Srividya Majeti

Assignment 10

CS 532: Introduction to Web Science
Dr. Michael Nelson
Spring 2016

April 30, 2016

Contents

1

Question 1

Using the data from A8:

- Consider each row in the blog-term matrix as a 500 dimension vector, corresponding to a blog. - From chapter 8, replace numpredict.euclidean() with cosine as the distance metric. In other words, you'll be computing the cosine between vectors of 500 dimensions.

- Use knnestimate() to compute the nearest neighbors for both:

`http://f-measure.blogspot.com/`

`http://ws-dl.blogspot.com/`

for k=1,2,5,10,20.

Following are the steps I have taken to the solve the problem:

- I stored each row in the blog-term matrix in JSON structure where 'blogname' is the 'key' and the 'word count' is the 'value'. This code is illustrated in the Listing 1.1. The input for the code is the 'blogdata.txt' generated in A8 this file is located at `https://github.com/majetisiri/cs532-s16/blob/master/a8/q1-blogdata.txt`. The output JSON structure is stored in 'dataAsVector'.
- To find the nearest neighbors for the URIs 'http://f-measure.blogspot.com/' and 'http://ws-dl.blogspot.com/' using knnestimate I used the PCI book code from 'chapter 8'.
- The function 'main()' reads the 'dataAsVector' file as input which is a JSON structure and selects the value that correspond to the key 'F-measure' and 'Web Science and Digital Libraries Research Group' and stores it as 'vec1'.
- The function 'getdistances()' iterates through all the other blogs and stores each blog as 'vec2'. The distance between 'vec1' and 'vec2' is calculated using function 'cosineDistance()'.
- All the above computed cosine distances between vectors of 500 dimensions are sorted in ascending order and the nearest neighbors for 'http://f-measure.blogspot.com/' and 'http://ws-dl.blogspot.com/' for k=1,2,5,10,20 is done using 'knnestimate()'. This is illustrated in Listing 1.2.
- The output for 'http://f-measure.blogspot.com/' is illustrated in Table below1.5:
- **k=1**

Table 1.1. Output for 'http://f-measure.blogspot.com/'

Blog Name	Distance
La Fotografa Efectista Abstracta. Fotos Abstractas. Abstract Photos.	0.0154444270953

- **k=2**

Table 1.2. Output for 'http://f-measure.blogspot.com/'

Blog Name	Distance
La Fotografa Efectista Abstracta. Fotos Abstractas. Abstract Photos.	0.0154444270953
Parish Radio	0.0306191242689

- **k=5**

Table 1.3. Output for 'http://f-measure.blogspot.com/'

Blog Name	Distance
La Fotografa Efectista Abstracta. Fotos Abstractas. Abstract Photos.	0.0154444270953
Parish Radio	0.0306191242689
PLATTENTIPPS	0.0513198672441
MARISOL	0.063556611459
MR. BEAUTIFUL TRASH ART	0.0708618273208

- **k=10**

Table 1.4. Output for 'http://f-measure.blogspot.com/'

Blog Name	Distance
La Fotografa Efectista Abstracta. Fotos Abstractas. Abstract Photos.	0.0154444270953
Parish Radio	0.0306191242689
PLATTENTIPPS	0.0513198672441
MARISOL	0.063556611459
MR. BEAUTIFUL TRASH ART	0.0708618273208
Stonehill Sketchbook	0.077138600469
Boggle Me Thursday	0.0798942002771
INDIEohren.!	0.0919836419218
IoTube :)	0.104650493144
THE HUB	0.105225114925

- **k=20**

Table 1.5. Output for 'http://f-measure.blogspot.com/'

Blog Name	Distance
La Fotografa Efectista Abstracta. Fotos Abstractas. Abstract Photos.	0.0154444270953
Parish Radio	0.0306191242689
PLATTENTIPPS	0.0513198672441
MARISOL	0.063556611459
MR. BEAUTIFUL TRASH ART	0.0708618273208
Stonehill Sketchbook	0.077138600469
Boggle Me Thursday	0.0798942002771
INDIEohren.!	0.0919836419218
IoTube :)	0.104650493144
THE HUB	0.105225114925
Room 19's Blog 2016	0.107930979076
ORGANMYTH	0.121948739579
Rod Shone	0.125346786751
FlowRadio Playlists (and Blog)	0.126749970237
One Stunning Single Egg	0.130875110969
mattgarman	0.145942366409
Flatbasset	0.146490784699
The Stearns Family	0.146995772609
jaaackie.	0.153804726796
A H T A P O T	0.161421621637

