

# 多媒體系統

PTT意見探勘實作

張家瑋 博士

國立臺中科技大學資訊工程系助理教授



NATURAL LANGUAGE PROCESSING

自然語言處理的原理與應用

# DATA PREPARATION

- Data preprocessing and cleaning
  - Preprocess data in order to reduce noise and handle missing values
  - 斷字, 斷詞
- Relevance analysis (feature selection)
  - Remove the irrelevant or redundant attributes
  - 移除stop words, 擷取有用資訊(TF-IDF)
- Data transformation
  - Generalize and/or normalize data
  - 轉成向量(Vector representation)

# SEGMENTATION

- Segment by word/ sentence
- Segment in English
  - In English, we can directly segment the word by space " "
  - Ex: I love machine learning. → [I, love, machine, learning]
- Segment in Chinese
  - In Chinese, we segment the word by meaningful word rather than directly segment by characters.
  - Ex: 我喜歡機器學習 → [我, 喜歡, 機器, 學習]  
rather than [我, 喜, 歡, 機, 器, 學, 習]



根據字詞結構將一句話斷字

Dear 小明,  
這是目前公司的最新技術，利用  
apples 和 pens 的特性可以讓產能最  
佳化.....



Dear, 小明, 這是, 目前, 公司, 的, 最新,  
技術, 利用, apples, 和, pens, 的, 特性,  
可以, 讓, 產能, 最佳化, .....

# REMOVING STOP WORDS

- Remove the word which is meaningless.
- Usually do after segment.
- Remove stop words in Chinese
  - Example of stop words: 的, 了, 且, 個, 是
  - Ex: 今天的空氣品質不好 → [今天, 空氣, 品質, 不好]
- Remove stop words in English
  - Example of stop words: is, the, an, and, a
  - Ex: Today 's air quality is not good → [Today's, air, quality, not, good]

Dear, 小明, 這是, 目前, 公司, 的, 最新, 技術,  
利用, apples, 和, pens, 的, 特性, 可以, 讓, 產  
能, 最佳化, .....



移除stop-word

小明, 目前, 最新, 技術, 利用, apples, pens, 特  
性, 產能, 最佳化, .....



# STEMMING

- Stemming is to transform the word into its original type by removing word endings such as -s , -ed and -ing.
  - "bikes" is replaced with "bike" ,
  - "raining" is replaced with "rain"
  - "tried" is replaced with "try"



小明, 目前, 最新, 技術, 利用, apples, pens, 特性, 產能, 最佳化, .....



stemming

小明, 目前, 最新, 技術, 利用, apple, pen, 特性, 產能, 最佳化, .....

# REPRESENTATION

- Select features from the data
- Transform data into vector model
- Ex)
  - WordNet
  - TF-IDF (Term Frequency - Inverse Document Frequency)
  - Word2Vec



自然語言理解

NATURAL LANGUAGE UNDERSTANDING

# SEMANTIC SIMILARITY MEASURES





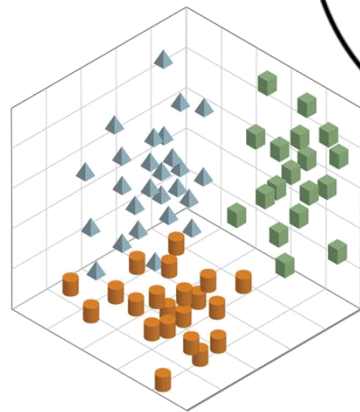
## 文字檔案

Input:  
one document



Lorem ipsum dolor  
sit amet, consete-  
tur sedipiscing elit,  
sed diam nonumy  
aliquid tempor  
invidunt ut labore  
et dolore magna  
aliquyam erat, sed  
diam voluptua. At  
vero eos et

word  
vectors



## word2vec

將被拆解成多個字元

Model:



vector space

解析成多元維度的向量

透過向量比對  
找出相似的資料

most\_similar('france'):

spain	0.678515
belgium	0.665923
netherlands	0.652428
italy	0.633130

highest cosine  
distance values  
in vector space  
of the nearest  
words

# VECTOR REPRESENTATION

	$w_1$	$w_2$	$w_3$	..	..	..	$w_{n-1}$	$w_n$	label
$D_1$	0.11	0.23	0	..	..	..	0.57	0	0
$D_2$	0	0	0	..	..	..	0.29	0.7	1
$D_3$	0	0.81	0.44	..	..	..	0	0	0
$D_4$	0	0.37	0	..	..	..	0	0.16	1
..	..	..	..	..	..	..	..	..	..
$D_k$	..	..	..	..	..	..	..	..	1

