Data mining course – lab2 – report

1. Data preprocessing:

(1) 首先把資料在 python 中打開，格式是 json 檔，用 list 的形式把 json 檔的 dictionary 儲存起來。

```
import pandas as pd
import json
from json import load


emotion = pd.read_csv('./emotion.csv')
data_identification = pd.read_csv('./data_identification.csv')
file = open("tweets_DM.json",'r',encoding='utf-8')
raw_data_json_all = []
for line in file.readlines():
    dic = json.loads(line)
    raw_data_json_all.append(dic)


raw_data_tweet = []
for i in raw_data_json_all:
    temp = i['_source']
    raw_data_tweet.append(temp)


raw_data_mix = []
for j in raw_data_tweet:
    temp2 = j['tweet']
    raw_data_mix.append(temp2)
```

(2) 再來把資料整理成進行 data preprocessing，把一些不是重要的符號過濾掉，以避免影響分析

```
def strip_all_entities(text):
    text = text.replace('\r', '').replace('\n', ' ').replace('\n', ' ').lower() #remove \n and \r and lowercase
    text = re.sub(r"(?:\@|https?\://)\S+", "", text) #remove links and mentions
    text = re.sub(r'[^\x00-\x7f]',r'', text) #remove non utf8/ascii characters such as '\x9a\x91\x97\x9a\x97'
    banned_list= string.punctuation + '’'+'‘'+'…'+'”'+'“'+'’'+'–'
    table = str.maketrans('', '', banned_list)
    text = text.translate(table)
    return text

#clean hashtags at the end of the sentence, and keep those in the middle of the sentence by removing just the # symbol
def clean_hashtags(tweet):
    new_tweet = " ".join(word.strip() for word in re.split('#(?!(?:hashtag)\b)[\w-]+(?=(?:\s+#[\w-]+)*\s*$)', tweet)) #remove last hashtags
    new_tweet2 = " ".join(word.strip() for word in re.split('#|_', new_tweet)) #remove hashtags symbol from words in the middle of the sentence
    return new_tweet2

#Filter special characters such as & and $ present in some words
def filter_chars(a):
    sent = []
    for word in a.split(' '):
        if ('$' in word) | ('&' in word):
            sent.append('')
        else:
            sent.append(word)
    return ' '.join(sent)

def remove_mult_spaces(text): # remove multiple spaces
    return re.sub("\s\s+" , " ", text)
```

(3) 資料過濾完以後，要避免出現沒有意義的字，所以先計算句子的長度，

並把長度小於 4 的句子給過濾掉

```
[19]
text_len_test = []
for text in predict_dataset.text_clean:
    tweet_len = len(text.split())
    text_len_test.append(tweet_len)

[20]
predict_dataset['text_len'] = text_len_test

[21]
print(f" DF SHAPE: {train_dataset.shape}")
print(f" DF TEST SHAPE: {predict_dataset.shape}")

...
 DF SHAPE: (1455563, 5)
 DF TEST SHAPE: (411972, 4)

[22]
df_train = train_dataset[train_dataset['text_len'] > 4]
df_test = predict_dataset[predict_dataset['text_len'] > 4]

[23]
print(f" DF SHAPE: {train_dataset.shape}")
print(f" DF TEST SHAPE: {predict_dataset.shape}")

...
 DF SHAPE: (1455563, 5)
 DF TEST SHAPE: (411972, 4)
```

(4) 再來把資料整理成 dataframe

```
[4]:
display(train_dataset)
```

	tweet_id	emotion	text	text_clean	text_len
0	0x3140b1	sadness	Why Chester? <LH> <LH>	why chester lh lh	4
1	0x368b73	disgust	@JaredLeto you are the fish that Jonah. Excep...	you are the fish that jonah except youre a fuc...	12
2	0x296183	anticipation	He is coming back again and gonna come again q...	he is coming back again and gonna come again q...	12
3	0x2bd6e1	joy	Dei is really such a beautiful person inside &...	dei is really such a beautiful person inside o...	24
4	0x2ee1dd	anticipation	Expressive praise is also an expression of fai...	expressive praise is also an expression of fai...	24
...	...	...	...	...	...
1455558	0x38dba0	joy	We would like to thank Errbody for wishing our...	we would like to thank errbody for wishing our...	21
1455559	0x300ea2	joy	@CherylShuman Thank Ms. Cheryl for retweeting ...	thank ms cheryl for retweeting michigan could ...	13
1455560	0x360b99	fear	Finally watching the new episode of <LH>	finally watching the new episode of lh	7
1455561	0x22eecf	joy	Doctor: I'm afraid you're going to need a tran...	doctor im afraid youre going to need a transpl...	22
1455562	0x2fb282	anticipation	What I wouldn't give for a @MacKinnonBrew grow...	what i wouldnt give for a growler right about ...	12