- The output for 'http://ws-dl.blogspot.com/' is illustrated in Table below 1.6:
- **k=1**

Table 1.6. Output for 'http://ws-dl.blogspot.com/'

Blog Name	Distance
PLATTENTIPPS	0.00203252382358

- **k=2**

Table 1.7. Output for 'http://ws-dl.blogspot.com/'

Blog Name	Distance
PLATTENTIPPS	0.00203252382358
MARISOL	0.00686746006616

- **k=5**

Table 1.8. Output for 'http://ws-dl.blogspot.com/'

Blog Name	Distance
PLATTENTIPPS	0.00203252382358
MARISOL	0.00686746006616
CRUZANDO EL UNIVERSO...	0.0213236027762
adrianoblog	0.0246165816012
La Fotografa Efectista Abstracta. Fotos Abstractas. Abstract Photos.	0.025829508375

- **k=10**

Table 1.9. Output for 'http://ws-dl.blogspot.com/'

Blog Name	Distance
PLATTENTIPPS	0.00203252382358
MARISOL	0.00686746006616
CRUZANDO EL UNIVERSO...	0.0213236027762
adrianoblog	0.0246165816012
La Fotografa Efectista Abstracta. Fotos Abstractas. Abstract Photos.	0.025829508375
IoTube :)	0.0456300688804
Boggle Me Thursday	0.0567821571865
If You Give a Girl a Camera...	0.0666690597973
Mystic Chords Of Memory	0.0673182085108
Flatbasset	0.0758444571954

- **k=20**

Table 1.10. Output for 'http://ws-dl.blogspot.com/'

Blog Name	Distance
PLATTENTIPPS	0.00203252382358
MARISOL	0.00686746006616
CRUZANDO EL UNIVERSO...	0.0213236027762
adrianoblog	0.0246165816012
La Fotografa Efectista Abstracta. Fotos Abstractas. Abstract Photos.	0.025829508375
IoTube :)	0.0456300688804
Boggle Me Thursday	0.0567821571865
If You Give a Girl a Camera...	0.0666690597973
Mystic Chords Of Memory	0.0673182085108
Flatbasset	0.0758444571954
INDIEohren.!	0.0767945198989
The Campus Buzz on WSOU	0.0918046411881
The Stearns Family	0.0934455499801
One Stunning Single Egg	0.0948170501912
Lo importante es que estes t bien	0.0948286097743
Lo importante es que estes t bien	0.0948286097743
THE HUB	0.0951894062985
jaaackie.	0.0965323470041
mattgarman	0.0981283528512
knstler treu	0.102859292064

Code Listing

```
import json
f= open('blogdata.txt','r')
output = open('dataAsVector','w')
list1 = []
count = 0
for line in f:
  count += 1
  if count >1:
    dict = {}
    line = line.strip()
    list = line.split('\t')
    blogName= list[0]
    list.pop(0)
    vector = list
    dict[blogName] = vector
    list1.append(dict)
output.write(json.dumps(list1))
```

Listing 1.1. python code for storing elements in each row of blog matrix in a vector.

