Introduction-to-Information-Retrieval-and-Text-Mining, HW 2

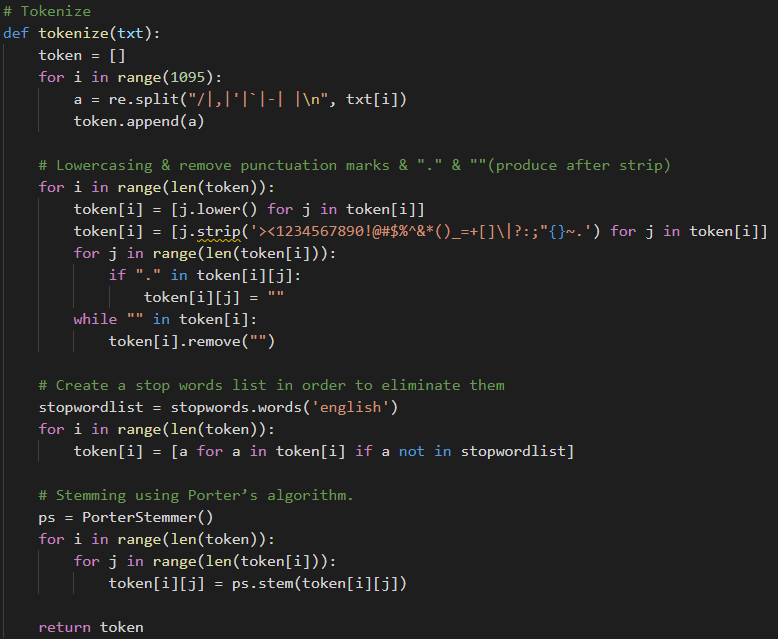
B07302230, 財金三 王博奕

1. 執行環境：Visual Studio Code
2. 程式語言：Python 3.8
3. 執行方式：
   1. Pip install nltk以便於接下來使用Porter’s algorithm 及 stopword。
   2. Pip install numpy 以計算consine similarity。
   3. Pip install math 以計算idf與tf-idf。
   4. Pip install re 以便切字。
4. 作業處理邏輯說明：

本次作業將延續作業一繼續往下進行，架構主要分為四部分，第一部分為Tokenize，第二部分為產生字典，第三部分為將字典裏面的Df轉變為idf以便計算各個文章的tf-idf，第四部分為cosine similarity的公式。以下將依各個部分分點說明。

1. **Tokenize:**

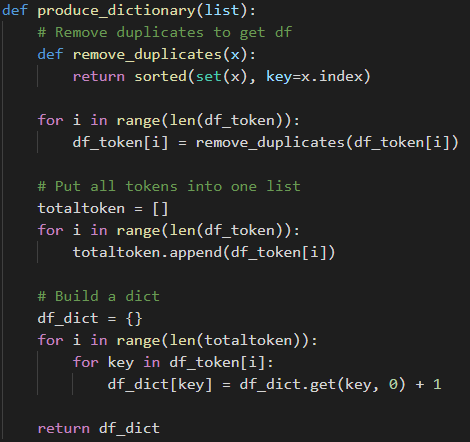
此處架構上與作業一基本上相似，但這裡我使用了re.split的方式，好處是可以一次針對多個來進行split。值得注意的是我把有”.”的term直接變成空格再把他remove掉，原因是實作上會出現u.s以及p.m的情況，我認為這些term也不是一個完整的詞，故就把移除掉。



1. **Produce\_dictionary:**

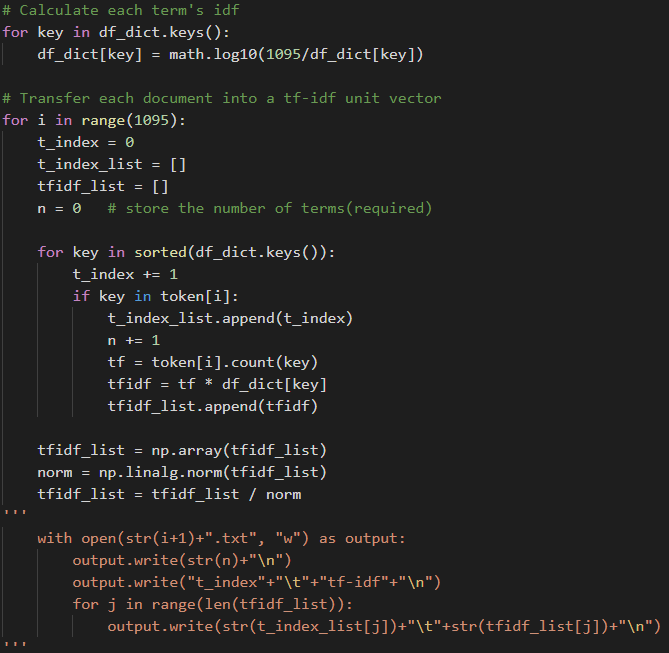
這裡我先寫一個將重複的term刪掉的函數，因為現在我們要先算df。假設一個文章出現兩次apple，就這個文章而言df只能算1。

接著我把每篇文章的token丟進一個totoaltoken的list，並且將其建成字典。



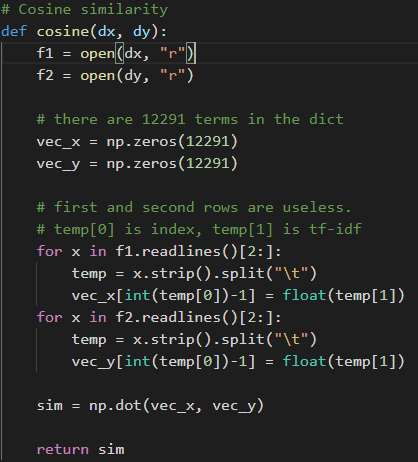
1. **Get each term’s tf-idf:**

這裡我將字典裡的df變成idf，並且再乘上tf即可得到tf-idf，再輸出每個文件的vector file。



1. **Cosine similarity:**

我計算cosine similarity的方式是先將原先output doc再輸入一個空的array，再把兩個array做內積即可算出cosine similarity。



最後，將輸出的tf-idf丟進cosine similarity的函數裡，跑出的相似度值為0.185。此處可以將document1和document2的路徑改成想要比較的文章的路徑即可算出cosine similarity。

