Bryan Carey
Assignment 10
Due May 1, 2017 11:59PM

For the totality of this assignment, I collaborated with my fellow classmates Michelle Graham & Breon Day.

1. Using the data from A8:

- Consider each row in the blog-term matrix as a 1000 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 1000 dimensions.

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

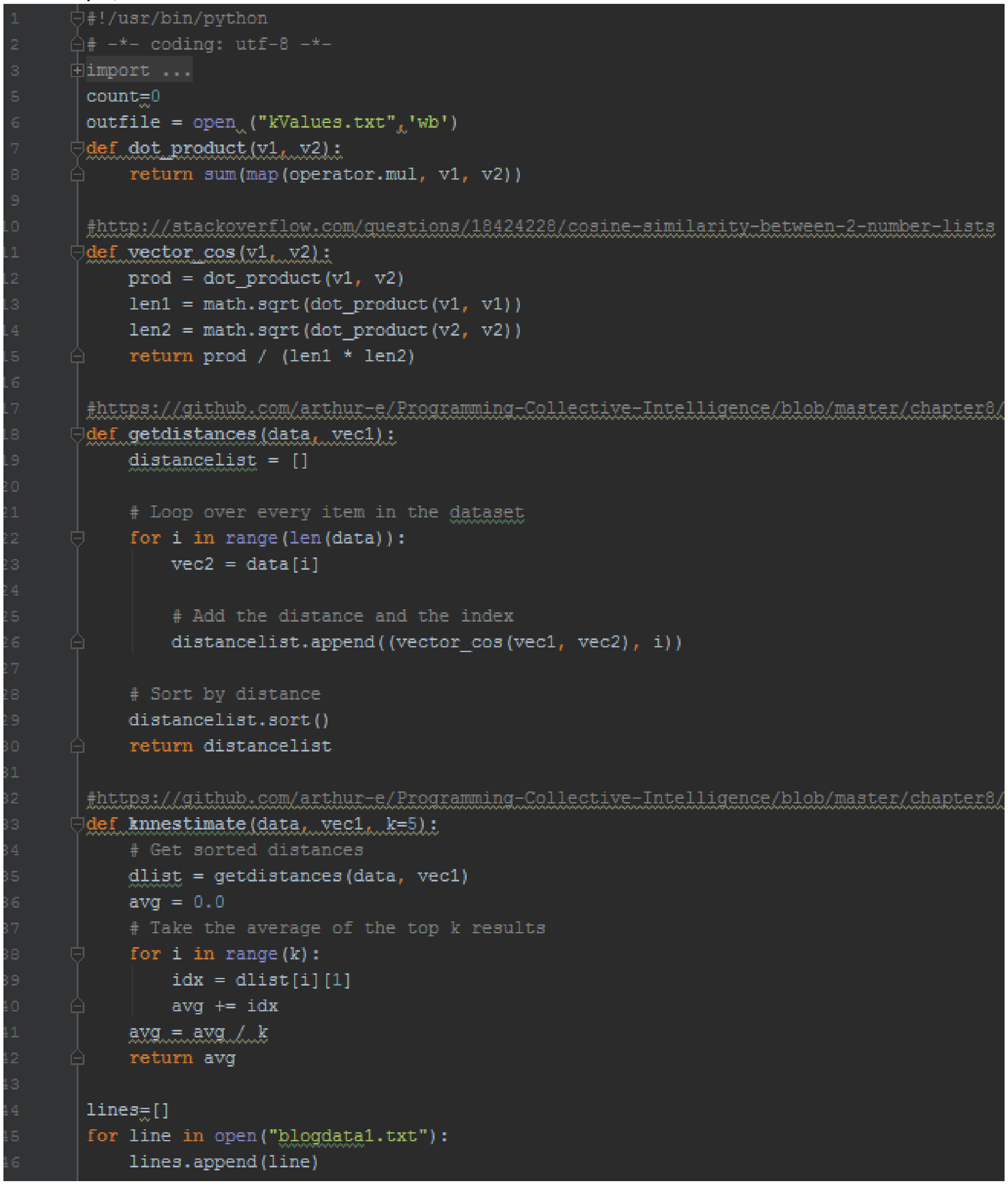
http://f-measure.blogspot.com/