Code Listing

```
from random import random,randint
import math
import json

def cosineDistance(v1,v2):
  "compute cosine similarity of v1 to v2: (v1 dot v2)/{||v1
      ||*||v2||)"
  sumxx, sumxy, sumyy = 0, 0, 0
  for i in range(0,len(v1)-1):
    x = int(v1[i]); y = int(v2[i])
    sumxx += x*x
    sumyy += y*y
    sumxy += x*y
  return sumxy/math.sqrt(sumxx*sumyy)


def getdistances(data,vec1):
  distancelist=[]

  for i in data:
    for subitem in i:
      if subitem != 'F-Measure':
        vec2= i[subitem]

    distancelist.append((cosineDistance(vec1,vec2),i))
  distancelist.sort()
```

```
26    return distancelist
27
28  def knnestimate(data,vec1,k=5):
29    dlist=getdistances(data,vec1)
30    avg=0.0
31    for i in range(k):
32      idx=dlist[i]
33      print idx
34
35  def main():
36    f= open('dataAsVector','r')
37    data = json.load(f)
38    for item in data:
39      for subitem in item:
40        if subitem == 'F-Measure':
41          vec1= item[subitem]
42          knnestimate(data,vec1)
43
44  main()
```

Listing 1.2. some functions in'numpredict.py' from PCI book code

2

Question 3

Re-download the 1000 TimeMaps from A2, Q2. Create a graph where the x-axis represents the 1000 TimeMaps. If a TimeMap has "shrunk", it will have a negative value below the x-axis corresponding to the size difference between the two TimeMaps. If it has stayed the same, it will have a "0" value. If it has grown, the value will be positive and correspond to the increase in size between the two TimeMaps.

As always, upload all the TimeMap data. If the A2 github has the original TimeMaps, then you can just point to where they are in the report.

Following are the steps I have taken to solve the problem:

- I repeated the same process that I did in Assignment 2 for getting the mementos data.
- With the help of ODU Memento Aggregator, I downloaded TimeMaps for all the URIs that are extracted in Assignment 2 question 1 using the following curl command:
  `curl-i--silenthttp://mementoproxy.cs.odu.edu/aggr/timemap/link/1/<uri>`
  These URIs are located at `https://github.com/majetisiri/cs532-s16/blob/master/a2/uri.json`.
- I stored the output produced by the cURL command into a file 'PagesList'.
- I processed the data from the above file to create a JSON with the original URI, memento URI , datetime and memento count. The memento count for each URI is derived by counting the URIs with rel="memento". This is outlined in Listing 2.1.

- Using 'getTimeMaps.py' I wrote the memento count for all the 1000 URIs for Assignment 2 and Assignment 10 in 2 different files 'allMementosA2' and 'allMementosA10' respectively.This code is listed in Listing 2.2.
- I calculated the difference between TimeMaps for Assignment 10 and Assignment 2 and stored the output in 'differenceTimemaps'. This code is illustrated in Listing 2.3
- The output graph where the x-axis represents the 1000 TimeMaps is illustrated in Figure 2.1.

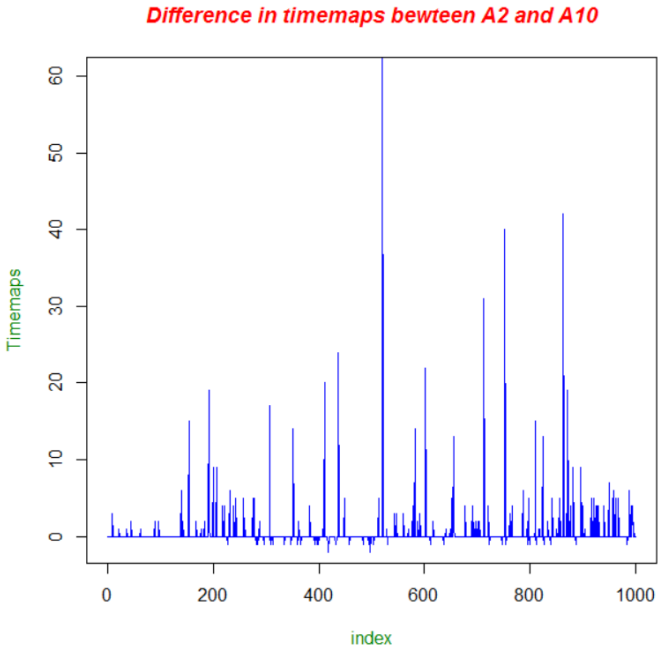

Fig. 2.1. Output graph with time maps on x axis

- From this graph we can conclude that most of the TimeMaps are 'grown' as the value is positive. But we can see few TimeMaps are 'shrunk' which have negative values.

