CS-432/532 Introduction to Web Science:
Assignment #8:
Clustering Algorithms

Due on Thursday, April 7, 2016

Dr. Michael L. Nelson

Plinio Vargas
pvargas@cs.odu.edu

Contents

List of Figures

Listings

Problem 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) 500 terms, this is *after* the criteria on p. 32 (slide 7) has been satisfied.

1.1 Approach

A great deal of code was obtain from [1] in order to complete this assignment. All the codes in [1] are implemented in Python 2.7, while the implementation in this work is based using Python 3.4. There were minor changes to adapt into a different version, and they will not be detailed; only major modifications will be pointed out.

BlogMatrix.py is the *Python* code implemented to answer this problem. To form the Matrix, we created a *set* object (*urilist*) containing 98 generated blog titles. Two required blogs were added in lines 42-43.

Listing 1: Creating Blog Matrix: BlogMatrix.py

```
37      urilist = set()
38      apcount = {}
39      wordcounts = {}
40      feedlist = []
41
42      urilist.add('http://f-measure.blogspot.com/')
43      urilist.add('http://ws-dl.blogspot.com/')
44
45      for uri in urilist:
46          title, wc = getallpages(uri)
47          if title:
48              wordcounts[title] = wc
49              print(title, wordcounts[title])
50              for word, count in wc.items():
51                  apcount.setdefault(word, 0)
52                  if count > 1:
53                      apcount[word] += 1
54              feedlist.append(uri)
```

Problem 1 continued on next page...

```
55              urilist.add(uri)
56
57      while len(urilist) < 100:
58          lasturi = set()
59          AddURI(lasturi, 'http://www.blogger.com/next-blog?navBar=true&blogID
                =3471633091411211117')
60
61          print('Getting word count....')
62          for uri in lasturi:
63              uri = re.match('(^.*\.com\/)(?!expref)', uri).group(0)
64              if uri not in urilist:
65                  title, wc = getallpages(uri)
66                  if title:
67                      wordcounts[title] = wc
68                      print(len(urilist), title, wordcounts[title])
69                      for word, count in wc.items():
70                          apcount.setdefault(word, 0)
71                          if count > 1:
72                              apcount[word] += 1
73                      feedlist.append(uri)
74                      urilist.add(uri)
75
76                  else:
77                      print('Discarding URI...')
78
79      wordlist = []
80      for w, bc in apcount.items():
81          frac = float(bc) / len(feedlist)
82          if 0.5 > frac > 0.1:  wordlist.append(w)
83
84      out = open('blogdata.txt', 'w')
85      out.write('Blog')
86      for word in wordlist[:500]: out.write('\t%s' % word)
87      out.write('\n')
88      for blog, wc in wordcounts.items():
89          print(blog)
90          out.write(blog)
91          for word in wordlist[:500]:
92              if word in wc: out.write('\t%d' % wc[word])
93              else: out.write('\t0')
94          out.write('\n')
95      out.close()
```

We proceeded to find the remaining URL(s) for our matrix. We re-utilized the code developed on assignment 2 to find the HTTP redirection ultimate location(line 59).

We removed undesired strings from the returned URL by using regular expression (line 63) and discarding any URL without a title (line 66). Finally, we included the URL in a *list* object (line 73) in order to provide all URLs used for this assignment. The remaining of the code was an extraction from [1].

A major difference between this implementation and [1] is that the latest does not account for all the web-pages in a blog. Only the first blog page is taken into consideration for word-counts.

Problem 1 continued on next page...

In order to account for all the pages in the blog, we created two functions in ***BlogMatrix.py***: *getallpages* and *linkpages*.

