INTRODUCTION TO WEB SCIENCES:
Assignment 10

Manoj Chandra Kompalli

29 April 2016

Contents

1 Question 1:

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.

1.1 Approach

- I have used my previous blog matrix which was in tab seperated format and made a similar one in JSON format
- I had used numpredict.py from Programming Collective intelligence and added some more functionality like cosine similarity between between vectors.
- By using sorted distances knnestimate() method computes the nearest neighbours
- I have used the knnestimate()to find the nearest neighbours of both the F-Measure blog and web science blog
- Output gives the nearest neighbours of F-measure and Web science blogs

1.2 Code Listing

1.2.1 makelist.py

```
import json
file1= open('blogmatrix.txt','r')
file2 = open('bloglist','w')
output = []
count = 0
for line in file1:
  count=count+ 1
  if count >1:
    mydict = {}
    line = line.strip()
    input = line.split('\t')
    # print input
    name= input[0]
    input.pop(0)
    generated = input
    # print name
    # print generated
    mydict[name] = generated
    output.append(mydict)
file2.write(json.dumps(output))
```

1.2.2 numpredict.py

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

def cosine_similarity(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=[]

  # Loop over every item in the dataset
  for i in data:
    # print item
    for subitem in i:
      if subitem != 'F-Measure':
        vec2= i[subitem]

    # Add the distance and the index
    distancelist.append((cosine_similarity(vec1,vec2),i))

  # Sort by distance
  distancelist.sort()
```

```
31
32    return distancelist
33
34  def knnestimate(data,vec1,k=20):
35    # Get sorted distances
36    print 'For Web Science blog and k=20'
37    dlist=getdistances(data,vec1)
38    avg=0.0
39    # print dlist
40    # Take the average of the top k results
41    for i in range(k):
42      idx=dlist[i]
43      value = idx[0]
44      for item in idx[1]:
45      blogname= item
46      print blogname +'\t'+ str(value)
47
48  def main():
49    file1= open('bloglist','r')
50    data = json.load(file1)
51
52    for data1 in data:
53
54      for data2 in data1:
55        if data2 == 'Web Science and Digital Libraries Research Group':
56          vec1= data1[data2]
57          knnestimate(data,vec1)
58
59  main()
```

1.3 Output

1.3.1 F-Measure blog output

```
atria:~/wsprograms/a10/q1> python numpredict.py
k=1
INDIEohren.!     0.0284258547722
atria:~/wsprograms/a10/q1> clear
atria:~/wsprograms/a10/q1> python numpredict.py
For F-Measure blog and k=1
INDIEohren.!     0.0284258547722
atria:~/wsprograms/a10/q1> python numpredict.py
For F-Measure blog and k=2
INDIEohren.!     0.0284258547722
MR. BEAUTIFUL TRASH ART 0.035023925836
atria:~/wsprograms/a10/q1> python numpredict.py
For F-Measure blog and k=5
INDIEohren.!     0.0284258547722
MR. BEAUTIFUL TRASH ART 0.035023925836
MARISOL 0.0421254990846
How to be an artist and still pass for normal   0.0615408984138
ORGANMYTH        0.063728972963
atria:~/wsprograms/a10/q1> clear
atria:~/wsprograms/a10/q1> python numpredict.py
For F-Measure blog and k=10
INDIEohren.!     0.0284258547722
MR. BEAUTIFUL TRASH ART 0.035023925836
MARISOL 0.0421254990846
How to be an artist and still pass for normal   0.0615408984138
ORGANMYTH        0.063728972963
IoTube     :)    0.076906363689
theindiefriend   0.0798683749435
A H T A P O T    0.118655531931
adrianoblog      0.123239578098
Rod Shone        0.133212524679
atria:~/wsprograms/a10/q1> python numpredict.py
For F-Measure blog and k=20
INDIEohren.!     0.0284258547722
MR. BEAUTIFUL TRASH ART 0.035023925836
MARISOL 0.0421254990846
How to be an artist and still pass for normal   0.0615408984138
ORGANMYTH        0.063728972963
IoTube     :)    0.076906363689
theindiefriend   0.0798683749435
A H T A P O T    0.118655531931
adrianoblog      0.123239578098
Rod Shone        0.133212524679
What Am I Doing?         0.137021765128
Stonehill Sketchbook     0.137508143489
"DANCING IN CIRCLES"     0.146334368545
sweeping the kitchen     0.14683876125
F-Measure        0.151098106517
                         0.151098106517
FlowRadio Playlists (and Blog)  0.154577490878
Spinitron Blog   0.15802910721


```

