COMP 6000I Search Engines and Applications
Fall 2018 Homework 2
ZHAO Zhuochan.   20551093

Report for hw2 programming

(i) Explanation of your design

1. Preprocess the documents and store the words in L_all(type: list[list[str]]). The elements in L_all are also lists, and each list contains all the words of one particular document.

Related function: preprocess();

code1: preprocess()

```
def preprocess():
    L = []                                         #type:List[List[str]]
    fo = open("collection-100.txt")
    r = '[’!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~]+'     # remove all the
blanks and punctuations
    for line in fo.readlines():
        line = line.strip()
        line = line.lower()
        line = re.sub(r, '', line)
        if line:
            L.append(line.split())
    for i in range(len(L)):                      # delete the ending 's'
        for j in L[i]:
            if j.endswith("s"):
                x = L[i].index(j)
                L[i][x] = j[0:len(j) - 1]
    for i in range(len(L)).                      # remove the words less
than 4 characters
        for j in L[i]:
            if len(j) < 4:
                L[i][L[i].index(j)] = ""
    for i in range(len(L)):
        while '' in L[i]:
            L[i].remove('')
    fo.close()
    return L
```

2. Create an index named inverted_file(type: dict[str, any]), which contains every keywords and its corresponding posting(including the DID, and the positions of the indexed word in the document).

Related function: create_an_index(), find_documents(t)

code2 :create_an_index()

```
    def find_documents(t):
    lis = []
    for i in range(len(L_all)):
        if t in L_all[i]:
            v = []
            v.append(i)
            for j in range(len(L_all[i])):
                if (j not in v[1:]) & (L_all[i][j] == t):
                    v.append(j)
            lis.append(v)
    return lis
def create_an_index():
    seq = []
    for i in range(len(L_all)):
        for j in L_all[i]:
            if j not in seq:
                seq.append(j)
    inverted_file = dict.fromkeys(seq)
    for i in inverted_file:
        inverted_file[i] = find_documents(i)
    return inverted_file
```

3. Compute the weights of all words and stored in weight (type: list[list[float]]), in the meanwhile, compute the number of unique words for each document and store in unique (type: list[int]).

Related function: compute_weight(l)

code3: compute_weight(l)

```
def compute_weight(l):
    tfreq = []
    dfreq = []
    weight = []
    for i in range(len(l)):
        tfreq.append(dict.fromkeys(l[i]))
        for j in tfreq[i]:
            tfreq[i][j] = l[i].count(j)
    for i in range(len(l)):
        dfreq.append(dict.fromkeys(l[i]))
        for k in dfreq[i]:
            num = 0
            for n in range(len(l)):
                if k in l[n]:
                    num += 1
                dfreq[i][k] = num

    for i in range(len(l)):
        weight.append(dict.fromkeys(l[i]))
        for j in weight[i]:
            weight[i][j]  =  (tfreq[i][j]  /  max(tfreq[i].values()))  *
math.log((100/dfreq[i][j]), 2)

    unique = []
    for i in range(len(tfreq)):
        num = 0
        for j in tfreq[i]:
            if tfreq[i][j] == 1:
                num += 1
        unique.append(num)
    return [weight, unique]
```

4. Compute the L2norm based on weight and store it in norm (type: list[int]).

Related function: compute_L2norm(w).

code4 : compute_L2norm(w)

```
def compute_L2norm(w):
    norm = []
    for i in range(len(w)):
        num = 0
        for j in w[i]:
            num += math.pow(w[i][j],2)
        norm.append(math.sqrt(num))
    return norm
```

5. Read the query file, and save the words in q. (type: list[list[str]])

The elements in q are also lists, and each list contains all the words of one particular query.

code5: read the query

```
query = open("query-10.txt")                  # read the queries
q = []
r = '[’!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~]+'
for line in query.readlines():
    line = line.strip()
    line = line.lower()
    line = re.sub(r, '', line)
    if line:
        q.append(line.split())
```

6. For each particular query, compute the similarity of this query and each document, and get the top 3 documents' index and similarity value stored in index(list[int]) and sim[list[float]].

