Assignment #7

Saturday, March 31TH, 2018

RANDY DO

Contents

[Date]

Problem 1

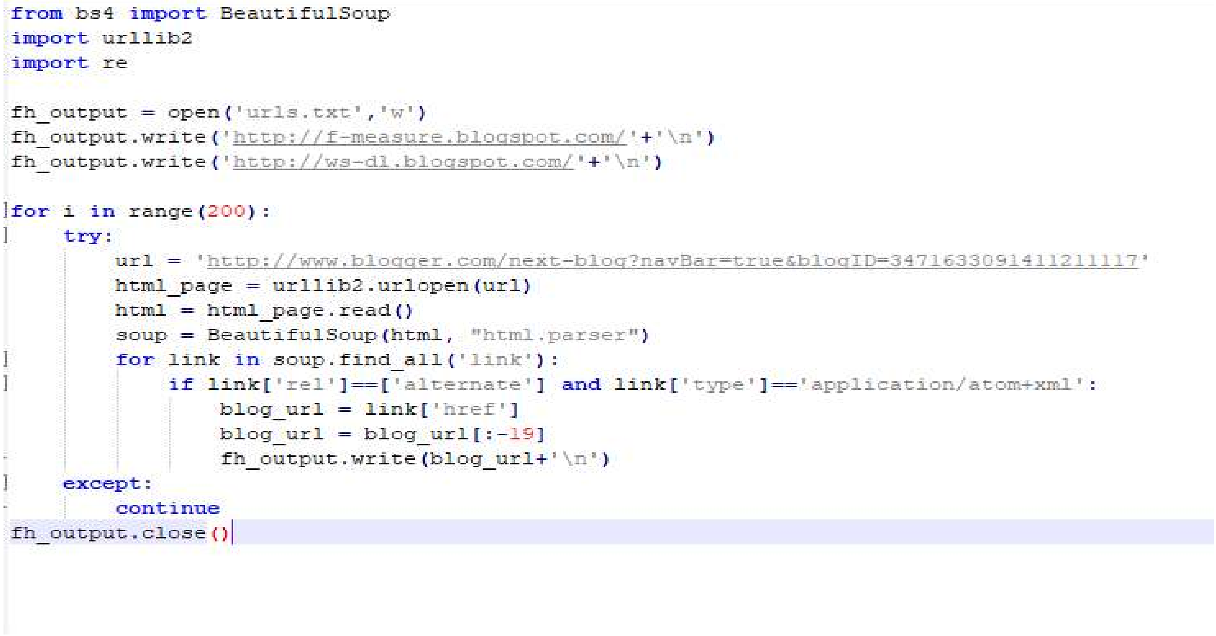
1. Create a blog-term matrix. Start by grabbing 100 blogs; include:

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

and grab 98 more as per the method shown in class. Note that this method randomly chooses blogs and each student will separately do this process, so it is unlikely that these 98 blogs will be shared among students. In other words, no sharing of blog data. Upload to github your code for grabbing the blogs and provide a list of blog URIs, both in the report and in github.

Use the blog title as the identifier for each blog (and row of the matrix). Use the terms from every item/title (RSS) or entry/title (Atom) for the columns of the matrix. The values are the frequency of occurrence. Essentially you are replicating the format of the "blogdata.txt" file included with the PCI book code. Limit the number of terms to the most "popular" (i.e., frequent) 1000 terms, this is *after* the criteria on p. 32 (slide 8) has been satisfied. Remember that blogs are paginated.

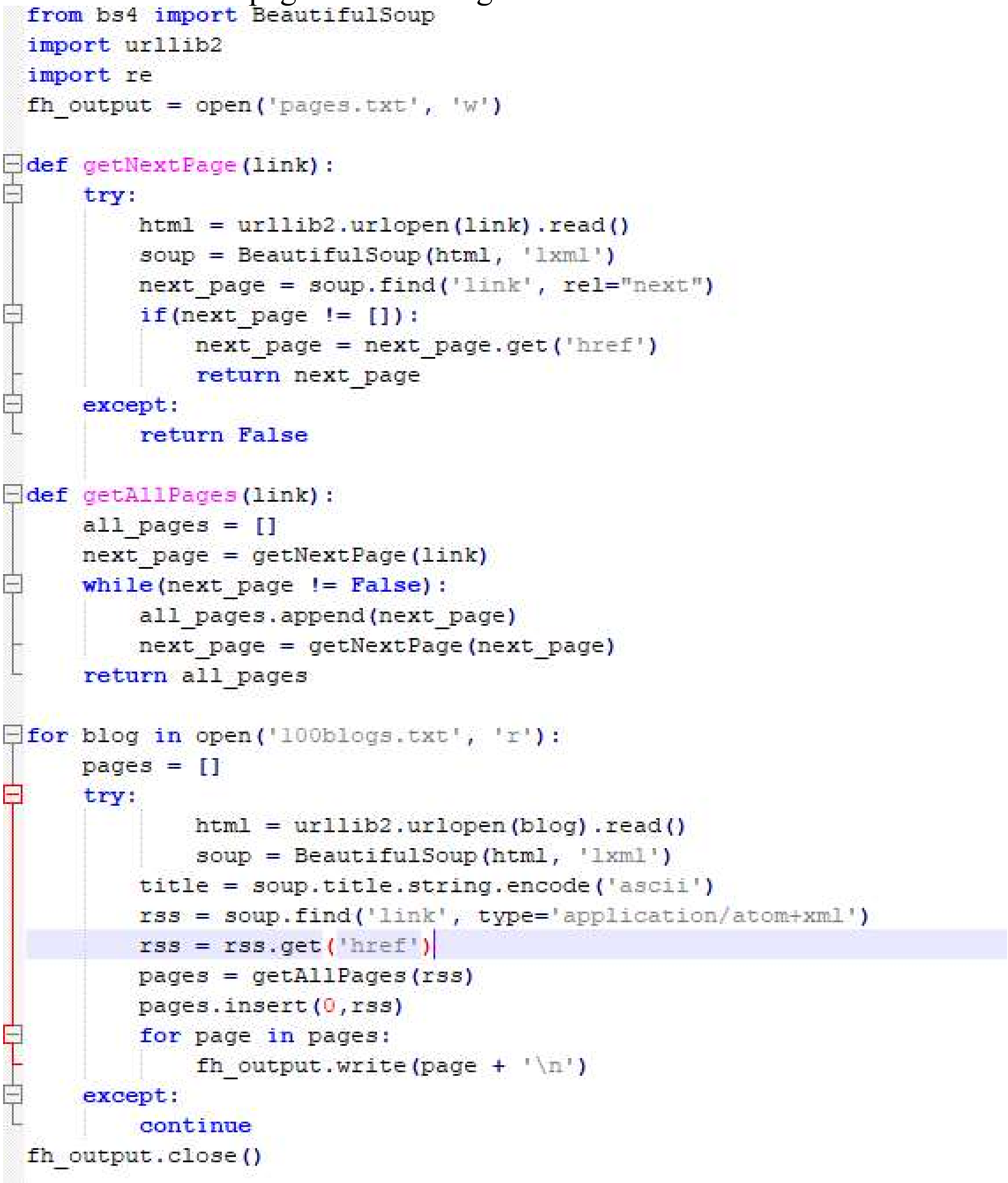
I will be using BeautifulSoup to obtain the urls. The program I will be using is called geturl.py which will out a text file called urls.py



```
from bs4 import BeautifulSoup
import urllib2
import re

fh_output = open('urls.txt','w')
fh_output.write('http://f-measure.blogspot.com/'+'\n')
fh_output.write('http://ws-dl.blogspot.com/'+'\n')

for i in range(200):
    try:
        url = 'http://www.blogger.com/next-blog?navBar=true&blogID=3471633091411211117'
        html_page = urllib2.urlopen(url)
        html = html_page.read()
        soup = BeautifulSoup(html, "html.parser")
        for link in soup.find_all('link'):
            if link['rel']==['alternate'] and link['type']=='application/atom+xml':
                blog_url = link['href']
                blog_url = blog_url[:-19]
                fh_output.write(blog_url+'\n')
    except:
        continue
fh_output.close()
```