```
                        0.161885545609
Boggle Me Thursday      0.174330461107
```

1.3.2 Web science blog output

```
atria:~/wsprograms/a10/q1> python numpredict.py
For Web Science blog and k=1
Rod Shone       0.0248283154805
atria:~/wsprograms/a10/q1> python numpredict.py
For Web Science blog and k=2
Rod Shone       0.0248283154805
adrianoblog     0.0601947926141
atria:~/wsprograms/a10/q1> python numpredict.py
For Web Science blog and k=5
Rod Shone       0.0248283154805
adrianoblog     0.0601947926141
INDIEohren.!    0.0609646307293
IoTube     :)   0.063726921167
MARISOL 0.0773923984209
atria:~/wsprograms/a10/q1> python numpredict.py
For Web Science blog and k=10
Rod Shone       0.0248283154805
adrianoblog     0.0601947926141
INDIEohren.!    0.0609646307293
IoTube     :)   0.063726921167
MARISOL 0.0773923984209
sweeping the kitchen    0.0837138528161
If You Give a Girl a Camera...  0.0874223257725


                        0.109439536637
theindiefriend  0.111459097808
"DANCING IN CIRCLES"    0.128160527359
atria:~/wsprograms/a10/q1> python numpredict.py
For Web Science blog and k=20
Rod Shone       0.0248283154805
adrianoblog     0.0601947926141
INDIEohren.!    0.0609646307293
IoTube     :)   0.063726921167
MARISOL 0.0773923984209
sweeping the kitchen    0.0837138528161
If You Give a Girl a Camera...  0.0874223257725


                        0.109439536637
theindiefriend  0.111459097808
"DANCING IN CIRCLES"    0.128160527359
How to be an artist and still pass for normal   0.135218576611
GLI Press       0.14290805507
Azul Valentina  0.14693001748
F-Measure       0.149696250775
                        0.149696250775
Boggle Me Thursday      0.150102315139
Lo importante es que estes t   bien     0.150707640184
MR. BEAUTIFUL TRASH ART 0.151538277269
What Am I Doing?        0.154256917386
Spinitron Blog  0.155694593625
```

2 Question 2:

2.1 Approach

Not Attempted

3 Question 3:

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.

3.1 Approach

- I have re-downloaded all the 1000 time maps from the previous assignment
- I have compared the mementos generated previously with the new mementos for each URI
- the generated graph shows that in general mementos have only increased and only in rare cases decreased by 1 or 2

3.2 Code Listing

3.2.1 memcount.py

```
#!/usr/local/bin/python3
import re
import sys
import urllib2
import json

mymementos = re.compile(r'rel.*?=.*?"memento".*?')#use regular expressions to find mementos
file3=open('abovezerocounts2.json','w')
file4=open('abovezerourls2.json','w')

def getTimeMap(url):
    mem_url = "http://mementoproxy.cs.odu.edu/aggr/timemap/link/1/" + url #plug in the url to a timemap
    try:
        response = urllib2.urlopen(mem_url)
        timemap = response.read()
    except urllib2.HTTPError:
        timemap = None
    return timemap

def countMementos(mem_url):
  time_map = getTimeMap(mem_url)
  if not time_map:# if no time maps
    count=0
  else:


    count=len(mymementos.findall(str(time_map)))#finds the count of all mementos per url
    if count>0:
      file3.write("%s\n"% count)
      file4.write("%s\n"% time_map)
    #print count
  return count

if __name__=="__main__":
  file1=open('output.json','r')# input a json file that contains 1000 urls
  file2=open('memcount2.json','w')
  #memcountlist=[]
  for line in file1.readlines():
    one_line = json.loads(line)# loads a json object
    link = one_line['link']
```