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

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

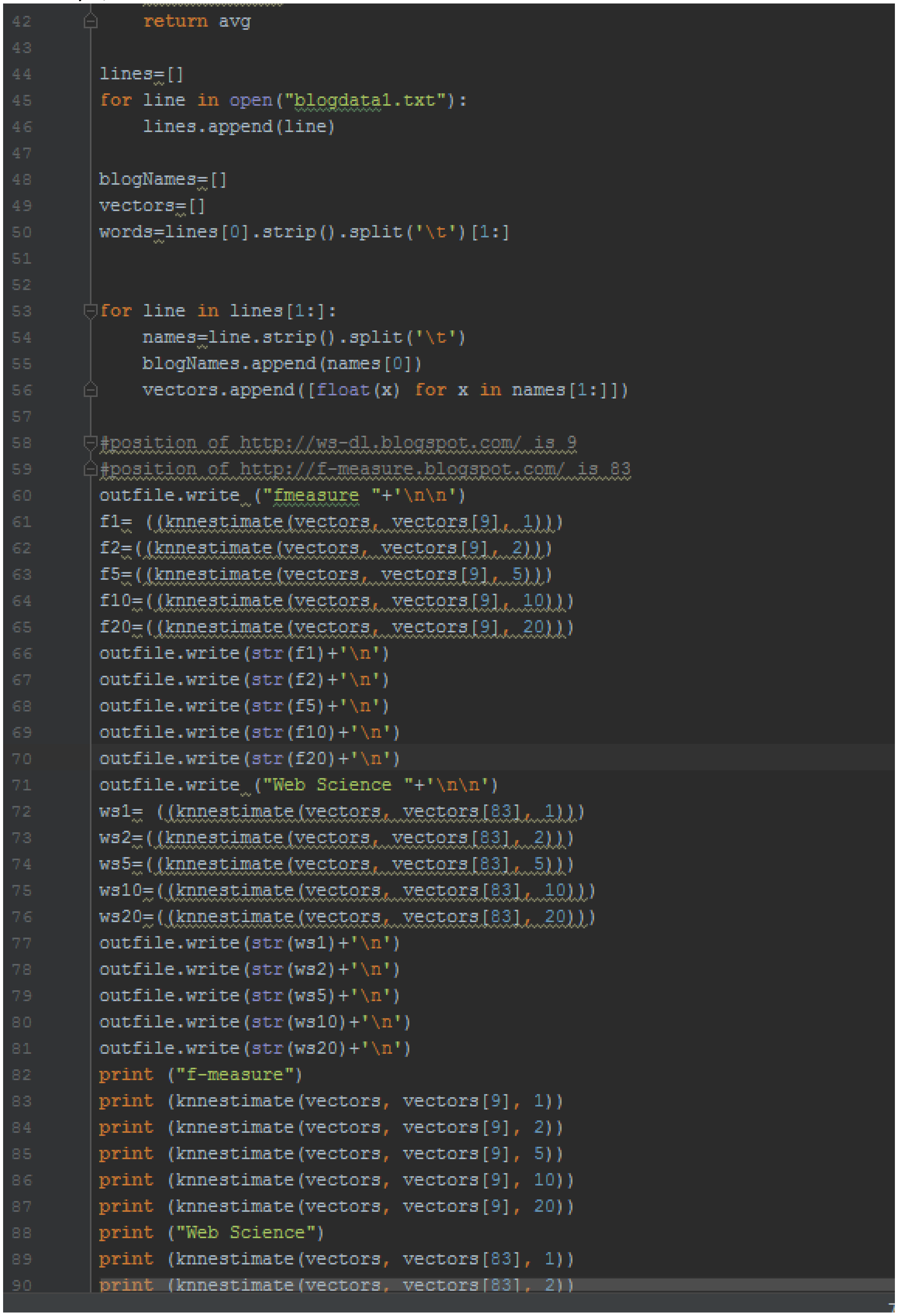
knnEsimate.py was constructed from pieces of code found in a few different locations, namely http://stackoverflow.com/questions/18424228/cosine-similarity-between-2-number-lists & https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter8/numpredict.py . The code is as follows-

