INTRODUCTION TO WEB SCIENCES:
Assignment 8

Manoj Chandra Kompalli

7 April 2016

Contents

1 Question 1: 2
- 1.1 Approach . . . 2
- 1.2 Code Listing . . . 3
  - 1.2.1 getBlogs.py to generate 100 URIs . . . 3
  - 1.2.2 countPages.py to find the number of pages in each blog . . . 4
  - 1.2.3 matrixcreate.py to create a blog term matrix . . . 5
- 1.3 Blog term matrix shown in Excel . . . 7

2 Question 2: 8
- 2.1 Approach . . . 8
- 2.2 Code Listing . . . 8
  - 2.2.1 dgram.py . . . 8
- 2.3 Output . . . 9
  - 2.3.1 JPEG Dendogram . . . 9
  - 2.3.2 ASCII dendogram . . . 10

3 Question 3: 14
- 3.1 Approach . . . 14
- 3.2 Code Listing . . . 14
  - 3.2.1 convert.py . . . 14
- 3.3 Output . . . 15
  - 3.3.1 Centroid Values and Iterations . . . 15

4 Question 4: 16
- 4.1 Approach . . . 16
- 4.2 Code Listing . . . 16
- 4.3 Output . . . 17
  - 4.3.1 JPEG of blogs using MPS . . . 17

5 Question 5: 18
- 5.1 Approach . . . 18
- 5.2 Code Listing . . . 18
- 5.3 Output . . . 22
  - 5.3.1 Output Blog term matrix with tfidf values . . . 22
  - 5.3.2 Output JPEG . . . 23

1 Question 1:

1.1 Approach

- I have extracted the 98 URIs initially and appended the required string
- I made sure there are no duplicates
- I have added the given 2 URIs in the question to make it 100
- I used feed parser to parse all the randomly generated URIs
- To get the most popular words I kept track of occurrence of each word in the blog
- I sorted all the words based on how many times they occurred and selected the top 500 words.
- Then I calculated the occurrence of every word in the respective blog
- The resulting matrix contains 501 columns and 101 rows

1.2 Code Listing

1.2.1 getBlogs.py to generate 100 URIs

```
import os
import requests
import feedparser
import re
import sys

def getBlogs():
  # link='http://www.blogger.com/next-blog?navBar=true&blogID=3471633091411211117'
  # cmd="curl -I -L "+""+link+""
  link='http://www.blogger.com/next-blog?navBar=true&blogID=3471633091411211117'


  durls=set()

  while len(durls)<98:
    req=requests.get(link)
    durls.add(req.url)
    # link=req.url
  # print durls
  # file1=open('urls','w')
  file2=open('urlsappended','w')
  # for item in durls:
    # #print item + '\n'
    # file1.write(item + '\n')
  file2.write('http://f-measure.blogspot.com'+'/feeds/posts/default?alt=rss' +'\n')
  file2.write('http://ws-dl.blogspot.com/'+'/feeds/posts/default?alt=rss' +'\n')


  for item in durls:
    #print item + '\n'
    file2.write(item.strip('/?expref=next-blog')+'/feeds/posts/default?alt=rss' +'\n')







if __name__== '__main__':
  getBlogs()
```

1.2.2 countPages.py to find the number of pages in each blog

```
#! /usr/bin/env python
import os
import sys
import urllib
import time
import feedparser

from bs4 import BeautifulSoup

def checkNextPage(url):

    f = urllib.urlopen(url)
    soup = BeautifulSoup(f.read(), from_encoding=f.info().getparam('charset'))

    try:
        link = soup.find('link', rel='next',href = True)['href']
    except TypeError:
        link = None
    return link


def main():
    feedlist   = open('urlsappended').readlines()

    for url in feedlist:
        try:
            d = feedparser.parse(url)

            title = d['feed']['title']

            count    = 1
            nextLink = checkNextPage( url )

            while nextLink:
                nextLink = checkNextPage( nextLink )
                count += 1

            print u'|'.join((str(count),title)).encode('utf-8').strip()



        except KeyError:
            pass

if __name__ == '__main__':
    main()
```