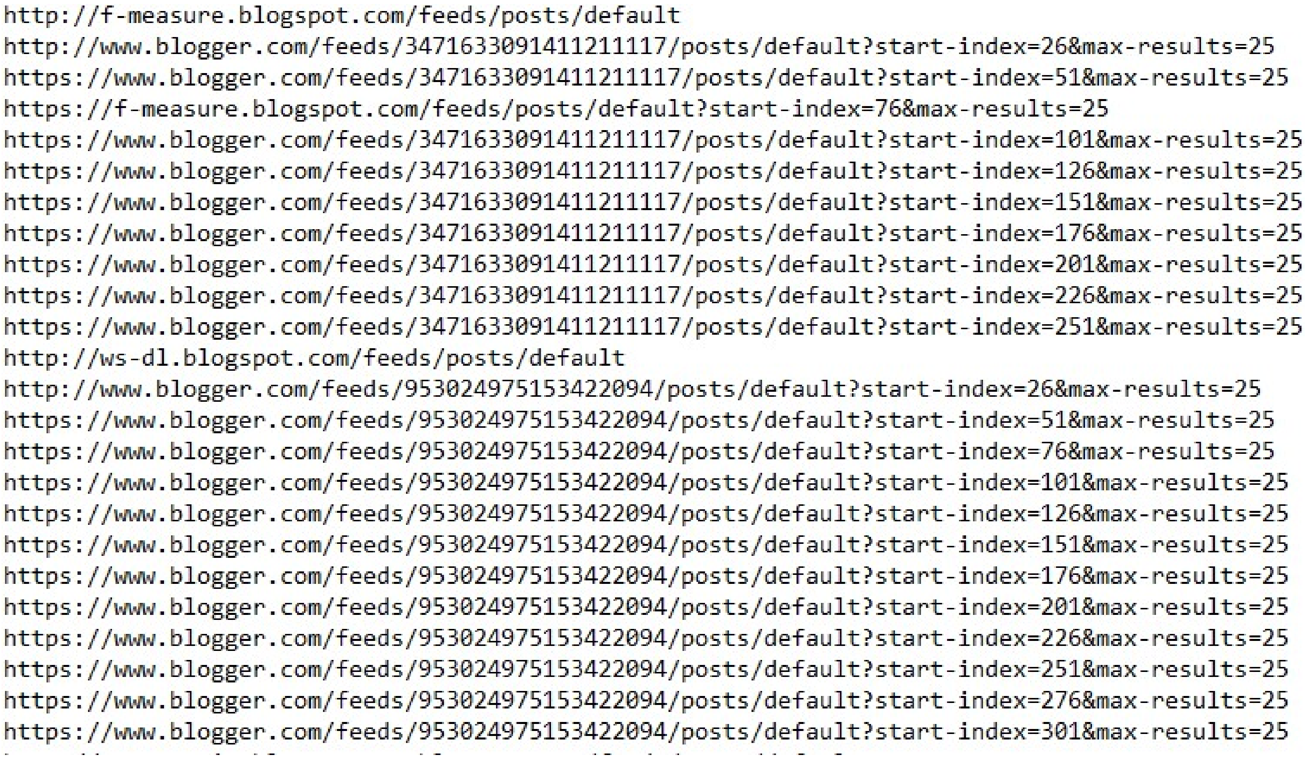
I got over 100 urls in my output. So, I reduced it down from 170 to 100 and named it 100blog.txt

Next is to find the pages in each blog website.



```
from bs4 import BeautifulSoup
import urllib2
import re
fh_output = open('pages.txt', 'w')

def getNextPage(link):
    try:
        html = urllib2.urlopen(link).read()
        soup = BeautifulSoup(html, 'lxml')
        next_page = soup.find('link', rel="next")
        if(next_page != []):
            next_page = next_page.get('href')
            return next_page
    except:
        return False

def getAllPages(link):
    all_pages = []
    next_page = getNextPage(link)
    while(next_page != False):
        all_pages.append(next_page)
        next_page = getNextPage(next_page)
    return all_pages

for blog in open('100blogs.txt', 'r'):
    pages = []
    try:
            html = urllib2.urlopen(blog).read()
            soup = BeautifulSoup(html, 'lxml')
        title = soup.title.string.encode('ascii')
        rss = soup.find('link', type='application/atom+xml')
        rss = rss.get('href')
        pages = getAllPages(rss)
        pages.insert(0,rss)
        for page in pages:
            fh_output.write(page + '\n')
    except:
        continue
fh_output.close()
```

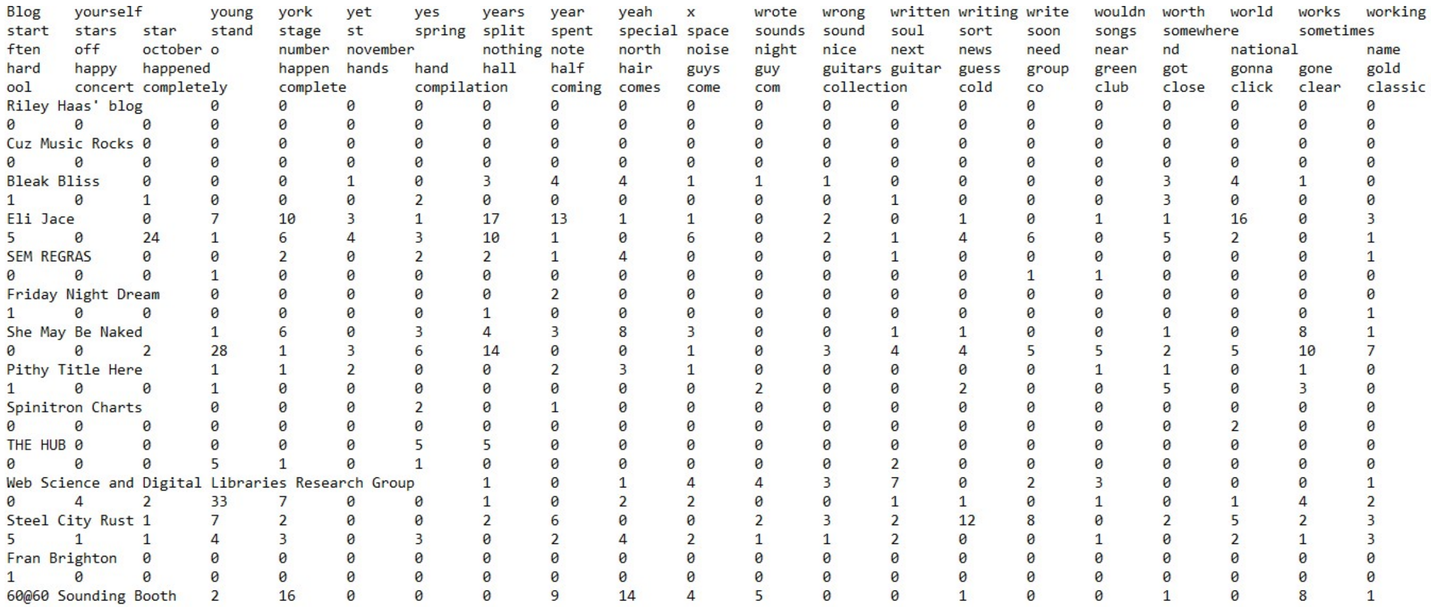
The program finds the pages and output it in the text file pages.txt. The output has over 1000+ lines. Here is an example of what it looks like.



```
http://f-measure.blogspot.com/feeds/posts/default
http://www.blogger.com/feeds/3471633091411211117/posts/default?start-index=26&max-results=25
https://www.blogger.com/feeds/3471633091411211117/posts/default?start-index=51&max-results=25
https://f-measure.blogspot.com/feeds/posts/default?start-index=76&max-results=25
https://www.blogger.com/feeds/3471633091411211117/posts/default?start-index=101&max-results=25
https://www.blogger.com/feeds/3471633091411211117/posts/default?start-index=126&max-results=25
https://www.blogger.com/feeds/3471633091411211117/posts/default?start-index=151&max-results=25
https://www.blogger.com/feeds/3471633091411211117/posts/default?start-index=176&max-results=25
https://www.blogger.com/feeds/3471633091411211117/posts/default?start-index=201&max-results=25
https://www.blogger.com/feeds/3471633091411211117/posts/default?start-index=226&max-results=25
https://www.blogger.com/feeds/3471633091411211117/posts/default?start-index=251&max-results=25
http://ws-dl.blogspot.com/feeds/posts/default
http://www.blogger.com/feeds/953024975153422094/posts/default?start-index=26&max-results=25
https://www.blogger.com/feeds/953024975153422094/posts/default?start-index=51&max-results=25
https://www.blogger.com/feeds/953024975153422094/posts/default?start-index=76&max-results=25
https://www.blogger.com/feeds/953024975153422094/posts/default?start-index=101&max-results=25
https://www.blogger.com/feeds/953024975153422094/posts/default?start-index=126&max-results=25
https://www.blogger.com/feeds/953024975153422094/posts/default?start-index=151&max-results=25
https://www.blogger.com/feeds/953024975153422094/posts/default?start-index=176&max-results=25
https://www.blogger.com/feeds/953024975153422094/posts/default?start-index=201&max-results=25
https://www.blogger.com/feeds/953024975153422094/posts/default?start-index=226&max-results=25
https://www.blogger.com/feeds/953024975153422094/posts/default?start-index=251&max-results=25
https://www.blogger.com/feeds/953024975153422094/posts/default?start-index=276&max-results=25
https://www.blogger.com/feeds/953024975153422094/posts/default?start-index=301&max-results=25
```

Lastly, we used the program generatefeedvector.py from https://github.com/arthur-e/Programming-Collective-Intelligence/tree/master/chapter3

