BDA_Homework2

Group_12
B04702077 徐嬿鎔 B04702010 廖文豪 B04702012 鄭皓 B04702091 陳宜君
P05751019 陳牧忠 B05705034 藤田教譽

方法

1.使用245個詞來建構向量空間，讓每一個維度代表每一個字的tfidf與chi square值，用以計算查詢文章與其他每篇文章之向量內積，最後取值最高之前3名，並扣除重複者。
2.製作GUI，使我們能隨時輸入任何一篇查詢，即得與其最相似的其他三篇文章。

過程

<前處理>
1.先Import所有所需要的套件。

```
import math
import pandas as pd
import numpy as np
from operator import itemgetter
import lib
import sys
from PyQt5.QtWidgets import QApplication, QMainWindow, QTableWidgetItem
from PyQt5.uic import loadUi
```

2.讀取所需的檔案，包含詞彙集與查詢集。

```
df_text_foxconn = pd.read_excel("hw1_text.xlsx",sheet_name='foxconn')
df_foxconn_keyword = pd.read_excel("bda2019_hw2_table.xlsx",sheet_name='L2_foxconn_keyword')
```

3.再將資料轉成List，並將標題以兩倍權重的方式展現在doc_list中

```
doc_list = df_text_foxconn.loc[:,'標題']+df_text_foxconn.loc[:,'標題']+df_text_foxconn.loc[:,'內容']
term_list = df_foxconn_keyword.loc[:,'term']
```

4.製作Word與ID互相轉換的列表（List與Dict)。

```
id2word = list() # 0 -> 'term'
word2id = dict() # 'term' -> 0
for num, item in enumerate(term_list):
    id2word.append(item)
    word2id[item] = num
```

5.制定TF與DF的Function。

```
def get_df(term_list, doc_list):
    df_list = list()
    df_set = {}
    for keyword in term_list:
        for n_text, text in enumerate(doc_list):
            if keyword in text:
                try:
                    df_set[keyword].add(n_text) # add doc_idx to df_set
                except:
                    df_set[keyword] = {n_text} # init df_set with doc_idx
    for item in df_set:
        df_list.append(len(df_set[item])) # doc freq list
    return df_list
```

```
def get_tf(term_list, doc_list):
    tf_list = list()
    for keyword in term_list:
        tf_tmp = list()
        for text in doc_list:
            tf_tmp.append(text.count(keyword)) # append tf for 1 doc
        tf_list.append(tf_tmp) # append tf of doc to tf_list
    return tf_list
```

6.制定TFIDF、Chi Square的Function。
特別可以注意，在if tf[n_row][n_col]>0 前提下，才套以公式計算，否則tfidf為0。一樣在 if tf[n_row][n_col]!=0 前提下，才套以公式計算，否則chi square 為0。

```
def get_tfidf(tf, df):
    tfidf_list = list()
    for n_row in range(len(df)):
        tfidf_tmp = list()
        for n_col in range(len(tf[0])):
            if tf[n_row][n_col]>0:
                tfidf = (1+math.log10(tf[n_row][n_col])) * math.log10(len(tf[0])/df[n_row]) # tfidf equation
            else:
                tfidf = 0
            tfidf_tmp.append(tfidf)
        tfidf_list.append(tfidf_tmp)
    return tfidf_list
```

```
def get_chi_square(tf, df):
    exp = list()
    for n_row in range(len(tf)):
        exp.append(0)
        for n_col in range(len(tf[0])):
            exp[n_row] = exp[n_row] + tf[n_row][n_col] # addup all tf for 1 term
        exp[n_row] = exp[n_row]/df[n_row] # averge tf
    chi_square_list = list()
    for n_row in range(len(tf)):
        chi_square_tmp = list()
        for n_col in range(len(tf[0])):
            if tf[n_row][n_col]!=0:
                sign = -1 if exp[n_row]>tf[n_row][n_col] else 1 # chi2 sign
                chi_square = (tf[n_row][n_col]-exp[n_row])**2 / exp[n_row] * sign # chi2 equation
            else:
                chi_square = 0
            chi_square_tmp.append(chi_square)
        chi_square_list.append(chi_square_tmp)
    return chi_square_list
```