Machine  
learning

# TF-IDF



# TF-IDF

- TF: term frequency: 
$$\text{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$
  - IDF: inverse document frequency: 
$$\text{idf}_i = \log \frac{|D|}{|\{j : t_i \in d_j\}|}$$
- where:
- $|D|$ : total number of documents in the corpus
  - $|\{j : t_i \in d_j\}|$  : number of documents where term  $t_i$  appears

Then:

- $$\text{tfidf}_{i,j} = \text{tf}_{i,j} \times \text{idf}_i$$



Document 1		Document 2	
Term	Term Count	Term	Term Count
this	1	this	1
is	1	is	1
a	2	another	2
sample	1	example	3

- The calculation of tf-idf for the term "this" is performed as follows:

$$\begin{aligned} \text{tf}(\text{"this"}, d_1) &= \frac{1}{5} = 0.2 \\ \text{tf}(\text{"this"}, d_2) &= \frac{1}{7} \approx 0.14 \end{aligned}$$

$$\text{idf}(\text{"this"}, D) = \log\left(\frac{2}{2}\right) = 0$$

- So tf-idf is zero for the word "this", which implies that the word is not very informative as it appears in all documents.

$$\begin{aligned} \text{tfidf}(\text{"this"}, d_1) &= 0.2 \times 0 = 0 \\ \text{tfidf}(\text{"this"}, d_2) &= 0.14 \times 0 = 0 \end{aligned}$$

Document 1		Document 2	
Term	Term Count	Term	Term Count
this	1	this	1
is	1	is	1
a	2	another	2
sample	1	example	3

- A slightly more interesting example arises from the word "example", which occurs three times only in the second document:

$$\text{tf}(\text{"example"}, d_1) = \frac{0}{5} = 0$$

$$\text{tf}(\text{"example"}, d_2) = \frac{3}{7} \approx 0.429$$

$$\text{idf}(\text{"example"}, D) = \log\left(\frac{2}{1}\right) = 0.301$$

$$\begin{aligned} \text{tfidf}(\text{"example"}, d_1) &= \text{tf}(\text{"example"}, d_1) \times \text{idf}(\text{"example"}, D) = 0 \times 0.301 = 0 \\ \text{tfidf}(\text{"example"}, d_2) &= \text{tf}(\text{"example"}, d_2) \times \text{idf}(\text{"example"}, D) = 0.429 \times 0.301 \approx 0.13 \end{aligned}$$

# PTT OPINION MINING

## PTT的輿情分析

張家瑋 博士

國立臺中科技大學資訊工程系助理教授

# PTT INTRODUCTION

作者 oppo5566 (5566)  
標題 [問卦] 發錢 預測大谷翔平本日打擊  
時間 Thu Apr 12 07:26:39 2018

大谷翔平 本日は先発打撃對上游騎兵左投Matt moore,

這是上次鄉民預測中的發錢名單

<https://i.imgur.com/t7xIGOK>. ※ 發信站: 批踢踢實業坊(ptt

## 以及收到P幣之後的感謝回信

<https://i.imgur.com/vEm8tme>.

那這次要預測的推文格式為：安

範例：3/1/1

前十位預測中的鄉民稅後各100F

PS.不用擔心錢不夠發，有朋友贊助

※發信站: 批踢踢實業坊(ptt)  
※文章網址: <https://www.ptt.cc/MessageDetail?msgid=386177>  
推 StBeer: 愛菜快遞  
推 LION: 4-0 3k  
推 badbadook: 好屌  
推 robinyu85: 2/0/1  
推 ATSEVEN: 0/0/0  
推 noff: 0/0/0  
※編輯: oppo5566 (45.77.49.82), 04/12/2018 07:28:49  
推 HAHAHUNG: 2/0/1  
噓 snsdakb48: 水桶88  
推 zaclai: 4/2/1  
推 loveinmars: 0/0/0  
推 randy101021: 3/1/1  
推 woook: 0/0/0  
→ Atwo: 事不過四 再打出全壘打我就  
推 dick8752: 1/0/0  
推 Jeff9453: 2/1/1  
推 wb7346: 1/0/0  
推 HC683150: 3/1/2  
推 andvalin: 1/1/2

【板主:Kay731/RS5566/Ra..】

[八卦板]置底協尋文，請大家幫忙！

看板《Gossiping》

<-[離開]

[>][閱讀]

[Ctrl-P]發表文章

[d]刪除

[z]精華區

[i]看板資訊/設定

[h]說明

編號

日期

作者

文

章

標

題

人氣:14099

786163 +

4/12

ahuang80919

R:

[新聞]

台灣人島國心態

柯P：電視很少國際新聞

786164 +

2 4/12

hachilou

R:

[問卦]

說中國古代文化在日本的是什麼人?