```
    counter=countMementos(link)# counter has count of the urls
    file2.write("%s"% counter)#outputs count of mementos of each url to a json file
    file2.write("\r\n")
  #for item in memcountlist:


file1.close()
file2.close()
```

3.3 Generating graph

3.3.1 Rcommands.R

```
d = read.table('dif2.json',col.name=c("mementos"))
plot(d$mementos,xlab="Number of URI's",ylab="Difference between Old and New Mementos",
   xlim=c(0,1000),ylim=c(-2,50),type="l")
```

3.4 Output Files

3.4.1 Output graph

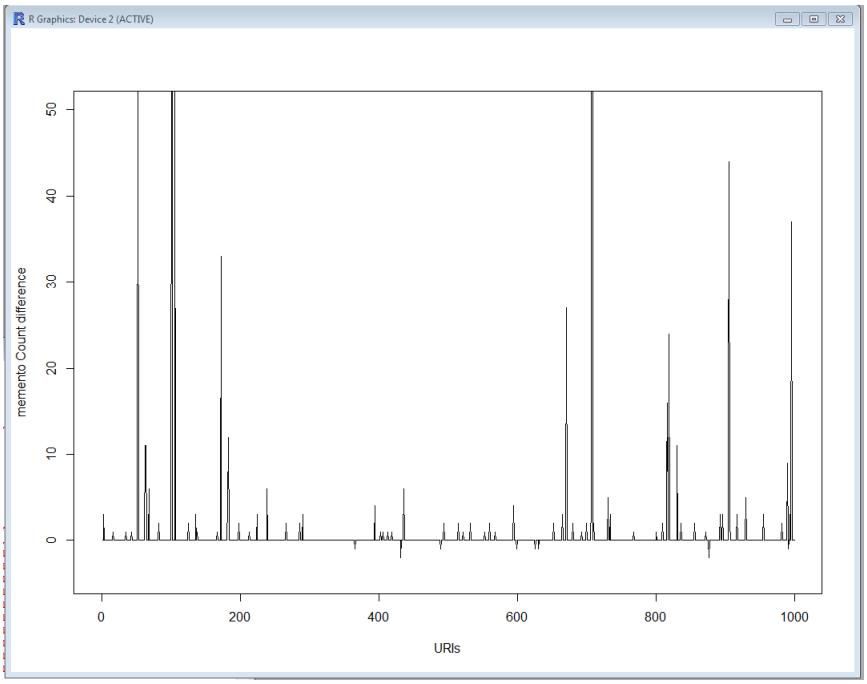

Figure 1: Memento difference

4 Question 4:

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.

4.1 Approach

- I have extracted raw and processed URIs again which returns status 200.
- I have found the size of old raw data, new raw data, old processed data, new processed data using size.py.
- I have found the difference in sizes of old raw data and new raw data and generated a graph.
- Similarly, I have found the difference in sizes of old processed data and new processed data and generated a graph .
- I wrote a program statuscode.py to find the status of the existing URIs.
- I have found that only 784 out of entire 1000 URIs had exited with status code 200.
- I had found the top three maximum distances of the old and new processed URIs using vim -d command.
- The comparision is shown in three seperate screenshots

4.2 Code Listing

4.2.1 For extracting raw and processed URIs (extract.py)

```
import re
import os
import json

if __name__=="__main__":
  f2name='raw.txt'
  f3name='processed.txt'
  count=0
  count1=0
  file1=open('links.json','r')# file which contains 1000 uris

  for line in file1.readlines():
    count=count+1
    newfile=str(count)+f2name #concatenates counter value to a string
    one_line = json.loads(line)
    link = one_line['link']

```

```
    cmd="curl -s -L "+ link+" >./rawurls/"+ newfile # shell script to print raw html content of each uri
    os.system(cmd)
  for line in file1.readlines():
    count1=count1+1
    newfile1=str(count1)+f3name #concatenates counter value to a string
    one_line1 = json.loads(line)
    link1 = one_line1['link']

    cmd1="lynx -dump -force_html "+ link1+" >./processedurls/"+ newfile1 # shell script to print processed html content of each uri
    os.system(cmd1)
```

4.2.2 To find the size of the size of either raw or processed files(size.py)

```
import os


path= "/home/mkompal/wsprograms/a10/q4/rawurls/"

file1 = open('newraw','w')
directory = os.listdir(path)

for file in directory:

  final = path + file
  print final
  size= os.path.getsize(final)
  file1.write(str(size)+'\n')
```

4.2.3 To find the number of URIs exiting with status 200 (statuscode.py)

```

import requests
import json

file2 = open("links.json","r")
file= open("codecount.txt","w")


count=0

for line in file2:

  short=line.strip()
  one_line = json.loads(line)
  link = one_line['link']

  try:
    info=requests.get(link)
    if info.status_code==200:

      count=count+1
      print count
  except Exception, e:
    print e
    continue
file.write(count)
```