1.2.3 matrixcreate.py to create a blog term matrix

```
#!/usr/bin/env python

import re
import sys
import feedparser
if __name__ == "__main__":

      matrixCreation()
def matrixCreation():
  nowords={}
  cap={}

  feedlist=[line for line in file('urlsappended')]
  for feedurl in feedlist:
    try:
      title,wc=getnowords(feedurl)
      nowords[title]=wc
      for word,count in wc.items():
        cap.setdefault(word,0)
        if count>1:
          cap[word]+=1
    except:
      print 'Parsing failed %s' % feedurl

  wordlist=[]
  repeatingWords = []
  for w,bc in cap.items():
    frac=float(bc)/len(feedlist)
    if frac>0.1 and frac<0.5:
      repeatingWords.append((w,bc))

  repeatingWords=sorted(repeatingWords,key=lambda x:x[1], reverse = True)

  for value in repeatingWords:

    value1 = value[0]

    value2 = value[1]
    length = len(wordlist)
    if(length < 500):
      wordlist.append(value1)
    else:
      break

  out=file('blogdata.txt','w')
  out.write('Blog')

  for word in wordlist:
    word1 = word.encode('UTF-8')
    out.write('\t%s' % word1)
  out.write('\n')

  for blog,wc in nowords.items():
    blogName = blog.encode('UTF-8')
    print blog
```

```
    out.write(blogName)
    for word in wordlist:
      if word in wc: out.write('\t%d' % wc[word])
      else: out.write('\t0')
    out.write('\n')
def getwords(html):

  txt=re.compile(r'<[^>]+>').sub('',html)


  words=re.compile(r'[^A-Z^a-z]+').split(txt)

  return [word.lower() for word in words if word!='']
def getnowords(url):

  d=feedparser.parse(url)
  wc={}


  for e in d.entries:
    if 'summary' in e: summary=e.summary
    else: summary=e.description


    words=getwords(e.title+' '+summary)
    for word in words:
      wc.setdefault(word,0)
      wc[word]+=1
  return d.feed.title,wc
```

1.3 Blog term matrix shown in Excel

	A	B	C	D	E	F	G	H	I	J	K	L	
1	Blog	yet	week	real	heart	both	him	times	once	such	did	without	doe
2	Flatbasset	14	38	7	3	13	22	1	3	1	7	2	
3	Riley Haas' blog	6	0	1	2	9	5	9	7	3	12	4	
4	(Insert World Problem Here) Sucks.	0	0	0	0	0	0	0	0	1	2	1	
5	SEM REGRAS	0	0	0	7	0	0	0	0	0	0	0	
6	Pithy Title Here	12	8	26	2	12	10	13	15	12	32	15	
7	Morgan's Blog	3	0	2	1	5	6	0	1	4	4	1	
8	MARISOL	0	0	0	0	0	0	0	0	0	0	0	
9	THE HUB	2	4	0	0	0	0	1	0	2	1	0	
10	Brian's Music Blog!!!	0	11	6	1	2	3	6	0	3	5	2	
11	Web Science and Digital Libraries Research Group	6	3	4	0	14	4	8	6	61	15	9	
12	Steel City Rust	3	2	7	1	3	7	2	3	10	4	8	
13	MR. BEAUTIFUL TRASH ART	0	0	0	0	0	0	0	0	0	0	0	
14	60@60 Sounding Booth	3	0	4	8	0	1	1	2	0	0	1	
15	ORGANMYTH	0	0	0	0	0	0	0	0	0	0	0	
16	MarkEOrtega's Journalism Portfolio	0	5	0	0	2	0	0	2	0	0	0	
17	Green Eggs and Ham Mondays 8-10am	0	0	2	4	0	0	1	0	0	0	1	
18	GLI Press	0	19	0	2	0	1	0	0	0	0	0	
19	turnitup!	2	1	6	6	3	1	6	4	4	2	5	
20	Stories From the City, Stories From the Sea	0	2	2	5	2	1	2	3	0	0	2	
21	A H T A P O T	0	0	1	0	0	0	0	0	0	0	0	
22	Floorshime Zipper Boots	3	0	1	0	2	0	4	0	2	0	0	
23	Did Not Chart	1	8	1	3	3	5	3	6	1	6	6	
24	How to be an artist and still pass for normal	0	0	0	0	0	0	0	0	1	0	0	
25	Party Full of Strangers	0	5	4	1	2	2	2	1	0	1	0	

Figure 1: JSON array containing the follower data

2 Question 2:

2.1 Approach

- I have used the clusters.py program mentioned in the presentation to produce the required dendogram
- Printclust prints the dendogram
- drawdendogram saves it in JPEG format
- There are numerous blogs similar to F-measure. The top one is the power of independent study
- The blog similar to web science and Digital libraries Research group is Ideal copy

