作業三

1.資料前處理 data preprocessing

a. 讀取csv前10000筆，保留text與score

```
In [1]: import pandas as pd
```

```
In [2]: df = pd.read_csv(r'Reviews.csv')
```

```
In [3]: df.head()
```

Out[3]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...

```
In [4]: sample_data = df[['Text','Score']][:10000]
        len(sample_data)
```

```
Out[4]: 10000
```

```
In [5]: sample_data.head()
```

Out[5]:

	Text	Score
0	I have bought several of the Vitality canned d...	5
1	Product arrived labeled as Jumbo Salted Peanut...	1
2	This is a confection that has been around a fe...	4
3	If you are looking for the secret ingredient i...	2
4	Great taffy at a great price. There was a wid...	5

將 "Score" 欄位內值大於等於4的轉成1(positive)，其餘轉成0(negative)

```
In [6]: sample_data['Score'] = sample_data['Score'].map(lambda x: 1 if x>=4 else 0) #此 map是pandas map不是一般python map
```

In [7]:

```
sample_data.head()
```

Out[7]:

	Text	Score
0	I have bought several of the Vitality canned d...	1
1	Product arrived labeled as Jumbo Salted Peanut...	0
2	This is a confection that has been around a fe...	1
3	If you are looking for the secret ingredient i...	0
4	Great taffy at a great price. There was a wid...	1

stop words處理

In [8]:

```
# python -m nltk.downloader all
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\wang\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\wang\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\wang\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\wang\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[8]:

```
True
```

In [10]:

```
def clean(text):
    wn = nltk.WordNetLemmatizer()
    stopword = nltk.corpus.stopwords.words('english')
    tokens = nltk.word_tokenize(text)
    lower = [word.lower() for word in tokens]
    no_stopwords = [word for word in lower if word not in stopword]
    no_alpha = [word for word in no_stopwords if word.isalpha()]
    lemm_text = [wn.lemmatize(word) for word in no_alpha]
    clean_text = ' '.join(lemm_text)
    return clean_text
```

In [11]:

```
sample_data['clean'] = sample_data['Text'].map(clean)
sample_data
```

Out[11]:

	Text	Score	clean
0	I have bought several of the Vitality canned d...	1	bought several vitality canned dog food produc...
1	Product arrived labeled as Jumbo Salted Peanut...	0	product arrived labeled jumbo salted peanut pe...
2	This is a confection that has been around a fe...	1	confection around century light pillowy citrus...
3	If you are looking for the secret ingredient i...	0	looking secret ingredient robitussin believe f...
4	Great taffy at a great price. There was a wid...	1	great taffy great price wide assortment yummy ...
...	...	...	...
9995	we switched from the advance similac to the or...	0	switched advance similac organic product think...
9996	Like the bad reviews say, the organic formula ...	1	like bad review say organic formula constipate...
9997	I wanted to solely breastfeed but was unable t...	1	wanted solely breastfeed unable keep supplemen...
9998	i love the fact that i can get this delieved t...	1	love fact get delieved house delievy hard find...
9999	We have a 7 week old... He had gas and constip...	1	week old gas constipation problem first week t...

10000 rows × 3 columns

In [12]:
```
from keras.preprocessing import sequence
from keras.preprocessing.text import Tokenizer
from keras.utils.vis_utils import plot_model
import nltk
import numpy as np
from keras.utils import pad_sequences
from keras.utils import to_categorical
from keras.layers import Dense, Input, GlobalMaxPooling1D
from keras.layers import Conv1D, MaxPooling1D, Embedding
from keras.models import Model
```

切割資料集

In [13]:
```
from sklearn.model_selection import train_test_split
```

In [14]:
```
X_train, X_validation = train_test_split(sample_data, test_size=0.2)
X_train = X_train.reset_index()
X_validation = X_validation.reset_index()
```

In [15]:
```
train_x = X_train.drop(['Score'],axis = 1)
train_y = X_train['Score']
```

In [16]:
```
test_x = X_validation.drop(['Score'],axis = 1)
test_y = X_validation['Score']
```

In [17]:
```
X_train.iloc[0][0]
```

Out[17]:
```
9599
```

In [18]:
```
X_train.iloc[0][1] #原始文字
```

Out[18]:
```
'I did a lot of research on this fondant and really wanted to like it however it just didn\'t work for me. When it arriv
ed it was hard as a rock, but there is a card inside that indicates about 30-50 seconds in the microwave will make it ea
sier to work with. This really did work well. It made it much more pliable and easier to work with. The fondant took col
or real well and rolled out nicely. It does say "no shortening or powdered sugar needed" that\'s pretty false. If you\'r
e coloring it you do need some while needing otherwise it will stick to your hands and your work surface. I DEFINITELY n
eeded powdered sugar while rolling it out. Otherwise it stuck to the silicon mat I use. Also, after coloring and putting
it in the microwave it definitely needed some time in the refrigerator before it could be rolled again and placed on the
cake. Otherwise it was far too gooey.<br /><br />Finally, the moment of truth arrived, laying the rolled out fondant on
the cake. It was much more difficult to work with than I imagined. It\'s very picky on the thickness. Too thick it\'s im
possible to lay flat, too thin and it would stretch and get holes. I will say it did get some \'elephant skin\' during a
pplication I was able to smooth it out using a smoothing tool and the final product looked fine. Perhaps I was a little
bit rusty, I used to teach a Wilton cake decorating class and have taught fondant decorating for many years but haven\'t
done a cake in the past 5-6 years. I just never remember the wilton brand, which tastes horrible, ever being this hard t
o work with.<br /><br />The flavor was fine on it\'s own, but as most fondants, it\'s incredibly sweet..incredibly. Desp
ite the improved flavor people were still pealing it off the cake and eating the cake and frosting.<br /><br />In the en
d the cake looked wonderful and people were pleased but not sure I\'ll be using this brand again.'
```

In [19]:
```
X_train.iloc[0][2] #label
```

Out[19]:
```
0
```

In [20]:
```
X_train.iloc[0][3] #stop words清掉
```

Out[20]:
```
'lot research fondant really wanted like however work arrived hard rock card inside indicates second microwave make easi
er work really work well made much pliable easier work fondant took color real well rolled nicely say shortening powdere
d sugar needed pretty false coloring need needing otherwise stick hand work surface definitely needed powdered sugar rol
ling otherwise stuck silicon mat use also coloring putting microwave definitely needed time refrigerator could rolled pl
aced cake otherwise far br br finally moment truth arrived laying rolled fondant cake much difficult work imagined picky
thickness thick impossible lay flat thin would stretch get hole say get skin application able smooth using smoothing too
l final product looked fine perhaps little bit rusty used teach wilton cake decorating class taught fondant decorating m
any year done cake past year never remember wilton brand taste horrible ever hard work br br flavor fine fondant incredi
bly sweet incredibly despite improved flavor people still pealing cake eating cake br br end cake looked wonderful peopl
e pleased sure using brand'
```

In [21]:
```
# 讀取所有資料
#X_train_text = []
X_train_clean = []
#X_validation_text = []
X_validation_clean = []
for i in range(len(X_train)):
    #X_train_text += [X_train.iloc[i,1]]
    X_train_clean += [X_train.iloc[i,3]]
for i in range(len(X_validation)):
    #X_validation_text += [X_validation.iloc[i,1]]
    X_validation_clean += [X_validation.iloc[i,3]]
```

In [22]:

```
# 建立 Token
token = Tokenizer(num_words = 4000)
token.fit_on_texts(X_train_clean)
token.word_index
# 文字轉數字(token)
X_train_seq = token.texts_to_sequences(X_train_clean)
X_test_seq = token.texts_to_sequences(X_validation_clean)
# 截長補短
X_train = pad_sequences(X_train_seq, maxlen = 100) #我不知道最大長度設多少，書上寫100
X_test = pad_sequences(X_test_seq, maxlen = 100)
```

In [23]:

```
X_train[0]
```

Out[23]:

```
array([  20, 1852,  992,  712,  116,  466,   17, 1504,   45, 1798, 1231,
        211,  659,  135,    1,    1,  474, 1465,  250, 1798,  973,  211,
         19,  548,   71, 3677,  529, 2411,  624, 1535, 1138, 1012,  573,
         11,   16, 1320,   53,   16,  706, 3396,  268,  194,  111, 2972,
       2412,    8,  458,  262,  678,   22,   62,   49, 2131,  211, 3678,
       1929,  973, 3678,   66,   47,  530,  211,  535,   47,   89,  618,
       2131,   36,    4,  744,  100,  128,   71,    1,    1,    6,  262,
        973, 1659,   55, 1659, 1139, 1749,    6,  138,   67,  211,  154,
        211,    1,    1,  386,  211,  458,  181,  138,  397,  117,  111,
         36])
```

In [24]:

```
from keras.layers.core import Dense,Dropout,Activation,Flatten
from keras.preprocessing import sequence
from keras.models import Sequential, Model
from keras.layers import Dense, Embedding, Input
from keras.layers import LSTM, Flatten, MaxPooling1D
from keras.layers import Conv1D
import numpy as np
```

In [25]:

```
#要改成np的資料型別，才塞得進去model
y_train = np.asarray(train_y).astype(np.float32)
y_test = np.asarray(test_y).astype(np.float32)
y_train
```

Out[25]:

```
array([0., 1., 1., ..., 1., 1., 1.], dtype=float32)
```

In [26]:

```
modelCNN = Sequential()
#Embedding層將「數字list」轉換成「向量list」
modelCNN.add(Embedding(output_dim=32,   #輸出的維度是32，希望將數字list轉換為32維度的向量
     input_dim=4000,                    #輸入的維度是4000，也就是我們之前建立的字典是4000字
     input_length=100))                 #數字list截長補短後都是100個數字
modelCNN.add(Conv1D(32, 2, activation="relu", input_shape=(100,1)))
modelCNN.add(MaxPooling1D())
modelCNN.add(Dropout(0.3)) #隨機在神經網路中放棄30%的神經元，避免overfitting
modelCNN.add(Conv1D(32, 2, activation="relu"))
modelCNN.add(MaxPooling1D())
modelCNN.add(Dropout(0.3))

modelCNN.add(Dense(16, activation="relu"))
modelCNN.add(Flatten())

modelCNN.add(Dense(3, activation = 'sigmoid'))
modelCNN.add(Dropout(0.3))
modelCNN.compile(loss = 'sparse_categorical_crossentropy' ,optimizer = "adam",metrics = ['accuracy'])
modelCNN.summary()

train_history = modelCNN.fit(X_train, y_train,epochs=15, batch_size=128, verbose=2, validation_split=0.2)

scores = modelCNN.evaluate(X_test, y_test,verbose=1)
scores[1]
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 32)           128000

 conv1d (Conv1D)             (None, 99, 32)            2080

 max_pooling1d (MaxPooling1D  (None, 49, 32)           0
 )

 dropout (Dropout)           (None, 49, 32)            0

 conv1d_1 (Conv1D)           (None, 48, 32)            2080

 max_pooling1d_1 (MaxPooling  (None, 24, 32)           0
 1D)