The input is pages.txt and output is blogdata.txt
Here is an example of what the output looks like



```
Blog    yourself        young   york    yet     yes     years   year    yeah    x       wrote   wrong   written writing write   wouldn  worth   world   works   working
start   stars   star    stand   stage   st      spring  split   spent   special space   sounds  sound   soul    sort    soon    songs   somewhere       sometimes
ften    off     october o       number  november        nothing note    north   noise   night   nice    next    news    need    near    nd      national        name
hard    happy   happened        happen  hands   hand    hall    half    hair    guys    guy     guitars guitar  guess   group   green   got     gonna   gone    gold
ool     concert completely      complete        compilation     coming  comes   come    com     collection      cold    co      club    close   click   clear   classic
Riley Haas' blog        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
Cuz Music Rocks 0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
Bleak Bliss     0       0       0       1       0       3       4       4       1       1       1       0       0       0       0       3       4       1       0
1       0       1       0       0       0       2       0       0       0       0       0       0       1       0       0       0       3       0       0       0
Eli Jace        0       7       10      3       1       17      13      1       1       0       2       0       1       0       1       1       16      0       3
5       0       24      1       6       4       3       10      1       0       6       0       2       1       4       6       0       5       2       0       1
SEM REGRAS      0       0       2       0       2       2       1       4       0       0       0       1       0       0       0       0       0       0       1
0       0       0       1       0       0       0       0       0       0       0       0       0       0       0       1       1       0       0       0       0
Friday Night Dream      0       0       0       0       0       2       0       0       0       0       0       0       0       0       0       0       0       0
1       0       0       0       0       0       0       1       0       0       0       0       0       0       0       0       0       0       0       0       1
She May Be Naked        1       6       0       3       4       3       8       3       0       0       1       1       0       0       1       0       8       1
0       0       2       28      1       3       6       14      0       0       1       0       3       4       4       5       5       2       5       10      7
Pithy Title Here        1       1       2       0       0       2       3       1       0       0       0       0       0       1       1       0       1       0
1       0       0       1       0       0       0       0       0       0       0       2       0       0       2       0       0       5       0       3       0
Spinitron Charts        0       0       0       2       0       1       0       0       0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       2       0       0
THE HUB 0       0       0       0       0       5       5       0       0       0       0       0       0       0       0       0       0       0       0       0
0       0       0       5       1       0       1       0       0       0       0       0       0       2       0       0       0       0       0       0       0
Web Science and Digital Libraries Research Group        1       0       1       4       4       3       7       0       2       3       0       0       0       1
0       4       2       33      7       0       0       1       0       2       2       0       0       1       1       0       1       0       1       4       2
Steel City Rust 1       7       2       0       0       2       6       0       0       2       3       2       12      8       0       2       5       2       3
5       1       1       4       3       0       3       0       2       4       2       1       1       2       0       0       1       0       2       1       3
Fran Brighton   0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
1       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
60@60 Sounding Booth    2       16      0       0       0       9       14      4       5       0       0       1       0       0       1       0       8       1
```

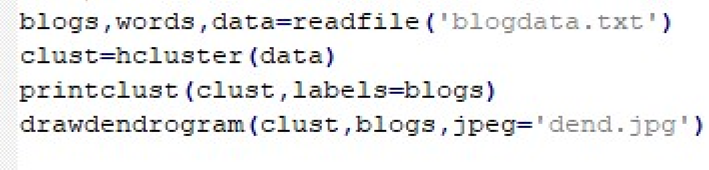
Problem 2

2. Create an ASCII and JPEG dendrogram that clusters (i.e., HAC) the most similar blogs (see slides 13 & 14). Include the JPEG in your report and upload the ascii file to github (it will be too unwieldy for inclusion in the report).

I used the program clusters.py from https://github.com/arthur-e/Programming-Collective-Intelligence/tree/master/chapter3

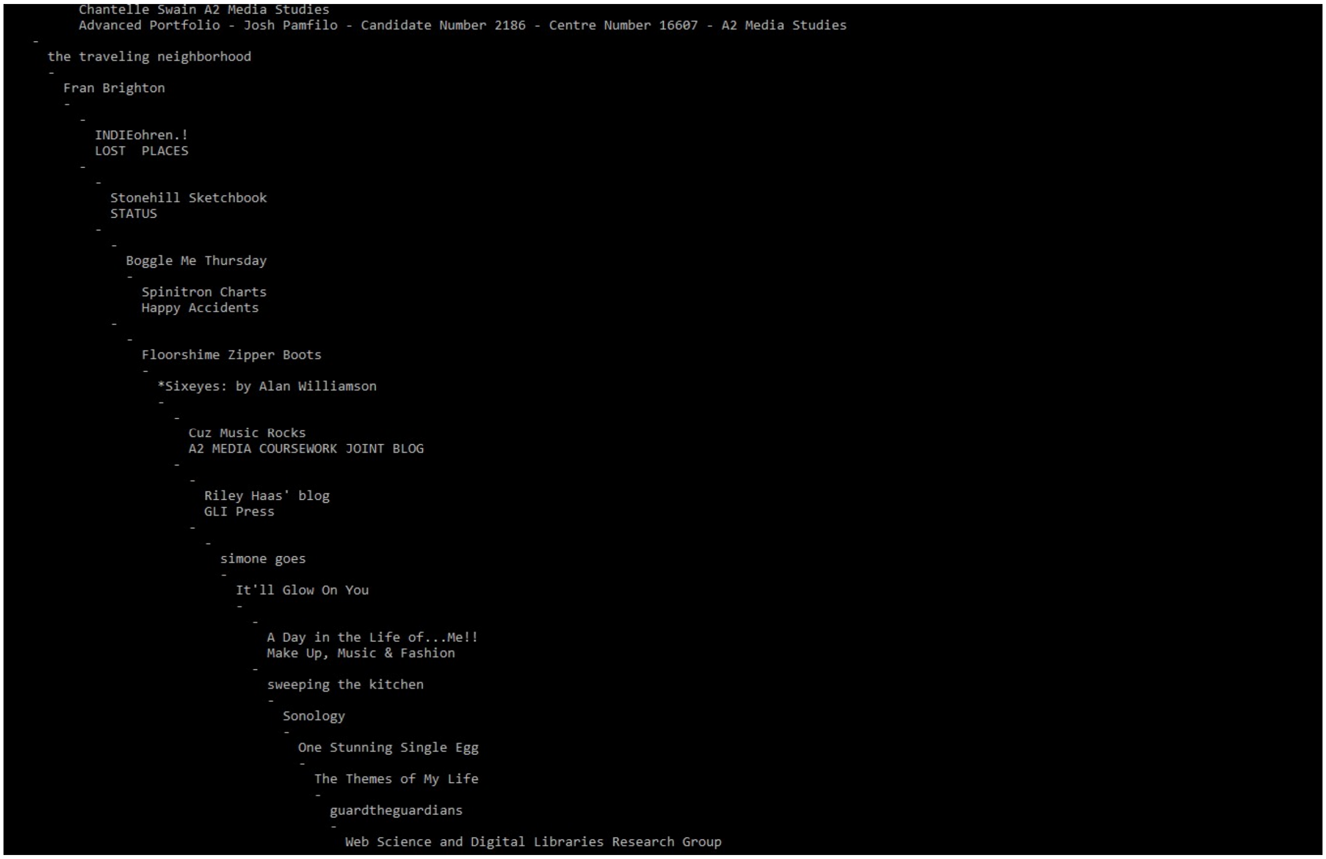
I also did a little modify on cluster.py

I added this



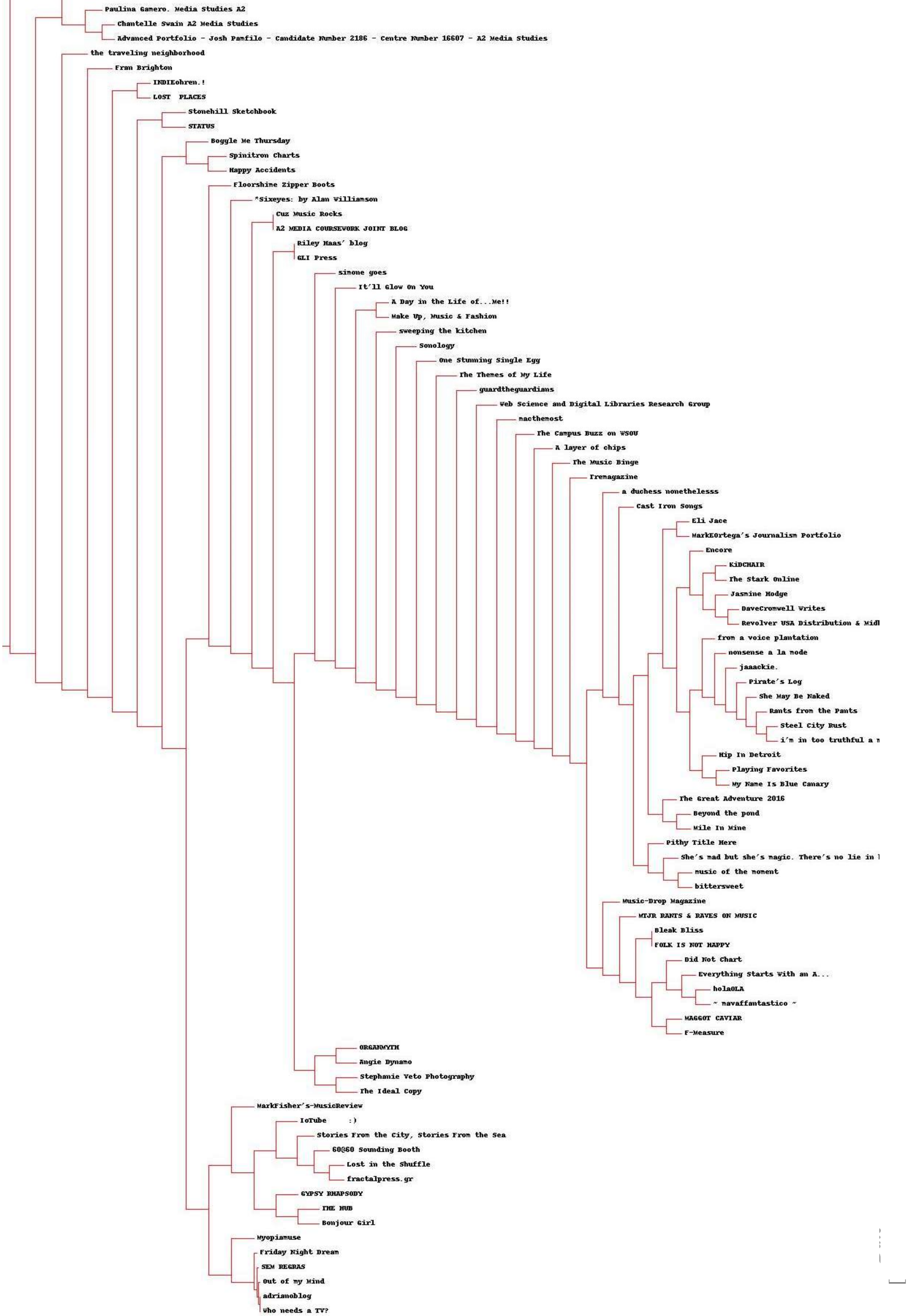
```
blogs,words,data=readfile('blogdata.txt')
clust=hcluster(data)
printclust(clust,labels=blogs)
drawdendrogram(clust,blogs,jpeg='dend.jpg')
```

Output from the cmd:



```
          Chantelle Swain A2 Media Studies
          Advanced Portfolio - Josh Pamfilo - Candidate Number 2186 - Centre Number 16607 - A2 Media Studies
    -
      the traveling neighborhood
      -
        Fran Brighton
        -
          -
            INDIEohren.!
            LOST  PLACES
          -
            -
              Stonehill Sketchbook
              STATUS
            -
              -
                Boggle Me Thursday
                -
                  Spinitron Charts
                  Happy Accidents
              -
                -
                  Floorshime Zipper Boots
                  -
                    *Sixeyes: by Alan Williamson
                    -
                      -
                        Cuz Music Rocks
                        A2 MEDIA COURSEWORK JOINT BLOG
                      -
                        -
                          Riley Haas' blog
                          GLI Press
                        -
                          -
                            simone goes
                            -
                              It'll Glow On You
                              -
                                -
                                  A Day in the Life of...Me!!
                                  Make Up, Music & Fashion
                                -
                                  sweeping the kitchen
                                  -
                                    Sonology
                                    -
                                      One Stunning Single Egg
                                      -
                                        The Themes of My Life
                                        -
                                          guardtheguardians
                                          -
                                            Web Science and Digital Libraries Research Group
```

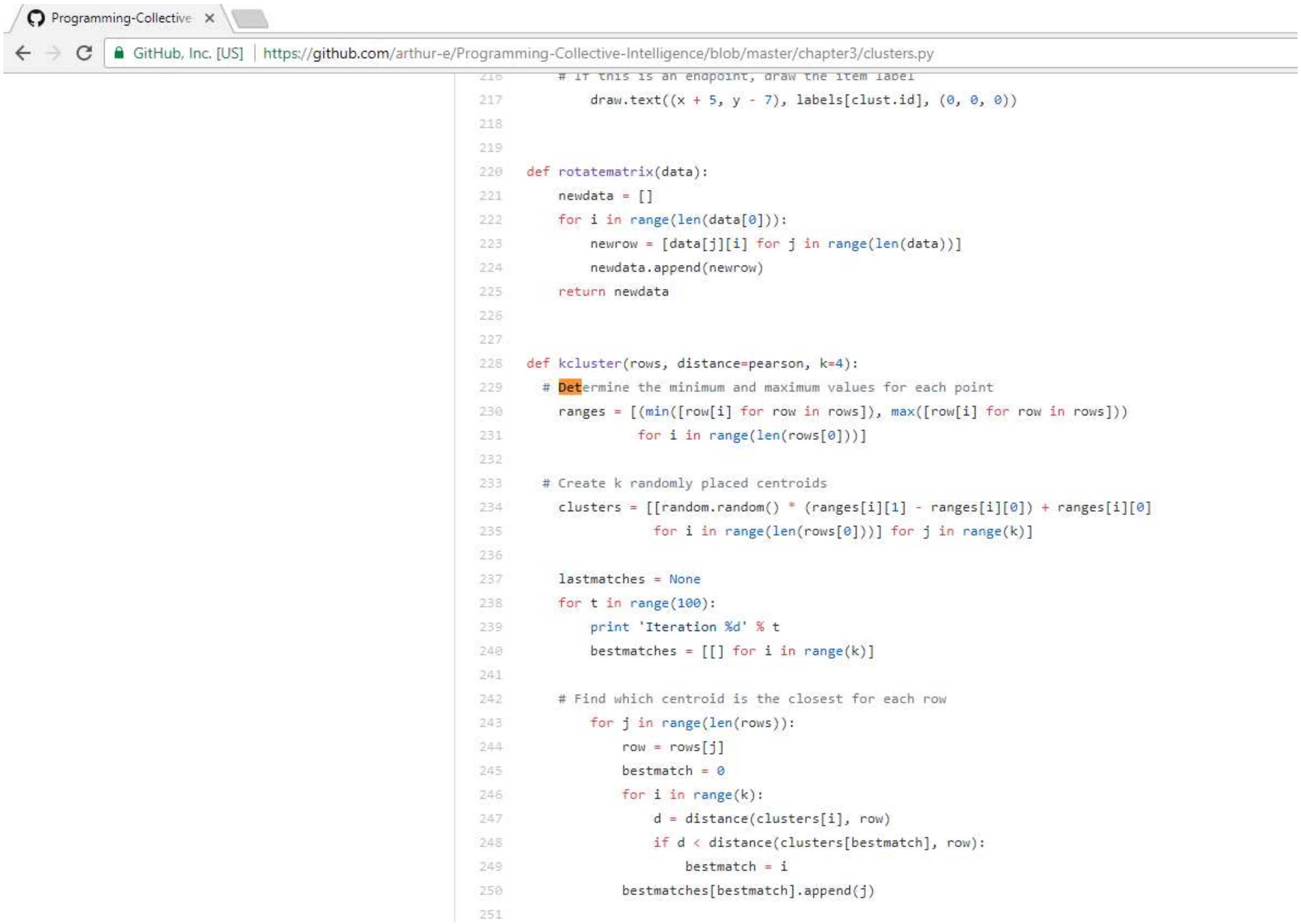
The results from cluster.py


Paulina Gamero. Media Studies A2
Chantelle Swain A2 Media Studies
Advanced Portfolio - Josh Pamfilo - Candidate Number 2186 - Centre Number 16607 - A2 Media Studies
the traveling neighborhood
Fran Brighton
INDIEohren.!
LOST PLACES
Stonehill Sketchbook
STATUS
Boggle Me Thursday
Spinitron Charts
Happy Accidents
Floorshine Zipper Boots
*Sixeyes: by Alan Williamson
Cuz Music Rocks
A2 MEDIA COURSEWORK JOINT BLOG
Riley Haas' blog
GLI Press
simone goes
It'll Glow On You
A Day in the Life of...Me!!
Make Up, Music & Fashion
sweeping the kitchen
Sonology
One Stunning Single Egg
The Themes of My Life
guardtheguardians
Web Science and Digital Libraries Research Group
macthemost
The Campus Buzz on WSOU
A layer of chips
The Music Binge
Tremagazine
a duchess nonethelesss
Cast Iron Songs
Eli Jace
MarkEOrtega's Journalism Portfolio
Encore
KiDCHAIR
The Stark Online
Jasmine Hodge
DaveCromwell Writes
Revolver USA Distribution & Midl
from a voice plantation
nonsense a la mode
jaaackie.
Pirate's Log
She May Be Naked
Rants from the Pants
Steel City Rust
i'm in too truthful a m
Hip In Detroit
Playing Favorites
My Name Is Blue Canary
The Great Adventure 2016
Beyond the pond
Mile In Mine
Pithy Title Here
She's mad but she's magic. There's no lie in l
music of the moment
bittersweet
Music-Drop Magazine
MTJR RANTS & RAVES ON MUSIC
Bleak Bliss
FOLK IS NOT HAPPY
Did Not Chart
Everything Starts With an A...
holaOLA
~ mavaffantastico ~
MAGGOT CAVIAR
F-Measure
ORGANMYTH
Angie Dynamo
Stephanie Veto Photography
The Ideal Copy
MarkFisher's-MusicReview
IoTube   :)
Stories From the City, Stories From the Sea
60@60 Sounding Booth
Lost in the Shuffle
fractalpress.gr
GYPSY RHAPSODY
THE HUB
Bonjour Girl
Myopiamuse
Friday Night Dream
SEM REGRAS
Out of my Mind
adrianoblog
Who needs a TV?

Problem 3

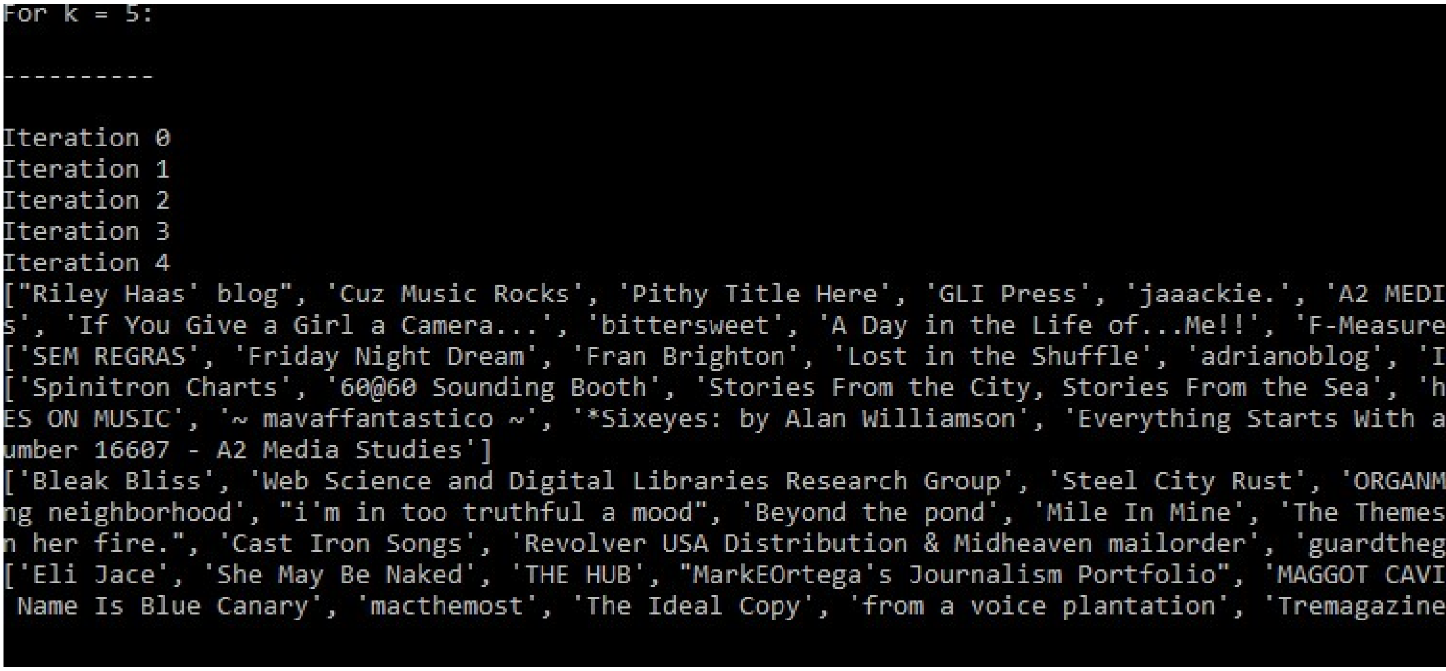
```
3.  Cluster the blogs using K-Means, using k=5,10,20. (see slide
25).  Print the values in each centroid, for each value of k.  How
many iterations were required for each value of k?
```

I will be using a function from clusters.py in this problem.


Programming-Collective
GitHub, Inc. [US]
https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter3/clusters.py

```
    # if this is an endpoint, draw the item label
        draw.text((x + 5, y - 7), labels[clust.id], (0, 0, 0))


def rotatematrix(data):
    newdata = []
    for i in range(len(data[0])):
        newrow = [data[j][i] for j in range(len(data))]
        newdata.append(newrow)
    return newdata


def kcluster(rows, distance=pearson, k=4):
  # Determine the minimum and maximum values for each point
    ranges = [(min([row[i] for row in rows]), max([row[i] for row in rows]))
              for i in range(len(rows[0]))]

  # Create k randomly placed centroids
    clusters = [[random.random() * (ranges[i][1] - ranges[i][0]) + ranges[i][0]
                for i in range(len(rows[0]))] for j in range(k)]

    lastmatches = None
    for t in range(100):
        print 'Iteration %d' % t
        bestmatches = [[] for i in range(k)]

    # Find which centroid is the closest for each row
        for j in range(len(rows)):
            row = rows[j]
            bestmatch = 0
            for i in range(k):
                d = distance(clusters[i], row)
                if d < distance(clusters[bestmatch], row):
                    bestmatch = i
            bestmatches[bestmatch].append(j)

```

The output picture is too big. Here is an example of what the output looks like. The rest is in p3 folder.



```
For k = 5:

----------

Iteration 0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
["Riley Haas' blog", 'Cuz Music Rocks', 'Pithy Title Here', 'GLI Press', 'jaaackie.', 'A2 MEDI
s', 'If You Give a Girl a Camera...', 'bittersweet', 'A Day in the Life of...Me!!', 'F-Measure
['SEM REGRAS', 'Friday Night Dream', 'Fran Brighton', 'Lost in the Shuffle', 'adrianoblog', 'I
['Spinitron Charts', '60@60 Sounding Booth', 'Stories From the City, Stories From the Sea', 'h
ES ON MUSIC', '~ mavaffantastico ~', '*Sixeyes: by Alan Williamson', 'Everything Starts With a
umber 16607 - A2 Media Studies']
['Bleak Bliss', 'Web Science and Digital Libraries Research Group', 'Steel City Rust', 'ORGANM
ng neighborhood', "i'm in too truthful a mood", 'Beyond the pond', 'Mile In Mine', 'The Themes
n her fire.", 'Cast Iron Songs', 'Revolver USA Distribution & Midheaven mailorder', 'guardtheg
['Eli Jace', 'She May Be Naked', 'THE HUB', "MarkEOrtega's Journalism Portfolio", 'MAGGOT CAVI
 Name Is Blue Canary', 'macthemost', 'The Ideal Copy', 'from a voice plantation', 'Tremagazine
```

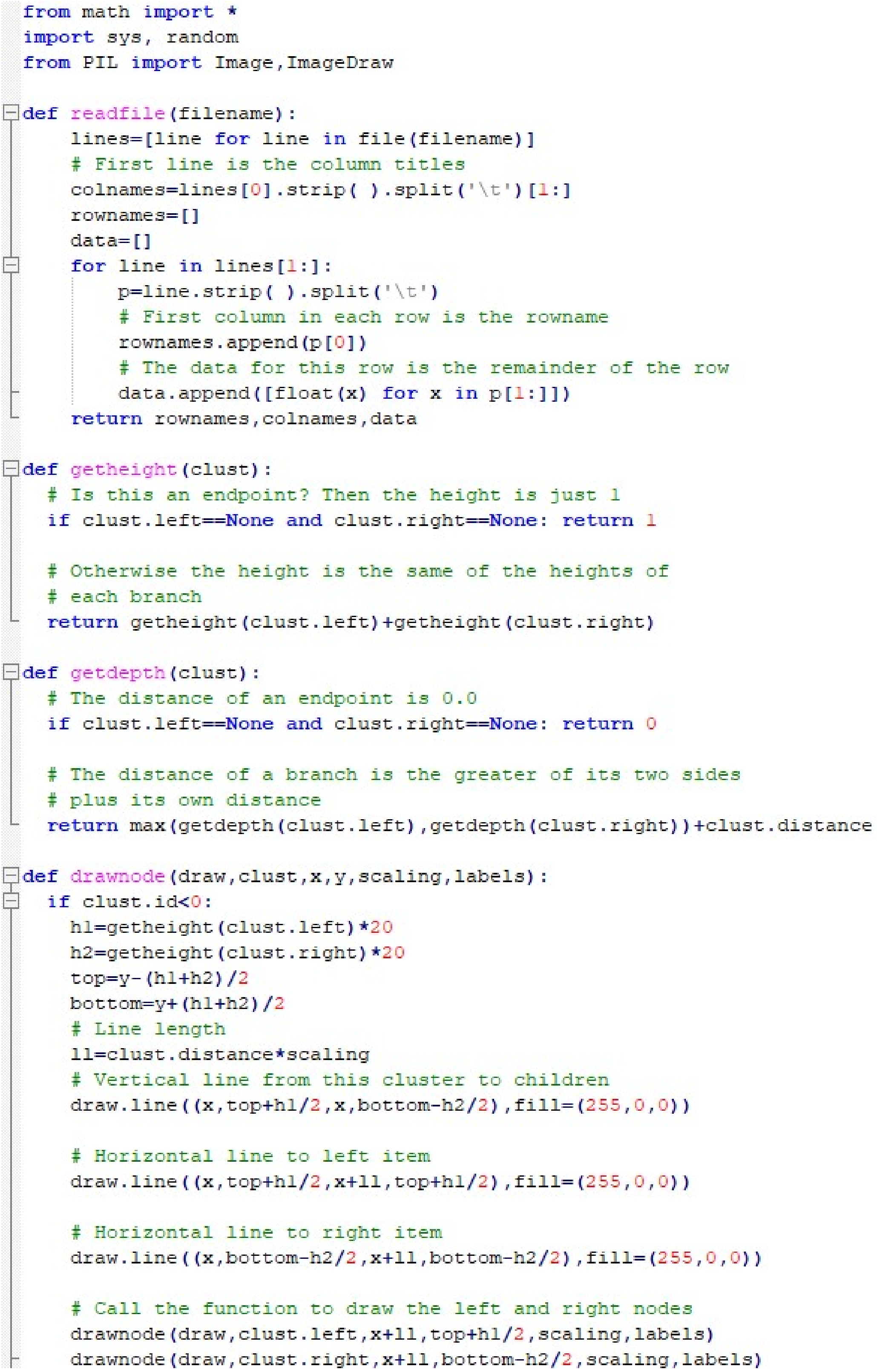
Problem 4

```
 4.  Use MDS to create a JPEG of the blogs similar to slide 29 of the
week 11 lecture.  How many iterations were required?
```

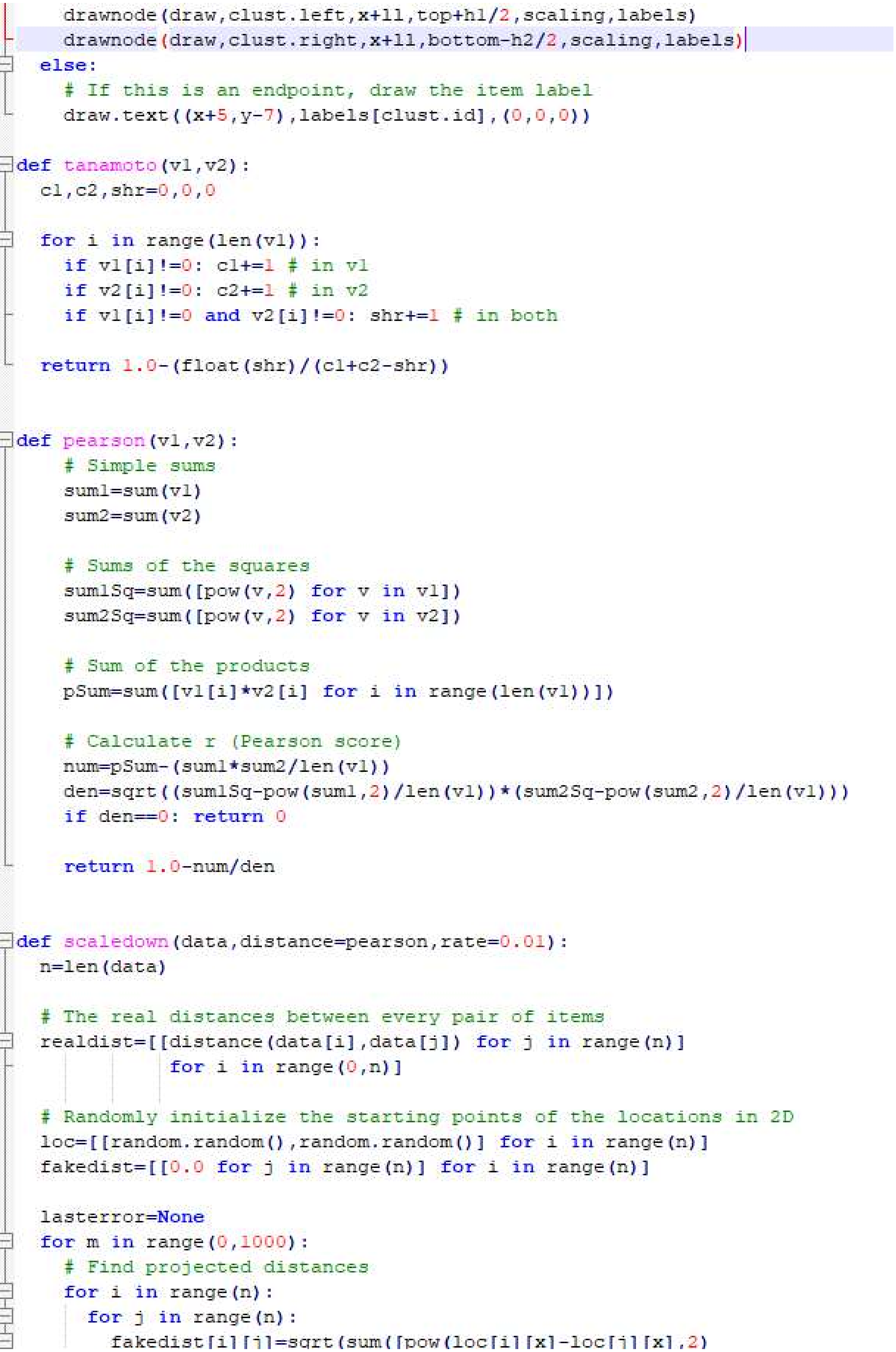
I used the function “scaledown” and “draw2d” from the script from https://github.com/arthur-e/Programming-Collective-Intelligence/tree/master/chapter3

The program is called 2d.py

I had an issue with this program, when I ran this program again, I got different results even if I use the same input. I am not sure what I am doing wrong.

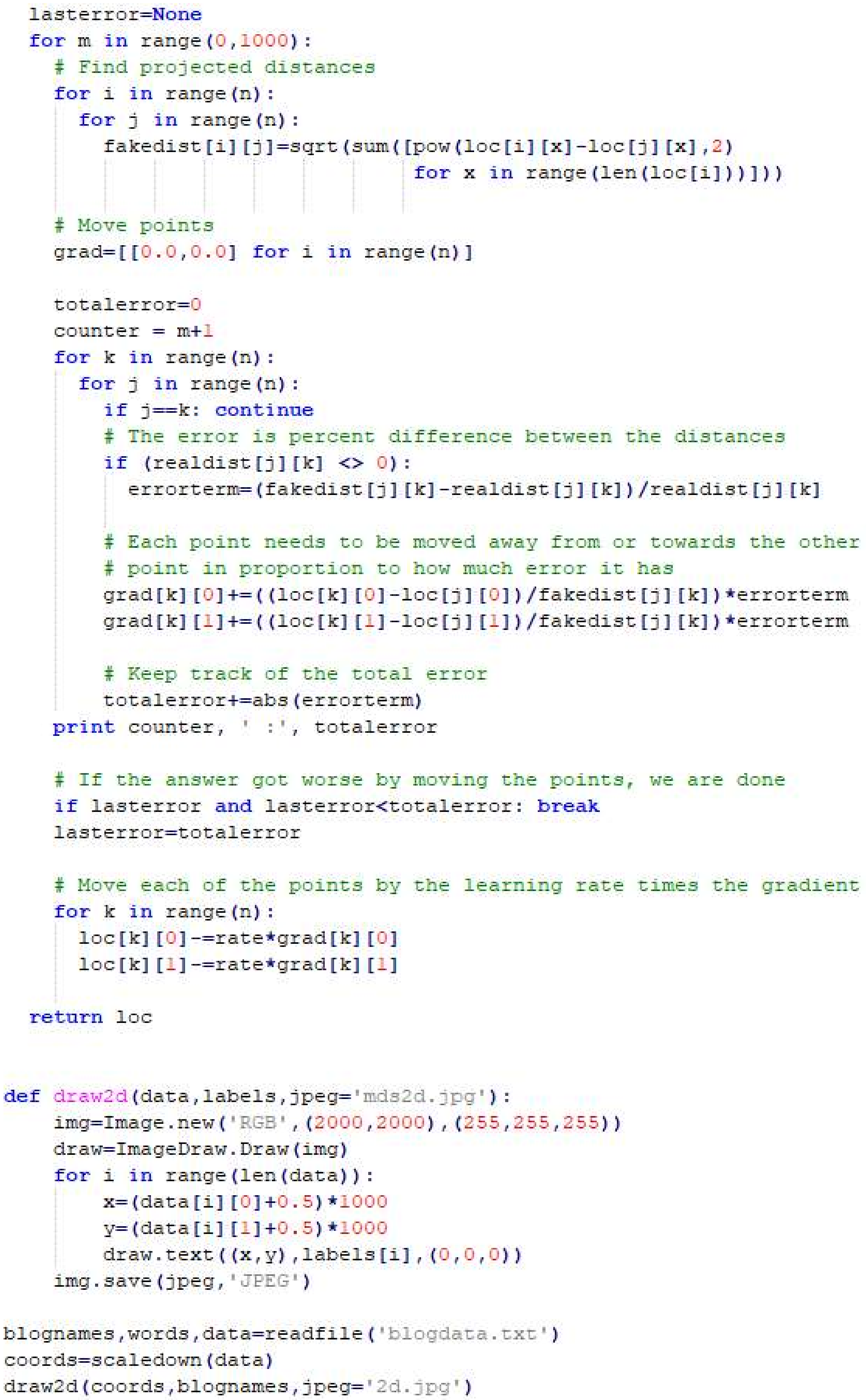


```
from math import *
import sys, random
from PIL import Image,ImageDraw

def readfile(filename):
    lines=[line for line in file(filename)]
    # First line is the column titles
    colnames=lines[0].strip( ).split('\t')[1:]
    rownames=[]
    data=[]
    for line in lines[1:]:
        p=line.strip( ).split('\t')
        # First column in each row is the rowname
        rownames.append(p[0])
        # The data for this row is the remainder of the row
        data.append([float(x) for x in p[1:]])
    return rownames,colnames,data

def getheight(clust):
  # Is this an endpoint? Then the height is just 1
  if clust.left==None and clust.right==None: return 1

  # Otherwise the height is the same of the heights of
  # each branch
  return getheight(clust.left)+getheight(clust.right)

def getdepth(clust):
  # The distance of an endpoint is 0.0
  if clust.left==None and clust.right==None: return 0

  # The distance of a branch is the greater of its two sides
  # plus its own distance
  return max(getdepth(clust.left),getdepth(clust.right))+clust.distance

def drawnode(draw,clust,x,y,scaling,labels):
  if clust.id<0:
    h1=getheight(clust.left)*20
    h2=getheight(clust.right)*20
    top=y-(h1+h2)/2
    bottom=y+(h1+h2)/2
    # Line length
    ll=clust.distance*scaling
    # Vertical line from this cluster to children
    draw.line((x,top+h1/2,x,bottom-h2/2),fill=(255,0,0))

    # Horizontal line to left item
    draw.line((x,top+h1/2,x+ll,top+h1/2),fill=(255,0,0))

    # Horizontal line to right item
    draw.line((x,bottom-h2/2,x+ll,bottom-h2/2),fill=(255,0,0))

    # Call the function to draw the left and right nodes
    drawnode(draw,clust.left,x+ll,top+h1/2,scaling,labels)
    drawnode(draw,clust.right,x+ll,bottom-h2/2,scaling,labels)
```



```
    drawnode(draw,clust.left,x+l1,top+h1/2,scaling,labels)
    drawnode(draw,clust.right,x+l1,bottom-h2/2,scaling,labels)
  else:
    # If this is an endpoint, draw the item label
    draw.text((x+5,y-7),labels[clust.id],(0,0,0))

def tanamoto(v1,v2):
  c1,c2,shr=0,0,0

  for i in range(len(v1)):
    if v1[i]!=0: c1+=1 # in v1
    if v2[i]!=0: c2+=1 # in v2
    if v1[i]!=0 and v2[i]!=0: shr+=1 # in both

  return 1.0-(float(shr)/(c1+c2-shr))


def pearson(v1,v2):
    # Simple sums
    sum1=sum(v1)
    sum2=sum(v2)

    # Sums of the squares
    sum1Sq=sum([pow(v,2) for v in v1])
    sum2Sq=sum([pow(v,2) for v in v2])

    # Sum of the products
    pSum=sum([v1[i]*v2[i] for i in range(len(v1))])

    # Calculate r (Pearson score)
    num=pSum-(sum1*sum2/len(v1))
    den=sqrt((sum1Sq-pow(sum1,2)/len(v1))*(sum2Sq-pow(sum2,2)/len(v1)))
    if den==0: return 0

    return 1.0-num/den


def scaledown(data,distance=pearson,rate=0.01):
  n=len(data)

  # The real distances between every pair of items
  realdist=[[distance(data[i],data[j]) for j in range(n)]
             for i in range(0,n)]

  # Randomly initialize the starting points of the locations in 2D
  loc=[[random.random(),random.random()] for i in range(n)]
  fakedist=[[0.0 for j in range(n)] for i in range(n)]

  lasterror=None
  for m in range(0,1000):
    # Find projected distances
    for i in range(n):
      for j in range(n):
        fakedist[i][j]=sqrt(sum([pow(loc[i][x]-loc[j][x],2)
```