2.2 Code Listing

2.2.1 dgram.py

```
import clusters
import sys

blognames,words,data=clusters.readfile('blogmatrix.txt')
clust = clusters.hcluster(data)

# print ASCII dendrogram
clusters.printclust(clust, labels=blognames)
sys.stdout = open('ascii.txt', 'w')

# save JPEG dendrogram
clusters.drawdendrogram(clust, blognames, jpeg='dendogram.jpg')
```

2.3 Output

2.3.1 JPEG Dendogram

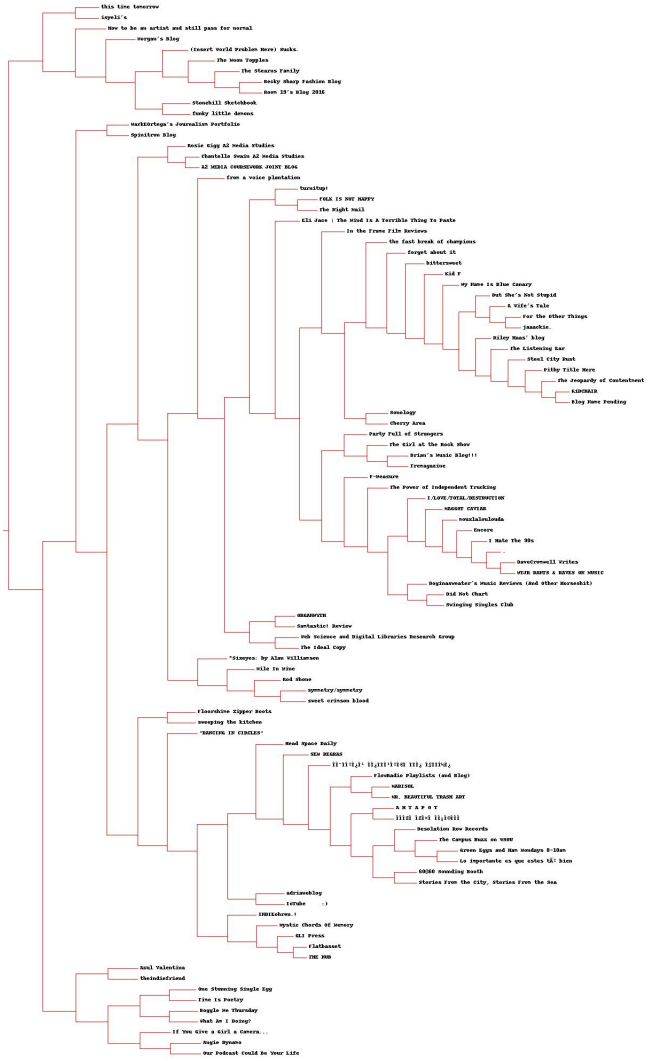

Figure 2: Initial graph split into 4 groups

2.3.2 ASCII dendogram

```
-
  -
    -
      this time tomorrow
      isyeli's
    -
      How to be an artist and still pass for normal
      -
        Morgan's Blog
        -
          -
            (Insert World Problem Here) Sucks.
            -
              The Moon Topples
              -
                The Stearns Family
                -
                  Becky Sharp Fashion Blog
                  Room 19's Blog 2016
          -
            Stonehill Sketchbook
            funky little demons
-
  -
    -
      MarkEOrtega's Journalism Portfolio
      Spinitron Blog
    -
      -
        -
          Rosie Gigg A2 Media Studies
          -
            Chantelle Swain A2 Media Studies
            A2 MEDIA COURSEWORK JOINT BLOG
        -
          -
            from a voice plantation
            -
              -
                -
                  turnitup!
                  -
                    FOLK IS NOT HAPPY
                    The Night Mail
                -
                  Eli Jace | The Mind Is A Terrible Thing To Paste
                  -
                    -
                      In the Frame Film Reviews
                      -
                        -
                          the fast break of champions
                          -
                            forget about it
                            -
```

```
                                bittersweet
                                –
                                  Kid F
                                  –
                                    My Name Is Blue Canary
                                    –
                                      –
                                        But She's Not Stupid
                                        –
                                          A Wife's Tale
                                          –
                                            For the Other Things
                                            jaaackie.
                                      –
                                        Riley Haas' blog
                                        –
                                          The Listening Ear
                                          –
                                            Steel City Rust
                                            –
                                              Pithy Title Here
                                              –
                                                The Jeopardy of Contentment
                                                –
                                                  KiDCHAIR
                                                  Blog Name Pending
                          –
                            Sonology
                            Cherry Area
                –
                  –
                    Party Full of Strangers
                    –
                      The Girl at the Rock Show
                      –
                        Brian's Music Blog!!!
                        Tremagazine
                  –
                    F–Measure
                    –
                      The Power of Independent Trucking
                      –
                        –
                          I/LOVE/TOTAL/DESTRUCTION
                          –
                            MAGGOT CAVIAR
                            –
                              mouxlaloulouda
                              –
                                Encore
                                –
                                  I Hate The 90s
                                  –
                                    .
                                    –
                                      DaveCromwell Writes
                                      MTJR RANTS & RAVES ON MUSIC