Listing 2: Accounting for all Blog Pages: BlogMatrix.py

```
107 def getallpages(uri):
108     twc = None
109     rss_feed = uri + 'feeds/posts/default/'
110     title, wc = getwordcounts(rss_feed)
111     if title:
112         print('Getting linkpages....', title)
113         twc = linkpages(wc, rss_feed)
114 
115     if not twc:
116         return '', ''
117 
118     return title, twc
119 
120 
121 def linkpages(wc, rss_feed):
122     twc = {}
123     for word, count in wc.items():
124         twc.setdefault(word, 0)
125         twc[word] += count
126 
127     flag = True
128     while flag:
129         page = requests.get(rss_feed).text
130         soup = BeautifulSoup(page, 'html.parser')
131         for link in soup.find_all('link'):
132             t = link.get('rel')
133             if t and t[0] == 'next':
134                 print('Getting %s ...' % link.get('href'))
135                 title, wc = getwordcounts(link.get('href'))
136 
137                 rss_feed = link.get('href')
138                 for word, count in wc.items():
139                     twc.setdefault(word, 0)
140                     twc[word] += count
141 
142                 print(twc)
143                 break
```

For each URL generated, a call is made to *getallpages*(see line 46 Listing 1) which has the purpose of passing all the words and title from a particular blog back to the main function(line 114).

linkpages actually counts all words and their frequencies in a blog. This is accomplished by inspecting the link tag with rel='next' and using the same scheme as [1], adding the initial values of the first page in the title word count variable *twc* (lines 119-121).

Problem 1 continued on next page...

Each blog page word count will be added to *twc* until there are no more rel='next' links (line 127-142). The iteration will continue until all generated URLs are processed. The remaining code is a replica from [1], but is worthy to mention that to find the 500 terms from the blog, we selected the first 500 columns by trunking the variable **wordlist** (line 86).

Problem 1 continued on next page...

1.2 Solution

a. File: **feedlist.txt** in github contains all the URLs feed for the assignment including:
`http://f-measure.blogspot.com/`
`http://ws-dl.blogspot.com/`

b. File **blogdata.tex**: contains the matrix with blog title and word counts, similar to [1].

Problem 2

Create an ASCII and JPEG dendrogram that clusters (i.e., HAC)the most similar blogs (see slides 12 & 13). Include the JPEG in your report and upload the ascii file to github (it will be too unwieldy for inclusion in the report).

2.1 Approach

We used **blogdata.tex** file, exact code in [1] to complete this problem. We added a file stream (line 19) in order to print **ascii** file.

