|  |
| --- |
|  |
| CS432 Spring 2018 |
| Assignment 7 |
|  |
| **Jonathan Kruszewski** |
| **28/04/2018** |

|  |
| --- |
|  |

CS 432/532 Web Science

Spring 2018

http://anwala.github.io/lectures/cs532-s18/

Assignment #7

Due: 11:59pm March 31

(10 points; 2 points for each question and 2 points for aesthetics)

Support your answer: include all relevant discussion, assumptions,

examples, etc.

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.

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).

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?

4. Use MDS to create a JPEG of the blogs similar to slide 29 of the

week 11 lecture. How many iterations were required?

**Part 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.

To accomplish this task I used a bash file to issue the curl command found in file “CurlBlogs.sh”, the curl command was repeated 498 times and saves the each result into an individual “curlFile[n].txt” into the folder “CurlFiles” and was then used on the two urls specified in the instructions.

filename='CurlFiles/curlFile'${i}'.txt'

curl -L -o /dev/null -w %{url\_effective} 'http://www.blogger.com/next-blog?navBar=true&blogID=3471633091411211117' > ${filename}

After this step “GetUrl.py” is run which reads each curlFile from the previous step and removes “/?expref=next-blog” then adds “/feeds/posts/default” to the end of each. Two txt files are then output, the first file: “unmodifiedURLS.txt” contains a complete list of blog urls, the second file: “feedlist.txt” contains the same list of urls with all duplicates removed.

urls = []for x in inF: tempStr = '' tempStr = x.rstrip('/?expref=next-blog') tempStr = tempStr + '/feeds/posts/default' urls.append(tempStr) print(tempStr)

After this step “generatefeedvector.py” from <https://github.com/arthur-e/Programming-Collective-Intelligence/tree/master/chapter3> is run with a small modification. The modified version functions the same as the version provided except that it outputs a file with a list of any feeds that failed to parse.

The blog data can be found in “blogdata1.txt” or the blogdata1 excel sheets in the containing folder.

**Part 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).

This task was accomplished by using “Dend.py” which used a slightly modified version of the “cluster.py” python code provided at <https://github.com/arthur-e/Programming-Collective-Intelligence/tree/master/chapter3> .“cluster.py” was modified by changing the printclust funtion to make it output a file “cluserASCII.txt” with the ASCII dendrogram. “Dend.py” calls the functions in “cluster.py” to get both the ASCII and JPEG dendrograms. The JPEG can be found at the end of this report.

**Part 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?

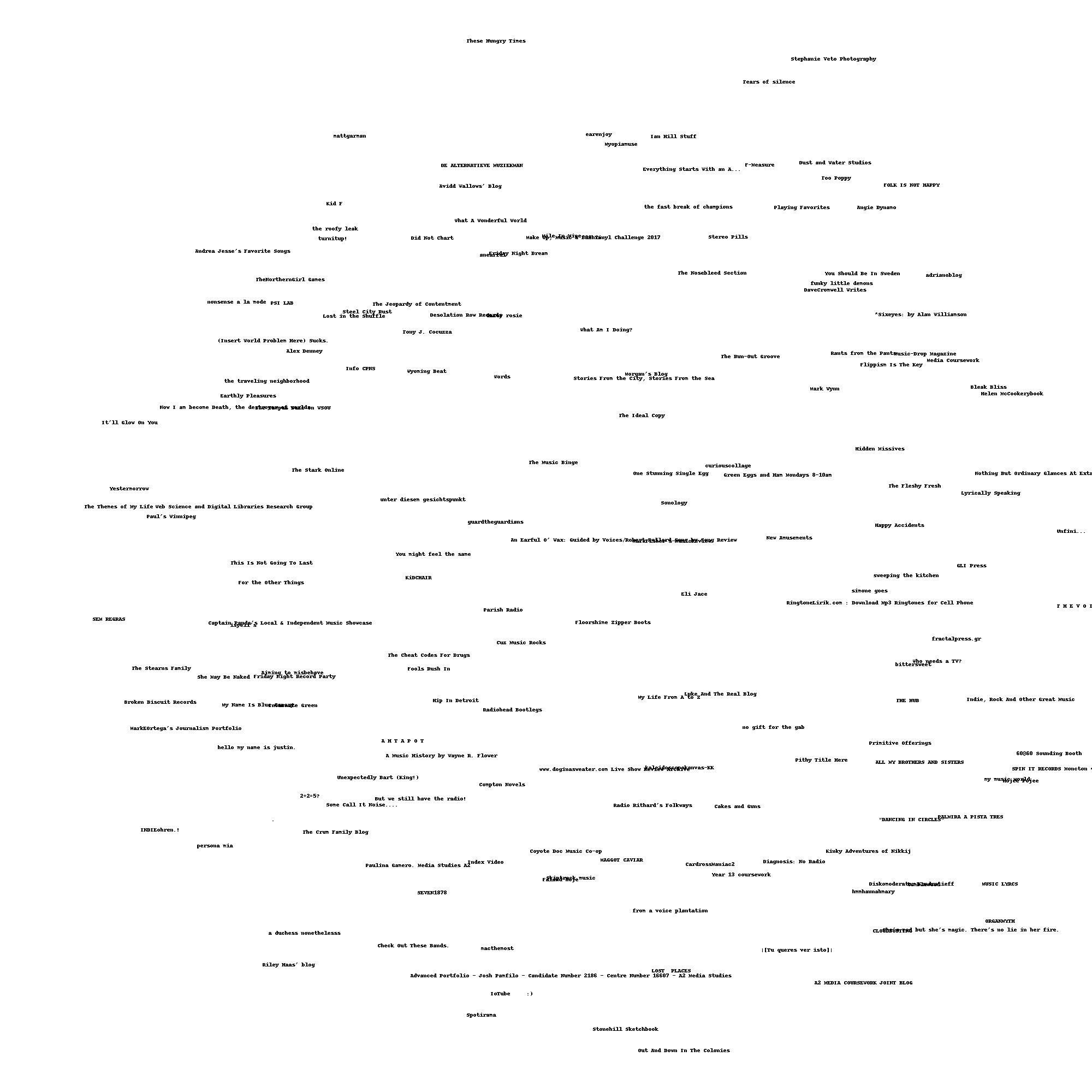
To accomish these tasks a python program “kmod.py” was used along with “cluster.py”. “kmod.py” called the kcluster function from “cluster.py” for k = 5, 10, 20. “cluster.py” was modified for this task to output to a file “kcluster[k].txt” after each iteration after which “kmod.py” would output the centroid for each to its corresponding file. K = 5 took 9 iterations, K = 10 took 8 iterations, and K = 20 took 7 iterations.