Related function: get_top3(q, w)

code6: get_top3(q,w)

```
def get_top3(q, w):
    s = []
    for i in range(len(w)):
        s.append(similarity(q, w[i]))
    top = heapq.nlargest(3, s)
    x = []
    y = []
    for i in range(3):
        for j in range(len(s)):
            if (s[j] == top[i]) & (j not in x):
                x.append(j)
                y.append(top[i])
    return [x[0:3], y[0:3]]
```

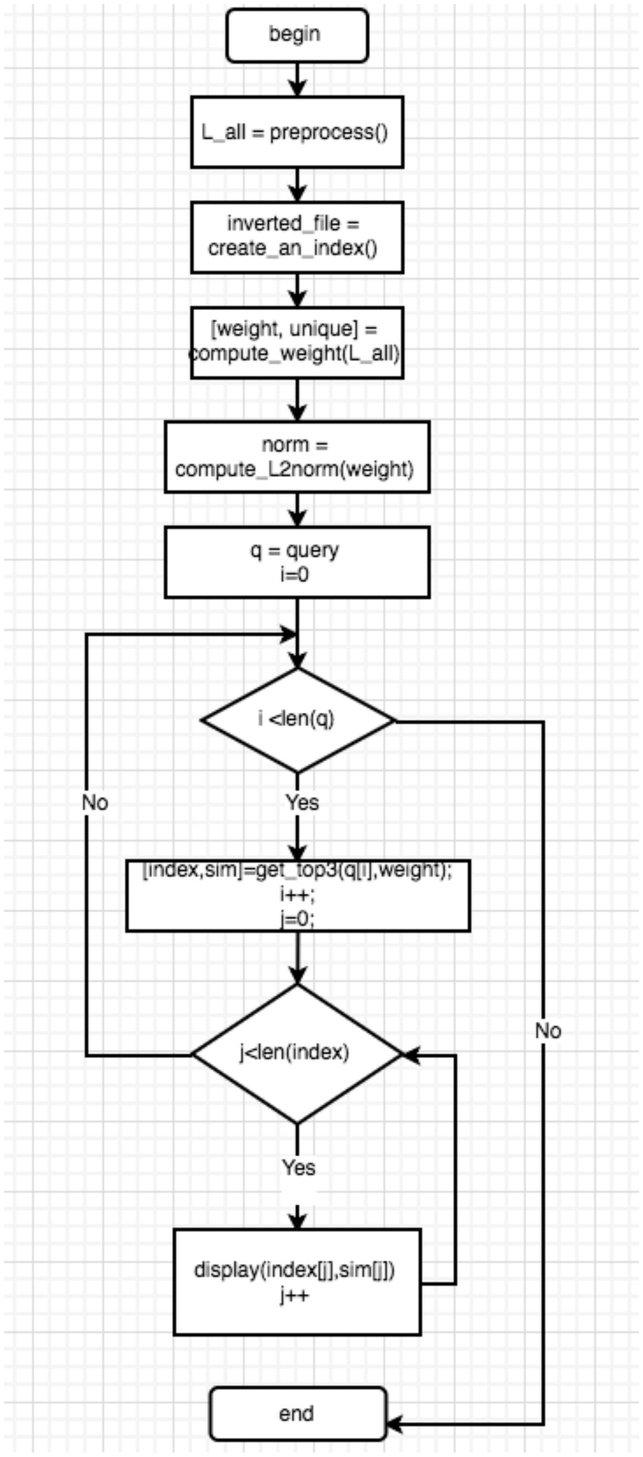
7. For each retrieved top 3 document, print the document ID first. Then get the 5 highest weighted keywords, and for each keyword, print it's posting lists. After that, print the number of unique keywords, L2norm and similarity score of the document.