786165 +X3

4/12

Safin

[新聞]

陳菊任府秘：現階段選舉不是重要工作

786166 +X2

4/12

borondawon

[F B]

陳沂：很多台男放任自己變肥宅

786167 +10

4/12

mike901003

[新聞]

法航開航 桃園機場飛巴黎浪漫百分百

786168 +

4/12

RonaldReagan

R:

[問卦]

有沒有台南奇美胸腔科謝俊民醫生的八卦

786169 +

4/12

lianpig5566

[問卦]

慟！耳機線卡到臉

786170 +

7 4/12

agh386690

[新聞]

南部某校畢旅連3天超時駕駛 遊覽車GPS還

●86171 +

7 4/12

IELTS

[問卦]

公司最近提供淋浴設備

86172 +

5 4/12

xjapan74269

[問卦]

原來弓箭可以切尾巴

86173 +

8 4/12

CORSA

R:

[問卦]

為啥越南/北韓不是中國不可分割的一部分?

86174 +

4/12

bota

[問卦]

要怎麼把川普這顆棋子效用最大化?

86175 +

5 4/12

Cocochia

[問卦]

臉書是不是快掛了?

86176 +99

4/12

Eliphalet

[新聞]

台北OL夢到被人偷摸！KTV醒來怒告男同事

86177 +

4/12

vmlinuz

[問卦]

台大校長與耶路撒冷 是不是很像?

86178 +

4/12

penisman

R:

[問卦]

有沒有台南奇美胸腔科謝俊民醫生的八

86179 +95

4/12

arrenwu

R:

[F B]

陳沂：很多台男放任自己變肥宅

86180 +

9 4/12

Pattaya

[新聞]

葉俊榮赴中講學 爆不止一次

86181 +

4/12

waymayday

[新聞]

賈納哥塔賺12億 妙天也曾挨告詐欺

86182 爆

4/12

oppo5566

[問卦]

發錢 預測大谷翔平本日打擊

文章標題

(y)回應(X)推文(^X)轉錄

(=|<>)相關主題(/?a)

找標題/作者

(b)進板畫面

04/12 07:27

04/12 07:27

04/12 07:28

04/12 07:28

.82), 04/12/2018 07:28:49

04/12 07:29

04/12 07:29

04/12 07:29

04/12 07:29

04/12 07:29

04/12 07:29

04/12 07:30

04/12 07:30

04/12 07:30

04/12 07:30

04/12 07:30

04/12 07:31

04/12 07:31



# 實作大綱

1. 繪製圖表(matplotlib、seaborn)
2. 中文斷詞(jieba)
  - 去除stop-words、斷詞
3. 機器學習(sklearn)
  - TFIDF、LinearSVC

# Code Structure

匯入函示庫



```
import library
import library1 as lib1
from library import sub-library as sublib
```

```
print('Hello World')
```

四個  
空白

```
for i in range(10):  
    ----print('Hi!')    #印出十次 'Hi!'
```

```
def sayhi():  
    ----print('Hi')
```

```
sayhi()    #呼叫 function sayhi()，印出一次 'Hi'
```

# Import Library

```
import json #用來讀取/產生 json 格式的套件  
import numpy as np #用來處理數值矩陣的套件
```

```
import matplotlib as mpl #用來繪製圖表的套件  
import matplotlib.pyplot as plt #為 matplotlib 的子套件，提供命令行式函數的集合  
import seaborn as sns #基於 matplotlib 的高階圖表的繪製套件
```

```
from collections import defaultdict #使用 dictionary 儲存資料
```

```
zhfont1 = mpl.font_manager.FontProperties(fname='DejaVuSans.ttf') #讀取中文字型
```

# Load Data

```
# load ptt posts  
  
path = 'gossip.json' #欲載入文檔之路徑  
  
with open(path, encoding='utf8') as f:  
    posts = json.load(f)
```



# Load Data

```
{ "author" : "morning3569" ,  
  "title" = "[協尋] 1/1台中清水早上八點多車禍" ,  
  "content" = "\n\n1/1 早上8點多\n\n台中清水紫雲巖....." ,  
  "comments" =  
    [{ "content" : "bad" , "score" :-1, "user" : "xxx" },  
      { "content" : "good" , "score" :1, "user" : "yyv" },  
      { "content" : "soso" , "score" :0, "user" : "zzz" },  
      ....  
    ]  
  , "score" :-244  
}
```

```
total_comments = defaultdict(int) #宣告 dict 儲存所有留言  
total_pushes = defaultdict(int) #宣告 dict 儲存所有推文  
total_hates = defaultdict(int) #宣告 dict 儲存所有噓文
```

