Data Mining Research and Practice HW4

Department: IAM Student ID: 309652008 Name: 廖家緯
GitHub Link: https://github.com/Jia-Wei-Liao/NLP_for_Yelp_Dataset

December 9, 2021

1 引入套件

本次作業使用 numpy、pandas、matplotlib、sklearn、tensorflow 等套件

```
import os
import re
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from matplotlib.ticker import MaxNLocator

import nltk
import gensim.downloader
from nltk.corpus import stopwords
from gensim.models import Word2Vec

import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Dense, LSTM, Dropout, Conv1D,
    MaxPooling1D, Flatten
```

2 讀取資料

```
df = pd.read_csv('yelp.csv')
df.head()
```

	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

Figure 1. Data Frame

1. 資料集包含 business_id、date、review_id、stars、text、type、user_id、cool、useful、funny，共 10 種屬性
2. stars 屬性為 1 到 5 的整數，大於等於 4 的表示顧客對餐廳感到滿意，否則為不滿意
3. 目標: 根據 text 屬性預測顧客對餐廳的評價為正面或負面
4. 資料來源: https://www.kaggle.com/omkarsabnis/yelp-reviews-dataset

3 資料前處理

3.1 前處理

1. 去除句子中 0-9、特殊標點符號
2. 單字大寫轉小寫
3. 將句子拆成單字陣列
4. 去除 stop word

```
nltk.download('stopwords')

def transforms(sentence):
  words = re.sub('[^a-zA-Z]',' ', sentence.lower()).split()
  stop_words = set(stopwords.words("english"))
  words = [w for w in words if not w in stop_words]

  return words
```

3.2 取出 data 與 label

- 定義 label 屬性為

$$\text{label}_i = \begin{cases} 1, & \text{start}_i \geq 4 \\ 0, & \text{otherwise} \end{cases}$$

- 有 6863 筆資料 label 為 1，有 3137 筆資料 label 為 0

```
data = pd.DataFrame([])
data.loc[:, 'sentence'] = df['text'].map(transforms)
data.loc[:, 'label'] = df['stars'].map(lambda x: int(x>=4))
data.head()
```

	sentence	label
0	[My, wife, took, me, here, on, my, birthday, f...	1
1	[I, have, no, idea, why, some, people, give, b...	1
2	[love, the, gyro, plate, Rice, is, so, good, a...	1
3	[Rosie, Dakota, and, I, LOVE, Chaparral, Dog, ...	1
4	[General, Manager, Scott, Petello, is, a, good...	1

Figure 2. Sentence and Label

3.3 分割訓練集及測試集

將資料隨機分成 80% training data 與 20% test data

```
X_train_word, X_test_word, y_train, y_test = train_test_split(data.sentence,
    data.label, test_size=0.2, random_state=42)
```

3.4 文字轉向量 (Word to Vector)

3.4.1 Input format

- 將單字依序轉成正整數，並做 padding ，統一向量的長度
- 經過前處理後，大部分句子長度 < 100，我們保守選擇，將句子調整至 150 維的向量

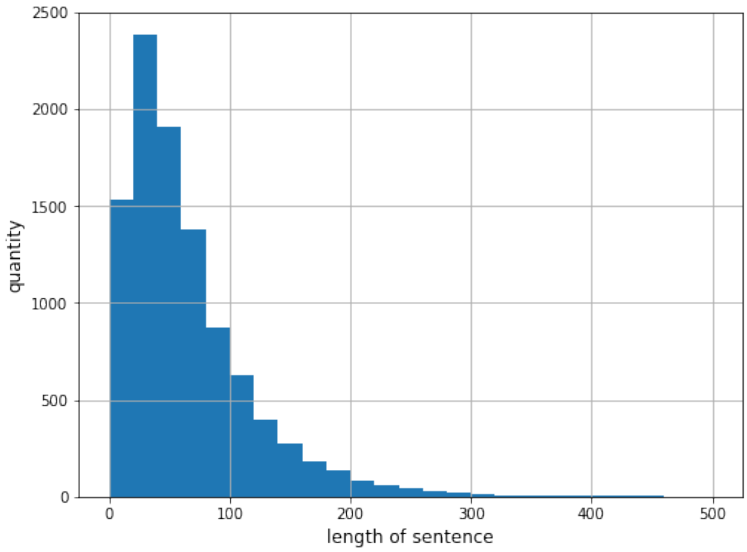

Figure 2. Length of sentence

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train_word)