Related function: display(index, sim) , get_5high(d).

code7: display()

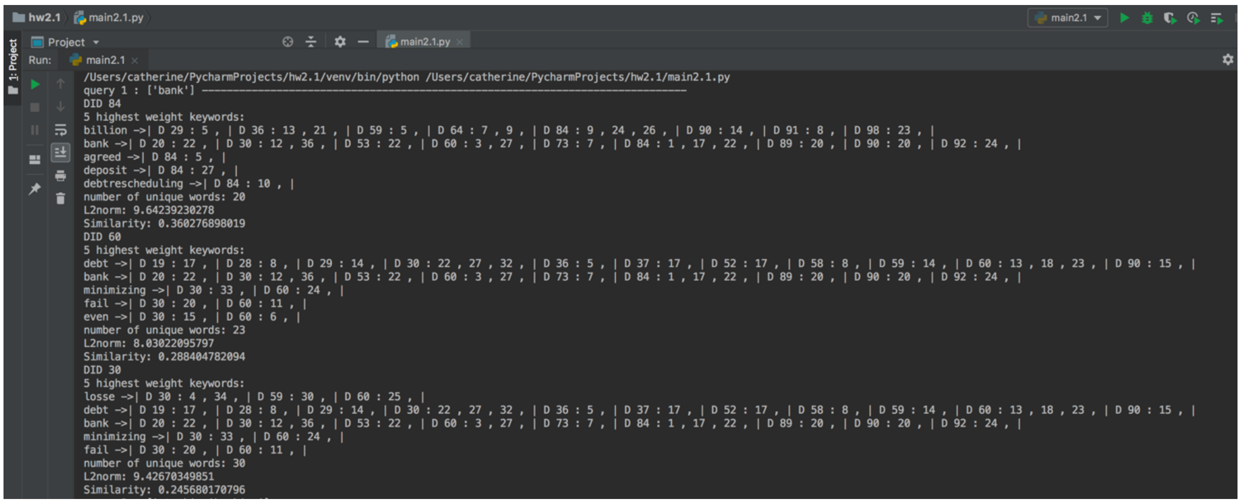
```
def get_5high(d):
    top = heapq.nlargest(5, d.values())
    highest = []
    for i in range(5):
        for j in d:
            if (d[j] == top[i]) & (~(j in highest)):
                highest.append(j)
    return highest[0:5]
def display(ind, sim):
    print "DID", ind+1
    print("5 highest weight keywords:")
    highest = get_5high(weight[ind])
    for i in range(5):
        lis = inverted_file[highest[i]]
        print highest[i], "->|",
        for j in range(len(lis)):
            print 'D', lis[j][0]+1, ":",
            for k in lis[j][1:]:
            print k, ",",
            print"|",
        print""
    print "number of unique words:", unique[ind]
    print "L2norm:", norm[ind]
    print "Similarity:", sim
```

(ii) A flowchart describing how the output in (c) is produced:


begin
L_all = preprocess()
inverted_file = create_an_index()
[weight, unique] = compute_weight(L_all)
norm = compute_L2norm(weight)
q = query
i=0
i <len(q)
Yes
No
[index,sim]=get_top3(q[i],weight);
i++;
j=0;
j<len(index)
Yes
No
display(index[j],sim[j])
j++
end