Bryan Carey
Assignment 10
Due May 1, 2017 11:59PM



```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import ...
count=0
outfile = open ("kValues.txt",'wb')
def dot_product(v1, v2):
    return sum(map(operator.mul, v1, v2))

#http://stackoverflow.com/questions/18424228/cosine-similarity-between-2-number-lists
def vector_cos(v1, v2):
    prod = dot_product(v1, v2)
    len1 = math.sqrt(dot_product(v1, v1))
    len2 = math.sqrt(dot_product(v2, v2))
    return prod / (len1 * len2)

#https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter8/
def getdistances(data, vec1):
    distancelist = []

    # Loop over every item in the dataset
    for i in range(len(data)):
        vec2 = data[i]

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

    # Sort by distance
    distancelist.sort()
    return distancelist

#https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter8/
def knnestimate(data, vec1, k=5):
    # Get sorted distances
    dlist = getdistances(data, vec1)
    avg = 0.0
    # Take the average of the top k results
    for i in range(k):
        idx = dlist[i][1]
        avg += idx
    avg = avg / k
    return avg

lines=[]
for line in open("blogdata1.txt"):
    lines.append(line)
```

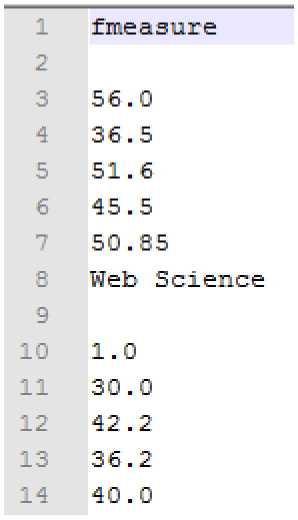
Bryan Carey
Assignment 10
Due May 1, 2017 11:59PM



```
    return avg

lines=[]
for line in open("blogdata1.txt"):
    lines.append(line)

blogNames=[]
vectors=[]
words=lines[0].strip().split('\t')[1:]


for line in lines[1:]:
    names=line.strip().split('\t')
    blogNames.append(names[0])
    vectors.append([float(x) for x in names[1:]])

#position of http://ws-dl.blogspot.com/ is 9
#position of http://f-measure.blogspot.com/ is 83
outfile.write ("fmeasure "+'\n\n')
f1= ((knnestimate(vectors, vectors[9], 1)))
f2=((knnestimate(vectors, vectors[9], 2)))
f5=((knnestimate(vectors, vectors[9], 5)))
f10=((knnestimate(vectors, vectors[9], 10)))
f20=((knnestimate(vectors, vectors[9], 20)))
outfile.write(str(f1)+'\n')
outfile.write(str(f2)+'\n')
outfile.write(str(f5)+'\n')
outfile.write(str(f10)+'\n')
outfile.write(str(f20)+'\n')
outfile.write ("Web Science "+'\n\n')
ws1= ((knnestimate(vectors, vectors[83], 1)))
ws2=((knnestimate(vectors, vectors[83], 2)))
ws5=((knnestimate(vectors, vectors[83], 5)))
ws10=((knnestimate(vectors, vectors[83], 10)))
ws20=((knnestimate(vectors, vectors[83], 20)))
outfile.write(str(ws1)+'\n')
outfile.write(str(ws2)+'\n')
outfile.write(str(ws5)+'\n')
outfile.write(str(ws10)+'\n')
outfile.write(str(ws20)+'\n')
print ("f-measure")
print (knnestimate(vectors, vectors[9], 1))
print (knnestimate(vectors, vectors[9], 2))
print (knnestimate(vectors, vectors[9], 5))
print (knnestimate(vectors, vectors[9], 10))
print (knnestimate(vectors, vectors[9], 20))
print ("Web Science")
print (knnestimate(vectors, vectors[83], 1))
print (knnestimate(vectors, vectors[83], 2))
```