 dropout_1 (Dropout)         (None, 24, 32)            0

 dense (Dense)               (None, 24, 16)            528

 flatten (Flatten)           (None, 384)               0

 dense_1 (Dense)             (None, 3)                 1155

 dropout_2 (Dropout)         (None, 3)                 0

=================================================================
Total params: 133,843
Trainable params: 133,843
Non-trainable params: 0
_________________________________________________________________
Epoch 1/15
50/50 - 3s - loss: 4.2717 - accuracy: 0.5723 - val_loss: 3.4175 - val_accuracy: 0.7700 - 3s/epoch - 51ms/step
Epoch 2/15
50/50 - 1s - loss: 2.9725 - accuracy: 0.5933 - val_loss: 2.6947 - val_accuracy: 0.7700 - 954ms/epoch - 19ms/step
Epoch 3/15
50/50 - 1s - loss: 2.8759 - accuracy: 0.5886 - val_loss: 2.5878 - val_accuracy: 0.7700 - 809ms/epoch - 16ms/step
Epoch 4/15
50/50 - 1s - loss: 2.7110 - accuracy: 0.5928 - val_loss: 2.4856 - val_accuracy: 0.7700 - 781ms/epoch - 16ms/step
Epoch 5/15
50/50 - 1s - loss: 2.6686 - accuracy: 0.5830 - val_loss: 2.0526 - val_accuracy: 0.7700 - 785ms/epoch - 16ms/step
Epoch 6/15
50/50 - 1s - loss: 2.2217 - accuracy: 0.5930 - val_loss: 1.6235 - val_accuracy: 0.7700 - 790ms/epoch - 16ms/step
Epoch 7/15
50/50 - 1s - loss: 2.0014 - accuracy: 0.5992 - val_loss: 1.4346 - val_accuracy: 0.7700 - 701ms/epoch - 14ms/step
Epoch 8/15
50/50 - 1s - loss: 1.9270 - accuracy: 0.5950 - val_loss: 1.3721 - val_accuracy: 0.7719 - 676ms/epoch - 14ms/step
Epoch 9/15
50/50 - 1s - loss: 1.8966 - accuracy: 0.5892 - val_loss: 1.4780 - val_accuracy: 0.7731 - 756ms/epoch - 15ms/step
Epoch 10/15
50/50 - 1s - loss: 1.8152 - accuracy: 0.5878 - val_loss: 1.4055 - val_accuracy: 0.7769 - 787ms/epoch - 16ms/step
Epoch 11/15
50/50 - 1s - loss: 1.7911 - accuracy: 0.5828 - val_loss: 1.9292 - val_accuracy: 0.7738 - 772ms/epoch - 15ms/step
Epoch 12/15
```

```
50/50 - 1s - loss: 1.7251 - accuracy: 0.6011 - val_loss: 1.4052 - val_accuracy: 0.7738 - 733ms/epoch - 15ms/step
Epoch 13/15
50/50 - 1s - loss: 1.6952 - accuracy: 0.5938 - val_loss: 1.3002 - val_accuracy: 0.7775 - 766ms/epoch - 15ms/step
Epoch 14/15
50/50 - 1s - loss: 1.7092 - accuracy: 0.5911 - val_loss: 1.6583 - val_accuracy: 0.7750 - 777ms/epoch - 16ms/step
Epoch 15/15
50/50 - 1s - loss: 1.6274 - accuracy: 0.6027 - val_loss: 1.7021 - val_accuracy: 0.7744 - 757ms/epoch - 15ms/step
63/63 [==============================] - 0s 2ms/step - loss: 1.8512 - accuracy: 0.7595
```

Out[26]:

```
0.7595000267028809
```

In [39]:

```
modelLSTM = Sequential()
modelLSTM.add(Embedding(output_dim = 32, input_dim = 4000, input_length = 100))
modelLSTM.add(Dropout(0.2))
modelLSTM.add(LSTM(32))
modelLSTM.add(Dense(units = 256, activation = 'relu'))
modelLSTM.add(Dropout(0.2))
modelLSTM.add(Dense(1, activation ='sigmoid'))
modelLSTM.compile(loss = 'binary_crossentropy' ,optimizer = "adam",metrics = ['accuracy'])
modelLSTM.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_8 (Embedding)     (None, 100, 32)           128000    
                                                                 
 dropout_17 (Dropout)        (None, 100, 32)           0         
                                                                 
 lstm_8 (LSTM)               (None, 32)                8320      
                                                                 
 dense_15 (Dense)            (None, 256)               8448      
                                                                 
 dropout_18 (Dropout)        (None, 256)               0         
                                                                 
 dense_16 (Dense)            (None, 1)                 257       
                                                                 
=================================================================
Total params: 145,025
Trainable params: 145,025
Non-trainable params: 0
_________________________________________________________________
```

In [40]:

```
scores2 = modelLSTM.evaluate(X_test, y_test,verbose=1)
scores2[1]
```

```
63/63 [==============================] - 1s 7ms/step - loss: 0.6927 - accuracy: 0.6095
```

Out[40]:

```
0.609499990940094
```

In [ ]: