推特仇恨言論偵測

何彩綺、黃振維、李若瑜

本次競賽以偵測推特(Twitter)上的仇恨言論為目標，將言論分為三種：Hateful(y=0)、Offensive(y=1)、Clean(y=2)。本組將分為三種方法，並訓練出幾種模型對言論分論進行預測。

首先列出 train.csv 資料集中每一列 tweet 內容，並判斷其文字長度。

```
# Adding text-length as a field in the df_train
# apply 是一個在 pandas dataframe 加入新列(Column)的指令
df_train['text length'] = df_train['tweet'].apply(len)
df_train
# print(df_train.head())
```

	class	tweet	text length
0	1	[9-1-13] 2:50 pm "son of a bitch ate my mac n ...	76
1	1	RT @BryceSerna: Don't be a pussy grab the boot...	86
2	2	RT @ClicquotSuave: bunch of rappers boutta flo...	79
3	2	@michigannews13 wow. Thats great language comi...	129
4	1	and this is why I'm single, I don't fuck with ...	77
...	...	...	...
14864	1	RT @Estellleeee: Lol bitches act hard when the...	102
14865	1	RT @yes_paul: I bitch about unnecessary things...	50
14866	1	RT @Im_Amy_Bitches: I'm not always a bitch, so...	61
14867	1	RT @dkiswinning: GamePlan: 1. Eat the pussy 2....	131
14868	1	RT @FeeelGreatness: You don't know where your ...	132

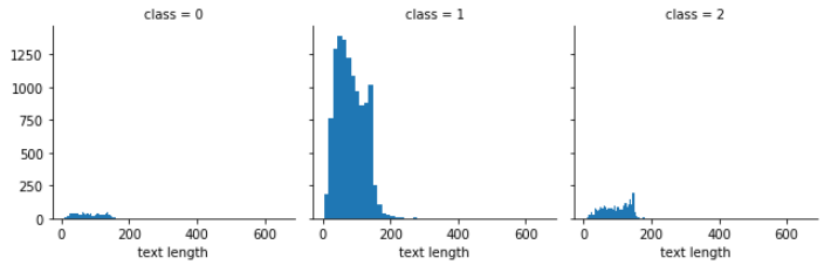
14869 rows × 4 columns

每列文字長度

依照 tweet 內容分類：class=0、class=1、class=2 別畫出直方圖與盒須圖，可以看出各分類數量與文字長度的對應關係。

```
# Basic visualization of data using histograms
# FacetGrid- Multi-plot grid for plotting conditional relationships
graph = sns.FacetGrid(data=df_train, col='class')
graph.map(plt.hist, 'text length', bins=50)
```

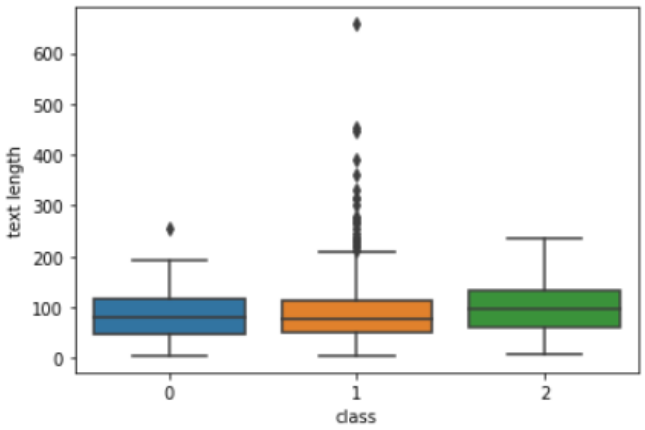
```
<seaborn.axisgrid.FacetGrid at 0x1bfdaf795b0>
```



▲ tweet 分類直方圖

```
# Box-plot visvualization
sns.boxplot(x='class', y='text length', data=df_train)
```

```
<AxesSubplot:xlabel='class', ylabel='text length'>
```



▲ tweet 分類盒須圖

看完內容分類與文字長度關係後，可以知道在 class 1 的 Offensive 內容文字敘述最長，而 class 0 的 Hateful 內容文字敘述最短。接下來我們針對文字進行處理，步驟為：

(1) Removal of punctuation and capitalization：移除標點符號並將字母都轉為小寫

(2) Tokenizing：斷詞

(3) Removal of stopwords：移除停頓詞

(4) Stemming：去除字尾

透過以上步驟清除文字中特殊符號並精簡內容，利用 processed-tweets 方便後續預測。

	tweet	processed_tweets
0	[9-1-13] 2:50 pm "son of a bitch ate my mac n ...	pm son bitch ate mac n chees
1	RT @BryceSerna: Don't be a pussy grab the boot...	pussi grab booti love booti appreci booti
2	RT @ClicquotSuave: bunch of rappers boutta flo...	bunch rapper boutta flood internet w trash remix
3	@michigannews13 wow. Thats great language comi...	wow that great languag come hs coach sure role...
4	and this is why I'm single, I don't fuck with ...	singl fuck bitch attitud foh
5	RT @__thaRealist: @Dono_44 yea that hoe was ro...	yea hoe rock friday last night
6	They should free all da real niccas n give the...	free da real nicca n give snitch da time
7	@bellaangeletti ur a fag	ur fag
8	@WendyDavisTexas @GregAbbott_TX a judge that w...	judg appoint obama doesnt see agenda fuck croo...
9	@HuntsTheWind I wonder if I have my very own N...	wonder nsa spook follow

▲ 文字處理後結果

針對處理後的文字，我們依序畫出全部 tweet 內容文字雲、Hateful 內容文字雲、Offensive 內容文字雲，可看出在該類別中常見單詞。

```
# visualizing which of the word is most commonly used in the twitter df_train
from wordcloud import WordCloud
# imshow-Display data as an image
# interpolation - https://matplotlib.org/3.2.1/gallery/images_contours_and_fields/interpolation_methods.html
all_words = ' '.join([text for text in df_train['processed_tweets'] ])
wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110).generate(all_words)
#random=0.30
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

▲ 全部 tweet 內容文字雲

```
# visualizing which of the word is most commonly used for hatred speech
hatred_words = ' '.join([text for text in df_train['processed_tweets'][df_train['class'] == 0]])
wordcloud = WordCloud(width=800, height=500,
random_state=21, max_font_size=110).generate(hatred_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

▲ Hateful 內容文字雲

```
# visualizing which of the word is most commonly used for offensive speech
offensive_words = ' '.join([text for text in df_train['processed_tweets'][df_train['class'] == 1]])
wordcloud = WordCloud(width=800, height=500,
random_state=21, max_font_size=110).generate(offensive_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

▲ Offensive 內容文字雲

最後將文字轉為向量模式，判斷該列是否具有對應單字，依據 TF(特定單字在

文件出現次數/文件總次數)與 IDF 之乘積，得到綜合分數 TFIDF。

	ab	abil	abl	abo	abort	absolut	abt	abus	accept	accid	...	young thug	your	yr	yr old	yu	yung	yup	zebra	zero	zone
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	0.0	0.0	0.0
1	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	0.0	0.0
2	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	0.0	0.0
3	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	0.0	0.0
4	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	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14864	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	0.0	0.0
14865	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	0.0	0.0
14866	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	0.0	0.0
14867	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	0.0	0.0
14868	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	0.0	0.0

14869 rows × 3997 columns

▲ 文字向量模式

(一) LogisticRegression

```
# If you don't specify the random_state in the code,
# then every time you run(execute) your code a new random value is generated
# and the train and test df_trains would have different values each time.
X = tfidf
y = df_train['class'].astype(int)
X_train_tfidf, X_test_tfidf, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.2)

model = LogisticRegression().fit(X_train_tfidf,y_train)
```

```
y_preds = model.predict(X_test_tfidf)
report = classification_report( y_test, y_preds )
print(report)
acc=accuracy_score(y_test,y_preds)
print("Logistic Regression, Accuracy Score:" , acc)
```

```
              precision    recall  f1-score   support

           0       0.73      0.09      0.16       209
           1       0.89      0.98      0.93      2867
           2       0.85      0.70      0.77       642

    accuracy                           0.88      3718
   macro avg       0.82      0.59      0.62      3718
weighted avg       0.87      0.88      0.86      3718

Logistic Regression, Accuracy Score: 0.8792361484669177
```

模型準確度 0.879

(二) Random Forest

```
y_preds = rf.predict(X_test_tfidf)
acc1=accuracy_score(y_test,y_preds)
report = classification_report( y_test, y_preds )
print(report)
print("Random Forest, Accuracy Score:",acc1)
```

```
              precision    recall  f1-score   support

           0       0.60      0.07      0.13       167
           1       0.89      0.97      0.93      2287
           2       0.84      0.72      0.78       520

    accuracy                           0.88      2974
   macro avg       0.78      0.59      0.61      2974
weighted avg       0.86      0.88      0.86      2974