```
file.close()
```

4.2.4 Generating graph

```
#--For Raw Files Size difference-------
d = read.table('rawdiff.txt' ,col.name=c("sizes"))
plot(d$sizes,xlab="URIs",ylab="Size difference in raw files",xlim=c(0,1000),ylim=c
    (-700000,700000),type="l")
#--For Processed Files Size difference----
d = read.table('processdiff.txt' ,col.name=c("sizes"))
plot(d$sizes,xlab="URIs",ylab="Size difference in processed files",xlim=c(0,1000),ylim
    =c(-200000,200000),type="l")
```

4.3 Output

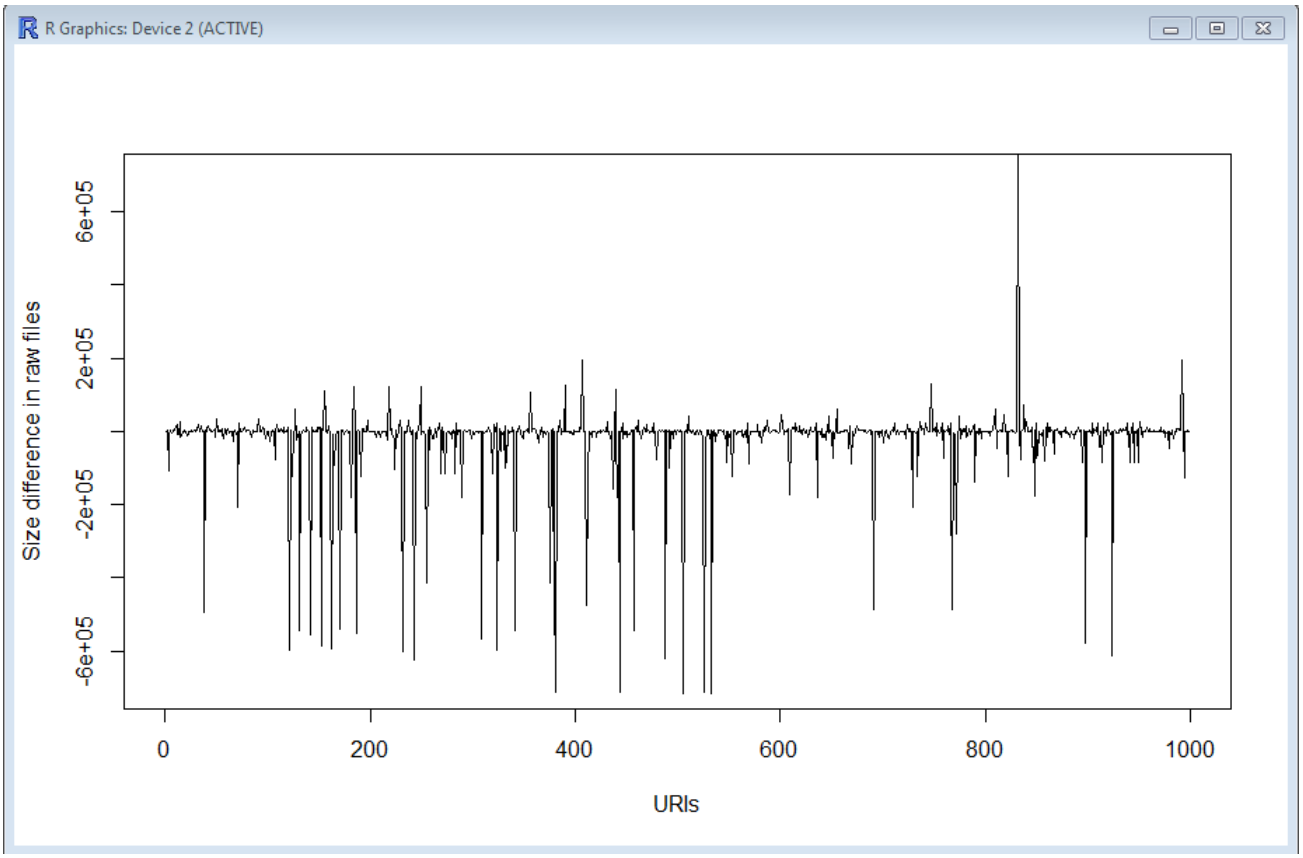

Figure 2: Raw Difference

