資訊檢索語文字探勘導論 PA4

1. 執行環境：Jupyter Notebook
2. 程式語言：Python 3.10.9
3. 執行方式：使用 Jupyter Notebook 執行 pa4.ipynb
4. 處理邏輯

（1）讀入所有 data 資料夾中的所有 txt 檔。

```
files = listdir(FILE_PATH)
files.sort(key=lambda x: int(x[:-4]))
doc_set = list()

for file in files:
    with open(FILE_PATH + file, "r") as f:
        document_id = str(file)[:-4]
        document = f.read()
        doc_set.append([document_id, document])
```

（2）利用作業二寫出的程式，計算出所有文件之間的 cosine similarity。

```
# Initialize cosine similarity and priority queue
def pre_clustering(doc_set):
    N = len(doc_set)
    cos_i_j = [[] for _ in range(N)]
    prior = []

    for i in range(N):
        pq = MaxPriorityQueue()
        for j in range(N):
            sim = cosine(i + 1, j + 1)
            cos_i_j[i].append([sim, j])
            if i != j:
                pq.insert(cos_i_j[i][j])
        prior.append(pq)
    return cos_i_j, prior
```

對應文章編號，將得出的結果存進屬於該文章編號的 priority queue

（3）實作 HAC 演算法，其中利用 complete-link，將新合成的 cluster 的 cosine similarity 設為 cluster 中距離最遠的兩者之 similarity。

共進行 len(document_set)-1 次 cluster。

```python
# Clustering for N-1 round
for _ in range(N - 1):
    max_sim = -1
    k1 = k2 = -1

    # Find the maximum cosine similarity
    for i in range(N):
        if alive[i] == 1 and not prior[i].is_empty():
            top = prior[i].peek()
            if top[0] > max_sim:
                max_sim = top[0]
                k1, k2 = i, top[1]

    # Record the merge docs
    merge.append([k1 + 1, k2 + 1])

    # Merge k2 into k1
    alive[k2] = 0
    prior[k1] = MaxPriorityQueue()

    # Update cosine similarity of every cluster
    for i in range(N):
        if alive[i] == 1 and i != k1:
            prior[i].remove(cos_i_j[i][k1])
            prior[i].remove(cos_i_j[i][k2])
            # Update cosine similarity for the new cluster
            new_sim = max(cosine(i + 1, k1 + 1), cosine(i + 1, k2 + 1))
            cos_i_j[i][k1][0] = new_sim
            prior[i].insert(cos_i_j[i][k1])

            cos_i_j[k1][i][0] = new_sim
            prior[k1].insert(cos_i_j[k1][i])
```

（4）將得到的結果進行處理，並分別將 K = 8, 13, 20 的結果存入 8.txt、13.txt、20.txt 中。

HAC 演算法中，存取各文章相關 similarity 的 priority queue，是使用 heap 實作