7.計算TF、DF後，導入TFIDF、Chi Square function 計算各值。最後，計算TFIDF的normalized weight 並導入VSM。

```
df = get_df(term_list, doc_list)
tf = get_tf(term_list, doc_list)
tfidf = get_tfidf(tf,df)
chi_square = get_chi_square(tf,df)
norm_weight = get_norm_weight(tfidf)
vsm, vsm_sorted_idx = get_vsm(norm_weight,219)
```

<計算排序>

get_norm_weight(weight):

是一個取normalized weight的function。Argument “weight”是計算出來的 tf-idf或chi2的值，是245*2081的向量。這個function裡有兩個list：norm_list 和 norm_weight_list。norm_list代表讓每列裡的每個元素平方後把每列的值加起來的list (norm = norm + weight[n_col][n_row]**2)，也就是說normalized factor list。norm_weight_list代表每個元素除於norm (norm_weight = weight[n_row][n_col]/norm_list[n_col])。然後return norm_weight_list。

get_vsm(weight,n_query):

是用來計算VSM的function。讓每列的每個元素平方後加起來的list，並且其中一個 weight用n_query指定哪一列的。(vsm = vsm + weight[n_row][n_col] * weight[n_row][n_query])。vsm_sorted_idx是把vsm由小到大排序後儲存那個index的list。

check_item(weight,n_col,id2word):

是把0的元素除掉的function(if weight[n_row][n_col] != 0: chk_list.append(id2word[n_row]))。

以下開始執行前項的function:

norm_weight = get_norm_weight(tfidf)

vsm, vsm_sorted_idx = get_vsm(norm_weight,219)

chk1 = check_item(norm_weight,vsm_sorted_idx[0],id2word)

chk2 = check_item(norm_weight,vsm_sorted_idx[2],id2word)

print('Chk1 Len = {}, Chk2 Len = {}, Matched Len = {}'.format(len(chk1), len(chk2), len(set(chk1) & set(chk2))))

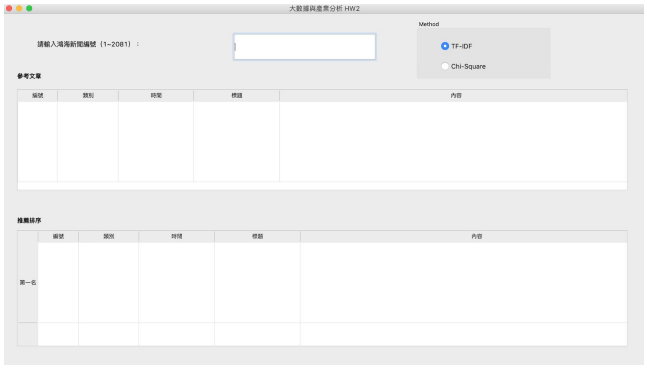
結果

參考文章	編號65	編號165	編號220
推薦文章	TF-IDF: 733 826 883 Chi-square: 927 926 1499	TF-IDF: 161 163 164 Chi-square: 1069 980 1199	TF-IDF: 1568 361 396 Chi-square: 285 1407 215

參考文章	編號586	編號1010	編號1658
推薦文章	TF-IDF: 587 381 1364 Chi-square: 1517 212 232	TF-IDF: 1011 1017 360 Chi-square: 1011 1017 1753	TF-IDF: 2063 768 409 Chi-square: 1582 1672 409

GUI

設計出GUI介面讓使用者互動，可選擇兩種推薦方法。先用qt designer 畫出基本介面，並存成hw.ui。接著在hw2_group.ipynb裡利用pyqt5讀取ui並對其做更進一步的操作 （ex. 架設欄位寬度高度及名稱等）。



在程式碼中有個function: on_line_entered()，會分別依照使用者輸入的新聞編號（最上面的空白欄）以及選用的指標（method: tf-idf / chi-square），進行文章推薦，過程如下：

1. 利用 get_tfidf(tf,df) / get_chi_square(tf,df) 回傳的list
2. 在get_vsm()中算出各個文章與參考文章的向量內積
3. 最後再將推薦的文章（內積值最高的三篇）的編號、類別、時間、標題、和內容塞進推薦排序中適當欄位，舉例如下：