```

```
                                -
                                  Doginasweater's Music Reviews (And Other Horseshit)
                                  -
                                    Did Not Chart
                                    Swinging Singles Club
                    -
                      -
                        ORGANMYTH
                        Samtastic! Review
                      -
                        Web Science and Digital Libraries Research Group
                        The Ideal Copy
                -
                  *Sixeyes: by Alan Williamson
                  -
                    Mile In Mine
                    -
                      Rod Shone
                      -
                        symmetry/symmetry
                        sweet crimson blood
        -
          -
            Floorshime Zipper Boots
            sweeping the kitchen
          -
            "DANCING IN CIRCLES"
            -
              -
                -
                  Head Space Daily
                  -
                    SEM REGRAS
                    -
                      -
                        -
                          FlowRadio Playlists (and Blog)
                          -
                            MARISOL
                            MR. BEAUTIFUL TRASH ART
                        -
                          -
                            A H T A P O T
                          -
                            -
                              Desolation Row Records
                              -
                                The Campus Buzz on WSOU
                                -
                                  Green Eggs and Ham Mondays 8-10am
                                  Lo importante es que estes  t   bien
                            -
                              60@60 Sounding Booth
```

```
                                Stories From the City, Stories From the Sea
              -
                adrianoblog
                IoTube      :)
            -
              INDIEohren.!
              -
                Mystic Chords Of Memory
                -
                  GLI Press
                  -
                    Flatbasset
                    THE HUB
-
  -
    Azul Valentina
    theindiefriend
  -
    -
      -
        One Stunning Single Egg
        Time Is Poetry
      -
        Boggle Me Thursday
        What Am I Doing?
    -
      If You Give a Girl a Camera...
      -
        Angie Dynamo
        Our Podcast Could Be Your Life
```

3 Question 3:

3.1 Approach

- Below code performs blog clustering based on clusters.py program using the method kcluster
- The program prints the iterations involved for each values of k (5,10,20).
- The program also print the values in each centroid for respective k values

3.2 Code Listing

3.2.1 convert.py

```
#!/usr/local/bin/python

import clusters

blognames,words,data=clusters.readfile('blogmatrix.txt')
print 'Iterations for varying k value '
print "For k=5"
kclust=clusters.kcluster(data, k=5)
print 'Centroid  values for first part are:'
for i in range(0,5):
  for item in kclust[i]:
    print blognames[item]
print

print "For k=10"
kclust=clusters.kcluster(data, k=10)
print 'Centroid  values for second part are:'
for i in range(0,10):
  for item in kclust[i]:
    print blognames[item]
print

print "For k=20"
kclust=clusters.kcluster(data, k=20)
print 'Centroid  values for third part are:'
for i in range(0,20):
  for item in kclust[i]:
    print blognames[item]
print
```

3.3 Output

3.3.1 Centroid Values and Iterations

```
For k=20
Iteration 0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Centroid  values for third part are:
The Night Mail
MR. BEAUTIFUL TRASH ART
ORGANMYTH
Samtastic! Review
from a voice plantation
Floorshime Zipper Boots
Did Not Chart
The Power of Independent Trucking
DaveCromwell Writes
FOLK IS NOT HAPPY
MAGGOT CAVIAR
MTJR RANTS & RAVES ON MUSIC
Doginasweater's Music Reviews (And Other Horseshit)
.
Blog Name Pending
Swinging Singles Club
mouxlaloulouda
I/LOVE/TOTAL/DESTRUCTION
Encore
I Hate The 90s
One Stunning Single Egg
this time tomorrow
Azul Valentina
Riley Haas' blog
(Insert World Problem Here) Sucks.
Pithy Title Here
Morgan's Blog
```

