變成0
```

In [7]:
```
#先全部轉成小寫 等等停頓詞處理有差
for i in range(df1.shape[0]):
    df1['text'][i] = df1['text'][i].lower()
```

In [8]:
```
df1.head(20)
```

Out[8]:

	stars	text
0	1	my wife took me here on my birthday for breakf...
1	1	i have no idea why some people give bad review...
2	1	love the gyro plate. rice is so good and i als...
3	1	rosie, dakota, and i love chaparral dog park!!...
4	1	general manager scott petello is a good egg!!!...
5	1	quiessence is, simply put, beautiful. full wi...
6	1	drop what you're doing and drive here. after i...
7	1	luckily, i didn't have to travel far to make m...
8	1	definitely come for happy hour! prices are ama...
9	1	nobuo shows his unique talents with everything...
10	1	the oldish man who owns the store is as sweet ...

d. 去除停頓詞(stop words)

常見去除停頓詞的方法有:

- Spacy / nltk / sklearn / 自定義停頓詞

我選擇使用 Spacy 來處理停頓詞，首先先看一下 Spacy 中的英文停頓詞詞庫大概有哪些英文詞

b. 去除停頓詞 stop words

- 停頓詞 (Stop Words) 的定義上是兩個集合：

1. 這個語言中出現非常頻繁的詞。
2. 文本資料中出現非常頻繁的詞。

網路上有3種做法:

a. spaCy

b. nltk

c. sklearn

In [3]:
```
#用 Python NLP 中的 spacy 進行stopwords 處理
#先看一下spacy中有哪些停頓詞
nlp = spacy.load('en_core_web_sm')
spacy_stopwords = spacy.lang.en.stop_words.STOP_WORDS
print('spaCy has {} stop words'.format(len(spacy_stopwords)))
print('The first twenty stop words are {}'.format(list(spacy_stopwords)[:20]))
```

```
spaCy has 326 stop words
The first twenty stop words are ["'d", 'nothing', 'why', 'once', 'who', 'say', 'anyone', 'nowhere', 'nobody', 'therefore', 'a
n', 'enough', 'therein', 'nor', 'when', 'whereby', 'five', 'that', 'this', 'whither']
```

接著開始去除停頓詞，將 text 切割跟 Spacy 詞庫內的詞作比較，留下沒有在詞庫的字。可以發現去除停頓詞後 text 變得比較簡潔，對於模型的預測可能會較準確。

```
In [12]:
"""
1. imdb_df['text'].apply將每個row的text值分別取出
2. apply(lambda x: rmsw_function(x, spacy_stopwords)) 這邊可以拆解成
    - 將imdb_df['text'] 取出，當作lambda function的 x
    - 然後丟進 rmsw_function 得到ruturn的值
    - 然後放到 imdb['spacy_rmsw_text_fancy']之中
"""
#nltk.download('punkt')
#def rmsw_function(text, stopword_list):
#    return ' '.join([word for word in word_tokenize(text) if word not in stopword_list])
#df1['rmsw_text'] = df1['text'].apply(lambda x: rmsw_function(x, spacy_stopwords))
df1['rmsw_text'] =  df1['text'].apply(lambda x:' '.join([word for word in x.split() if word not in spacy_stopwords]))

In [13]:
print(df1['rmsw_text'][0])

wife took birthday breakfast excellent. weather perfect sitting outside overlooking grounds absolute pleasure. waitress excelle
nt food arrived quickly semi-busy saturday morning. looked like place fills pretty quickly earlier better. favor bloody mary. p
henomenal simply best i've had. i'm pretty sure use ingredients garden blend fresh order it. amazing. menu looks excellent, whi
te truffle scrambled eggs vegetable skillet tasty delicious. came 2 pieces griddled bread amazing absolutely meal complete. bes
t "toast" i've had. anyway, can't wait back!

