資訊檢索與文字探勘導論

HOMEWORK 4

資管三吳驊祐 B09705009

2022-12-26

1 Environment

Using Jupyter Noteook

2 Language

Python3

3 Execute

import request, copy, sys, nltk, math, pandas, numpy,
`from nltk stem import PorterStemmer`
`from sklearn metrics import pairwise distances`
`from scipy spatial distance import cosine`
-> Use jupyter notebook to run the ipynb file.

In [1]:
```
import sys
import math
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rcParams
import seaborn as sns

import re
import nltk
from nltk.stem import PorterStemmer

import warnings
warnings.filterwarnings('ignore')
plt.rcParams['axes.labelsize'] = 14
plt.rcParams['axes.titlesize'] = 16
plt.rcParams['figure.titlesize'] = 16
plt.rcParams['xtick.labelsize'] = 12
plt.rcParams['ytick.labelsize'] = 12
plt.rcParams['legend.fontsize'] = 10
```

In [2]:
```
ps = PorterStemmer()
np.set_printoptions(threshold=sys.maxsize)

#stopwords from txt
my_file = open("stopwords.txt", "r")
stopwords = my_file.read()
stopwords = [stopwords.translate({ ord(c): None for c in '" ' }).split(",") ][0]#delete punctuation
#stopwords
```

In [3]:
```
length_all_doc = 1095 #實際1095
```

In [4]:
```
def tokenize(text): #順序很有差！以下都用空白取代
    text = re.sub(r'\|\|\|', r' ', text) #照數據格式切成一段一段
    text = re.sub(r'http\S+', r' ', text) #刪除網址
    text = re.sub('\S+com$', ' ', text) #刪除網站
    text = re.sub('\d+',' ', text) #刪除所有數字
    text = re.sub(r'[^\w\s]', ' ', text) #刪除所有標點符號 避免如ideas/thought -> ideasthought
    #print(text)
    #text = [i.translate({ ord(c): None for c in '"1234567890' }) for i in text] #delete punctuation
    #text = [i.translate({ ord(c): ' ' for c in "#$%&'()*+,-./:;!<=>?@[\]^_`{|~} " }) for i in text]
    text = re.sub("[^a-zA-Z]", " ", text) #只保留文字
    text = re.sub(' +', ' ', text) # Remove spaces > 1

    text = [i.strip() for i in text.split(' ')]
    text = [i.lower() for i in text]
    text = [ps.stem(i) for i in text] #porter's algo
    text = [i for i in text if i not in stopwords] #stopwords delete
    text = list(filter(None, text)) #去除空字元
    stringtext = ' '.join([str(item) for item in text])
    #print(stringtext)
    return stringtext
```

In [5]:
```
Allwords = [] #一個 總體Dictionary
def buildDict(text): #只算有無出現, 弄成list
    for word in str(text).split(' '):
        #print(word)
        if word not in Allwords:
            Allwords.append(word)

for i in range(1, length_all_doc+1):
    with open(f'./data/{i}.txt') as f:
        data=f.read()
        text = tokenize(data)
        #print(text)
        buildDict(text)

Allwords_dict = sorted(Allwords) #照字母排
AllwordSize = len(Allwords_dict) #總term數
AllwordSize
```

`Allwords_dict` 為總共有哪些字的字典，AllwordsSize為字典大小

計算每個字出現在多少文件裡面（`IDF`）-> wd_appear

In [6]:
```
wd_appear = {k: 0 for k in Allwords_dict}
for i in range(1, length_all_doc+1):
    tmp_appear = {k: 0 for k in Allwords_dict} #看每篇文章有沒有出現字，最高1
    with open(f'./data/{i}.txt') as f:
        data=f.read()
        text = tokenize(data)
        for wd in text.split(' '):
            tmp_appear[wd] = 1
        for wd, num in tmp_appear.items():
            wd_appear[wd] += tmp_appear[wd]
for wd, num in wd_appear.items():
    wd_appear[wd] = math.log(length_all_doc+1/wd_appear[wd]+1) #算IDF值對於字典每個字
```

計算tf在每個文件的每個字 `tf` : wd_cnt

- 同時結合 idf 後標準化成此文章對於每個字的陣列

In [7]:
```
allVector = [] #保存所有文章的vector，總共有文章數*字典字數 個參數
for i in range(1, length_all_doc+1):
    wd_cnt = {k: 0 for k in Allwords_dict}
    with open(f'./data/{i}.txt') as f:
        data=f.read()
        text = tokenize(data)
        for wd in text.split(' '):
            wd_cnt[wd] += 1

    arr = []
    for wd, cnt in wd_cnt.items():
        arr.append(cnt*wd_appear[wd]) #tf*idf
    normal_V = arr/np.linalg.norm(arr)
    allVector.append(normal_V) #標準化成每個文章對於每個字的vector
#allVector
```

In [8]:
```
from sklearn.metrics import pairwise_distances
from scipy.spatial.distance import cosine
vMatrix = 1-pairwise_distances(allVector, metric="cosine")
#vMatrix
```

先從 0 ，輸出時再加一回來

In [11]:
```
#使用complete link clustering
def calculateResult(vecMatrix, kclusters=3, doclength=10):
    mergeStep=[]
    alive_person = [i for i in range(doclength)] #initial alive 活著會在此顯示1
    dead_person  = [] #initial dead 死了則會在此顯示1
    clusters     = [[i] for i in range(doclength)] #initial cluster

    for i in range(doclength-kclusters): #要合併幾次
        maxCosine = 0
        maxpair = []
        for first in alive_person:
            for last in alive_person:
                if last<=first: continue #之後要把last合併到first
                if vecMatrix[first][last]>=maxCosine:
                    maxCosine = vecMatrix[first][last]
                    maxpair = [first, last]
        #決定這次要合併誰：last合併到first
        #print(maxpair)
        mergeStep.append((maxpair[0],maxpair[1]))
        alive_person.remove(maxpair[1])
        dead_person.append(maxpair[1])
        dead_person.append(maxpair[1])
        for j in alive_person: #更新群間距離
            vecMatrix[maxpair[0]][j] = min(vecMatrix[maxpair[0]][j], vecMatrix[maxpair[1]][j])
            vecMatrix[j][maxpair[0]] = min(vecMatrix[maxpair[0]][j], vecMatrix[maxpair[1]][j])
        for doc in clusters[maxpair[1]]: #把那類全部合併過來
            clusters[maxpair[0]].append(doc)
        clusters[maxpair[1]] = [] #刪除該類別

#         print(f"mergeStep: {mergeStep}")
#         print(f"clusters: {clusters}")
#         print(vecMatrix)
#         print()

    return clusters
```

n [12]:
```
for k in [8, 13, 20]:
    res = calculateResult(vMatrix, kclusters=k, doclength=length_all_doc)
    res = [ele for ele in res if ele != []]
    with open(f'./output/{k}.txt', 'w') as wf:
        for num in res:
            for item in sorted(num):
                wf.write(f'{item+1}\n')
            wf.write('\n')
```

4 Program Logic