HW4 Documentation 資管碩一 r06725015 李尚恩

1. 執行環境：Windows，Jupyter
2. 程式語言：Python 3.6.2，需要 numpy套件
3. 執行方式：（C.npy、I.npy與doc\_vector.npy需在同個資料夾）

先使用 pip 安裝 numpy

在 command line 上輸入：

python HW4.py 或 python3 HW4.py 或 python3.6 HW4.py

執行完畢後，可在同個資料夾底下看到8.txt，13.txt，20.txt。

1. 作業處理邏輯說明：

以下邏輯處理以程式段落順序說明：

1. 第一段：定義文章數量，讀入文章的 tf-idf vector、 C 與 I 矩陣。此部分接續上次 HW2 的結果，將 HW2 的結果處理完後存成 npy 檔直接讀入以加速程式運行。要特別注意的是，這部分的 index 為 0~1094 是實際的文章編號減一。
2. 第二段：Sim 為 cosine similarity 的 function，即將兩篇文章的 tf-idf vector dot 取得 similarity。find\_max\_sim 這個 function 是去尋找目前最大的 similarity pair，其中也會判斷該兩篇文章是不是還沒有被合併過，回傳配對的文章編號。
3. 第三段：按照老師投影片的 code 寫出 HAC 演算法。我是使用 complete link 的方式來更新 similarity，也就是取 pair 中 similarity 較小(距離較遠)的作為更新。

1. 第四段：定義寫檔的 function，將分群的結果依照作業要求格式分別寫入 K.txt 的檔案中。
2. 第五段：將 merge pairs A 的結果組合起來，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 A 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。
3. 心得：

這次作業的重點在於對於演算法的了解，透過這樣的實作的確讓我們更加清楚演算法的運作，一步一步將群組堆疊起來的感覺。理解上比較困難的可能是在更新距離(Similarity)的那一部分，每一輪選到一組 Similarity 最大的 pair 後，要去對所有文章與這組群集的距離作調整，剛開始會有點混亂不知道怎麼寫才是對的，還要稍微畫圖去看一下，可能是因為每次更新完之後，都會有一個文章被移除，而該文章就合併到某個群組中，每一個群組跟每一個群組之間的距離又都是由不同文章跟不同文章算出來的，才會覺得有點複雜。