Programming Assignment 4

執行環境: Visual Studio Code
程式語言:Python 3.11.5
作業架構:

```
R12725048
├── report.pdf
├── 8.txt
├── 13.txt
├── 20.txt
├── pa4.py
```

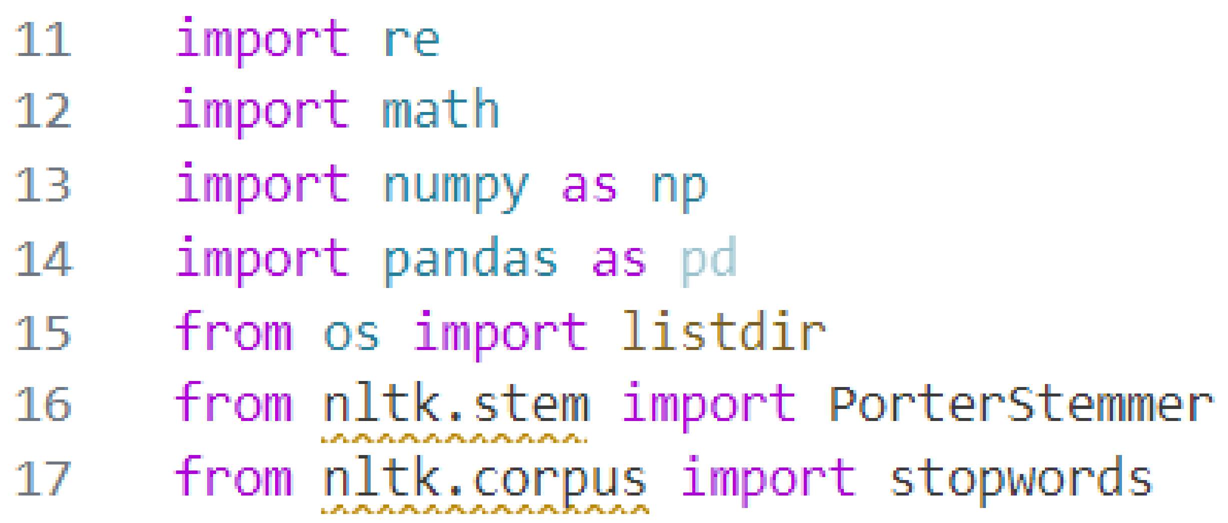
資管所碩一 R12725048 楊喻妃

執行方式

- 使用VS code跑pa4.py檔
- 需要下載的套件有:
  - `pip install nltk`: 刪除stopwords及使用Porter’s algorithm.所需
- 直接按全部執行即可

處理邏輯

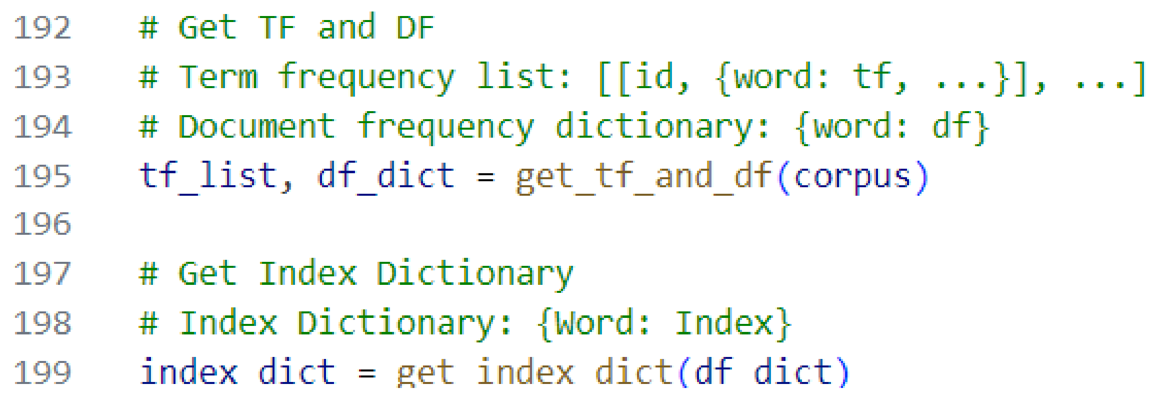
- Step 1 : import 所需套件



```
import re
import math
import numpy as np
import pandas as pd
from os import listdir
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
```

處理邏輯

- Step 2 : 根據前作業 一樣先依據上次的dataset為每個doc建立其tfidf vector
  - ➢ 此步驟會做資料的前處理、建好dictionary、tf-idf table
  - ➢ 皆是上次作業的內容，因此簡報就不多花篇幅敘述