Code Listing

```
import commands
import re
import json
import sys

def geturi():
  file=open("uri.json","r")
  uricounter = 0
  for line in file:
    if uricounter <1000:
      f = open('PagesList','w')
      response = getPages(line)
      f.write(response)
      f.close()
      finalCount =getMementosData()
      uricounter += 1
  file.close()

def getPages(uri):
  timemapUri = "http://mementoproxy.cs.odu.edu/aggr/timemap/link/1/"
  command ="curl -i --silent " + timemapUri + str(uri).strip()

  pageList = commands.getoutput(command)
  return pageList

def getMementosData():
  getMementosDataFile = open("PagesList","r")
  outputfile= open('mementoData.json', 'a')
  mementoList= []
  mementoJson ={}
  Json ={}
  count = 0
  for line in getMementosDataFile:
    if 'rel="original"' in line:
      count = 0
      originalLink = (re.findall(r'(https?://[^\s]+>)', line))[0][:-1]
      Json['originaluri'] = originalLink
      print Json
    if 'rel="memento"' in line:
      count += 1
      link = ""
      if re.findall(r'(https?://[^\s]+>)', line):
        link = (re.findall(r'(https?://[^\s]+>)', line))[0][:-1]
```

```
44          elif re.findall(r'(www.[^\s]+>)', line):
45            link = (re.findall(r'(www.[^\s]+>)', line))[0][:-1]
46          else:
47            print line
48          mementoId = count
49          mementoJson['mementouri'] = link
50          mementoJson['id'] =mementoId
51          if(line.find('datetime="') > -1):
52            datetime = (line.split('datetime="'))[1].split('"')
                  [0]
53            mementoJson['datetime'] =datetime
54          mementoList.append(mementoJson)
55          Json['memento'] = mementoList
56    finalCount = str(count)
57    Json['count'] = finalCount
58    outputfile.write(json.dumps(Json) + "\n")
59
60  geturi()
```

Listing 2.1. Python code for getting mementos data.

Code Listing

```
1  import json
2  import datetime
3  import dateutil.parser
4
5  f= open('mementoData.json','r')
6  f1 = open('allMementosA10','w')
7  count =0
8
9  for line in f:
10   count+=1
11   if count <1001:
12     data= json.loads(line)
13     f1.write(data['count']+'\n')
14
15 f1.close()
16
17 f= open('allMementosA10','r')
18 f1= open('allMementosA2','r')
```

Listing 2.2. python code for getting time maps from mementos data.

Code Listing

```
1  from operator import sub
2  f1= open('allMementosA10','r')
3  f2= open('allMementosA2', 'r')
4  f3= open('differenceTimeMaps', 'w')
5
6  list = []
7  list1 =[]
8  resultList= []
9
10 for line in f1:
11   list.append(int(line.strip()))
12
13 print list
14
15 for line in f2:
16   list1.append(int(line.strip()))
17
18 print list1
19
20 resultList = [a - b for a, b in zip(list, list1)]
21 print resultList
22
23 for item in resultList:
24   f3.write(str(item)+'\n')
```

Listing 2.3. python code for getting difference in time maps for mementos in Assignment 2 and Assignment 10.

3

Question 4

Repeat A3, Q1. Compare the resulting text from February to the text you have now. Do all 1000 URIs still return a "200 OK" as their final response (i.e., at the end of possible redirects)?

Create two graphs similar to that described in Q3, except this time the y-axis corresponds to difference in bytes (and not difference in TimeMap magnitudes). For the first graph, use the difference in the raw (unprocessed) results. For the second graph, use the difference in the processed (as per A3, Q1) results.

Of the URIs that still terminate in a "200 OK" response, pick the top 3 most changed (processed) pairs of pages and use the Unix "diff" command to explore the differences in the version pairs.

