Introduction-to-Information-Retrieval-and-Text-Mining, HW 4

B07302230, 財金三 王博奕

1. 執行環境：Visual Studio Code
2. 程式語言：Python 3.8
3. 執行方式：
   A. Pip install np 來計算 cosine similarity
4. 作業處理邏輯說明：

Step1: 讀取作業二的解答，並且裝進一個 1095(文章數)*12291(字典)的矩陣。其中每個文章都已變成 tfidf 的向量。

```
tfidf_matrix = np.zeros((1095, 12291))

# create a matrix recording normal vector of each doc by using the result of hw2
for i in range(1,1096):
    f = open("C:\\Users\\asdfg\\OneDrive\\桌面\\IRTM_hw2_result\\"+str(i)+".txt",'r')
    # remove title
    next(f)
    next(f)
    vec = np.array([0.0]*12291)
    for x in f.readlines():
        temp = x.strip().split("\t")
        vec[int(temp[0])-1] = float(temp[1])
        # temp[0] is term index
        # temp[1] is tfidf
    tfidf_matrix[i-1] = vec
```

Step2: 創出一個矩陣 sim_matrix，裡面放各文章彼此的相似度，此外紀錄該文章是否被併入。

```
# in order to create a matrix recording similarity
# existence = 1 means exist
def cos_sim(dx, dy):
    return (tfidf_matrix[dx]*tfidf_matrix[dy]).sum()

sim_matrix = np.zeros((doc_size, doc_size))
existence = np.zeros(doc_size)
for i in range(doc_size):
    for j in range(doc_size):
        sim_matrix[i][j] = cos_sim(i, j)
    existence[i] = 1
```

Step3: 定義出一個找出 sim_matrix 裡最大值的函數，同時尚未被併，並且記錄兩者的編號。之後以 complete link 的方式來更新 sim_matrix 的值。

```
# HAC
# complete link
def find_max_sim(sim_matrix, existence):
    max_sim = -1
    index_i = -1
    index_j = -1
    for i in range(doc_size):
        if existence[i] == 1:
            for j in range(doc_size):
                if existence[j] == 1 and j != i:
                    if max_sim < sim_matrix[i][j]:
                        max_sim = sim_matrix[i][j]
                        index_i = i
                        index_j = j
    return max_sim, index_i, index_j
```

```
record = []
for x in range(doc_size - 1):
    max_sim, i, j = find_max_sim(sim_matrix, existence)
    record.append([i ,j])
    for k in range(doc_size):
        sim_matrix[i][k] = min(cos_sim(i, k), cos_sim(j, k))
        sim_matrix[k][i] = sim_matrix[i][k]
    existence[j] = 0
```

Step4: 依據作業要求定義寫檔的方式。先定義一個 dictionary 其中讓每一篇文章 ID 作為 key，而 value 就是目前的分群結果，初始值即為每一篇文章自己的 ID。將 record 中的每一個 pair 照順序讀入，一一將他們合併起來，當合併結果的群組共有 20、13、8 時就寫檔。

```
def write_cluster(cluster_dict, K):
    with open(str(K) + '.txt', 'w') as cluster_file:
        for key, l in cluster_dict.items():
            doc_list = np.sort(l)
            for doc_id in doc_list:
                cluster_file.write(str(doc_id+1) + '\n')
            cluster_file.write('\n')
```

```
cluster_dict = {}
for i in range(doc_size):
    cluster_dict[str(i)] = [i]

for i, j in record:
    temp = cluster_dict[str(j)]
    cluster_dict.pop(str(j), None)
    cluster_dict[str(i)] += temp
    if len(cluster_dict) == 20:
        write_cluster(cluster_dict, 20)
    elif len(cluster_dict) == 13:
        write_cluster(cluster_dict, 13)
    elif len(cluster_dict) == 8:
        write_cluster(cluster_dict, 8)
```