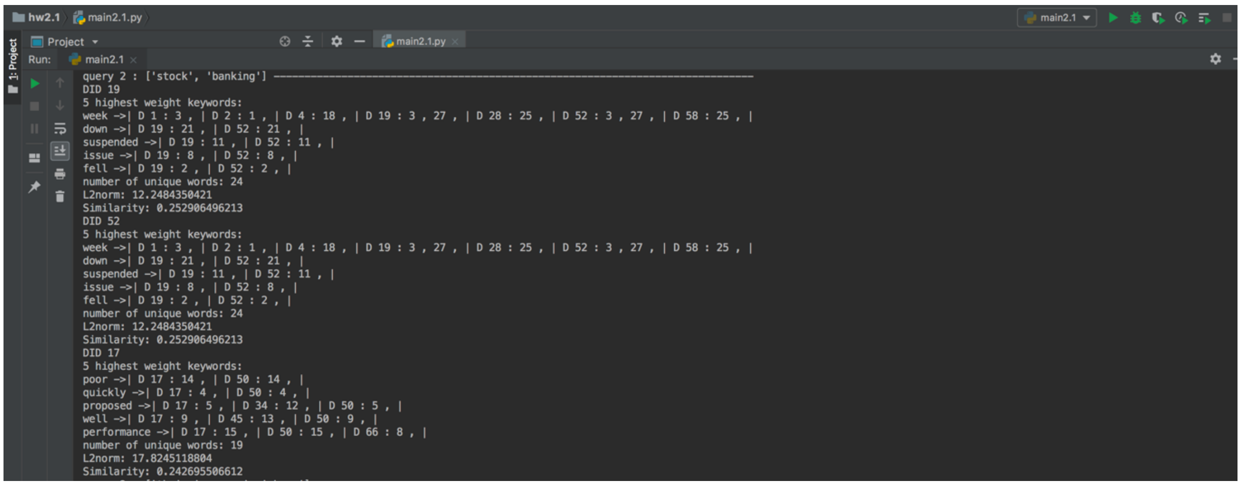
(iii) Answer to (c),(d) and (e) above.

(c): query 1:



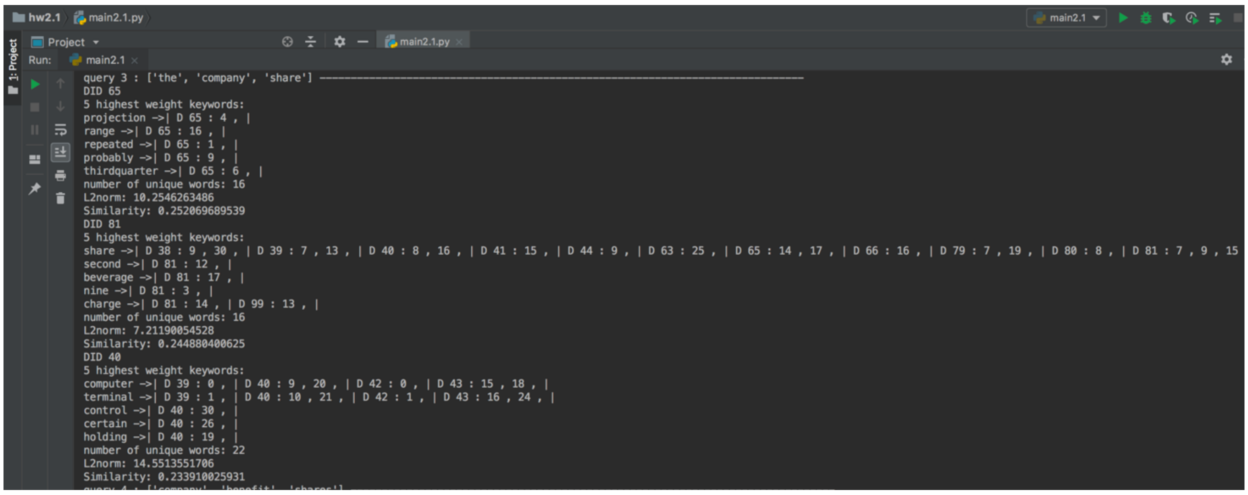
```
/Users/catherine/PycharmProjects/hw2.1/venv/bin/python /Users/catherine/PycharmProjects/hw2.1/main2.1.py
query 1 : ['bank'] ----------------------------------------------------------------------------------
DID 84
5 highest weight keywords:
billion ->| D 29 : 5 , | D 36 : 13 , 21 , | D 59 : 5 , | D 64 : 7 , 9 , | D 84 : 9 , 24 , 26 , | D 90 : 14 , | D 91 : 8 , | D 98 : 23 , |
bank ->| D 20 : 22 , | D 30 : 12 , 36 , | D 53 : 22 , | D 60 : 3 , 27 , | D 73 : 7 , | D 84 : 1 , 17 , 22 , | D 89 : 20 , | D 90 : 20 , | D 92 : 24 , |
agreed ->| D 84 : 5 , |
deposit ->| D 84 : 27 , |
debtrescheduling ->| D 84 : 10 , |
number of unique words: 20
L2norm: 9.64239230278
Similarity: 0.360276898019
DID 60
5 highest weight keywords:
debt ->| D 19 : 17 , | D 28 : 8 , | D 29 : 14 , | D 30 : 22 , 27 , 32 , | D 36 : 5 , | D 37 : 17 , | D 52 : 17 , | D 58 : 8 , | D 59 : 14 , | D 60 : 13 , 18 , 23 , | D 90 : 15 , |
bank ->| D 20 : 22 , | D 30 : 12 , 36 , | D 53 : 22 , | D 60 : 3 , 27 , | D 73 : 7 , | D 84 : 1 , 17 , 22 , | D 89 : 20 , | D 90 : 20 , | D 92 : 24 , |
minimizing ->| D 30 : 33 , | D 60 : 24 , |
fail ->| D 30 : 20 , | D 60 : 11 , |
even ->| D 30 : 15 , | D 60 : 6 , |
number of unique words: 23
L2norm: 8.03022095797
Similarity: 0.288404782094
DID 30
5 highest weight keywords:
losse ->| D 30 : 4 , 34 , | D 59 : 30 , | D 60 : 25 , |
debt ->| D 19 : 17 , | D 28 : 8 , | D 29 : 14 , | D 30 : 22 , 27 , 32 , | D 36 : 5 , | D 37 : 17 , | D 52 : 17 , | D 58 : 8 , | D 59 : 14 , | D 60 : 13 , 18 , 23 , | D 90 : 15 , |
bank ->| D 20 : 22 , | D 30 : 12 , 36 , | D 53 : 22 , | D 60 : 3 , 27 , | D 73 : 7 , | D 84 : 1 , 17 , 22 , | D 89 : 20 , | D 90 : 20 , | D 92 : 24 , |
minimizing ->| D 30 : 33 , | D 60 : 24 , |
fail ->| D 30 : 20 , | D 60 : 11 , |
number of unique words: 30
L2norm: 9.42670349851
Similarity: 0.245680170796
```

query 2:



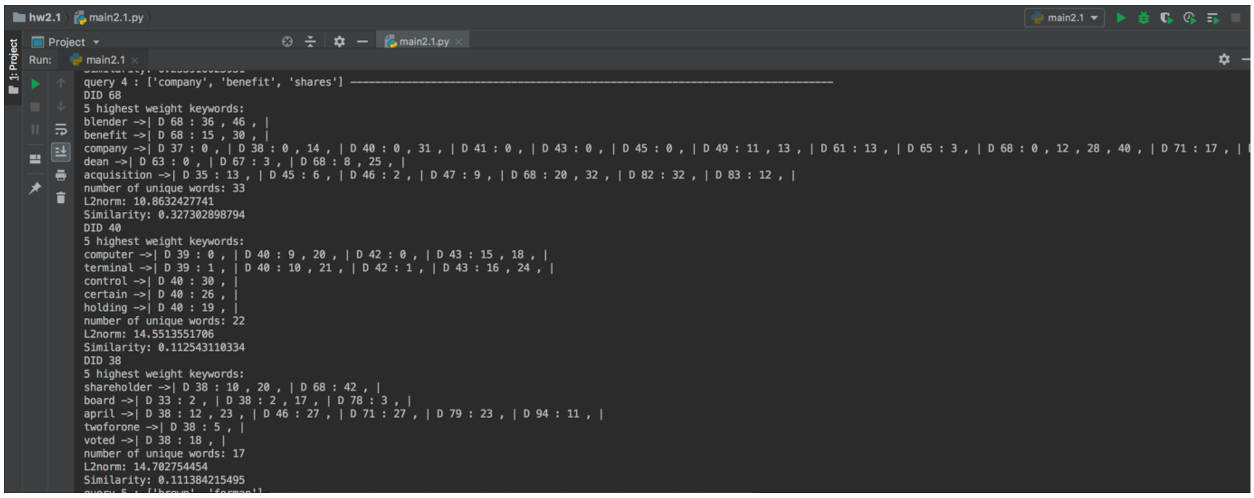
```
query 2 : ['stock', 'banking'] ----------------------------------------------------------------------------------
DID 19
5 highest weight keywords:
week ->| D 1 : 3 , | D 2 : 1 , | D 4 : 18 , | D 19 : 3 , 27 , | D 28 : 25 , | D 52 : 3 , 27 , | D 58 : 25 , |
down ->| D 19 : 21 , | D 52 : 21 , |
suspended ->| D 19 : 11 , | D 52 : 11 , |
issue ->| D 19 : 8 , | D 52 : 8 , |
fell ->| D 19 : 2 , | D 52 : 2 , |
number of unique words: 24
L2norm: 12.2484350421
Similarity: 0.252906496213
DID 52
5 highest weight keywords:
week ->| D 1 : 3 , | D 2 : 1 , | D 4 : 18 , | D 19 : 3 , 27 , | D 28 : 25 , | D 52 : 3 , 27 , | D 58 : 25 , |
down ->| D 19 : 21 , | D 52 : 21 , |
suspended ->| D 19 : 11 , | D 52 : 11 , |
issue ->| D 19 : 8 , | D 52 : 8 , |
fell ->| D 19 : 2 , | D 52 : 2 , |
number of unique words: 24
L2norm: 12.2484350421
Similarity: 0.252906496213
DID 17
5 highest weight keywords:
poor ->| D 17 : 14 , | D 50 : 14 , |
quickly ->| D 17 : 4 , | D 50 : 4 , |
proposed ->| D 17 : 5 , | D 34 : 12 , | D 50 : 5 , |
well ->| D 17 : 9 , | D 45 : 13 , | D 50 : 9 , |
performance ->| D 17 : 15 , | D 50 : 15 , | D 66 : 8 , |
number of unique words: 19
L2norm: 17.8245118804
Similarity: 0.242695506612
```

query 3:



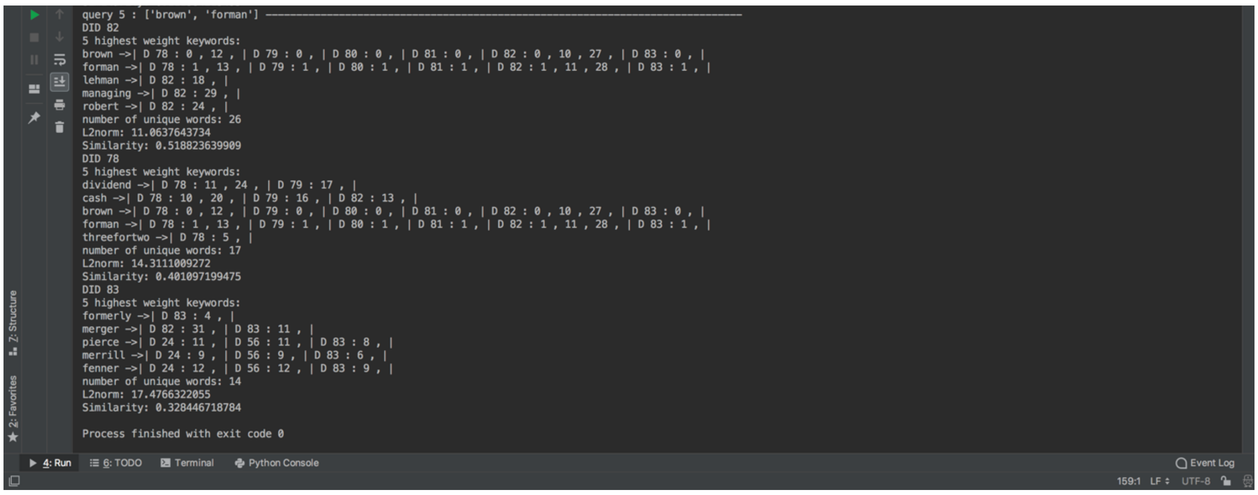
```
query 3 : ['the', 'company', 'share'] ----------------------------------------------------------------------------------
DID 65
5 highest weight keywords:
projection ->| D 65 : 4 , |
range ->| D 65 : 16 , |
repeated ->| D 65 : 1 , |
probably ->| D 65 : 9 , |
thirdquarter ->| D 65 : 6 , |
number of unique words: 16
L2norm: 10.2546263486
Similarity: 0.252069689539
DID 81
5 highest weight keywords:
share ->| D 38 : 9 , 30 , | D 39 : 7 , 13 , | D 40 : 8 , 16 , | D 41 : 15 , | D 44 : 9 , | D 63 : 25 , | D 65 : 14 , 17 , | D 66 : 16 , | D 79 : 7 , 19 , | D 80 : 8 , | D 81 : 7 , 9 , 15
second ->| D 81 : 12 , |
beverage ->| D 81 : 17 , |
nine ->| D 81 : 3 , |
charge ->| D 81 : 14 , | D 99 : 13 , |
number of unique words: 16
L2norm: 7.21190054528
Similarity: 0.244880400625
DID 40
5 highest weight keywords:
computer ->| D 39 : 0 , | D 40 : 9 , 20 , | D 42 : 0 , | D 43 : 15 , 18 , |
terminal ->| D 39 : 1 , | D 40 : 10 , 21 , | D 42 : 1 , | D 43 : 16 , 24 , |
control ->| D 40 : 30 , |
certain ->| D 40 : 26 , |
holding ->| D 40 : 19 , |
number of unique words: 22
L2norm: 14.5513551706
Similarity: 0.233910025931
```

query 4:



```
query 4 : ['company', 'benefit', 'shares'] ------------------------------------------------------------------------------
DID 68
5 highest weight keywords:
blender ->| D 68 : 36 , 46 , |
benefit ->| D 68 : 15 , 30 , |
company ->| D 37 : 0 , | D 38 : 0 , 14 , | D 40 : 0 , 31 , | D 41 : 0 , | D 43 : 0 , | D 45 : 0 , | D 49 : 11 , 13 , | D 61 : 13 , | D 65 : 3 , | D 68 : 0 , 12 , 28 , 40 , | D 71 : 17 , |
dean ->| D 63 : 0 , | D 67 : 3 , | D 68 : 8 , 25 , |
acquisition ->| D 35 : 13 , | D 45 : 6 , | D 46 : 2 , | D 47 : 9 , | D 68 : 20 , 32 , | D 82 : 32 , | D 83 : 12 , |
number of unique words: 33
L2norm: 10.8632427741
Similarity: 0.327302898794
DID 40
5 highest weight keywords:
computer ->| D 39 : 0 , | D 40 : 9 , 20 , | D 42 : 0 , | D 43 : 15 , 18 , |
terminal ->| D 39 : 1 , | D 40 : 10 , 21 , | D 42 : 1 , | D 43 : 16 , 24 , |
control ->| D 40 : 30 , |
certain ->| D 40 : 26 , |
holding ->| D 40 : 19 , |
number of unique words: 22
L2norm: 14.5513551706
Similarity: 0.112543110334
DID 38
5 highest weight keywords:
shareholder ->| D 38 : 10 , 20 , | D 68 : 42 , |
board ->| D 33 : 2 , | D 38 : 2 , 17 , | D 78 : 3 , |
april ->| D 38 : 12 , 23 , | D 46 : 27 , | D 71 : 27 , | D 79 : 23 , | D 94 : 11 , |
twoforone ->| D 38 : 5 , |
voted ->| D 38 : 18 , |
number of unique words: 17
L2norm: 14.702754454
Similarity: 0.111384215495
```