Listing 3: Generating ASCII and JPEG Dendrogram: PrintClusters.py

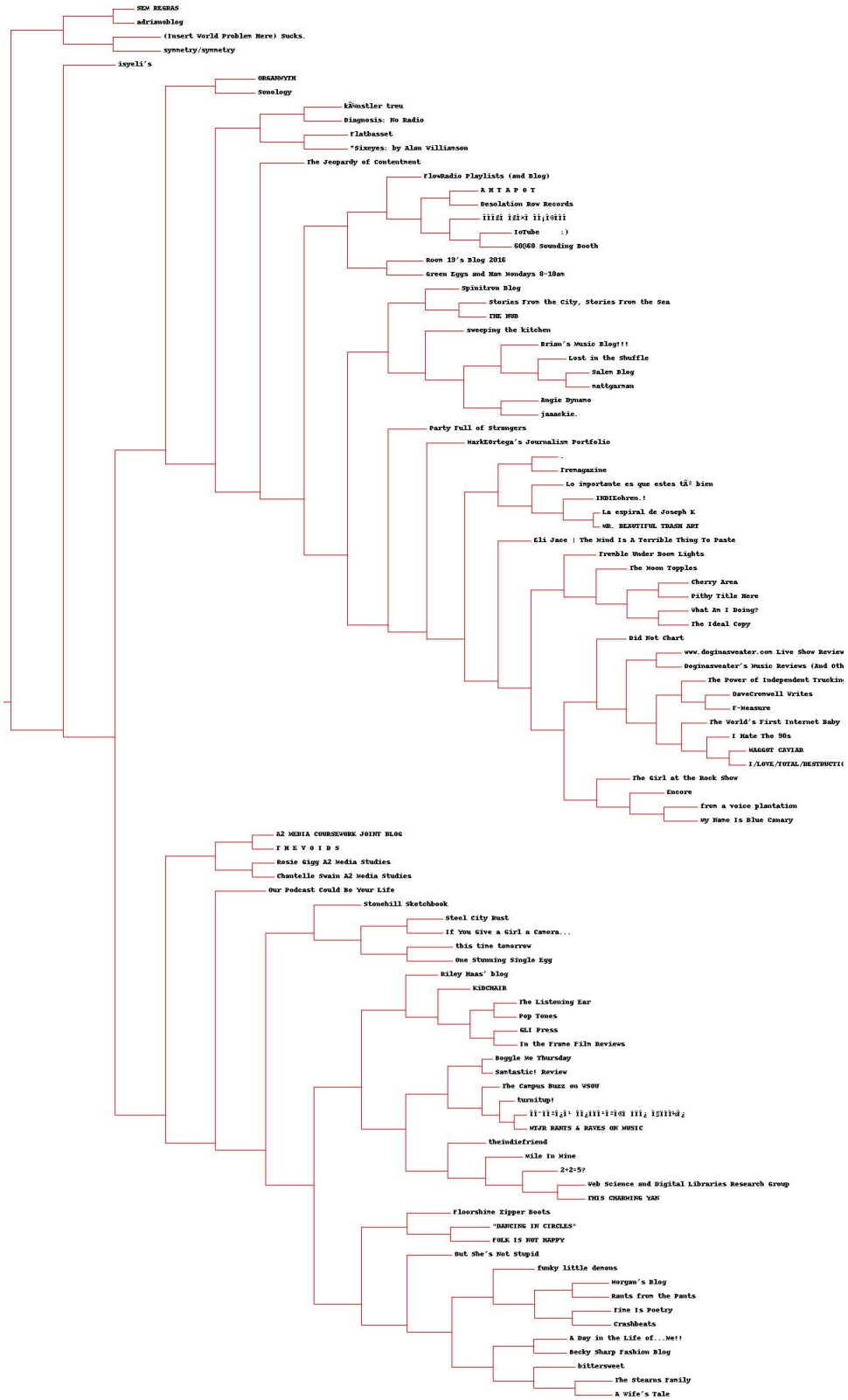
```
13 import lib.PCI_Code_Folder.chapter3.clusters as clusters
14
15 def main():
16     blognames,words,data=clusters.readfile('blogdata.txt')
17     clust=clusters.hcluster(data)
18     outfile = open('ascii-dendogram.txt', 'w')
19     clusters.printclust(clust, labels=blognames, file=outfile)
20     outfile.close()
21     clusters.drawdendrogram(clust,blognames,jpeg='blogclust.jpg')
22
23     return
```

2.2 Solution

a. File: **ascii-dendogram.txt** in github contains ASCII dendrogram that clusters the most similar blogs.

b. File **blogclust.jpeg**: contains JPEG dendrogram that clusters the most similar blogs. See example below.

Problem 2 continued on next page...


SEM REGRAS
adrianwobley
(Insert World Problem Here) Sucks.
symmetry/symmetry
isyeli's
ORGANOVYM
Sonology
kÃ¤nstler treu
Diagnosis: No Radio
flatbasset
"Sixeyes: by Alan Williamson
The Jeopardy of Contentment
FlowRadio Playlists (and Blog)
A M T A P O T
Desolation Row Records
ÎÎÎÍÍ ÎÍÍ×Î ÎÎ¡Î¤ÎÎÎ
IoTube :)
60Q60 Sounding Booth
Room 19's Blog 2016
Green Eggs and Man Mondays 8-10am
Spinitron Blog
Stories From the City, Stories From the Sea
THE HUB
sweeping the kitchen
Brian's Music Blog!!!
Lost in the Shuffle
Salem Blog
mattgarman
Angie Dynamo
jaaackie.
Party Full of Strangers
MarkEOrtega's Journalism Portfolio
.
Frenagazine
Lo importante es que estes tÃ³ bien
INDIEohren.!
La espiral de Joseph K
MR. BEAUTIFUL TRASH ART
Eli Jace | The Mind Is A Terrible Thing To Paste
Trenble Under Boon Lights
The Moon Topples
Cherry Area
Pithy Title Here
What Am I Doing?
The Ideal Copy
Did Not Chart
www.doginasweater.com Live Show Review
Doginasweater's Music Reviews (And Oth
The Power of Independent Truckin
DaveCronwell Writes
F-Measure
The World's First Internet Baby
I Hate The 90s
MAGGOT CAVIAR
I/LOVE/TOTAL/DESTRUCTI(
The Girl at the Rock Show
Encore
from a voice plantation
My Name Is Blue Canary
A2 MEDIA COURSEWORK JOINT BLOG
I M E Y 0 I D S
Rosie Gigg A2 Media Studies
Chantelle Swain A2 Media Studies
Our Podcast Could Be Your Life
Stonehill Sketchbook
Steel City Rust
If You Give a Girl a Camera...
this time tomorrow
One Stunning Single Egg
Riley Haas' blog
KiDCHAIR
The Listening Ear
Pop Tones
GLI Press
In the Frame Film Reviews
Boggle Me Thursday
Santastic! Review
The Campus Buzz on WSOU
turnitup!
ÎÎ¯ÎÎ²¿ÎÌ ÎÎ¿ÎÎÎ¹Î²ÎÎ ÎÎÎ¿ ÎÎÎ¿ÎÎÎÎ¼¿
MIJR RANTS & RAVES ON MUSIC
theindiefriend
Wile In Wine
2+2=5?
Web Science and Digital Libraries Research Group
THIS CHARMING VAN
Floorshine Zipper Boots
"DANCING IN CIRCLES"
FOLK IS NOT HAPPY
But She's Not Stupid
funky little demons
Horgan's Blog
Rants from the Pants
Fine Is Poetry
Crashbeats
A Day in the Life of...Me!!
Becky Sharp Fashion Blog
bittersweet
The Stearns Family
A Wife's Tale

Problem 3

Cluster the blogs using K-Means, using k=5,10,20. (see slide 18). Print the values in each centroid, for each value of k. How many iterations were required for each value of k?

3.1 Approach

For this problem *PrntK-Cluster.py* was implemented, using

Listing 4: Using K-Means: PrintK-Clusters.py

```
13 import lib.PCI_Code_Folder.chapter3.clusters as clusters
14
15 def main():
16     # record running time
17     start = time()
18     print('Starting Time: %s' % strftime("%a,  %b %d, %Y at %H:%M:%S", localtime()
       ))
19
20     outfile = open('k-clusters.txt', 'w')
21     blognames, words, data=clusters.readfile('blogdata.txt')
22     k_means = [5, 10, 20]
23     for k in k_means:
24         kclust, n_iterations = clusters.kcluster(data, k=k)
25
26         counter = 0
27         for i in range(len(kclust)):
28             outfile.write('Cluster-%d:\n%s\n' % (i, ('-' * 9)))
29             print('Cluster-%d:\n%s' % (i, ('-' * 9)))
30             for r in kclust[i]:
31                 counter += 1
32                 outfile.write('%s,' % blognames[r])
33                 print(blognames[r], end=', ')
34                 if counter % 5 == 0:
35                     outfile.write('\n')
36                     print()
37             outfile.write('\n')
38             print('\n')
39
40         outfile.write('Number of iterations for k=%d is %d.\n\n\n' % (k,
           n_iterations + 1))
41         print('Number of iterations for k=%d is %d.\n\n\n' % (k, n_iterations + 1)
           )
42
43     print('\nEnd Time:  %s' % strftime("%a,  %b %d, %Y at %H:%M:%S", localtime()))
44     print('Execution Time: %.2f seconds' % (time()-start))
```

3.2 Solution

```
Cluster-0:
---------
```

```
A2 MEDIA COURSEWORK JOINT BLOG,A Day in the Life of...Me!!,La espiral de Joseph
    K,theindiefriend,GLI Press,
symmetry/symmetry,MR. BEAUTIFUL TRASH ART,What Am I Doing?,A Wife's Tale,Morgan's Blog,
.,The Girl at the Rock Show,KiDCHAIR,2+2=5?,MarkEOrtega's Journalism Portfolio,
isyeli's,Steel City Rust,Tremagazine,jaaackie.,The Ideal Copy,
from a voice plantation,In the Frame Film Reviews,The Jeopardy of Contentment,My Name Is Blue
    Canary,The Listening Ear,
Cherry Area,Web Science and Digital Libraries Research Group,Becky Sharp Fashion Blog,Our Podcast
    Could Be Your Life,If You Give a Girl a Camera...,
THIS CHARMING YAN,Mile In Mine,Pop Tones,
Cluster-1:
---------
Riley Haas' blog,
Cluster-2:
---------
(Insert World Problem Here) Sucks.,
ORGANMYTH,The Stearns Family,One Stunning Single Egg,Time Is Poetry,Sonology,
funky little demons,But She's Not Stupid,Floorshime Zipper Boots,Room 19's Blog 2016,Crashbeats,
The Moon Topples,Pithy Title Here,Rants from the Pants,Tremble Under Boom Lights,bittersweet,

Cluster-3:
---------

Salem Blog,this time tomorrow,,A H T A P O T,IoTube :),
Lo importante es que estes tu bien,The Campus Buzz on WSOU,Angie Dynamo,FlowRadio Playlists (and
    Blog), ,
Chantelle Swain A2 Media Studies,Stories From the City, Stories From the Sea,60\@60 Sounding
    Booth,Party Full of Strangers,THE HUB,
Stonehill Sketchbook,Lost in the Shuffle,Spinitron Blog,SEM REGRAS,INDIEohren.!,
Desolation Row Records,sweeping the kitchen,mattgarman,Brian's Music Blog!!!,Green Eggs and Ham
    Mondays 8-10am,
adrianoblog,
Cluster-4:
---------
MAGGOT CAVIAR,The World's First Internet Baby,Rosie Gigg A2 Media Studies,"DANCING IN CIRCLES",
T H E V O I D S,kunstler treu,Boggle Me Thursday,DaveCromwell Writes,turnitup!,
Encore,Flatbasset,FOLK IS NOT HAPPY,MTJR RANTS & RAVES ON MUSIC,I Hate The 90s,
Diagnosis: No Radio,Samtastic! Review,*Sixeyes: by Alan Williamson,www.doginasweater.com Live Show
    Review Archive,F-Measure,
I/LOVE/TOTAL/DESTRUCTION,The Power of Independent Trucking,Did Not Chart,Doginasweater's Music
    Reviews (And Other Horseshit),Eli Jace | The Mind Is A Terrible Thing To Paste,

Number of iterations for k=5 is 5.


Cluster-0:
---------

Cluster-1:
---------
MAGGOT CAVIAR,The World's First Internet Baby,symmetry/symmetry,"DANCING IN CIRCLES",Sonology,
DaveCromwell Writes,Encore,from a voice plantation,My Name Is Blue Canary,I Hate The 90s,
```

```
*Sixeyes: by Alan Williamson,www.doginasweater.com Live Show Review
    Archive,F-Measure,I/LOVE/TOTAL/DESTRUCTION,The Power of Independent Trucking,
Did Not Chart,Tremble Under Boom Lights,Doginasweater's Music Reviews (And Other
    Horseshit),mattgarman,Eli Jace | The Mind Is A Terrible Thing To Paste,

Cluster-2:
---------
Riley Haas' blog,
Cluster-3:
---------
Unicode Character,The Campus Buzz on WSOU,Boggle Me Thursday,turnitup!,
MTJR RANTS & RAVES ON MUSIC,Samtastic! Review,
Cluster-4:
---------
La espiral de Joseph K,MR. BEAUTIFUL TRASH ART,Lo importante es que estes tu bien,
.,But She's Not Stupid,MarkEOrtega's Journalism Portfolio,Tremagazine,FOLK IS NOT HAPPY,
SEM REGRAS,INDIEohren.!,adrianoblog,
Cluster-5:
---------
GLI Press,kunstler treu,
KiDCHAIR,Flatbasset,In the Frame Film Reviews,The Jeopardy of Contentment,The Listening Ear,
Diagnosis: No Radio,Our Podcast Could Be Your Life,Pop Tones,
Cluster-6:
---------
A2 MEDIA COURSEWORK JOINT BLOG,ORGANMYTH,
theindiefriend,Rosie Gigg A2 Media Studies,T H E V O I D S,Floorshime Zipper Boots,2+2=5?,
Chantelle Swain A2 Media Studies,Web Science and Digital Libraries Research Group,THIS CHARMING
    YAN,
Cluster-7:
---------

Cluster-8:
---------
this time tomorrow,A H T A P O T,
IoTube    :),FlowRadio Playlists (and Blog),UnicodeCharacters,Stories From the City, Stories From
    the Sea,60@60 Sounding Booth,
Party Full of Strangers,THE HUB,Lost in the Shuffle,Spinitron Blog,Desolation Row Records,
sweeping the kitchen,Brian's Music Blog!!!,Green Eggs and Ham Mondays 8-10am,
Cluster-9:
---------
Salem Blog,(Insert World Problem Here) Sucks.,
A Day in the Life of...Me!!,The Stearns Family,One Stunning Single Egg,What Am I Doing?,A Wife's
    Tale,
Time Is Poetry,Morgan's Blog,Angie Dynamo,funky little demons,The Girl at the Rock Show,
isyeli's,Steel City Rust,jaaackie.,The Ideal Copy,Room 19's Blog 2016,
Crashbeats,The Moon Topples,Cherry Area,Becky Sharp Fashion Blog,Stonehill Sketchbook,
If You Give a Girl a Camera...,Pithy Title Here,Rants from the Pants,Mile In Mine,bittersweet,

Number of iterations for k=10 is 11.


Cluster-0:
---------
```

```
Salem Blog,IoTube   :),Stonehill Sketchbook,Lost in the Shuffle,sweeping the kitchen,
mattgarman,Brian's Music Blog!!!,
Cluster-1:
---------
MAGGOT CAVIAR,The World's First Internet Baby,"DANCING IN CIRCLES",
.,Floorshime Zipper Boots,DaveCromwell Writes,FOLK IS NOT HAPPY,I Hate The 90s,
Web Science and Digital Libraries Research Group,F-Measure,I/LOVE/TOTAL/DESTRUCTION,THIS CHARMING
    YAN,The Power of Independent Trucking,
Did Not Chart,Doginasweater's Music Reviews (And Other Horseshit),Eli Jace | The Mind Is A
    Terrible Thing To Paste,
Cluster-2:
---------
A Day in the Life of...Me!!,ORGANMYTH,
The Stearns Family,A Wife's Tale,Sonology,Angie Dynamo,jaaackie.,
Room 19's Blog 2016,Crashbeats,Diagnosis: No Radio,Cherry Area,Pithy Title Here,
Mile In Mine,Green Eggs and Ham Mondays 8-10am,bittersweet,
Cluster-3:
---------
Riley Haas' blog,
Cluster-4:
---------

Cluster-5:
---------

Cluster-6:
---------
kunstler treu,
Flatbasset,*Sixeyes: by Alan Williamson,
Cluster-7:
---------
(Insert World Problem Here) Sucks.,
Cluster-8:
---------
A2 MEDIA COURSEWORK JOINT BLOG,Rosie Gigg A2 Media Studies,
T H E V O I D S,Chantelle Swain A2 Media Studies,
Cluster-9:
---------
Unicode-Characters,The Campus Buzz on WSOU,turnitup!,
MTJR RANTS & RAVES ON MUSIC,
Cluster-10:
---------
this time tomorrow,theindiefriend,One Stunning Single Egg,What Am I Doing?,
funky little demons,2+2=5?,The Ideal Copy,The Moon Topples,If You Give a Girl a Camera...,
Spinitron Blog,Rants from the Pants,
Cluster-11:
---------

Cluster-12:
---------
La espiral de Joseph K,MR. BEAUTIFUL TRASH ART,Lo importante es que estes tu bien,
MarkEOrtega's Journalism Portfolio,The Jeopardy of Contentment,SEM REGRAS,INDIEohren.!,adrianoblog,
```

Problem 3 continued on next page...

```
Cluster-13:
---------

Cluster-14:
---------
A H T A P O T,FlowRadio Playlists (and Blog),Unicode-Characters,Stories From the City, Stories
    From the Sea,60@60 Sounding Booth,
Party Full of Strangers,THE HUB,Desolation Row Records,
Cluster-15:
---------
Boggle Me Thursday,Samtastic! Review,

Cluster-16:
---------
symmetry/symmetry,The Girl at the Rock Show,Encore,from a voice plantation,My Name Is Blue Canary,
www.doginasweater.com Live Show Review Archive,Becky Sharp Fashion Blog,Tremble Under Boom Lights,
Cluster-17:
---------
Time Is Poetry,But She's Not Stupid,
isyeli's,Tremagazine,
Cluster-18:
---------

Cluster-19:
---------
GLI Press,Morgan's Blog,KiDCHAIR,
Steel City Rust,In the Frame Film Reviews,The Listening Ear,Our Podcast Could Be Your Life,Pop
    Tones,

Number of iterations for k=20 is 6.
```

```
Number of iterations for k=5 is 5.
Number of iterations for k=10 is 11.
Number of iterations for k=20 is 6.
```

Problem 4

Use MDS to create a JPEG of the blogs similar to slide 29. How many iterations were required?

4.1 Approach

For this problem *CreateMDS.py* was implemented, using

Listing 5: Creating MDS with: CreateMDS.py

```
15 def main():
16     # record running time
17     start = time()
18     print('Starting Time: %s' % strftime("%a,  %b %d, %Y at %H:%M:%S", localtime()
           ))
```

```
19
20      blognames, words, data=clusters.readfile('blogdata.txt')
21      coords, n_iterations = clusters.scaledown(data)
22      clusters.draw2d(coords, blognames, jpeg='blogs2d.jpg')
23
24      print("Number of iterations is %d" % n_iterations)
25
26      print('\nEnd Time:  %s' % strftime("%a,  %b %d, %Y at %H:%M:%S", localtime()))
27      print('Execution Time: %.2f seconds' % (time()-start))
28      return
```

Above code was extracted from [1]

4.2 Solution

File name: *blogclust.jpg*


```
Number of iterations required was 424
```

Problem 5 - Extra Credit

Re-run question 2, but this time with proper TFIDF calculations instead of the hack discussed on slide 7 (p. 32). Use the same 500 words, but this time replace their frequency count with TFIDF scores as computed in assignment #3. Document the code, techniques, methods, etc. used to generate these TFIDF values. Upload the new data file to github.

Compare and contrast the resulting dendrogram with the dendrogram from question #2.

Note: ideally you would not reuse the same 500 terms and instead come up with TFIDF scores for all the terms and then choose the top 500 from that list, but I'm trying to limit the amount of work necessary.

5.1 Approach

The approach is the same as in problem 2, but we substituted the matrix data calculating with TFIDF values. Our **Total Document Corpus** is accumulated as we read the matrix in line 38. **Document Term** values are accumulated in line 37.

