HW2 Documentation 資管碩一 r06725015 李尚恩

1. 執行環境：MacOS，Jupyter & Terminal
2. 程式語言：Python 3.6.2，需要 nltk 與 collections 套件
3. 執行方式：（IRTM 需在同個資料夾）

在 command line 上輸入，：

python HW2.py 或 python3 HW2.py 或 python3.6 HW2.py

執行完畢後，可在同個資料夾底下看到 dictionary.txt，然後可在 vector 資料夾中看到每個文章的 txt 檔，而裡面就是該文章的 tf-idf vector。

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：import 所需要使用的 python 套件，並從 nltk 中取得 stop words 的清單。
2. 第二段：定義全域變數，包含文章總數（doc\_num），存 df 的 dictionary（df\_dict），存字詞 index 的 dictionary（index\_dict），存各文章字詞的清單（doc\_words），還有檔案路徑。
3. 第三段：對全部文章進行 tokenization、stemming 與刪除 stop words，並同時統計 df 存入 dictionary 中，然後也將每個文章個別出現的詞存入清單。
4. 第四段：將統計完的 df 依照字母進行排序，然後再依照題目的格式將結果存入 dictionary.txt，完成作業的第一小題。
5. 第五段：再來我們處理每篇文章的字詞清單，統計每個字詞的 frequency，然後計算出它們 tf-idf，並將 vectors 標準化，依照題目格式輸出成 txt 存入 vector 資料夾中，完成作業的第二小題。
6. 第六段：依照題目的函式設計實際寫出 cosine function，分別讀入兩個文章的標準化 vectors 後，將兩個 vectors 相乘取總和，即可得到兩個 vectors 的 cosine similarity。完成作業的第三小題。文章 1, 2 的 cosine 值為 0.188265。
7. 心得：

我覺得這次作業與上次的感覺不太相同，上次的作業比較著重在資料上的處理，正確地讀取及切割資料等等，而這次雖然也有接續上次的部分，但更著重在資料結構上，要怎麼規劃及定義儲存資料的結構，才能儲存所有需要的數值。經過這次作業也更加靠近資料檢索的核心，實作了 cosine similarity 大概可以體會，那些搜尋引擎是以怎樣的方式來比較不同文章、字句之間的關係。