# HW6

A. 組員

a. 統計113 H24096037 張幼澄

b. 統計113 H24091299 張瑞恩

c. 統計113 H24091223 陳彥亨

B. 競賽敘述與目標

a. 我們要做的是有關仇恨言論的推特貼文的語言處理，目標是判斷推特上的貼文是否是含有仇恨言論的內容，含有仇恨言論的內容中又分為Hateful與Offensive兩類。

C. 資料前處理

a. 將不具有意義的推特貼文內容去除：Stopwords, Emoji, html, url,” rt”

b. 將大寫轉成小寫

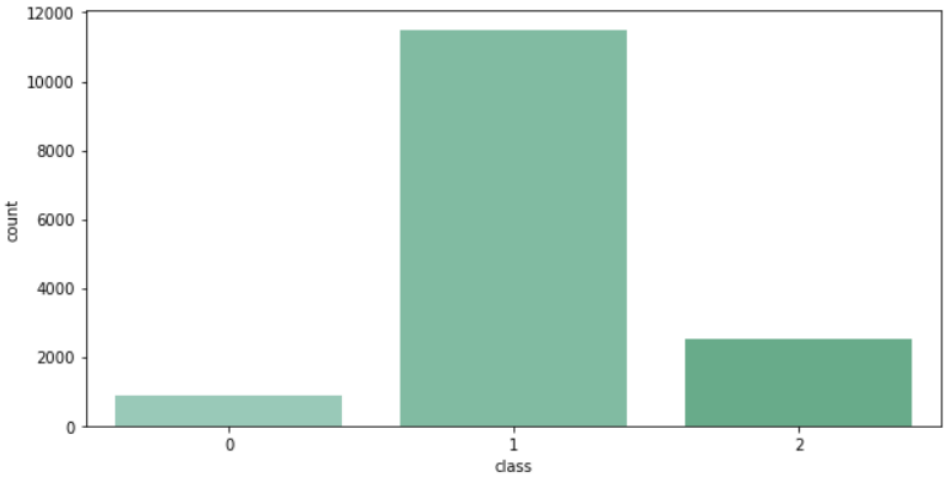
```
def remove_url(text):
    url = re.compile(r'https?://\S+|www\.\S+')
    return url.sub(r'', text)


def remove_emoji(text):
    emoji_pattern = re.compile(
        '['
        u'\U0001F600-\U0001F64F'  # emoticons
        u'\U0001F300-\U0001F5FF'  # symbols & pictographs
        u'\U0001F680-\U0001F6FF'  # transport & map symbols
        u'\U0001F1E0-\U0001F1FF'  # flags (iOS)
        u'\U00002702-\U000027B0'
        u'\U000024C2-\U0001F251'
        ']+',
        flags=re.UNICODE)
    return emoji_pattern.sub(r'', text)

def remove_html(text):
    html = re.compile(r'<.*?>|&([a-z0-9]+|#[0-9]{1,6}|#x[0-9a-f]{1,6});')
    return re.sub(html, '', text)

def clean_text(text):
    '''Make text lowercase, remove text in square brackets,remove links,remove punctuation
    and remove words containing numbers.'''
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub(
        'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+',
        '',
        text
```

D. 特徵處理與分析

a. 可以看出Offensive佔資料中的大多數

```
plt.figure(figsize=(10,5))
sns.set_palette(sns.color_palette("BuGn_d",8))
sns.countplot(x="class",data=train)
```

```
<AxesSubplot:xlabel='class', ylabel='count'>
```




b. 各分群中的出現頻率最高的詞

i. Hateful

```
In [10]: from collections import defaultdict
corpus_disaster_tweets = create_corpus_df(train, 1)

dic=defaultdict(int)

for word in corpus_disaster_tweets:
    dic[word]+=1

top_real=sorted(dic.items(), key=lambda x:x[1],reverse=True)[:10]
top_real
```

```
Out[10]: [('bitch', 6485),
 ('hoe', 2460),
 ('like', 1450),
 ('pussi', 1254),
 ('fuck', 1181),
 ('nigga', 1104),
 ('dont', 878),
 ('ass', 874),
 ('get', 857),
 ('shit', 736)]
```

ii. Offensive

```
from collections import defaultdict
corpus_disaster_tweets = create_corpus_df(train, 1)

dic=defaultdict(int)

for word in corpus_disaster_tweets:
    dic[word]+=1

top_real=sorted(dic.items(), key=lambda x:x[1],reverse=True)[:10]
top_real
```

```
[('bitch', 6485),
 ('rt', 3607),
 ('hoe', 2460),
 ('like', 1450),
 ('pussi', 1254),
 ('fuck', 1181),
 ('nigga', 1104),
 ('dont', 878),
 ('ass', 874),
 ('get', 857)]
```