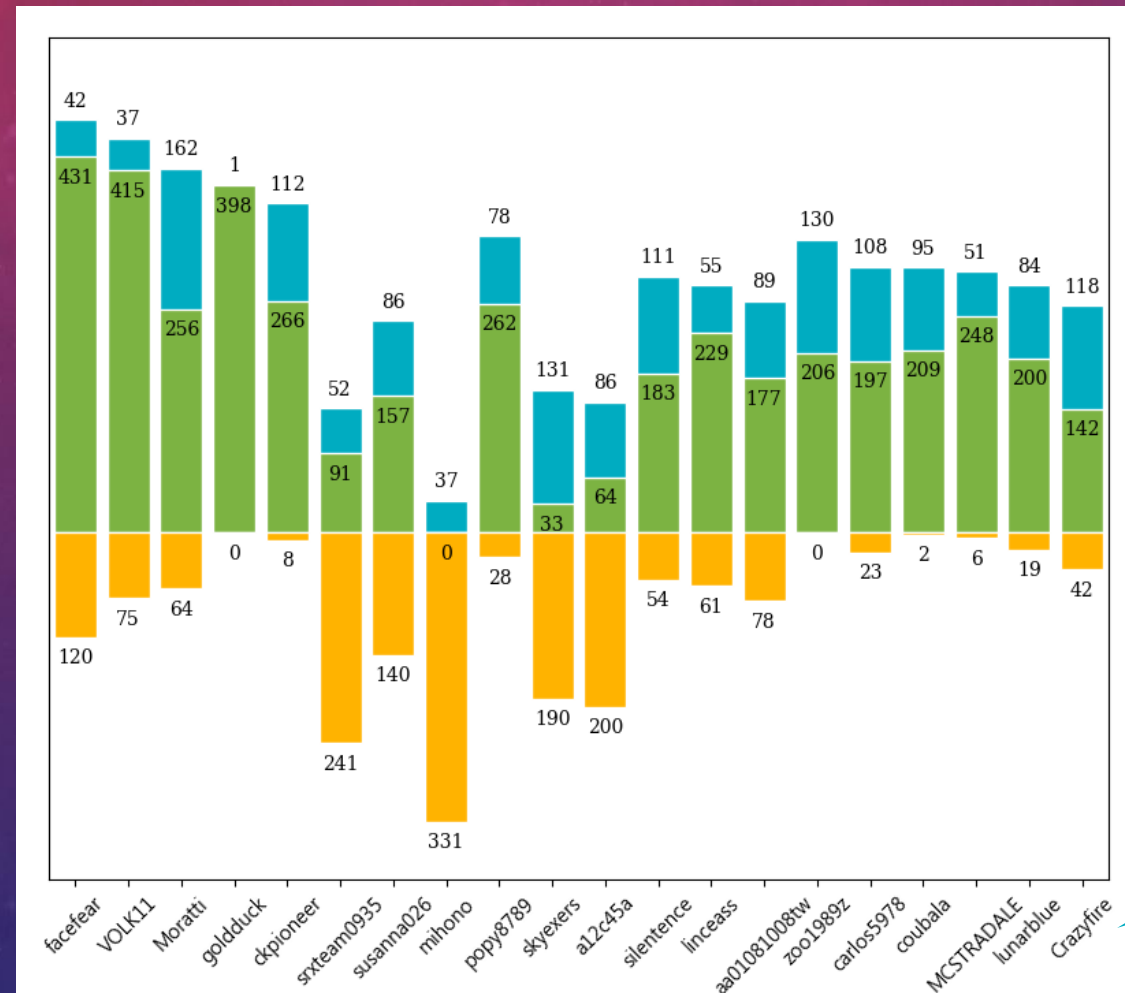
```
for post in posts: #逐一讀取 json 中的所有八卦版文章  
    for comment in post['comments']: #抓出該篇文章的所有留言  
        user = comment['user'] #抓出該則留言的鄉民帳號  
        total_comments[user] += 1 #該名user的留言次數+1  
        if comment['score'] > 0: #score 大於 0 代表是推文  
            total_pushes[user] += 1 #該名user的推文次數+1  
        elif comment['score'] < 0: #score 小於 0 代表是噓文  
            total_hates[user] += 1 #該名user的噓文次數+1
```