1455563 rows × 5 columns

(5) 如此 data preprocessing 就算完成了

2. Feature engineering:

(1) 進行 tokenizer

```
import nltk

# build analyzers (bag-of-words)
BOW_500 = CountVectorizer(max_features=500, tokenizer=nltk.word_tokenize)

# apply analyzer to training data
BOW_500.fit(train_df['text_clean'].astype('U'))

train_data_BOW_features_500 = BOW_500.transform(train_df['text_clean'].astype('U'))

## check dimension
train_data_BOW_features_500.shape
```

```
# for a classificaiton problem, you need to provide both training & testing data
X_train = BOW_500.transform(train_df['text_clean'].astype('U'))
y_train = train_df['emotion']

X_test = BOW_500.transform(test_df['text_clean'].astype('U'))
y_test = test_df['emotion']

## take a look at data dimension is a good habit  :)
print('X_train.shape: ', X_train.shape)
print('y_train.shape: ', y_train.shape)
print('X_test.shape: ', X_test.shape)
print('y_test.shape: ', y_test.shape)
```

(2) 把 token 進行 one-hot encoding

```
## deal with label (string -> one-hot)
import keras
import keras.utils
from keras import utils as np_utils
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
label_encoder.fit(y_train)
print('check label: ', label_encoder.classes_)
print('\n## Before convert')
print('y_train[0:7]:\n', y_train[0:7])
print('\ny_train.shape: ', y_train.shape)
print('y_test.shape: ', y_test.shape)

def label_encode(le, labels):
    enc = le.transform(labels)
    return keras.utils.to_categorical(enc)

def label_decode(le, one_hot_label):
    dec = np.argmax(one_hot_label, axis=1)
    return le.inverse_transform(dec)

y_train = label_encode(label_encoder, y_train)
y_test = label_encode(label_encoder, y_test)

print('\n\n## After convert')
print('y_train[0:7]:\n', y_train[0:7])
print('\ny_train.shape: ', y_train.shape)
print('y_test.shape: ', y_test.shape)
```

3. Mode explanation:

(1) 建立神經網路

使用最基本的神經網路進行 classification，hidden layer 設定了三層，並使用 leakyrelu，比起 relu 有更好的表現，最後用 softmax 完成分類的任務，optimizer 使用 AdamW，因為其具有 decay 的效果，是 adam 的優化版，loss 則是用 entropy 來判斷，整體模型的結構大概如下圖。

```
from keras.models import Model
from keras.layers import Input, Dense
from keras.layers import ReLU, Softmax
# input layer
model_input = Input(shape=(input_shape, ))  # 500
X = model_input
X_W0 = Dense(units=256)(X)  # 256
H0 = LeakyReLU()(X_W0)
# 1st hidden layer
X_W1 = Dense(units=128)(H0)  # 128
H1 = LeakyReLU()(X_W1)
# 2nd hidden layer
H1_W2 = Dense(units=64)(H1)  # 64
H2 = LeakyReLU()(H1_W2)
# output layer
H2_W3 = Dense(units=output_shape)(H2)  # 8
H3 = Softmax()(H2_W3)
model_output = H3
# create model
model = Model(inputs=[model_input], outputs=[model_output])
# loss function & optimizer
model.compile(optimizer=keras.optimizers.AdamW(),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
# show model construction
model.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 500)]             0

 dense_4 (Dense)             (None, 256)               128256

 leaky_re_lu_3 (LeakyReLU)   (None, 256)               0

 dense_5 (Dense)             (None, 128)               32896

 leaky_re_lu_4 (LeakyReLU)   (None, 128)               0

 dense_6 (Dense)             (None, 64)                8256

 leaky_re_lu_5 (LeakyReLU)   (None, 64)                0

 dense_7 (Dense)             (None, 8)                 520

 softmax_1 (Softmax)         (None, 8)                 0

=================================================================
Total params: 169928 (663.78 KB)
Trainable params: 169928 (663.78 KB)
Non-trainable params: 0 (0.00 Byte)
```

(2) 進行 training

```
# training setting
epochs = 20
batch_size = 32

# training!
history = model.fit(X_train, y_train,
                    epochs=epochs,
                    batch_size=batch_size,
                    validation_data = (X_test, y_test))
print('training finish')
```

(3) Training 完成以後，進行預測

[23]:
```
## predict
pred_test = model.predict(X_test, batch_size=128)
pred_test[:5]
```

```
2/2 [==============================] - 0s 4ms/step
```