max_length = 150
X_train = pad_sequences(tokenizer.texts_to_sequences(X_train_word),
    maxlen=max_length)
X_test = pad_sequences(tokenizer.texts_to_sequences(X_test_word),
    maxlen=max_length)
```

- shape of training data: (8000, 150)
- shape of test data: (2000, 150)

Word	my	wife	took	me	here	⋯
Index		384	135			⋯
Padding	$[0, 0, \cdots, 0, 384, 135, \cdots]$					

Table 1. Example of word index

3.4.2 Word Embedding Model

- word embedding model 有 CBOW、skip-gram 兩種架構，我們採用 skip-gram
- gensim 裡有許多 pretrain 好的模型，但實驗後發現 accuracy 並不高 (7 成左右)，因此選擇自己 train word embedding model
- NLP 的研究者通常會將 embedding 的維度設成 200 維到 300 維，我們折衷選擇 250 維

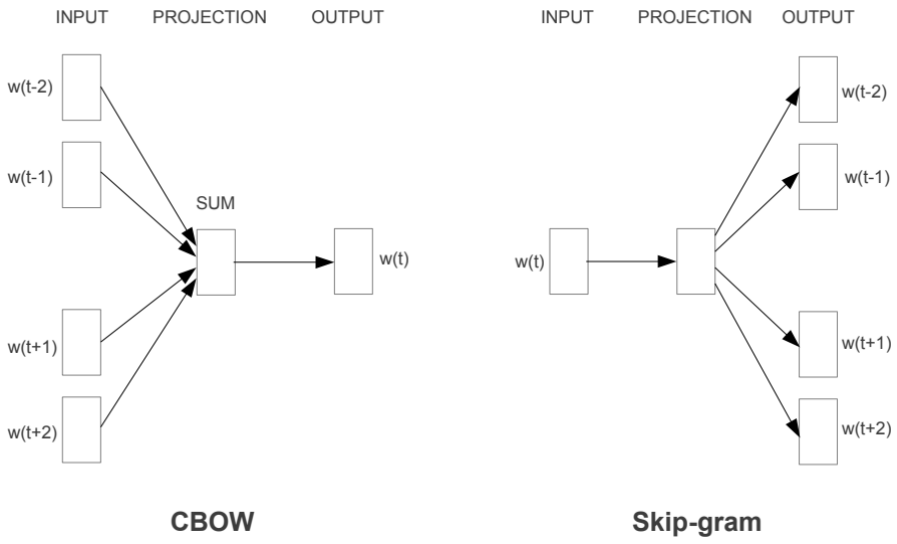

Figure 3. Word embedding model

```
embedding_dim = 250
w2v_model = Word2Vec(X_train_word, min_count=1, size=embedding_dim, iter=10, sg=1)
```

Word	wife	took
Similar word	husbands	taking
Cos-similarity	0.710834	0.671477

Table 2. Example of similar word

3.4.3 建構 Embedding Matrix

將所有的單字做 embedding，並依序排列成 25873×250 大小的矩陣，其中 25873 為單字的數量 (最大的 index)，250 為 embedding 的維度

```
vocab_size = len(tokenizer.word_index)+1
embedding_matrix = np.zeros((vocab_size, max_length))
for word, i in tokenizer.word_index.items():
  if word in w2v_model.wv:
    embedding_matrix[i] = w2v_model.wv[word]
```

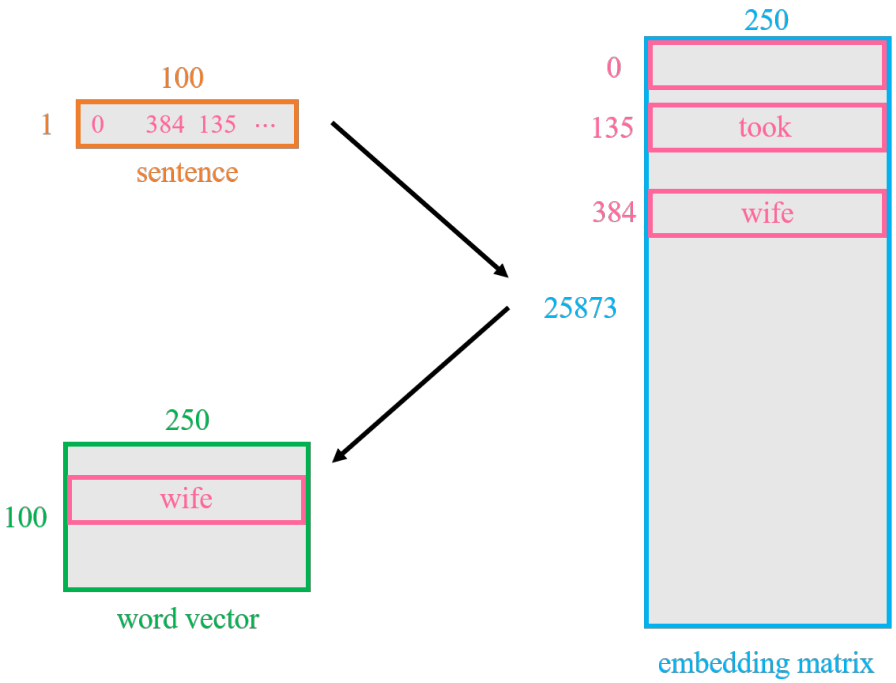

Figure 4. Embedding matrix

4 模型建立及預測

4.1 Convolutional Neural Network (CNN)