iii. Clean

```
corpus_disaster_tweets = create_corpus_df(train, 2)

dic=defaultdict(int)
for word in corpus_disaster_tweets:
    dic[word]+=1

top_fake=sorted(dic.items(), key=lambda x:x[1],reverse=True)[:10]
top_fake
```

```
[('rt', 817),
 ('trash', 426),
 ('bird', 281),
 ('yanke', 207),
 ('like', 167),
 ('charli', 167),
 ('yellow', 146),
 ('get', 125),
 ('dont', 102),
 ('one', 97)]
```

E. 預測訓練模型

a. Pipeline

i. Pipeline 可以把多個“處理數據的節點”按順序打包在一起，數據在前一個節點處理之後的結果，轉到下一個節點處理。除最後一個節點外，其他節點都必須實現’ fit()’ 和’ transform()’ 方法， 最後一個節點需要實現fit()方法即可。當訓練樣本數據送進Pipeline進行處理時， 它會逐個調用節點的fit()和transform()方法，然後點用最後一個節點的fit()方法來擬合數據。

ii. 我們使用 CountVectorizer, TfidfTransformer, XGBoost ，這三個模型

iii. 使用 x_train 訓練模型後，再用訓練過後的模型去預測Test數據中的Target

```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
import xgboost as xgb

pipe = Pipeline([
    ('bow', CountVectorizer()),
    ('tfid', TfidfTransformer()),
    ('model', xgb.XGBClassifier(
        learning_rate=0.1,
        max_depth=7,
        n_estimators=100,
        use_label_encoder=False,
        eval_metric='auc',
    ))
])
```

iv. 從Train當中切割出來的Train 跟Test的 Accuracy 數據分別為0.918跟0.899

```
# Fit the pipeline with the data
pipe.fit(x_train, y_train)

y_pred_class = pipe.predict(x_test)
y_pred_train = pipe.predict(x_train)

print('Train: {}'.format(metrics.accuracy_score(y_train, y_pred_train)))
print('Test: {}'.format(metrics.accuracy_score(y_test, y_pred_class)))
```

```
Train: 0.9181239350730876
Test: 0.899677245831092
```

b. LSTM

i. 先將class為1或0的合併為1，class為2的編為0

ii. 再用Tokenizer將剩下的文字轉成數字，並將每列的長度單一化。

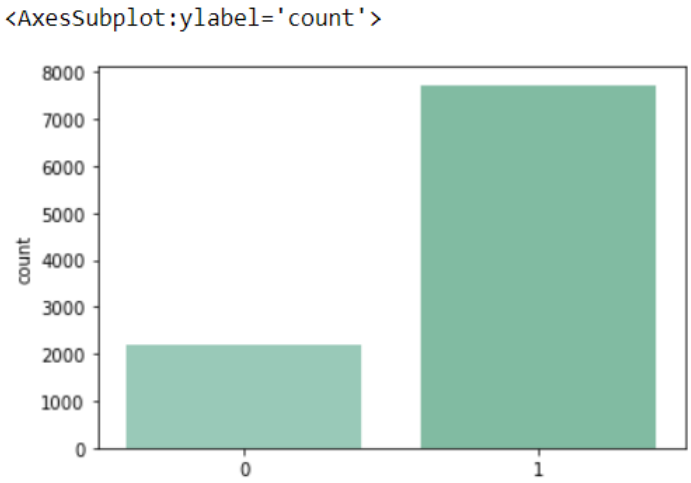
iii. 從GloVe 官網上下載語料庫(將許多常用單字編為100維的語料庫)，並與有出現在推文中的詞作比較，最後生成Embedding matrix。