Random Forest, Accuracy Score: 0.8786146603900471
```

模型準確度 0.878

(三) Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
X_train_tfidf, X_test_tfidf, y_train, y_test = train_test_split(X.toarray(), y, random_state=42, test_size=0.2)
nb=GaussianNB()
nb.fit(X_train_tfidf,y_train)
y_preds = nb.predict(X_test_tfidf)
acc2=accuracy_score(y_test,y_preds)
report = classification_report( y_test, y_preds )
print(report)
print("Naive Bayes, Accuracy Score:",acc2)
```

```
              precision    recall  f1-score   support

           0       0.14      0.25      0.18       167
           1       0.86      0.85      0.85      2287
           2       0.65      0.50      0.56       520

    accuracy                           0.76      2974
   macro avg       0.55      0.53      0.53      2974
weighted avg       0.78      0.76      0.77      2974

Naive Bayes, Accuracy Score: 0.7568930733019502
```

模型準確度 0.756

最後我們選擇準確度最高也最熟悉的 svm 模型，繼續進行分析

```
from sklearn.model_selection import train_test_split

# extract the labels from the train data
y = train['class']

# use 70% for the training and 30% for the test
x_train, x_test, y_train, y_test = train_test_split(train.clean_tweet.values, y,
                                                    stratify=y,
                                                    random_state=1,
                                                    test_size=0.3, shuffle=True)
```

先將 train 的資料 split 為 train 與 test

```
from sklearn.feature_extraction.text import CountVectorizer

# vectorize tweets for model building
vectorizer = CountVectorizer(binary=True, stop_words='english')

# learn a vocabulary dictionary of all tokens in the raw documents
vectorizer.fit(list(x_train) + list(x_test))

# transform documents to document-term matrix
x_train_vec = vectorizer.transform(x_train)
x_test_vec = vectorizer.transform(x_test)
```

再將 x_train 與 x_test transform

```
from sklearn import svm
# classify using support vector classifier
svm = svm.SVC(kernel = 'linear', probability=True)

# fit the SVC model based on the given training data
prob = svm.fit(x_train_vec, y_train).predict_proba(x_test_vec)

# perform classification and prediction on samples in x_test
y_pred_svm = svm.predict(x_test_vec)
```

```
from sklearn.metrics import accuracy_score
print("Accuracy score for SVC is: ", accuracy_score(y_test, y_pred_svm) * 100, '%')
```

```
Accuracy score for SVC is:  88.47791974893522 %
```

```
output=pd.DataFrame({'class':y_pred_svm})
output
```

最後 train 出 model 之後將 test 的資料丟入 model

何彩綺

本次是學期最後一次競賽，處理的資料類別是先前作業和競賽中沒有出現過的文字資料集，因此在前處理的部分花比較多時間。因為 tweet 內容有轉發等機制，因此會出現很多特殊文字符，再加上語言變化多端，需要考慮到很多元素並將其刪除。先前並沒有在數字以外型態的資料加以著墨，所以也找了很多方法，是蠻不錯的經驗。而回顧整學期的作業與競賽，每一次都會有不一樣的收穫，例如這次的練習讓我對文字處理更為熟悉，分析不同資料集並得出結果。是讓我獲益良多的課程，也謝謝老師與助教指教。

李若瑜

在這次的競賽中我多研究了幾種 model，比較少用但是準確度也不低，其中有常見的 logistic regression 跟 svm，其他也有比較少見的 random forest 跟 naive bayes，其中我測出準確度最高的是 svm，不過後來又遇到一個困擾我很久的問題，就是無法把 test 的資料丟進 model 裡面預測，會一直顯示筆數不符合，後來是參考組員的作法才弄出來。其實競賽本身蠻有趣的，只是當模型做出來之後一直卡在預測的部分其實有點挫折，但總結來說這堂課讓我受益最多的是如何整理資料，並做出預測模型，但對於丟入 test 的部分似乎還是沒有那麼熟練。

黃振維

這次的競賽我們組試了許多不同的預測模型，其中以 SVM 的預測準確率最高，這次競賽與以往不同的部分是資料皆為 categorical 而非 numeric，我對類別變數的操作還並未十分熟練，因此花了許多時間在做 encoding 與過濾特殊符號如@,!等等。這次的競賽除了程式建構模型外，希望之後若有機會能再更了解這些模型的統計與機率原理。我覺得這學期學到了許多不同的技能，從一開始的 clustering, 爬蟲再到 git, machine learning，都是十分實用的技能，謝謝教授與助教們的用心教導。