```
cnn_model = Sequential()

embedding_layer = Embedding(
    vocab_size, embedding_dim,
    weights=[embedding_matrix],
    input_length=max_length,
    trainable=False)

cnn_model.add(embedding_layer)
cnn_model.add(Dropout(0.5))
cnn_model.add(Conv1D(128, 16, activation='relu'))
cnn_model.add(Dropout(0.2))
cnn_model.add(Conv1D(128, 16, activation='relu'))
cnn_model.add(Dropout(0.2))
cnn_model.add(Conv1D(128, 16, activation='relu'))
cnn_model.add(Flatten())
cnn_model.add(Dense(2, activation='softmax'))
```

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 150, 250)	6468250
dropout (Dropout)	(None, 150, 250)	0
conv1d (Conv1D)	(None, 135, 128)	512128
dropout_1 (Dropout)	(None, 135, 128)	0
conv1d_1 (Conv1D)	(None, 120, 128)	262272
dropout_2 (Dropout)	(None, 120, 128)	0
conv1d_2 (Conv1D)	(None, 105, 128)	262272
flatten (Flatten)	(None, 13440)	0
dense (Dense)	(None, 2)	26882

Total params: 7,531,804
Trainable params: 1,063,554
Non-trainable params: 6,468,250

Figure 5. CNN model summary

```
def lr_schedule(epoch, lr):
  if epoch % 5 == 4:
    lr *= 0.8

  return lr

callback = tf.keras.callbacks.LearningRateScheduler(lr_schedule)
```

```
cnn_model.compile(
    loss='sparse_categorical_crossentropy',
    optimizer="adam",
    metrics=['accuracy'])