Bryan Carey
Assignment 10
Due May 1, 2017 11:59PM

The program once run with my previous data, blogdata1.txt generated the following output and was store in kValues.txt.



```
fmeasure

56.0
36.5
51.6
45.5
50.85
Web Science

1.0
30.0
42.2
36.2
40.0
```

2. Rerun A9, Q2 but this time using LIBSVM. If you have n categories, you'll have to run it n times. For example, if you're classifying music and have the categories: metal, electronic, ambient, folk, hip-hop, pop you'll have to classify things as:

Use the 1000 term vectors describing each blog as the features, and

your mannally assigned classifications as the true values. Use

10-fold cross-validation (as per slide 46, which shows 4-fold

cross-validation) and report the percentage correct for

each of your categories.

Getting ahold of the library was difficult, as LIBSVM no longer worked for Python until Breon was somehow able to download it on his personal computer at home. From there he uploaded it to GitHub and instructed me download his repository. Upon doing so I was able to copy the LIBSVM library in its entirety to my Q2 folder. This became my new workspace.

A .txt file, stopWordList contains a list of words that are insignificant. This is passed into the program as a parameter and prevents excessive common words from being taken into account.

I ran A10GenerateFeedVectorQ2.py with my blogdata1.txt as a parameter and the output of the program was stored in blogdataQ2.txt. The matrix is too large to display, it can be found in my repository.

Bryan Carey
Assignment 10
Due May 1, 2017 11:59PM

========The questions below is for 3 points extra credit===========

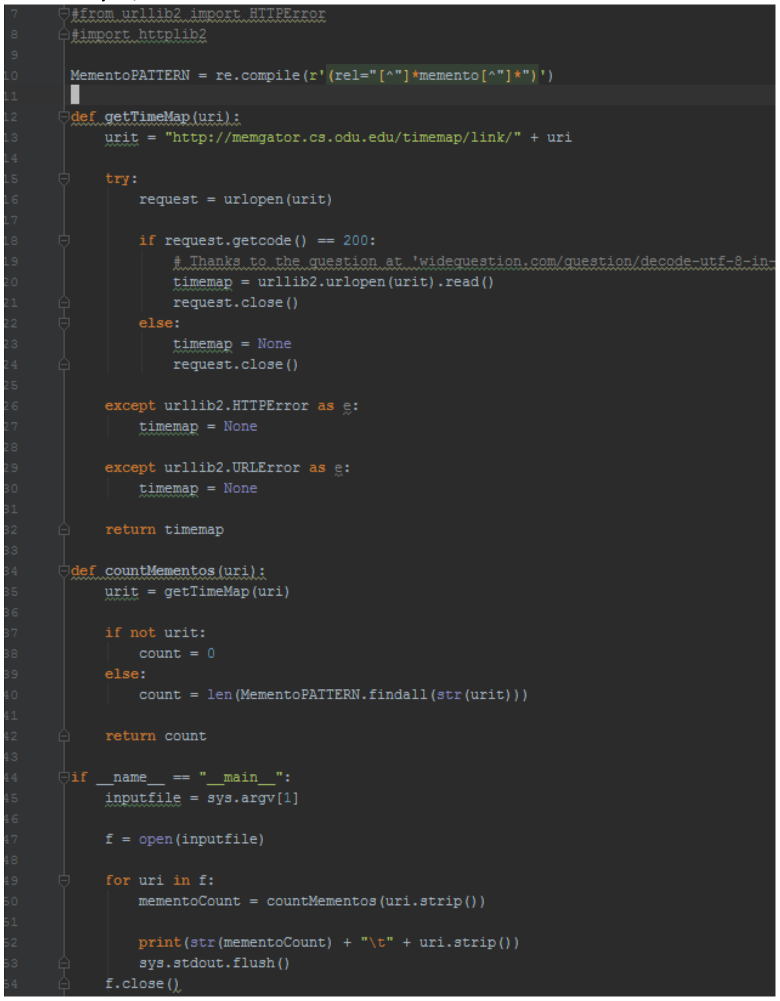
==

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.