同在第二個  
for loop  
的生命週期中

# 繪製圖表

- matplotlib
- seaborn

# Top 20 踴躍留言者分析的圖表實作



推

→

噓

Top 20  
的鄉民ID

# Top 20 踴躍留言者分析的圖表實作

counts.items() 取得所有在案鄉民的ID與其留言次數。如下：  
{[account, times], [account1, times1], [account2, times2]...}

```
def show_distributions(counts, pushes, hates):
```

```
    sorted_cnts = [t[0] for t in sorted(counts.items(), key= lambda x: -x[1])][:20]  
    #取前20個最踴躍回覆者之ID
```

```
    usernames = [u for u in sorted_cnts]           #依序取得前20名的鄉民ID  
    total_y = [counts[u] for u in sorted_cnts]      #依序取得前20名鄉民的總留言數  
    y_pushes = [pushes[u] for u in sorted_cnts]     #依序取得前20名鄉民的推文數  
    y_hates = [hates[u] for u in sorted_cnts]       #依序取得前20名鄉民的噓文數
```

```
    y_neutral = np.asarray(total_y) - np.asarray(y_pushes) - np.asarray(y_hates)  
    #依序取得前20名鄉民的箭頭(中立)留言數
```

```
    y_NandP = y_neutral + np.asarray(y_pushes)  
    #依序將前20名鄉民的箭頭(中立)留言數與推文數相加
```

```
    ....
```

```
    ....
```




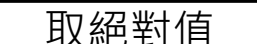
因sorted()預設是遞增，所以實作技巧上可以將次數都先加上負號，再取前20個。



# Top 20 踴躍留言者分析的圖表實作

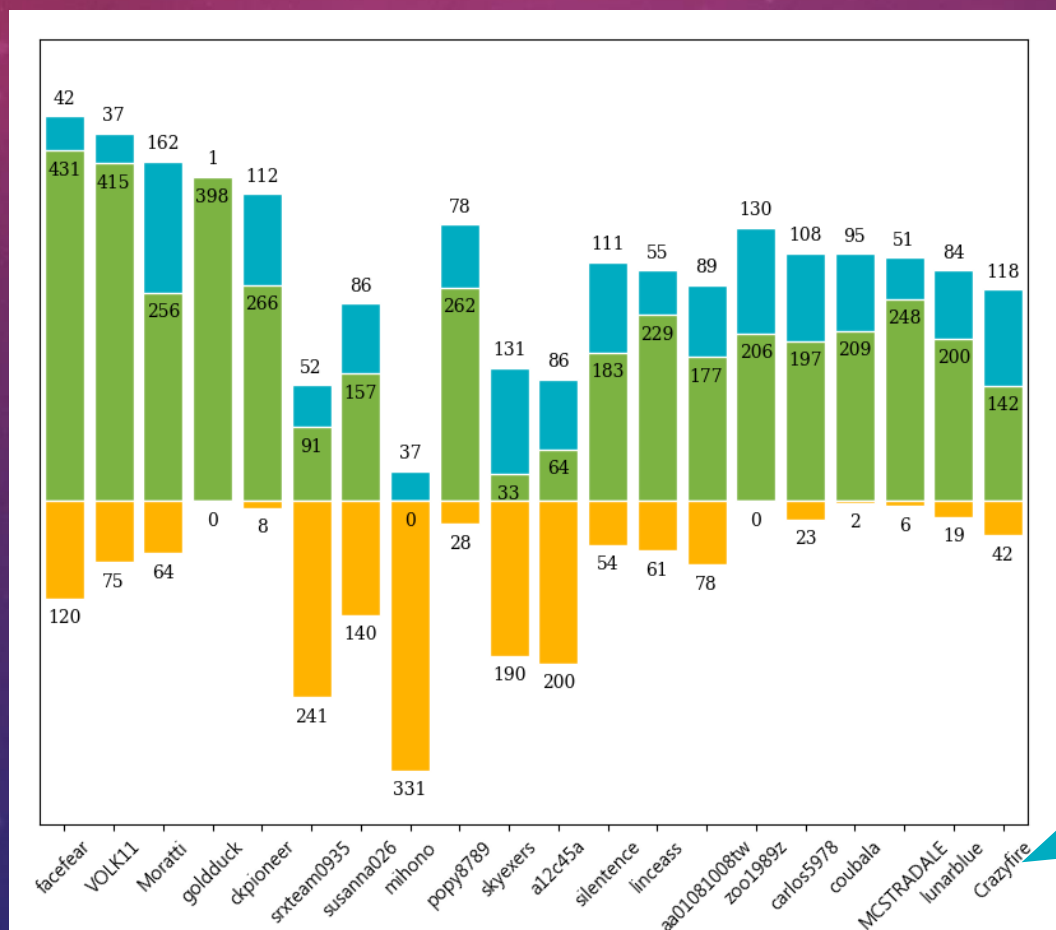
```
def show_distributions(counts, pushes, hates):  
    ....  
    X = np.arange(20) #生成 0-19 的矩陣(array)，代表 Top20 的鄉民  
    fig, ax = plt.subplots(figsize=(10,8))  
  
    plt.bar(X, np.asarray(y_pushes)+np.asarray(y_neutral), facecolor='#00ACC1', edgecolor='white')  
    #將推文數與中立留言數相加，依照 Top20 的 ID 順序繪圖，該顏色代表鄉民的推文數  
    plt.bar(X, np.asarray(y_neutral), facecolor='#7CB342', edgecolor='white')  
    #依照 Top20 的 ID 順序繪圖，該顏色代表鄉民的中立留言數  
    plt.bar(X, -np.asarray(y_hates), facecolor='#FFB300', edgecolor='white')  
    #依照 Top20 的 ID 順序繪圖，該顏色代表鄉民的噓文數。Y軸之值加上負號，讓噓文在另一象限顯示  
  
    plt.xlim(-0.5, 19.5) #設定本圖的 X 軸邊界，左右多 0.5 是為了美觀留有空間  
    plt.ylim(-max(y_hates)*1.2, max(y_NandP)*1.2) #設定本圖的Y軸邊界，分別以上下象限的最大值得1.2倍  
    plt.yticks() #去除 Y 軸的標籤  
    ax.set_xticks(X) #設定 X 軸的 0-19 的軸距標記  
    ax.set_xticklabels(usernames, rotation=45, fontsize=12, fontproperties=zhfont1)  
    #在已設定的 X 軸標記上，將鄉民的ID標記上。設定 ID 文字傾斜45度，文字大小 12 並使用 DejaVu Sans 字體。  
    ....
```

