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：
   A. Pip install nltk 以便於接下來使用 Porter's algorithm 及 stopword。
   B. Pip install numpy 以計算 consine similarity。
   C. Pip install math 以計算 idf 與 tf-idf。
   D. 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。

A. 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。

```
# Tokenize
def tokenize(txt):
    token = []
    for i in range(1095):
        a = re.split("/|,|'|`|-| |\n", txt[i])
        token.append(a)

    # Lowercasing & remove punctuation marks & "." & ""(produce after strip)
    for i in range(len(token)):
        token[i] = [j.lower() for j in token[i]]
        token[i] = [j.strip('><1234567890!@#$%^&*()_=+[]\|?:;"{}~.') for j in token[i]]
        for j in range(len(token[i])):
            if "." in token[i][j]:
                token[i][j] = ""
        while "" in token[i]:
            token[i].remove("")

    # Create a stop words list in order to eliminate them
    stopwordlist = stopwords.words('english')
    for i in range(len(token)):
        token[i] = [a for a in token[i] if a not in stopwordlist]

    # Stemming using Porter's algorithm.
    ps = PorterStemmer()
    for i in range(len(token)):
        for j in range(len(token[i])):
            token[i][j] = ps.stem(token[i][j])

    return token
```

B. 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。

```
def produce_dictionary(list):
    # Remove duplicates to get df
    def remove_duplicates(x):
        return sorted(set(x), key=x.index)

    for i in range(len(df_token)):
        df_token[i] = remove_duplicates(df_token[i])

    # Put all tokens into one list
    totaltoken = []
    for i in range(len(df_token)):
        totaltoken.append(df_token[i])

    # Build a dict
    df_dict = {}
    for i in range(len(totaltoken)):
        for key in df_token[i]:
            df_dict[key] = df_dict.get(key, 0) + 1

    return df_dict
```

C. 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。

```
# Calculate each term's idf
for key in df_dict.keys():
    df_dict[key] = math.log10(1095/df_dict[key])

# Transfer each document into a tf-idf unit vector
for i in range(1095):
    t_index = 0
    t_index_list = []
    tfidf_list = []
    n = 0   # store the number of terms(required)

    for key in sorted(df_dict.keys()):
        t_index += 1
        if key in token[i]:
            t_index_list.append(t_index)
            n += 1
            tf = token[i].count(key)
            tfidf = tf * df_dict[key]
            tfidf_list.append(tfidf)

    tfidf_list = np.array(tfidf_list)
    norm = np.linalg.norm(tfidf_list)
    tfidf_list = tfidf_list / norm
'''
    with open(str(i+1)+".txt", "w") as output:
        output.write(str(n)+"\n")
        output.write("t_index"+"\t"+"tf-idf"+"\n")
        for j in range(len(tfidf_list)):
            output.write(str(t_index_list[j])+"\t"+str(tfidf_list[j])+"\n")
'''
```

D. 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。

```
# Cosine similarity
def cosine(dx, dy):
    f1 = open(dx, "r")
    f2 = open(dy, "r")

    # there are 12291 terms in the dict
    vec_x = np.zeros(12291)
    vec_y = np.zeros(12291)

    # first and second rows are useless.
    # temp[0] is index, temp[1] is tf-idf
    for x in f1.readlines()[2:]:
        temp = x.strip().split("\t")
        vec_x[int(temp[0])-1] = float(temp[1])
    for x in f2.readlines()[2:]:
        temp = x.strip().split("\t")
        vec_y[int(temp[0])-1] = float(temp[1])

    sim = np.dot(vec_x, vec_y)

    return sim
```

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。

```
document1 = "C:\\Users\\asdfg\\OneDrive - g.ntu.edu.tw\\NTU\\109-1\\109-1IRTM\
document2 = "C:\\Users\\asdfg\\OneDrive - g.ntu.edu.tw\\NTU\\109-1\\109-1IRTM\

sim = cosine(document1, document2)
print("The cosine similarity of document1 and document2 is: " + str(sim))
```

```
The cosine similarity of document1 and document2 is: 0.185363411922876
```