cnn_record = cnn_model.fit(
    X_train, y_train,
    batch_size=1024, epochs=30,
    validation_split=0.2,
    callbacks=[callback], verbose=1)
```

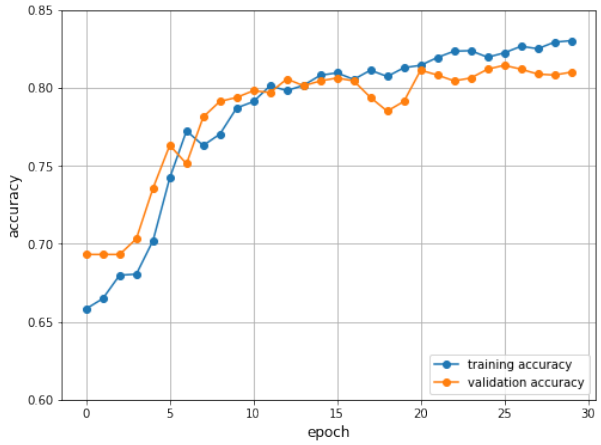

Figure 7. Accuracy of CNN w/ dropout

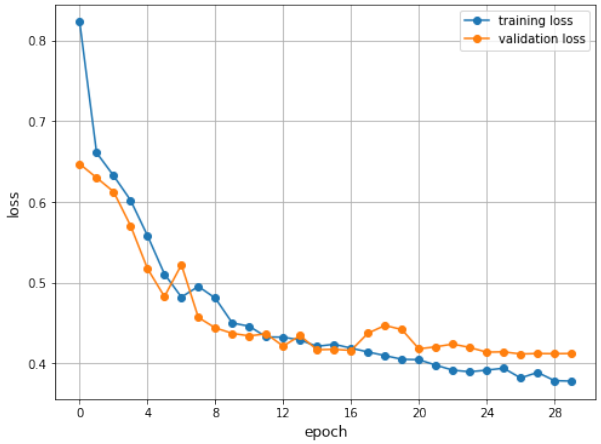

Figure 8. Loss of CNN w/ dropout

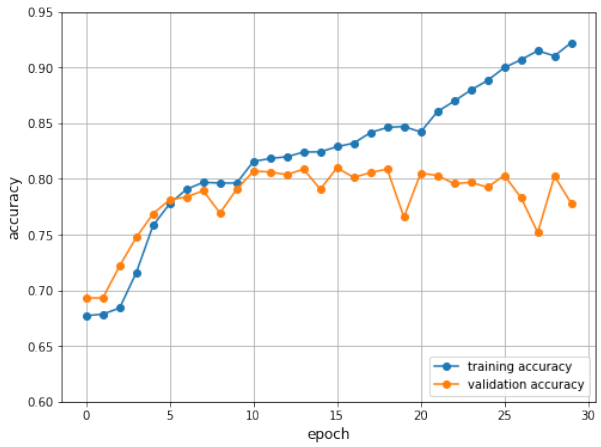

Figure 9. Accuracy of CNN w/o dropout

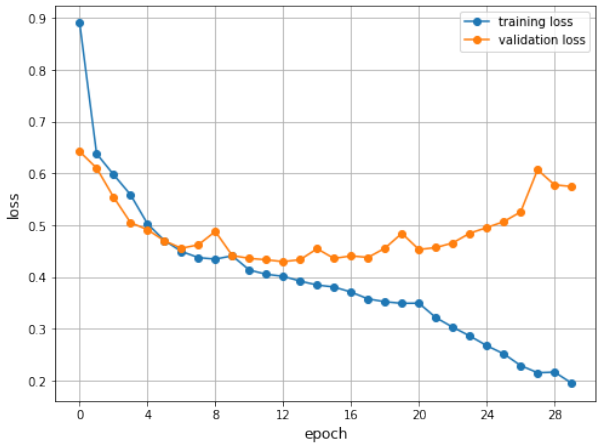

Figure 10. Loss of CNN w/o dropout

首先我們採用 CNN model 進行訓練，因為 model 本身不大，用 GPU 訓練快速，這裡我選擇不做 maximum pooling 防止特徵的損失。為了讓參數做最佳化到最佳的地方，我們採取 learning rate schedule，在最佳化的過程中，調整 learning rate，希望能收斂到好的 minimum。我們還比較加入 dropout 的結果，從上圖可以發現，沒有加 dropout 到 20 個 epoch 後產生嚴重的 overfitting，有加 dropout 的曲線會比較穩定。

4.2 Long Short Term Memory (LSTM)

```
lstm_model = Sequential()

embedding_layer = Embedding(
    vocab_size, embedding_dim,
    weights=[embedding_matrix],
    input_length=max_length, trainable=False)

lstm_model.add(embedding_layer)
lstm_model.add(Dropout(0.5))
lstm_model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
lstm_model.add(Dense(1, activation='sigmoid'))
```

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 150, 250)	6468250
dropout_3 (Dropout)	(None, 150, 250)	0
lstm (LSTM)	(None, 100)	140400
dense_1 (Dense)	(None, 1)	101

Total params: 6,608,751
Trainable params: 140,501
Non-trainable params: 6,468,250

Figure 6. LSTM model summary

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

lstm_record = lstm_model.fit(X_train, y_train, batch_size=1024, epochs=30,
    validation_split=0.2, verbose=1)
```