Following are the steps I have taken to solve the problem:

- To find if all the 1000 URIs still return '200 OK', I checked the history of each URI and created a JSON structure which stores the 'status code' as 'key' and the 'count' as value. I have written this JSON structure in a file 'status_CodeWithUriCount.json'. This code is listed in Listing 3.1. The output for this is illustrated in Table below: 3.1

Table 3.1. Status code and count

Status code	count
417	1
200	819
429	3
403	52
404	80
503	43
500	1
410	1

- From the above table we can see that '819' URIs returns a '200OK' as their final response at the end of possible redirects.
- To get the raw data and processed data I repeated the steps in A3, Q1. I got the raw data for all the 1000 unique URIs that I collected in assignment 2, using the following cURL command:
  curl <URI> > < output filename >.
  These URIs are located at `https://github.com/majetisiri/cs532-s16/blob/master/a2/uri.json`
- I stored the raw HTML output generated by the cURL command in separate files for each URI and named the files based on their index. This code is listed in Listing 3.2
- Then I got the processed HTML and stored the output in separate files for each URI using the command:
  lynx -dump -force_html <URI> > < output filename >.
  I named the files with the URI index followed by a hyphen and the word 'processed'. This code is listed in Listing 3.3
- I calculated the size of each file in 'raw data' and 'processed data' in bytes for both Assignment 2 and Assignment 10. The output is stored in 'q4rawDataSizeA2', 'q4rawDataSizeA10', 'q4processedDataSizeA2' and 'q4processedDataSizeA10'.
- Furthermore I calculated the difference between the size of the files for 'rawdata' and 'ProcessedData' generated in 'Assignment 10' and 'Assignment 2'. The difference output is stored in 'q4differenceRawDataSize' and 'q4differenceProcessedDataSize' respectively.

- The output graph where the y-axis corresponds to the difference in bytes of the raw data is illustrated in Figure 3.1.

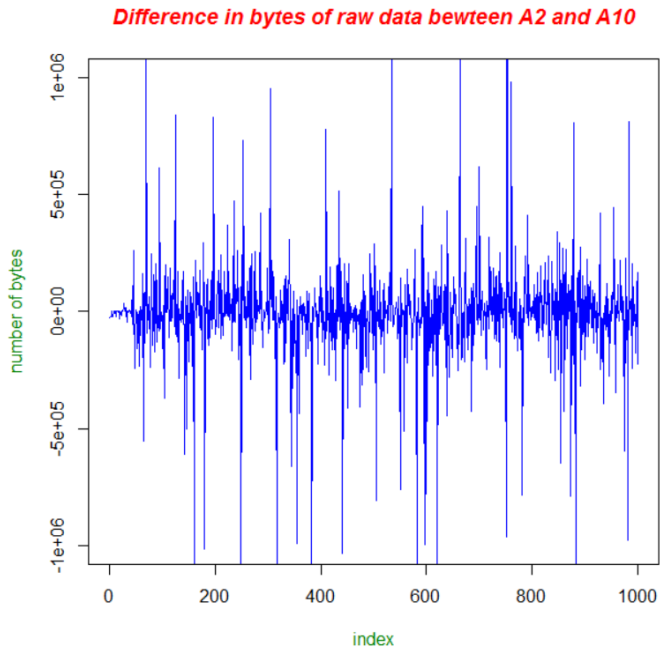

Fig. 3.1. Output graph with difference in bytes of raw data

- The output graph where the y-axis corresponds to the difference in bytes of the processed data is illustrated in Figure 3.2.