# Top 20 踴躍留言者分析的圖表實作

```
def show_distributions(counts, pushes, hates):  
    ....  
    ....  
    #以下設定推、噓、中立留言的次數所顯示的位置  
  
    for x, y, z in zip(X, np.asarray(y_pushes)+np.asarray(y_neutral), np.asarray(y_pushes)):  
        plt.text(x, y+10, z, ha='center', va='bottom')  
          
          
          
    for x, y in zip(X, np.asarray(y_neutral)):  
        plt.text(x, y-35, y, ha='center', va='bottom')  
  
    for x, y in zip(X, -np.asarray(y_hates)):  
        plt.text(x, y-35, abs(y), ha='center', va='bottom')  
          
    plt.show(fig) #顯示圖表
```

# Top 20 踴躍留言者的簡易分析

show\_distributions(total\_comments, total\_pushes, total\_hates)



推

→

噓

Top 20  
的鄉民ID

# 中文斷詞

- jieba
- remove stop-words



# Jieba 中文斷詞

```
import jieba #用來處理中文斷詞的套件
```

```
for w in jieba.cut( "我來到台南成功大學" ):  
    print(w)
```

---

我來  
到  
台南  
成功  
大學

# [實作] PTT鄉民用語分析

```
# 預處理鄉民留言之用語 (斷詞與計算次數) – Start
```

```
c_words = []
```

```
c_scores = []
```

```
for post in posts:
```

```
    for comment in post['comments']: #取得八卦文文章之鄉民留言
```

```
        l = comment['content'].strip() #去頭去尾換行之類的字符
```

```
        if l and comment['score'] != 0:
```

```
            d = defaultdict(int)
```

```
            for w in jieba.cut(l): # w 是針對 l 中的文字斷詞後所得之詞語
```

```
                d[w] += 1
```

```
            if len(d) > 0:
```

```
                c_scores.append(1 if comment['score'] > 0 else 0) #每則留言之標記(推/噓)
```

```
                c_words.append(d)
```

```
#預處理鄉民留言之用語 (斷詞與計算次數) - End
```

```
{ "author" : "morning3569",  
  "title" = "[協尋] 1/1台中清水早上八點多車禍",  
  "content" = "\n\n1/1 早上8點多\n\n台中清水紫雲巖.....",  
  "comments" =  
    [{ "content" : "bad" , "score" :-1, "user" : "xxx" },  
      { "content" : "good" , "score" :1, "user" : "vv" },  
      { "content" : "soso" , "score" :0, "user" : "zzz" },  
  
      ....  
    ]  
  , "score" : -244  
}
```

# 機器學習

- TF-IDF
- Vector Representation
- LinearSVC

# TF-IDF

## 單篇文章的詞頻統計

{'1': 2, '/': 1, ' ': 1, '早上': 1, '8': 1, '點多': 1, '台': 1, '中': 1, '清水': 1, '紫': 1, '雲': 1, '巖': 1, '外': 1, '中山路': 1, '那邊': 1, '的': 3, '7': 1, '-': 1, '11': 1, '附近': 1, '我': 1, '同學': 1, '阿嬤出': 1, '嚴重': 1, '車禍': 1, '肇事者': 1, '到現': 1, '在': 1, '都': 1, '還沒': 1, '出面': 1, '現在': 1, '還在': 1, '加護': 1, '病房': 1, '如果': 1, '有': 2, '路口': 1, '監視器': 1, '影像': 1, '或是': 1, '行車紀': 1, '錄器': 1, '拍': 1, '到': 1, '懇請': 1, '提供': 1, '麻煩': 1, '八卦': 1, '板': 1, '各位': 1, '幫高調': 1, '謝謝': 1, '！': 2})