In [14]:
print(df1['text'][0])

my wife took me here on my birthday for breakfast and it was excellent.  the weather was perfect which made sitting outside ove
rlooking their grounds an absolute pleasure.  our waitress was excellent and our food arrived quickly on the semi-busy saturday
morning.  it looked like the place fills up pretty quickly so the earlier you get here the better.

do yourself a favor and get their bloody mary.  it was phenomenal and simply the best i've ever had.  i'm pretty sure they only
use ingredients from their garden and blend them fresh when you order it.  it was amazing.

while everything on the menu looks excellent, i had the white truffle scrambled eggs vegetable skillet and it was tasty and del
icious.  it came with 2 pieces of their griddled bread with was amazing and it absolutely made the meal complete.  it was the b
est "toast" i've ever had.

anyway, i can't wait to go back!
```

e. 文字探勘前處理，將文字轉換成向量(使用 tf-idf 與 word2vec 並比較)

➔ Tf-idf: 包含詞頻(tf)與逆向文件頻率(idf)，詞頻指某一個特定詞語在該文件中出現的頻率，而逆向文件頻率用來處理常用字的問題

Tf-idf = tf*idf

➔ Word2vec: 透過學習大量文本資料，將字詞用數學向量的方式來代表他們的語意。並將字詞嵌入到一個空間後，讓語意相似的單字可以有較近的距離。主要有 CBOW 與 Skip-gram 兩種模型。從直觀上來理解， Skip-gram 是給定輸入字詞後，來預測上下文；CBOW 則是給定上下文，來預測輸入的字詞

A. Tf-idf

Tf-idf

```
In [17]: # td-idf
         #vectorizer = CountVectorizer(stop_words = spacy_stopwords)
         tfidf_vectorizer = TfidfVectorizer()
         tfidf = tfidf_vectorizer.fit_transform(df1['rmsw_text'])
         weight = tfidf.toarray()
         print(weight.shape)
         print(weight)
         df1['tfidf'] = tfidf
```

```
(10000, 29160)
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

B. Word2vec

Word2Vec

```
In [19]: tokenized_tweet = df1['rmsw_text'].apply(lambda x: x.split()) # tokenizing

         model_w2v = gensim.models.Word2Vec(
                     tokenized_tweet,
                     vector_size=200, # desired no. of features/independent variables
                     window=5, # context window size
                     min_count=2, # Ignores all words with total frequency lower than 2.
                     sg = 1, # 1 for skip-gram model
                     hs = 0,
                     negative = 10, # for negative sampling
                     workers= 32, # no.of cores
                     seed = 34
         )
```

```
In [20]: model_w2v.build_vocab(tokenized_tweet, progress_per = 1000)
```

```
In [21]: model_w2v.epochs
Out[21]: 5
```

```
In [22]: model_w2v.corpus_count
Out[22]: 10000
```

```
In [23]: model_w2v.train(tokenized_tweet, total_examples =model_w2v.corpus_count, epochs = model_w2v.epochs)
Out[23]: (2865961, 3167730)
```

```
In [24]: model_w2v.save("./w2v_model")
```

```
In [25]: model_w2v.wv.most_similar('bad')
Out[25]: [('terrible', 0.6191313862800598),
          ('negative', 0.6006874442100525),
          ('ruined', 0.5913141369819641),
          ('either.', 0.5795227885246277),
          ('worse', 0.5695444345474243),
          ('previous', 0.5682346820831299),
          ('coup,', 0.5675501823425293),
          ('problems', 0.566647469997406),
          ('compelled', 0.5659807920455933),
          ("people's", 0.5637657046318054)]
```

可以看到跟 bad 相關的詞可能有 ruined, terrible, negative…的負面詞

但我們的資料是一整個評論而非單單僅是單詞，所以需要想辦法使用

word2vec 模型中的詞向量為整個評論創建向量表示

⇨ 解決方法: 取評論中所有詞向量的平均，對資料中的所有評論重複相同的過程獲得它們的向量

→得到每一個評論的向量並轉成 array 的形式

```
In [27]: def word_vector(tokens, size):
             vec = np.zeros(size).reshape((1, size))
             count = 0
             for word in tokens:
                 try:
                     vec += model_w2v.wv[word].reshape((1, size))
                     count += 1.
                 except KeyError:  # handling the case where the token is not in vocabulary
                     continue
             if count != 0:
                 vec /= count
             return vec
```

```
In [28]: wordvec_arrays = np.zeros((len(tokenized_tweet), 100))
         for i in range(len(tokenized_tweet)):
             wordvec_arrays[i,:] = word_vector(tokenized_tweet[i], 100)
         wordvec_df = pd.DataFrame(wordvec_arrays)
         # wordvec_df.shape
         print(len(wordvec_arrays))
```

```
10000
```