[23]:
```
array([[0.02021115, 0.20385256, 0.09527053, 0.03063533, 0.36522603,
        0.10942964, 0.02884967, 0.14652519],
       [0.00652276, 0.18279281, 0.04406465, 0.05112736, 0.2763366 ,
        0.08413881, 0.02034392, 0.33467302],
       [0.02988647, 0.16422996, 0.13352525, 0.03836708, 0.29311782,
        0.14154156, 0.0392299 , 0.16010188],
       [0.00219282, 0.04376331, 0.00853515, 0.00724519, 0.38742498,
        0.02207759, 0.0078688 , 0.52089214],
       [0.00152432, 0.09448581, 0.00579758, 0.02753259, 0.57002366,
        0.02558098, 0.01306097, 0.26199418]], dtype=float32)
```

(4) 最後把它弄成可以繳交的格式即完成了

```
sample_sub= pd.read_csv('/kaggle/input/sample-submission/sampleSubmission.csv')
```

```
prediction_raw = pred_result
```

```
print(type(prediction_raw))
```

```
<class 'numpy.ndarray'>
```

```
predict_dataset_outcome = predict_dataset
predict_dataset_outcome['emotion'] = prediction_raw
predict_dataset_outcome.drop(columns='text',inplace=True)
predict_dataset_outcome.drop(columns='text_clean',inplace=True)
predict_dataset_outcome.drop(columns='text_len',inplace=True)
```

```
predict_dataset_outcome.rename(columns={'tweet_id': 'id'},inplace=True)
```

[33]:
```
sample_sub.drop(columns='emotion',inplace=True)
```

[34]:
```
predict_dataset_outcome
```

[34]:

	id	emotion
0	0x28b412	joy
1	0x2de201	anticipation
2	0x218443	joy
3	0x2939d5	joy
4	0x26289a	anticipation
...	...	...
411967	0x2913b4	joy
411968	0x2a980e	anticipation
411969	0x316b80	disgust
411970	0x29d0cb	joy
411971	0x2a6a4f	sadness

411972 rows × 2 columns

35]:
```
sample_sub
```

35]:

	id
0	0x2c7743
1	0x2c1eed
2	0x2826ea
3	0x356d9a
4	0x20fd95
...	...
411967	0x351857
411968	0x2c028e
411969	0x1f2430
411970	0x2be24e
411971	0x35802a

411972 rows × 1 columns

+ Code + Markdown

36]:
```
submission1 = pd.merge(sample_sub,predict_dataset_outcome,how='left',on='id')
```

[37]:
```
submission1
```

[37]:

	id	emotion
0	0x2c7743	joy
1	0x2c1eed	joy
2	0x2826ea	joy
3	0x356d9a	joy
4	0x20fd95	joy
...	...	...
411967	0x351857	joy
411968	0x2c028e	disgust
411969	0x1f2430	anticipation
411970	0x2be24e	joy
411971	0x35802a	joy

411972 rows × 2 columns

+ Code + Markdown

[38]:
```
submission1.to_csv("submission.csv", index=False)
```

4. Future insight:

(1) 使用 pre-train 模型進行資料訓練:
在網路上有看到很多人使用 pre-train 模型進行 sentiment analysis，像是 BERT、Glove 之類的，但如何使用該模型我甚是不熟，如果下次有時間我想我可以鑽研看看，應該會讓精準度有所提升。

(2) 對於 tokenize 的方法應該還有更適合這個 dataset 的方式，像是可以使用 BERT tokenizer 並且和 BERT 一起搭配使用；或是使用 word2vec 的方式讓模型去理解 text，都不失為一個好方法。

(3) 在此次訓練模型的過程中，我沒有使用圖表來觀察訓練過程的 loss 及精準度，這樣無法判斷模型是否有 overfitting 的現象，如果能再重來一次，我想我會把視覺化這個部分弄好一些，這樣我對於模型的修改會更有方向。

(4) 我在這次作業一開始是使用 decision tree 和 random forest 的方式，用分類的概念去解這道題目，但我發現效果不是很好，分數才 0.28 多而已，我想情緒分系用 NLP 的方法去處理效果應該更好。

(5) Data preprocessing 我參考很多人的做法，但我發現大家的做法都大同小異，就跟老師教的重點差不多。值得一提的是，在經過 data preprocessing 以後，我的資料出現了一些空值但我沒有發現，導致我苦惱了一些時間，因為後續的分析一直報錯，我一直找不到問題在哪，幸好最後看很久的 stackoverflow 才找到問題。