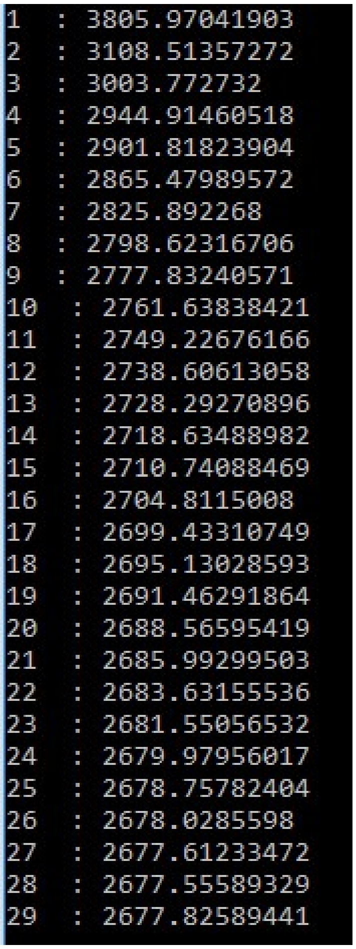
Date



```
  lasterror=None
  for m in range(0,1000):
    # Find projected distances
    for i in range(n):
      for j in range(n):
        fakedist[i][j]=sqrt(sum([pow(loc[i][x]-loc[j][x],2)
                                 for x in range(len(loc[i]))]))

    # Move points
    grad=[[0.0,0.0] for i in range(n)]

    totalerror=0
    counter = m+1
    for k in range(n):
      for j in range(n):
        if j==k: continue
        # The error is percent difference between the distances
        if (realdist[j][k] <> 0):
          errorterm=(fakedist[j][k]-realdist[j][k])/realdist[j][k]

        # Each point needs to be moved away from or towards the other
        # point in proportion to how much error it has
        grad[k][0]+=((loc[k][0]-loc[j][0])/fakedist[j][k])*errorterm
        grad[k][1]+=((loc[k][1]-loc[j][1])/fakedist[j][k])*errorterm

        # Keep track of the total error
        totalerror+=abs(errorterm)
    print counter, ' :', totalerror

    # If the answer got worse by moving the points, we are done
    if lasterror and lasterror<totalerror: break
    lasterror=totalerror

    # Move each of the points by the learning rate times the gradient
    for k in range(n):
      loc[k][0]-=rate*grad[k][0]
      loc[k][1]-=rate*grad[k][1]

  return loc


def draw2d(data,labels,jpeg='mds2d.jpg'):
    img=Image.new('RGB',(2000,2000),(255,255,255))
    draw=ImageDraw.Draw(img)
    for i in range(len(data)):
        x=(data[i][0]+0.5)*1000
        y=(data[i][1]+0.5)*1000
        draw.text((x,y),labels[i],(0,0,0))
    img.save(jpeg,'JPEG')

blognames,words,data=readfile('blogdata.txt')
coords=scaledown(data)
draw2d(coords,blognames,jpeg='2d.jpg')
```

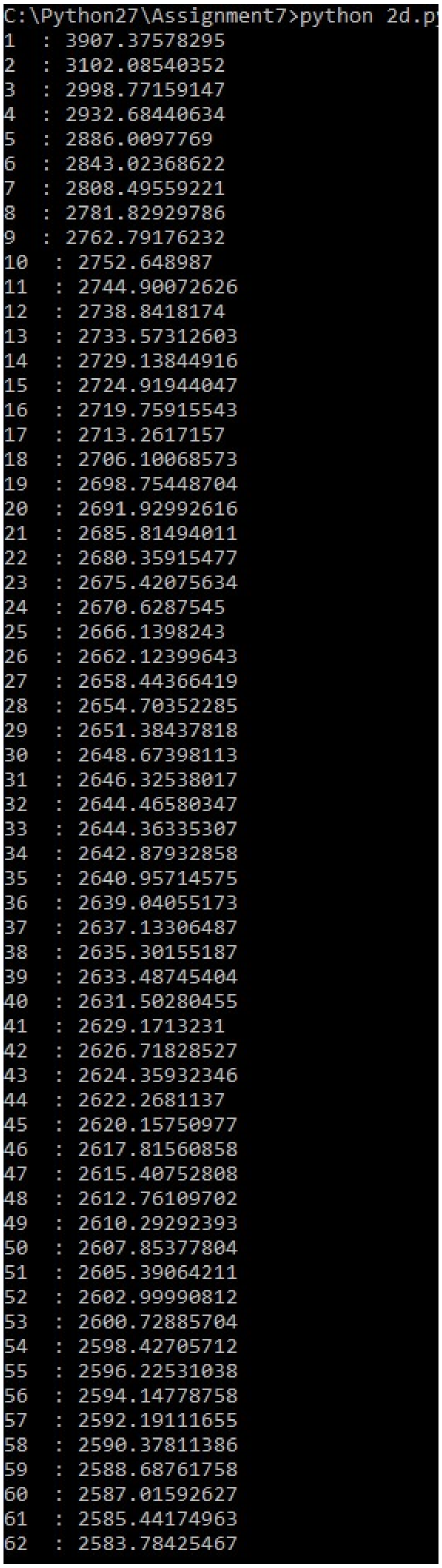
[Date]

The 1st output:

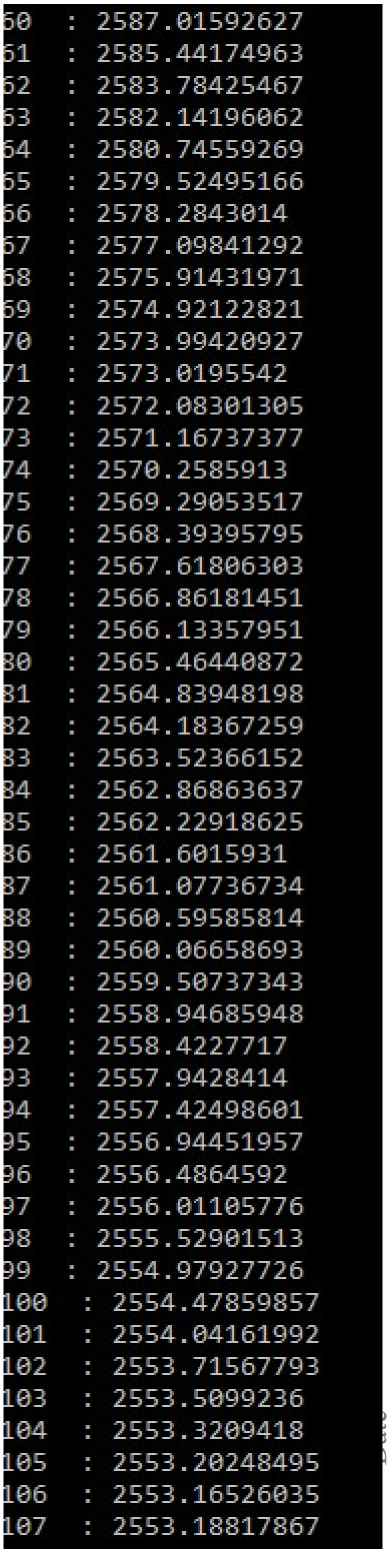


```
1   : 3805.97041903
2   : 3108.51357272
3   : 3003.772732
4   : 2944.91460518
5   : 2901.81823904
6   : 2865.47989572
7   : 2825.892268
8   : 2798.62316706
9   : 2777.83240571
10   : 2761.63838421
11   : 2749.22676166
12   : 2738.60613058
13   : 2728.29270896
14   : 2718.63488982
15   : 2710.74088469
16   : 2704.8115008
17   : 2699.43310749
18   : 2695.13028593
19   : 2691.46291864
20   : 2688.56595419
21   : 2685.99299503
22   : 2683.63155536
23   : 2681.55056532
24   : 2679.97956017
25   : 2678.75782404
26   : 2678.0285598
27   : 2677.61233472
28   : 2677.55589329
29   : 2677.82589441
```