三. Create model

a. 先用 random forest tree 單純跑一次看看

Create Model

a. 使用 Random forest進行分類

```
In [33]: y=df1['stars']
         x=tfidf
         x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=42,test_size=0.2)
```

```
In [34]: rf= RandomForestClassifier(criterion= 'entropy', max_depth= None, max_features= 'sqrt', n_estimators= 500, oob_score = True)
         rf.fit(x_train,y_train)
         y_pred= rf.predict(x_test)
         accuracy_score(y_pred,y_test)
         #先單純跑一次試試看
```

```
Out[34]: 0.7995
```

b. 用套件先試試看跑 K-fold cross validation 準確度

```
In [32]: #先用套件跑跑看效果
         from sklearn.model_selection import cross_val_score
         x = tfidf
         y = df1['stars']
         #rf= RandomForestClassifier(criterion= 'entropy', max_depth= None, max_features= 'sqrt', n_estimators= 100, oob_score = True)
         rf = RandomForestClassifier(criterion= 'entropy', max_features= 'sqrt', n_estimators= 500, oob_score = True)
         accuracy = cross_val_score(rf, x, y, cv=4, scoring='accuracy')
         print(accuracy)
         print(accuracy.mean()*100,'%')
```

```
[0.7808 0.7944 0.7884 0.7944]
78.95 %
```

c. 建立 cross_validation_split，切割等等要丟入 K-fold cross-validation function 中的 train data 與 test data

k-fold cross-validation

```
In [38]: import random
         model= RandomForestClassifier(criterion= 'entropy', max_features= 'sqrt', n_estimators= 100, oob_score = True)
         def cross_validation_split(dataset, folds):
             dataset_split = []
             df_copy = dataset
             fold_size = int(df_copy.shape[0] / folds)

             # for loop to save each fold
             for i in range(folds):
                 fold = []
                 # while loop to add elements to the folds
                 while len(fold) < fold_size:
                     # select a random element
                     r = random.randrange(df_copy.shape[0])
                     # determine the index of this element
                     index = df_copy.index[r]
                     # save the randomly selected line
                     fold.append(df_copy.loc[index].values.tolist())
                     # delete the randomly selected line from
                     # dataframe not to select again
                     df_copy = df_copy.drop(index)
                 # save the fold
                 dataset_split.append(np.asarray(fold))

             return dataset_split
```

d. 建立 K-fold cross-validation function，並將 tf-idf 處理的 rmsw_text(remove stop words 後的 text)與 word2vec 處理的 rmsw_text 分別丟入 model

In [39]:
```
def K_fold_CV(dataset, f = 4):
    data=cross_validation_split(dataset,f)
    Accuracy=0
    # determine training and test sets
    for i in range(f):
        r = list(range(f))
        r.pop(i) #i是testing set
        test=data[i]

        for j in r :
            if j == r[0]:
                cv = data[j]
            else:
                cv=np.concatenate((cv,data[j]), axis=0)

        # apply RandomForest model
        X = cv[:,1:]
        Y = cv[:,0]
        X_test = test[:,1:]
        y_test = test[:,0]
        model.fit(X,Y)
        y_pred=model.predict(X_test)
        Accuracy=Accuracy+metrics.accuracy_score(y_test, y_pred)
    return Accuracy/f
```

```
K_fold_CV(df3) #word2vec
```

```
0.7772
```

```
K_fold_CV(df2) #tfidf
```

```
0.7901
```

結果比較: tf-idf 跑出來的結果相較於 word2vec 好

(執行多次結果都是這樣)