For this question I received permission from Ross Reelachart, to use his old data from assignment two as well as his code to handle the data. I downloaded getMemento.py and the files containing the uri's with the mementos from earlier in the semester- as I had not completed that portion of Assignment2 so I have no time maps for my 1000 uri's to compare them too. Per the instructions I received and following the Ross's report on how to use his getMemento.py.

Bryan Carey
Assignment 10
Due May 1, 2017 11:59PM



```python
#from urllib2 import HTTPError
#import httplib2

MementoPATTERN = re.compile(r'(rel="[^"]*memento[^"]*")')

def getTimeMap(uri):
    urit = "http://memgator.cs.odu.edu/timemap/link/" + uri

    try:
        request = urlopen(urit)

        if request.getcode() == 200:
            # Thanks to the question at 'widequestion.com/question/decode-utf-8-in-
            timemap = urllib2.urlopen(urit).read()
            request.close()
        else:
            timemap = None
            request.close()

    except urllib2.HTTPError as e:
        timemap = None

    except urllib2.URLError as e:
        timemap = None

    return timemap

def countMementos(uri):
    urit = getTimeMap(uri)

    if not urit:
        count = 0
    else:
        count = len(MementoPATTERN.findall(str(urit)))

    return count

if __name__ == "__main__":
    inputfile = sys.argv[1]

    f = open(inputfile)

    for uri in f:
        mementoCount = countMementos(uri.strip())

        print(str(mementoCount) + "\t" + uri.strip())
        sys.stdout.flush()
    f.close()
```

1) I took Ross's 1000Links.txt and split it into 5 separate files for the sake of workability(easier to process several small files instead of one large one). The files were as follows (Linksaa…Linksae). These files were generated with the command **'split-| 200 1000Links.txt'**

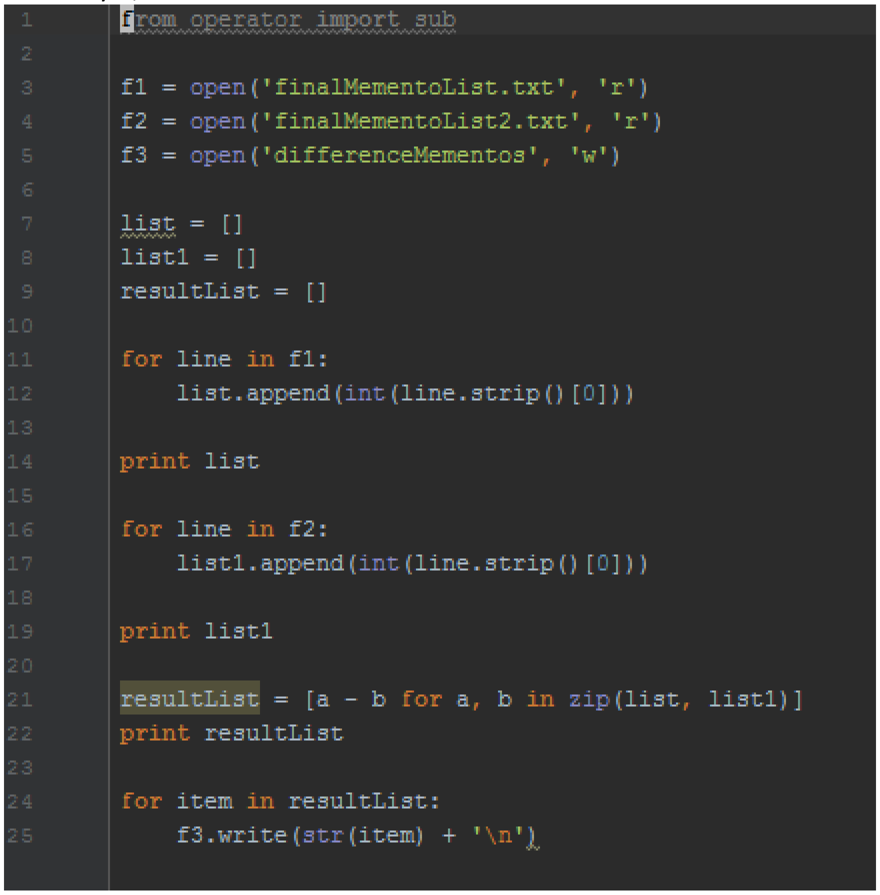
2) I ran each of the generated files through getMemento.py with putty. This calculated the memento and spat out a .txt file containing the new memento as well as the uri.