單  
篇  
的  
向  
量  
轉  
換

(0, 0)	1.0	(0, 29)	1.0
(0, 1)	1.0	(0, 30)	2.0
(0, 2)	1.0	(0, 31)	1.0
(0, 3)	2.0	(0, 32)	1.0
(0, 4)	1.0	(0, 33)	1.0
(0, 5)	1.0	(0, 34)	1.0
(0, 6)	1.0	(0, 35)	3.0
(0, 7)	1.0	(0, 36)	1.0
(0, 8)	1.0	(0, 37)	1.0
(0, 9)	1.0	(0, 38)	1.0
(0, 10)	1.0	(0, 39)	1.0
(0, 11)	1.0	(0, 40)	1.0
(0, 12)	1.0	(0, 41)	1.0
(0, 13)	1.0	(0, 42)	1.0
(0, 14)	1.0	(0, 43)	1.0
(0, 15)	1.0	(0, 44)	1.0
(0, 16)	1.0	(0, 45)	1.0
(0, 17)	1.0	(0, 46)	1.0
(0, 18)	1.0	(0, 47)	1.0
(0, 19)	1.0	(0, 48)	1.0
(0, 20)	1.0	(0, 49)	1.0
(0, 21)	1.0	(0, 50)	1.0
(0, 22)	1.0	(0, 51)	1.0
(0, 23)	1.0	(0, 52)	1.0
(0, 24)	1.0	(0, 53)	2.0

:



# TF-IDF

## 單篇的 TF-IDF (Sparse Matrix) 單篇的 Vector Representation

### 單篇統計詞頻

(0, 0)	1.0	(0, 29)	1.0
(0, 1)	1.0	(0, 30)	2.0
(0, 2)	1.0	(0, 31)	1.0
(0, 3)	2.0	(0, 32)	1.0
(0, 4)	1.0	(0, 33)	1.0
(0, 5)	1.0	(0, 34)	1.0
(0, 6)	1.0	(0, 35)	3.0
(0, 7)	1.0	(0, 36)	1.0
(0, 8)	1.0	(0, 37)	1.0
(0, 9)	1.0	(0, 38)	1.0
(0, 10)	1.0	(0, 39)	1.0
(0, 11)	1.0	(0, 40)	1.0
(0, 12)	1.0	(0, 41)	1.0
(0, 13)	1.0	(0, 42)	1.0
(0, 14)	1.0	(0, 43)	1.0
(0, 15)	1.0	(0, 44)	1.0
(0, 16)	1.0	(0, 45)	1.0
(0, 17)	1.0	(0, 46)	1.0
(0, 18)	1.0	(0, 47)	1.0
(0, 19)	1.0	(0, 48)	1.0
(0, 20)	1.0	(0, 49)	1.0
(0, 21)	1.0	(0, 50)	1.0
(0, 22)	1.0	(0, 51)	1.0
(0, 23)	1.0	(0, 52)	1.0
(0, 24)	1.0	(0, 53)	2.0

單篇的  
TF-IDF  
計算

(0, 53)	0.23735633163877065	(0, 24)	0.11867816581938533
(0, 52)	0.11867816581938533	(0, 23)	0.11867816581938533
(0, 51)	0.11867816581938533	(0, 22)	0.11867816581938533
(0, 50)	0.11867816581938533	(0, 21)	0.11867816581938533
(0, 49)	0.11867816581938533	(0, 20)	0.11867816581938533
(0, 48)	0.11867816581938533	(0, 19)	0.11867816581938533
(0, 47)	0.11867816581938533	(0, 18)	0.11867816581938533
(0, 46)	0.11867816581938533	(0, 17)	0.11867816581938533
(0, 45)	0.11867816581938533	(0, 16)	0.11867816581938533
(0, 44)	0.11867816581938533	(0, 15)	0.11867816581938533
(0, 43)	0.11867816581938533	(0, 14)	0.11867816581938533
(0, 42)	0.11867816581938533	(0, 13)	0.11867816581938533
(0, 41)	0.11867816581938533	(0, 12)	0.11867816581938533
(0, 40)	0.11867816581938533	(0, 11)	0.11867816581938533
(0, 39)	0.11867816581938533	(0, 10)	0.11867816581938533
(0, 38)	0.11867816581938533	(0, 9)	0.11867816581938533
(0, 37)	0.11867816581938533	(0, 8)	0.11867816581938533
(0, 36)	0.11867816581938533	(0, 7)	0.11867816581938533
(0, 35)	0.35603449745815596	(0, 6)	0.11867816581938533
(0, 34)	0.11867816581938533	(0, 5)	0.11867816581938533
(0, 33)	0.11867816581938533	(0, 4)	0.11867816581938533
(0, 32)	0.11867816581938533	(0, 3)	0.23735633163877065
(0, 31)	0.11867816581938533	(0, 2)	0.11867816581938533
(0, 30)	0.23735633163877065	(0, 1)	0.11867816581938533
(0, 29)	0.11867816581938533	(0, 0)	0.11867816581938533