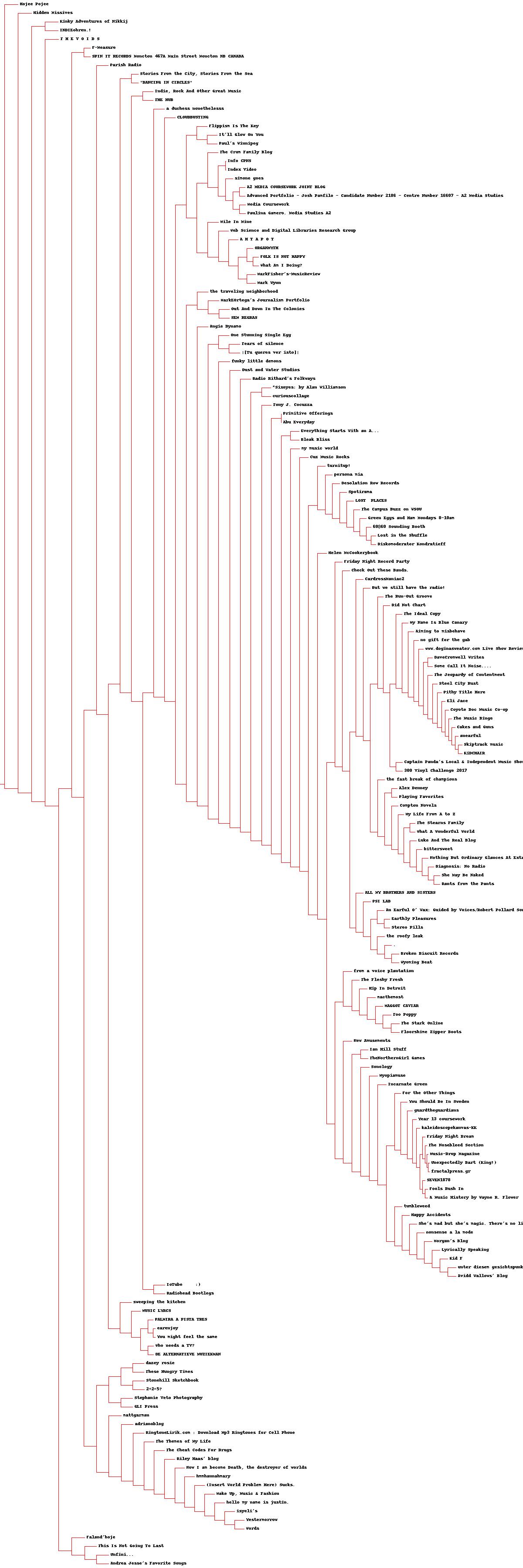
**Part 4**

4. Use MDS to create a JPEG of the blogs similar to slide 29 of the

week 11 lecture. How many iterations were required?

To accomish these tasks a python program “mds.py” was used along with “cluster.py”. It took 2 iterations. The mds image can be viewed below or found sepretly as “mds.jpg”



Dendrogram: “blogClusetr.jpg”