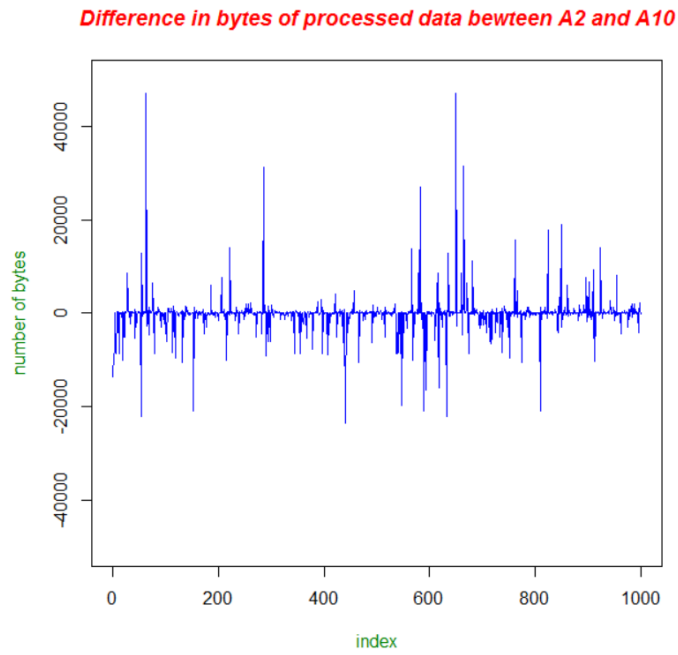

Fig. 3.2. Output graph with difference in bytes of processed data

Code Listing

```
1  from sets import Set
2  import requests
3  import json
4
5  f = open("1000uri.json","r")
6  file= open("status_CodeWithUriCount.json","w")
7  UriSet = Set()
8  uricounter = 0
9  codeDict = {}
10 for line in f:
11   if uricounter<1000:
12     link= line.strip()
13     print link
14     try:
15       r = requests.get(link)
16       print '[%s] %s' % (r.status_code, r.url)
17       if r.status_code in codeDict:
18         codeDict[r.status_code] += 1
19       else:
20         codeDict[r.status_code] = 1
```

```
21      except Exception, e:
22        print e
23        continue
24    uricounter += 1
25
26  file.write(json.dumps(codeDict))
27  file.close()
```

Listing 3.1. Python code for finding count for each status code

Code Listing

```
1  import commands
2  import os
3
4  count = 0
5  input=open('1000uri.json','r')
6  t=open('command','w')
7  for line in input:
8    count +=1
9    if count <1001:
10     filename= str(count)
11     command ='curl ' + '"' + str(line).rstrip('\n') + '"'+ '
           > ./rawData/' + filename
12     output = commands.getoutput(command)
13 input.close()
```

Listing 3.2. python code for getting processed data for each blog URI

Code Listing

```
1  import commands
2  import os
3
4  count = 0
5  input=open('1000uri.json','r')
6  t=open('command','w')
7  for line in input:
8    count +=1
9    if count <1001:
10     filename= str(count) + '-processed'
11     command ='lynx -dump -force_html ' + '"'+ str(line).
           rstrip('\n') + '"'+'> ./processedData/' + filename
12     print command
13     output = commands.getoutput(command)
14 input.close()
```

Listing 3.3. python code for getting raw data for each blog URI

Code Listing

```
1  import os
2
3  # path = "/home/smajeti/coursework/cs532/a10/q4/rawData/"
4  path = "/home/smajeti/coursework/cs532/a10/q4/processedData/
       "
5  file1 = open('processedDataSize','w')
6  dirs = os.listdir( path )
7
8  for file in dirs:
9    path1 = path + file
10   print path1
11   size= os.path.getsize(path1)
12   file1.write(str(size)+'\n')
```

Listing 3.4. Python code for getting bytes of each blog file in raw data and processed data

Code Listing

```
1  from operator import sub
2  f1= open('processedDataSize','r')
3  f2= open('A2/processedDataSize', 'r')
4  f3= open('differenceProcessedDataSize', 'w')
5
6  list = []
7  list1 =[]
8  resultList= []
9
10 for line in f1:
11   list.append(int(line.strip()))
12
13 for line in f2:
14   list1.append(int(line.strip()))
15
16 resultList = [a - b for a, b in zip(list, list1)]
17
18 for item in resultList:
19   f3.write(str(item)+'\n')
```

Listing 3.5. python code for getting difference between bytes of raw/processed data in assignmnet 2 and assignment 10

References

1. Python code knnestimate.py from PCI book. Igraph Tutorial. `https://github.com/cataska/programming-collective-intelligence-code/tree/master/chapter8`, 2007, Toby Segaran
2. Python code docclass.py from PCI book. Download GraphML for Karate Club. `https://github.com/uolter/PCI/blob/master/chapter6/docclass.py`, 2007, Toby Segaran