: :

# Scikit-learn (sklearn)

```
from sklearn.feature_extraction import DictVectorizer  
#用於轉換 dict 為 sklearn estimators 可用的向量
```

```
from sklearn.feature_extraction.text import TfidfTransformer  
#將矩陣轉換為 TF 或 TF-IDF 表示
```

```
from sklearn.svm import LinearSVC  
#以 LinearSVC 演算法為例
```

# 以 LinearSVC 提取留言的用語特徵

```
# 將詞語及其出現次數轉換成向量
```

```
c_dvec = DictVectorizer() #宣告向量轉換方法
```

```
c_tfidf = TfidfTransformer() #宣告TFIDF方法
```

```
c_X = c_tfidf.fit_transform(c_dvec.fit_transform(c_words))
```

```
#將所有的留言中的詞語矩陣，轉成向量並計算tf-idf
```

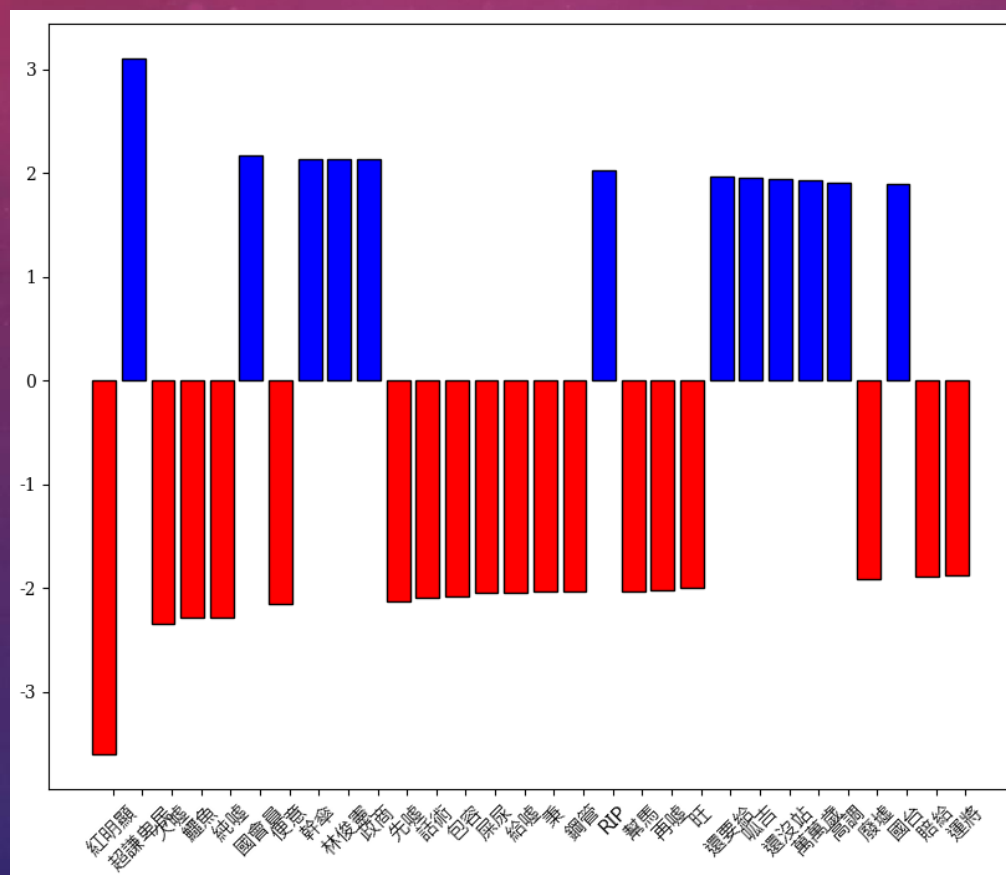
```
c_svc = LinearSVC() #宣告 LinearSVC 方法
```

```
c_svc.fit(c_X, c_scores) #餵入訓練資料 c_X 以及資料標籤 c_scores
```

```
c_svc.coef_[0] #取得留言用語的權重係數，值越大代表越有代表性
```

## 以 LinearSVC 提取留言的前三十大用語

```
display_top_features(c_svc.coef_[0], c_dvec.get_feature_names(), 30)
```



正面用語

# 負面用語





THANK YOU