iv. 建立LSTM 的模型並訓練

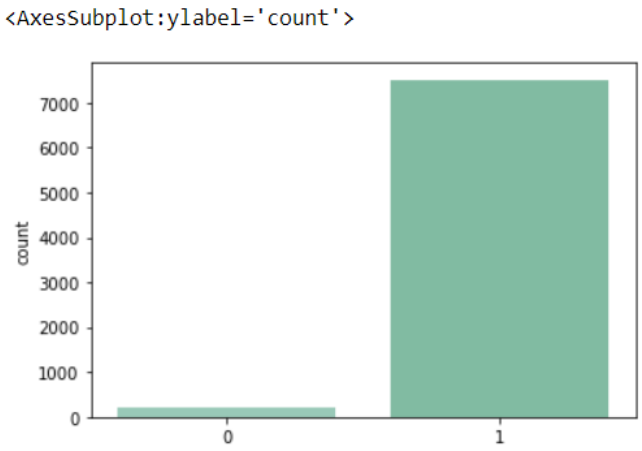
```
def glove_lstm():
    model = Sequential()

    model.add(Embedding(
        input_dim=embedding_matrix.shape[0],
        output_dim=embedding_matrix.shape[1],
        weights = [embedding_matrix],
        input_length=length_long_sentence
    ))
    model.add(Bidirectional(LSTM(
        length_long_sentence,
        return_sequences = True,
        recurrent_dropout=0.2
    )))

    model.add(GlobalMaxPool1D())
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(length_long_sentence, activation = "relu"))
    model.add(Dropout(0.5))
    model.add(Dense(length_long_sentence, activation = "relu"))
    model.add(Dropout(0.5))
    model.add(Dense(1, activation = 'sigmoid'))
    model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])

    return model

model = glove_lstm()
model.summary()
```

```
model=glove_lstm()

from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau

checkpoint = ModelCheckpoint(
    'model.h5',
    monitor = 'val_loss',
    verbose = 1,
    save_best_only = True
)
reduce_lr = ReduceLROnPlateau(
    monitor = 'val_loss',
    factor = 0.2,
    verbose = 1,
    patience = 5,
    min_lr = 0.001
)
history = model.fit(
    X_train,
    y_train,
    epochs = 6,
    batch_size = 32,
    validation_data = (X_test, y_test),
    verbose = 1,
    callbacks = [reduce_lr, checkpoint]
)
```

v. 獲得預測的結果(此時的結果為1與0, 1 為hateful與offensive的, 0為clean)




vi. 再訓練一次模型，將1的結果分為hateful跟offensive (0為hateful,1 為offensive)




vii. 合併數據

F. 預測結果分析

| | HateF | ALLF | Final |
|---|---|---|---|
| Pipeline | 0.7108 | 0.7108 | 0.7108 |
| LSTM | 0.699 | 0.699 | 0.699 |

G. 感想與心得

a. 統計113 H24091223 陳彥亨

這次的作業是有關於如何判斷推特貼文有沒有包含攻擊性言論。重點就是先找出三種不同言論（hateful, offensive, clean)中出現評率最高的一些詞，並將其當成關鍵字，隨後再透過不同模組去預測最後的結果，當然其中有透過一些語言處理的方式（例如tokenizer 跟GloVe去為文字編碼跟做比較）以及設定參數去提高準確性。其實這次的報告跟我們的final project 蠻類似的，但比較不一樣的是這次的作業在分類上又多了一項，所以該如何在模組中設定問題中的0、1、2便是一個問題點，因為模組基本上都是以0跟1做區分，我們的作法則是將clean 言論與其他兩種分開先預測一次，接下來在offensive 與hateful其中再預測一次。那因為這次作業期間我比較忙，所以基本上都是另外兩位組員在處理，還請老師斟酌扣分。

b. 統計113 張瑞恩

這次的作業剛好跟期末做的報告一樣，都是屬於關於推特內容的分群判斷，所以在資料前處理的時候就比上一次的作業好上手許多，像是處理表情符號，網址及停用詞，都可以根據上次的報告來做，節省不少時間。

這次的作業讓我學習到了怎麼透過機器學習的方法來分析文字訊息，這次學習到的東西與之前課程中的網路爬蟲結合，就可以從社群網站中的各種貼文和留言了解當前的趨勢，所以我覺得這次的報告所學到的技巧和知識非常的實用，值得好好學習。

c. 統計113 張幼澄

這次作業六的題目剛好跟我們的Final Project 類似，都是處理字串並去訓練模型，但我認為更難了一些，因為資料的結果總共有三個，不是二元分法，因此在準確度上面跟作業五的數據就差了很多，當中也遇到許多瓶頸，不知道該如何提高數據，也找不到更好參數去提升準確度。不過我覺得LSTM這個模型十分的酷炫，可以從自己的數據當中不斷去刪除掉不準的東西，提升準確度，是個不錯用的模型，感覺之後可以花更多時間去了解各個參數的意義。

這次因為剛好卡到期末考還有Final Project的Deadline，沒有花太多時間在這次的作業上，但我認為還是學到不少東西，這次的競賽十分有趣，相信未來會有機會可以用到這堂課所學的東西。