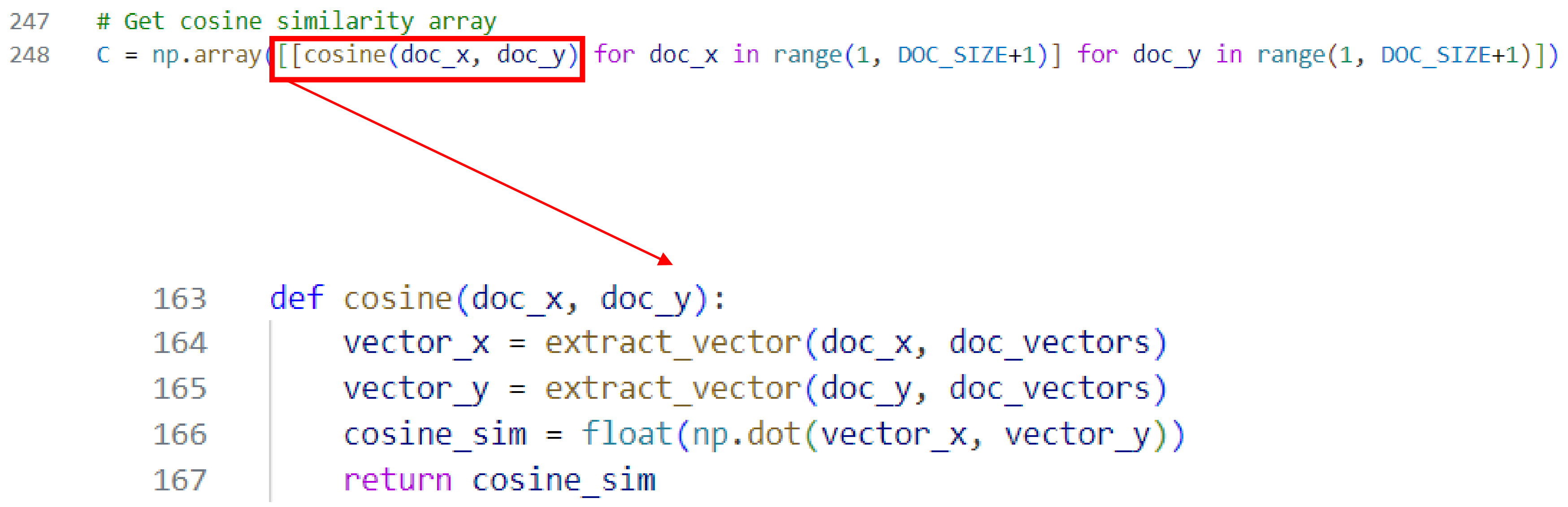


```
# Get TF and DF
# Term frequency list: [[id, {word: tf, ...}], ...]
# Document frequency dictionary: {word: df}
tf_list, df_dict = get_tf_and_df(corpus)

# Get Index Dictionary
# Index Dictionary: {Word: Index}
index_dict = get_index_dict(df_dict)
```

處理邏輯

- Step 3 : 計算兩兩documents的cosine similarity
    - ➢記錄到C Matrix

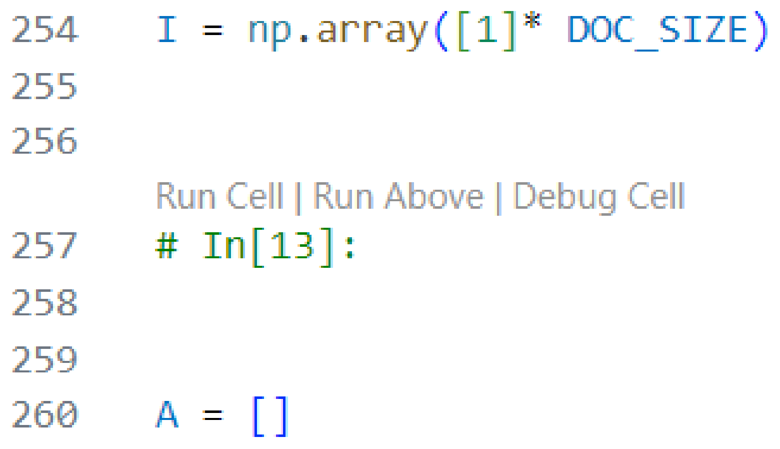


```python
# Get cosine similarity array
C = np.array([[cosine(doc_x, doc_y) for doc_x in range(1, DOC_SIZE+1)] for doc_y in range(1, DOC_SIZE+1)])
```

```python
def cosine(doc_x, doc_y):
    vector_x = extract_vector(doc_x, doc_vectors)
    vector_y = extract_vector(doc_y, doc_vectors)
    cosine_sim = float(np.dot(vector_x, vector_y))
    return cosine_sim
```