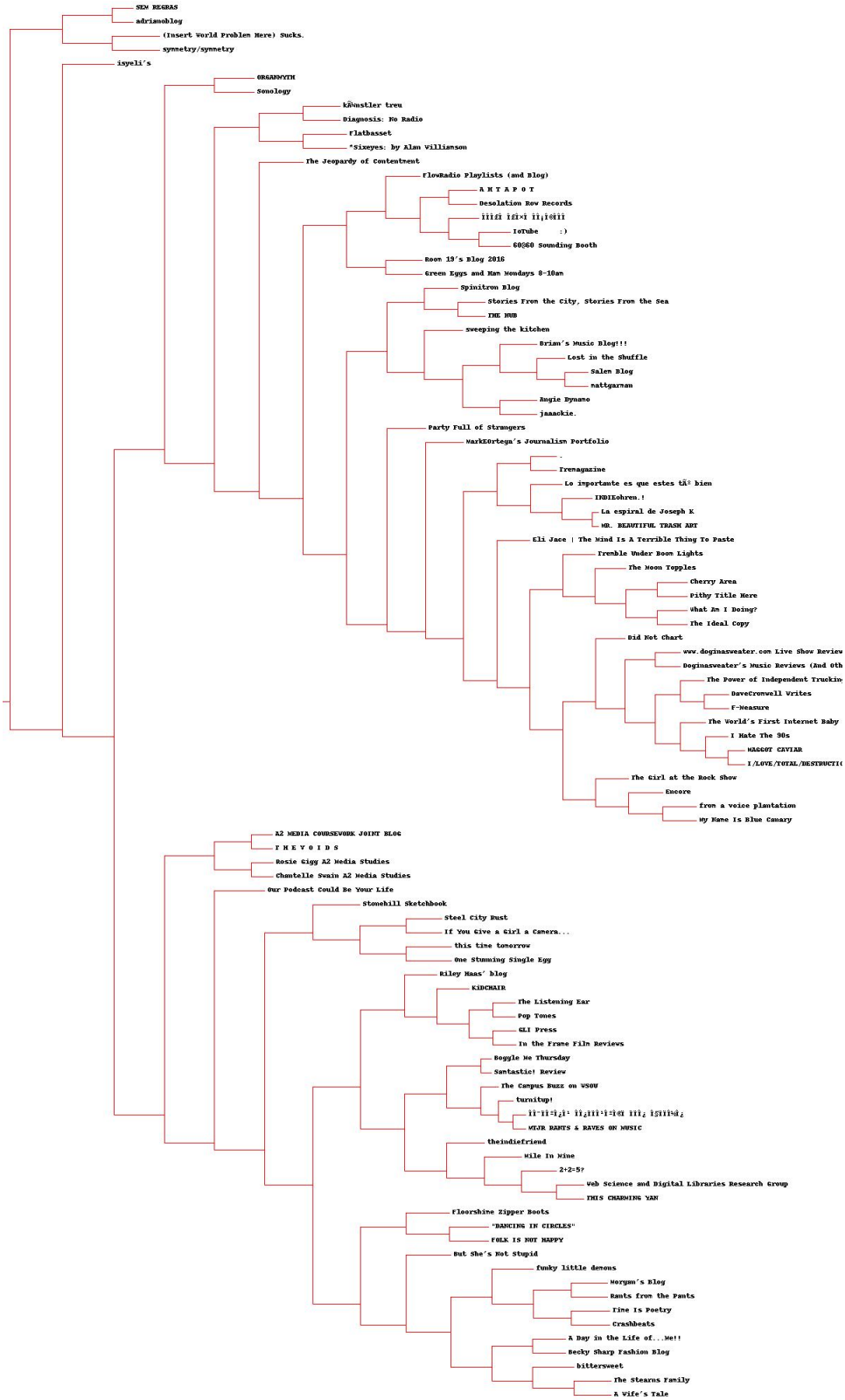
Listing 6: Using TFIDF: CalcTFIDF.py

```
17 def main():
18
19     n = 500
20     m = 100
21     matrix = []
22     blognames = []
23     matrix_total = [0] * n
24     matrix_TDIF = [[0] * n for i in range(m)]
25     total_doc_corpus = 0
26     counter = 0
27     with open('blogdata.txt', 'r') as infile:
28         for line in infile:
29             terms = line.strip().split('\t')
30             if counter > 0:
31                 row = []
32                 row.append(terms[0])
33                 blognames.append(terms[0])
34                 i = 0
35                 for size in terms[1:]:
36                     row.append(size)
37                     matrix_total[i] += int(size)
38                     total_doc_corpus += int(size)
39                     i += 1
40                 matrix.append(row)
41             counter += 1
42
43     total_doc_corpus                    # total docs in corpus
44     for i in range(m):
45         for k in range(n):
46             matrix_TDIF[i][k] = math.log(total_doc_corpus / matrix_total[k], 2) *
                   int(matrix[i][k + 1])
47
```

```
48      clust=clusters.hcluster(matrix_TDIF)
49      outfile = open('ascii-dendogramP5.txt', 'w')
50      clusters.printclust(clust, labels=blognames, file=outfile)
51      outfile.close()
52      clusters.drawdendrogram(clust,blognames,jpeg='blogclustP2.jpg')
53
54      return
```

Lines 44 through 47 iterates through the matrix to calculate the TFIDF for each blog term. The remaining of the code is a copy of problem 2.

5.2 Solution



The result is very interesting. A great deal of the blogs are clustering with similarity, but they seem to be following on a different categories. For example, blogs F-Measure, DaveCrownell Writes, I hate the 90's,

The girl at the Rock Show in problem 2 are clustering together in a flatter way while using TFIDF makes the relationship much hierarchical.

References

[1] Segarn, Toby. Programming Collective Intelligence. *Building Smart Web 2.0 Application*. (pp 29-53). Sebastopol, CA: O'Reilly Media.