Figure 3: Centroid Values and Iterations

4 Question 4:

4.1 Approach

- Blog space is generated by the multi dimensional scaling using the code mds.py.
- Again we make use of clusters.py for the scaledown method.
- The output shows a graph which is circular unlike the previous question

4.2 Code Listing

```
#!/usr/local/bin/python

import clusters

blognames,words,data=clusters.readfile('blogmatrix.txt')

coords = clusters.scaledown(data)

clusters.draw2d(coords, blognames, jpeg='draw2d.jpg')
```

4.3 Output

4.3.1 JPEG of blogs using MPS

Figure 4: JPEG of blogs using MPS

5 Question 5:

5.1 Approach

- I wrote almost the same code except for the few lines which calculate the tfidf values
- Here I have ordered the matrix based on tfidf values of the respective words.
- I have extracted the words with high tfidf values
- I generated dendogram again just like how I did before.
- From the current dendogram F-Measure is similar to the blog Desolation Row Records
- WebScience and Digital Libraries Research Search group is similar to Maggot Caviar and blog I love total destruction
- We can observe that the hierarchy of the blogs has changed. Somehow, F-Measure now has a lot less similar blogs

5.2 Code Listing

```
import feedparser
import collections
import re
import operator
import math
def wordsRetrieve(html):
    text  = re.compile(r'<[^>]+>').sub('', html)
    words = re.compile(r'[^A-z^a-z]+').split(text)

    return [word.lower() for word in words if word]

input  = "urlsappended"
outpput = "blogmatrixnew.txt"

def countRetrieve(url):

    fd = feedparser.parse(url)
    wc        = collections.defaultdict(int)
    stopwords = []

    stopWordList = open('stopWordList.txt').readlines()
    pages = len(fd['entries'])

    for stopWord in stopWordList:
        stopWord = stopWord.strip()
        stopwords.append(stopWord)

    for e in fd.entries:
        if 'summary' in e:
            summary = e.summary
        else:
            summary = e.description
        words = wordsRetrieve('%s %s' % (e.title, summary))
```

```
34
35          for word in words:
36              if word not in stopwords:
37                  wc[word] += 1
38
39      if pages == 500:
40          next_link = url + "?start-index=501"
41          d         = feedparser.parse(next_link)
42          pages     = len(d['entries'])
43          for e in d.entries:
44              if 'summary' in e:
45                  summary = e.summary
46              else:
47                  summary = e.description
48
49              words = wordsRetrieve('%s %s' % (e.title, summary))
50
51              for word in words:
52                  if word not in stopwords:
53                      #print word
54
55                      wc[word] += 1
56
57          if pages == 500:
58              next_link = url + "?start-index=1001"
59              for e in d.entries:
60                  if 'summary' in e:
61                      summary = e.summary
62                  else:
63                      summary = e.description
64
65                  words = wordsRetrieve('%s %s' % (e.title, summary))
66
67                  for word in words:
68                      if word not in stopwords:
69                          #print word
70
71                          wc[word] += 1
72              if pages == 500:
73                  next_link = url + "?start-index=1501"
74                  for e in d.entries:
75                      if 'summary' in e:
76                          summary = e.summary
77                      else:
78                          summary = e.description
79
80                      words = wordsRetrieve('%s %s' % (e.title, summary))
81
82                      for word in words:
83                          if word not in stopwords:
84                              wc[word] += 1
85
86      if 'title' not in fd.feed:
87          print 'Invalid url', url
88          return 'bogus data', wc
89
90      return fd.feed.title, wc
```

```
91
92
93  def matrixCreation():
94
95
96
97      apcount    = collections.defaultdict(int)
98      wordcounts = {}
99      feedlist   = open( input ).readlines()
100     totalWordCount = {}
101
102     for url in feedlist:
103         title, wc = countRetrieve(url)
104         wordcounts[title] = wc
105
106         for word, count in wc.iteritems():
107             if count > 1:
108                 apcount[word] += 1
109
110                 try:
111                     totalWordCount[word] += count
112                 except KeyError:
113                     totalWordCount[word] = count
114
115     wordlist = []
116
117
118     for w, bc in apcount.iteritems():
119         frac = float(bc)/len(feedlist)
120         #print frac
121         if frac > 0.1 and frac < 0.5:
122             wordlist.append(w)
123
124     countOfWords = []
125
126     for word in wordlist:
127         countOfWords.append((word,totalWordCount[word]))
128
129     countOfWords.sort(key=lambda rating: rating[1], reverse = True )
130
131     countOfWords = countOfWords[0:500]
132
133     out = file(outpput, 'w')
134     out.write('Blog')
135
136     idfWordCount = {}
137
138     for w in countOfWords:
139         word = w[0]
140         noOfBlogs = 0
141         for blogname, counts in wordcounts.iteritems():
142
143             if word in counts:
144                 noOfBlogs += 1
145
146         idf = math.log( 100.0 / noOfBlogs   , 2 )
147
```

```
148
149         idfWordCount[word]  = idf
150
151
152     for w in countOfWords:
153
154         out.write('\t' + w[0])
155
156     out.write('\n')
157
158     for blogname, counts in wordcounts.iteritems():
159         blogname = blogname.encode('UTF-8')
160         out.write(blogname)
161
162         for w in countOfWords:
163             word = w[0]
164             occurance = w[1]
165
166             tf = float(counts[word]) / occurance
167             tfidf = tf * idfWordCount[word]
168
169
170             out.write('\t%f' % tfidf)
171
172         out.write('\n')
173
174     out.close()
175
176 if __name__ == '__main__':
177     matrixCreation()
```

5.3 Output

5.3.1 Output Blog term matrix with tfidf values

	A	B	C	D	E	F	G	H	I	J	K	L
1	Blog	vocal	track	mso	pop	guitar	musical	http	review	style	voice	set
2	Flatbasset	0.003204	0.011198	0	0.02513	0.007064	0	0	0.002698	0.006353	0.001713	0.005788
3	Riley Haas	0.003204	0.005168	0	0.001047	0	0	0	0.010791	0.007941	0.001713	0.002894
4	(Insert Wo	0	0	0	0	0	0	0	0	0	0	0
5	SEM REGR	0	0	0	0.003141	0	0	0	0	0	0	0
6	Pithy Title	0	0.023258	0.057151	0.034554	0.018365	0.009801	0	0.002698	0.019058	0.015414	0.039068
7	Morgan's I	0.001602	0.001723	0	0	0.002825	0.00196	0	0	0	0	0.002894
8	MARISOL	0	0	0	0	0	0	0	0	0	0	0
9	THE HUB	0	0	0.078207	0.002094	0	0	0	0	0.015882	0	0
10	Brian's Mu	0	0.005168	0	0.012565	0.002825	0.003921	0.021079	0	0.003176	0.008563	0.001447
11	Web Scier	0	0.006891	0	0	0	0	0.274021	0.018884	0.007941	0	0.023152
12	Steel City	0.003204	0.009475	0	0.010471	0.007064	0.005881	0	0.005395	0.003176	0.005138	0.001447
13	MR. BEAU	0	0	0	0.006282	0	0	0.003513	0	0	0	0
14	60@60 So	0	0	0	0	0.001413	0	0	0	0.001588	0	0.010129
15	ORGANM	0	0	0	0	0	0	0	0	0.001588	0	0
16	MarkEOrte	0.001602	0.000861	0	0.010471	0	0	0	0	0	0	0.008682
17	Green Egg	0	0.002584	0.042111	0.001047	0	0	0	0.002698	0.007941	0	0
18	GLI Press	0	0.004307	0	0	0	0	0	0	0	0.001713	0
19	turnitup!	0.003204	0.005168	0	0.0178	0.007064	0.021563	0	0	0.004765	0.001713	0.001447
20	Stories Fr	0	0.000861	0	0.009424	0	0	0.024592	0	0	0	0.011576
21	A H T A P (	0	0	0	0	0.001413	0	0	0	0	0	0
22	Floorshim	0.001602	0.006891	0	0.005235	0.002825	0.003921	0.003513	0	0.001588	0	0
23	Did Not Cl	0.004807	0.001723	0	0.021989	0.009889	0.00196	0	0	0.007941	0.001713	0.004341
24	How to be	0	0	0	0	0	0	0	0	0	0	0
25	Party Full	0.001602	0.009475	0	0.00733	0.001413	0.003921	0	0	0	0.005138	0.001447

Figure 5: Output Blog term matrix with tfidf values

5.3.2 Output JPEG

Figure 6: Output generated by mds.py

References

[1] Feed Parser https://github.com/Unode/python-feedparser/tree/master/feedparser,

[2] Finding TFIDF value http://en.wikipedia.org/wiki/Tfidf,

[3] Python for beginners http://www.pythonforbeginners.com/feedparser/using-feedparser-in-python

[]