query 5:



```
query 5 : ['brown', 'forman'] ------------------------------------------------------------------------------
DID 82
5 highest weight keywords:
brown ->| D 78 : 0 , 12 , | D 79 : 0 , | D 80 : 0 , | D 81 : 0 , | D 82 : 0 , 10 , 27 , | D 83 : 0 , |
forman ->| D 78 : 1 , 13 , | D 79 : 1 , | D 80 : 1 , | D 81 : 1 , | D 82 : 1 , 11 , 28 , | D 83 : 1 , |
lehman ->| D 82 : 18 , |
managing ->| D 82 : 29 , |
robert ->| D 82 : 24 , |
number of unique words: 26
L2norm: 11.0637643734
Similarity: 0.518823639909
DID 78
5 highest weight keywords:
dividend ->| D 78 : 11 , 24 , | D 79 : 17 , |
cash ->| D 78 : 10 , 20 , | D 79 : 16 , | D 82 : 13 , |
brown ->| D 78 : 0 , 12 , | D 79 : 0 , | D 80 : 0 , | D 81 : 0 , | D 82 : 0 , 10 , 27 , | D 83 : 0 , |
forman ->| D 78 : 1 , 13 , | D 79 : 1 , | D 80 : 1 , | D 81 : 1 , | D 82 : 1 , 11 , 28 , | D 83 : 1 , |
threefortwo ->| D 78 : 5 , |
number of unique words: 17
L2norm: 14.3111009272
Similarity: 0.401097199475
DID 83
5 highest weight keywords:
formerly ->| D 83 : 4 , |
merger ->| D 82 : 31 , | D 83 : 11 , |
pierce ->| D 24 : 11 , | D 56 : 11 , | D 83 : 8 , |
merrill ->| D 24 : 9 , | D 56 : 9 , | D 83 : 6 , |
fenner ->| D 24 : 12 , | D 56 : 12 , | D 83 : 9 , |
number of unique words: 14
L2norm: 17.4766322055
Similarity: 0.328446718784

Process finished with exit code 0
```

(d):data structures used in this program:

1. type: list[list[str]]. name: L_all

The elements in L_all are also lists, and each list contains all the words of one particular document.

2. type:dict[str, any]. name: inverted_file

Inverted file is a dictionary. It's keys are all the terms in the collection, and value is the corresponding posting(including the DID, and the positions of the indexed word in the document).

3. type: list[list[float]]. name: weight

The elements in weight are also lists, and each list represents a document in the collection. Every list in L_all is consist of all words' weight of the corresponding document.

4. type: list[int]. name: uique

Each element in unique represents the number of unique words in corresponding document.

5. type: list[float]. name: norm

Each element in norm represents the L2norm of the corresponding document.

6. type: list[list[str]]. name: q

The elements in q are also lists, and each list contains all the words of one particular query.

(e) If the text passages are not updated, how would you design your program to speed up the computation of the similarity values?

Answer: Compute the weights of all the keywords in the documents and store them. When it comes a new query, compute the similarity of the query and each document.

(iv) Programming language and version, the OS, and libraries/packages used, and how to run the programs.

Answer:

Programming language: Python

Version: 2.7

OS: MAC OS

The modules I used: “division”, “re”, “math” and “heapq”.

How to run the programs: Open the folder named SE-hw2, and run the main.py file.