3) The resulting .txt files were then concatenated together using the putty command '**cat Linksaa.txt Linksab.txt….Linksae.txt > finalMementoList2.txt**'

4) This file, finalMementoList2.txt was compared to the previous results of Ross's Assignment2, the original finalMementoList.txt. Adopting Ross's method, getDifference.py was created. I used this to find the difference in between the mementos from finalMementoList.txt and finalMementoList2.txt.

5) The result of getDifference.py was stored in differenceMemento.txt.

Bryan Carey
Assignment 10
Due May 1, 2017 11:59PM

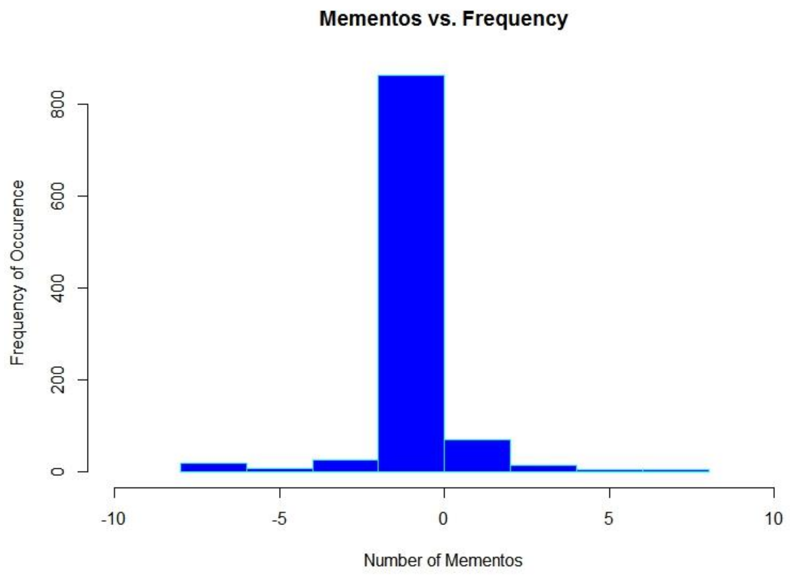


```
from operator import sub

f1 = open('finalMementoList.txt', 'r')
f2 = open('finalMementoList2.txt', 'r')
f3 = open('differenceMementos', 'w')

list = []
list1 = []
resultList = []

for line in f1:
    list.append(int(line.strip()[0]))

print list

for line in f2:
    list1.append(int(line.strip()[0]))

print list1

resultList = [a - b for a, b in zip(list, list1)]
print resultList

for item in resultList:
    f3.write(str(item) + '\n')
```

6) This data used to construct the histogram in R below. Curiously, some of the mementos or the sites had gone down, which I thought was not possible as mementos as they reflect the number of times that the site was visited. The R Script:

```
histData <- read.table("C:/CS432/A10/Q3/differenceTimeMaps")
hist(histData$V1, xlab='Number of Mementos', xlim = range(c(-10,10)),
     ylab='Frequency of Occurence', main='Mementos vs. Frequency',
     ty =4, col ='blue', border = 'cyan')
```

Bryan Carey
Assignment 10
Due May 1, 2017 11:59PM



===

========The questions below is for 3 points extra credit===========

===

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.