HW2 Report B05702095 會計四 黃禹翔

1. 執行環境：

Windows Powershell

2. 程式語言：

Python 3.7

3. 執行方式：

在執行程式之前，需要安裝以下幾個套件：

$ pip install numpy nltk num2words

其中，numpy 是 numerical python，用以協助數值運算；nltk 是 natural language tookit，用以協助資料前處理；num2words 是一個協助擴充數字英文停用字的小套件。

執行方式：

```
PS D:\python code\IR-Text-Mining\hw2\hw2> python .\hw2-b05702095.py
```

$ python hw2-b05702095.py

由於這次報告是要對特定的資料集做運算，因此我在程式裡寫死檔案路徑，直接執行即可。我將資料集包在同一個資料夾中，執行完會產生一個檔案、一個路徑，包含：

- dictionary.txt：
  字典，格式為：
  t_index term df
  1 aan 1
  2 aaron 2
  3 ab 1
  …

- IRTMhw2tfidfVec/：
  包含所有以 TFIDF 形式儲存的文件，每份文件格式舉例如下：
  120
  t_index tf-idf
  66 0.06218456522882724
  201 0.04755538725119509
  333 0.023723387086798435
  …
  其中，第一行為 term 的種類數

為防萬一助教有重製的需求，我將我原始資料集包含在壓縮檔中，若助教需要重製，則將 IRTMhw2tfidfVec 目錄內的所有檔案包含目錄移出，重新執行程式即可，

4. 作業處理邏輯說明：

首先，先遍歷過資料集中所有的文件，透過統一的前處理後(由作業一)計算每個 term 的 document frequency 並存成字典(dictionary.txt)。接著，利用建立好的字典去計算每個文件中各格 term 的 tfidf 值，在這裡，我採用了 Sublinear TF Scaling 的方式平滑，因為我滿認同課堂上所講，一個 term 多出現一次並不會使得 term 在文章中的重要性突然暴增兩倍。最後，計算出所有文件的 TFIDF，並轉成單位向量、排序詞 ID 的形式存儲。

計算兩個文件之間 cosine 相似度的時候，由於事先排序過詞 ID 以及轉成單位向量，因此可以用 O(m)，其中 m 為 term 較多的文件 TFIDF 向量的長度。
計算過後，文件 1 和文件 2 的相似度為 0.15328，相較於沒有做 TFIDF 平滑的 0.16 多稍微下降，但頗為相近，而我相信這更好的反應了兩者的相似度，理由如前述。

5. 遇到的困難：

目前我比較困惑的是建立字典的部分，建立字典需要遍歷過所有文件，之後計算各文件 TFIDF 向量的時候又會遍歷過一次，不知道有沒有辦法節省這部分的手續或時間呢？