處理邏輯

- Step 4 : 初始化A、I array
    - I:紀錄每個doc的存活狀態
    - A:紀錄每次merge是將哪兩個cluster合併的



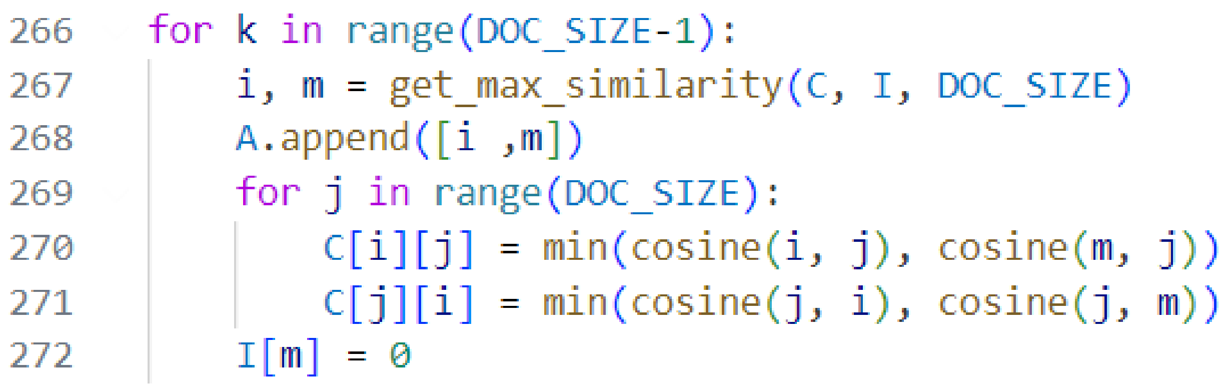
```
I = np.array([1]* DOC_SIZE)


Run Cell | Run Above | Debug Cell
# In[13]:


A = []
```

處理邏輯

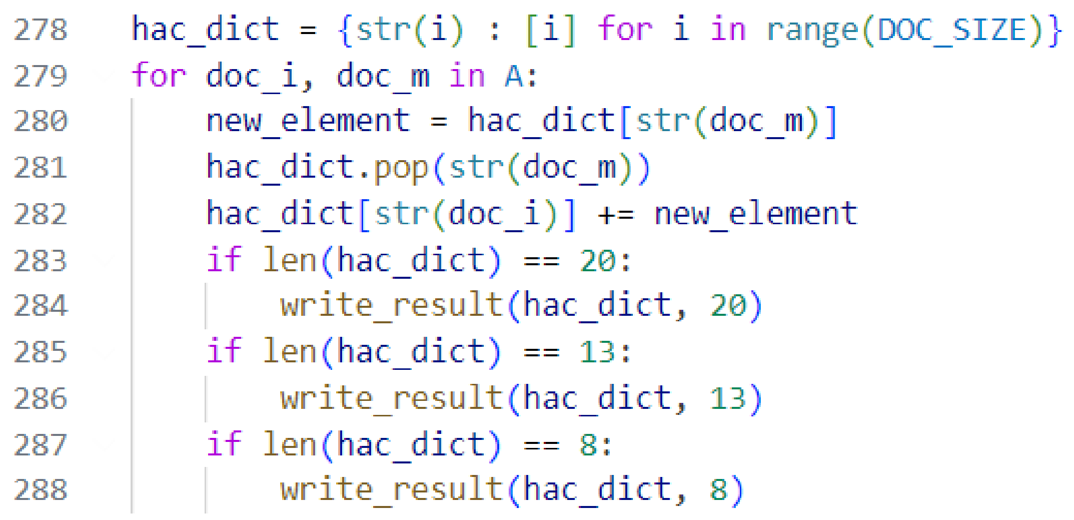
- Step 5 :並用complete link的方式建構HAC
  - ➢先取出具有最大相似的的兩群
  - ➢再更新C Matrx:用這兩群中最遠的兩點作為新的相似度
  - ➢將I[m]設為0，因為已經被併到cluster i了



```
for k in range(DOC_SIZE-1):
    i, m = get_max_similarity(C, I, DOC_SIZE)
    A.append([i ,m])
    for j in range(DOC_SIZE):
        C[i][j] = min(cosine(i, j), cosine(m, j))
        C[j][i] = min(cosine(j, i), cosine(j, m))
    I[m] = 0
```

處理邏輯

- Step 6 :以bottom-up的方式建構HAC，並將cluster數分別為K = 8, 13, and 20的時候輸出txt檔



```python
hac_dict = {str(i) : [i] for i in range(DOC_SIZE)}
for doc_i, doc_m in A:
    new_element = hac_dict[str(doc_m)]
    hac_dict.pop(str(doc_m))
    hac_dict[str(doc_i)] += new_element
    if len(hac_dict) == 20:
        write_result(hac_dict, 20)
    if len(hac_dict) == 13:
        write_result(hac_dict, 13)
    if len(hac_dict) == 8:
        write_result(hac_dict, 8)
```