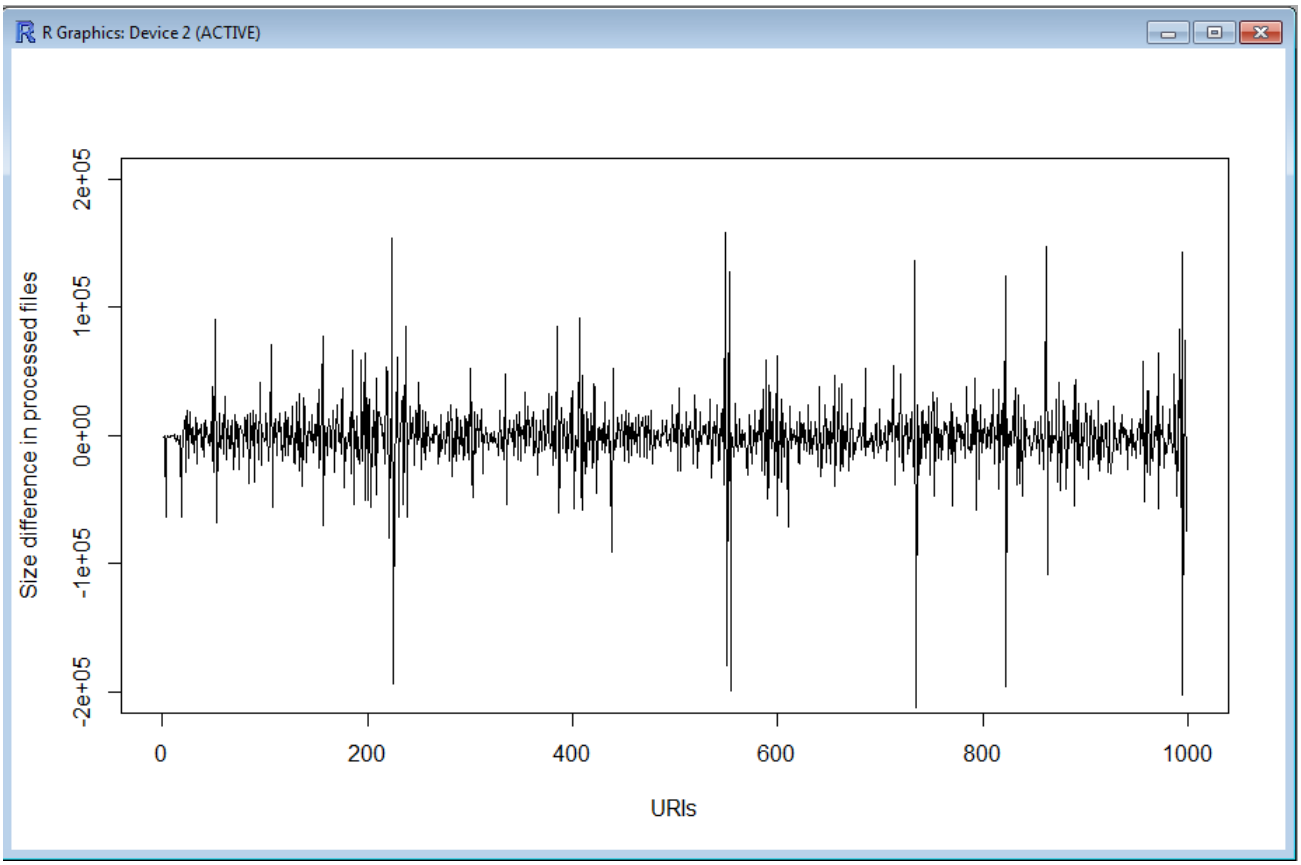

Figure 3: Processed Difference

Figure 4: Comparing Processed data of maximum size difference

Figure 5: Comparing Processed data of second maximum size difference

Figure 6: Comparing Processed data of second maximum size difference

References

[1] Cosine distance. http://stackoverflow.com/questions/18424228/cosine-similarity-between-2-number-lists.

[2] Status Checker. http://cairographics.org/pycairo.

[3] numpredict.py. https://github.com/manojchandrak/Programming-Collective-Intelligence/blob/master/chapter8/numpredict.py.

[]