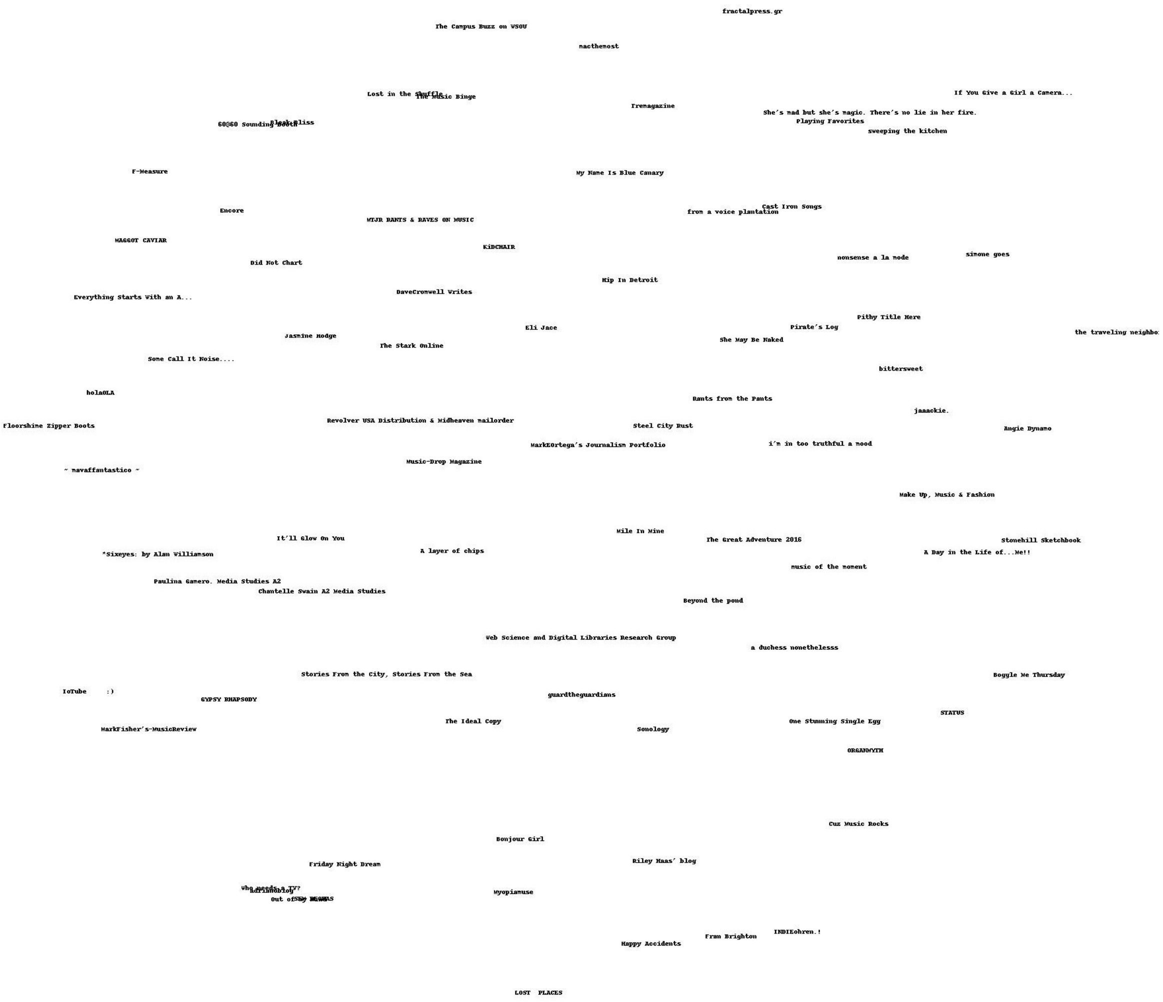
2nd time running this:



```
C:\Python27\Assignment7>python 2d.p
1   : 3907.37578295
2   : 3102.08540352
3   : 2998.77159147
4   : 2932.68440634
5   : 2886.0097769
6   : 2843.02368622
7   : 2808.49559221
8   : 2781.82929786
9   : 2762.79176232
10   : 2752.648987
11   : 2744.90072626
12   : 2738.8418174
13   : 2733.57312603
14   : 2729.13844916
15   : 2724.91944047
16   : 2719.75915543
17   : 2713.2617157
18   : 2706.10068573
19   : 2698.75448704
20   : 2691.92992616
21   : 2685.81494011
22   : 2680.35915477
23   : 2675.42075634
24   : 2670.6287545
25   : 2666.1398243
26   : 2662.12399643
27   : 2658.44366419
28   : 2654.70352285
29   : 2651.38437818
30   : 2648.67398113
31   : 2646.32538017
32   : 2644.46580347
33   : 2644.36335307
34   : 2642.87932858
35   : 2640.95714575
36   : 2639.04055173
37   : 2637.13306487
38   : 2635.30155187
39   : 2633.48745404
40   : 2631.50280455
41   : 2629.1713231
42   : 2626.71828527
43   : 2624.35932346
44   : 2622.2681137
45   : 2620.15750977
46   : 2617.81560858
47   : 2615.40752808
48   : 2612.76109702
49   : 2610.29292393
50   : 2607.85377804
51   : 2605.39064211
52   : 2602.99990812
53   : 2600.72885704
54   : 2598.42705712
55   : 2596.22531038
56   : 2594.14778758
57   : 2592.19111655
58   : 2590.37811386
59   : 2588.68761758
60   : 2587.01592627
61   : 2585.44174963
62   : 2583.78425467
```



```
60   : 2587.01592627
61   : 2585.44174963
62   : 2583.78425467
63   : 2582.14196062
64   : 2580.74559269
65   : 2579.52495166
66   : 2578.2843014
67   : 2577.09841292
68   : 2575.91431971
69   : 2574.92122821
70   : 2573.99420927
71   : 2573.0195542
72   : 2572.08301305
73   : 2571.16737377
74   : 2570.2585913
75   : 2569.29053517
76   : 2568.39395795
77   : 2567.61806303
78   : 2566.86181451
79   : 2566.13357951
80   : 2565.46440872
81   : 2564.83948198
82   : 2564.18367259
83   : 2563.52366152
84   : 2562.86863637
85   : 2562.22918625
86   : 2561.6015931
87   : 2561.07736734
88   : 2560.59585814
89   : 2560.06658693
90   : 2559.50737343
91   : 2558.94685948
92   : 2558.4227717
93   : 2557.9428414
94   : 2557.42498601
95   : 2556.94451957
96   : 2556.4864592
97   : 2556.01105776
98   : 2555.52901513
99   : 2554.97927726
100   : 2554.47859857
101   : 2554.04161992
102   : 2553.71567793
103   : 2553.5099236
104   : 2553.3209418
105   : 2553.20248495
106   : 2553.16526035
107   : 2553.18817867
```


fractalpress.gr
The Campus Buzz on WSOU
macthemost
Lost in the Shuffle
The Music Binge
If You Give a Girl a Camera...
Tremagazine
She’s mad but she’s magic. There’s no lie in her fire.
Playing Favorites
60@60 Sounding Booth
Bleak Bliss
sweeping the kitchen
F-Measure
My Name Is Blue Canary
Cast Iron Songs
Encore
from a voice plantation
MTJR RANTS & RAVES ON MUSIC
MAGGOT CAVIAR
KiDCHAIR
nonsense a la mode
simone goes
Did Not Chart
Hip In Detroit
DaveCromwell Writes
Everything Starts With an A...
Pithy Title Here
Eli Jace
Pirate’s Log
the traveling neighbo
Jasmine Hodge
She May Be Naked
The Stark Online
Some Call It Noise....
bittersweet
holaOLA
Rants from the Pants
jaaackie.
Revolver USA Distribution & Midheaven mailorder
Floorshime Zipper Boots
Steel City Rust
Angie Dynamo
i’m in too truthful a mood
MarkEOrtega’s Journalism Portfolio
Music-Drop Magazine
~ mavaffantastico ~
Make Up, Music & Fashion
Mile In Mine
It’ll Glow On You
The Great Adventure 2016
Stonehill Sketchbook
A layer of chips
*Sixeyes: by Alan Williamson
A Day in the Life of...Me!!
music of the moment
Paulina Gamero. Media Studies A2
Chantelle Swain A2 Media Studies
Beyond the pond
Web Science and Digital Libraries Research Group
a duchess nonethelesss
Stories From the City, Stories From the Sea
Boggle Me Thursday
IoTube :)
guardtheguardians
GYPSY RHAPSODY
STATUS
The Ideal Copy
One Stunning Single Egg
MarkFisher’s-MusicReview
Sonology
ORGANMYTH
Cuz Music Rocks
Bonjour Girl
Friday Night Dream
Riley Haas’ blog
Who needs a TV?
adrianoblog
Out of my Mind
Myopiamuse
INDIEohren.!
Fran Brighton
Happy Accidents
LOST PLACES

Reference:

https://github.com/arthur-e/Programming-Collective-Intelligence/tree/master/chapter3