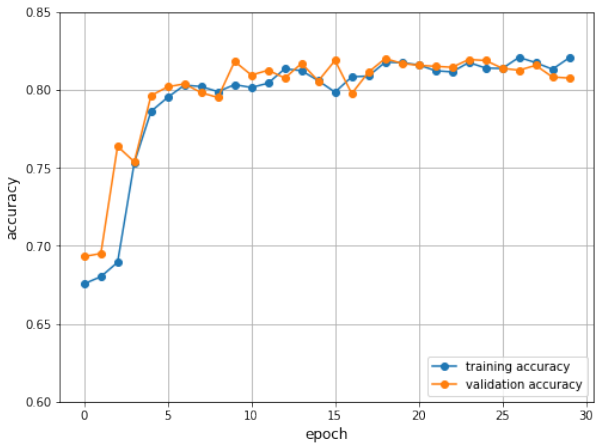

Figure 11. Accuracy of LSTM w/ dropout

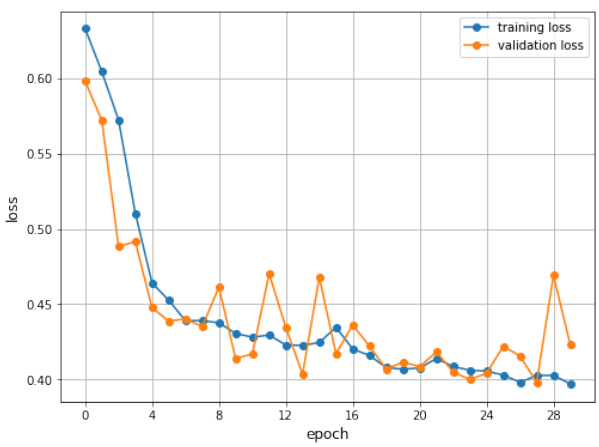

Figure 12. Loss of LSTM w/ dropout

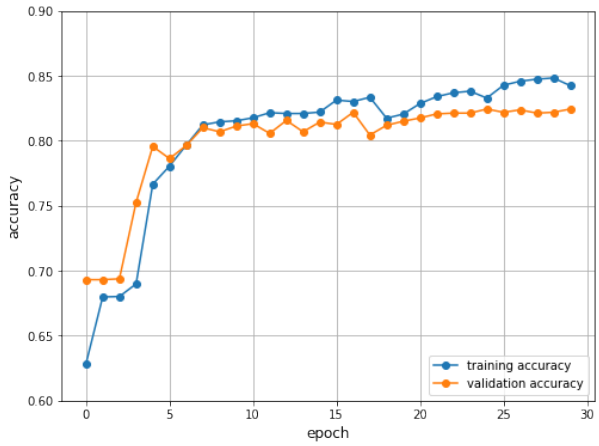

Figure 13. Accuracy of LSTM w/o dropout

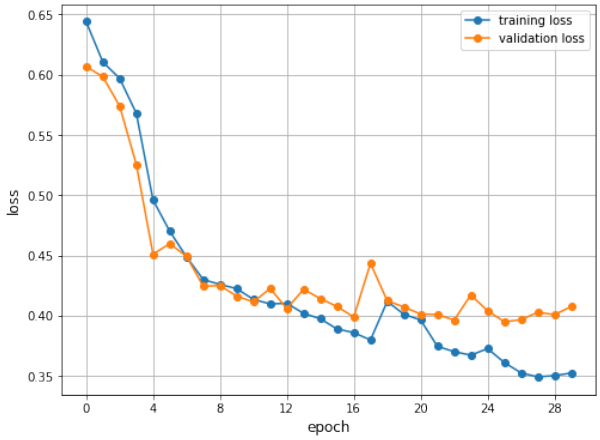

Figure 14. Loss of LSTM w/o dropout

再來我們使用 LSTM model，因為 LSTM 的 error surface 比較崎嶇，導致 loss 比較不穩定，從上圖可以發現 LSTM 的 loss 曲線震盪較 CNN 大。比較 dropout 加入後的改變，雖然從上圖可以發現沒加 dropout 的 validation accuracy 會稍微高一點，這個可能是 initial 不同所導致的，不過比較 training 與 validation 兩條曲線可以發現有加 dropout 會使得兩條曲線比較貼近，而沒加 dropout 的則是從 20 個 epoch 差異逐漸增大，如果再 train 下去，就會 overfitting 了。

5 模型評估

我們將 CNN 與 LSTM model train 了 30 個 epoch 進行比較

Model	CNN	LSTM
parameter	1,063,554	140,501
epoch	30	30
training time	90 (s)	210 (s)
test accuracy (w/o dropout)	0.7785	0.8185
test accuracy (w/ dropout)	0.8160 (+0.0375)	0.8035 (−0.015)

Table 3. Accuracy of test set

Test set	Predicted Positive	Predicted Negative
Actual Positive	1277	126
Actual Negative	242	355

Table 4. Confusion matrix of CNN prediction w/ dropout

Test set	Predicted Positive	Predicted Negative
Actual Positive	1356	47
Actual Negative	346	251

Table 5. Confusion matrix of LSTM prediction w/ dropout

Model	Precision	Recall	Accuracy
CNN w/ dropout	0.9102	0.8407	0.8160
LSTM w/ dropout	0.9665	0.7967	0.8035

Table 6. Evaluation of models

6 總結

這次作業實作 CNN 與 LSTM 對 yelp 資料集進行好感度預測。首先使用 word embedding model，將文字轉成向量，我們發現自己 train 的 model 比使用 pretrain 在最後能得到更好的 performance，我們猜測這個和當初 pretrain 的 dataset 很有關，這次的資料集為美食評論，和大多數資料集不一樣。

關於效能，雖然 CNN 的參數量比 LSTM 多很多，但在使用 GPU 下，CNN 比 LSTM 更快速，收斂也比較穩，不過根據 NLP 領域的經驗，LSTM 若能使用適當的參數與好的 initial weight，performance 通常能超過 CNN (可見好的參數可遇不可求 XD)。