**Assignment Eight**

Jacob Berlin

CS432 – Spring 2016

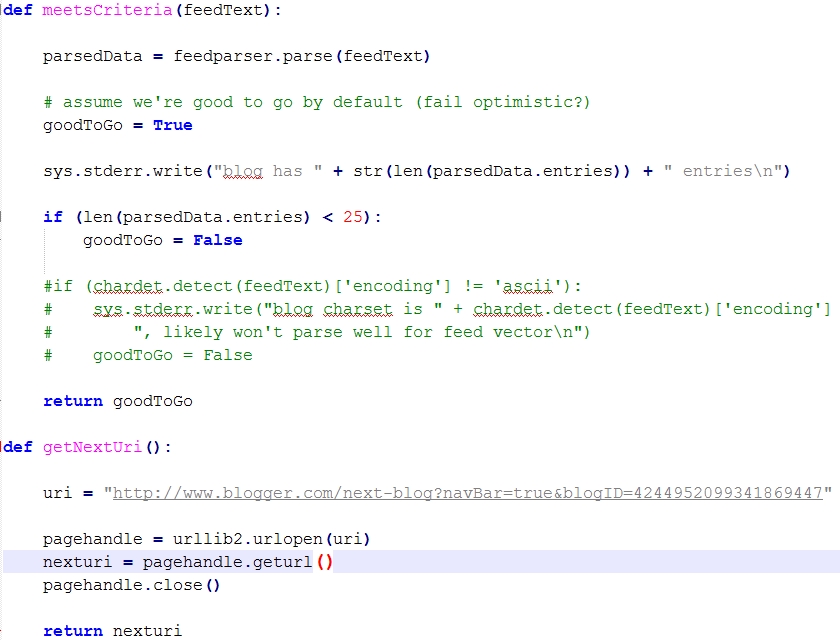
1. Create a blog-term matrix. Start by grabbing 100 blogs; include:

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

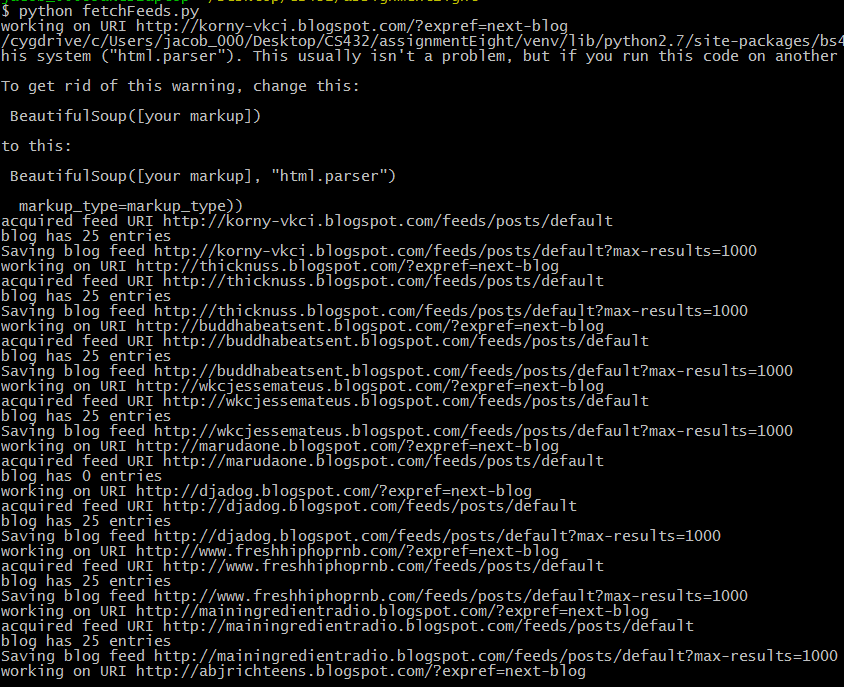
http://ws-dl.blogspot.com/

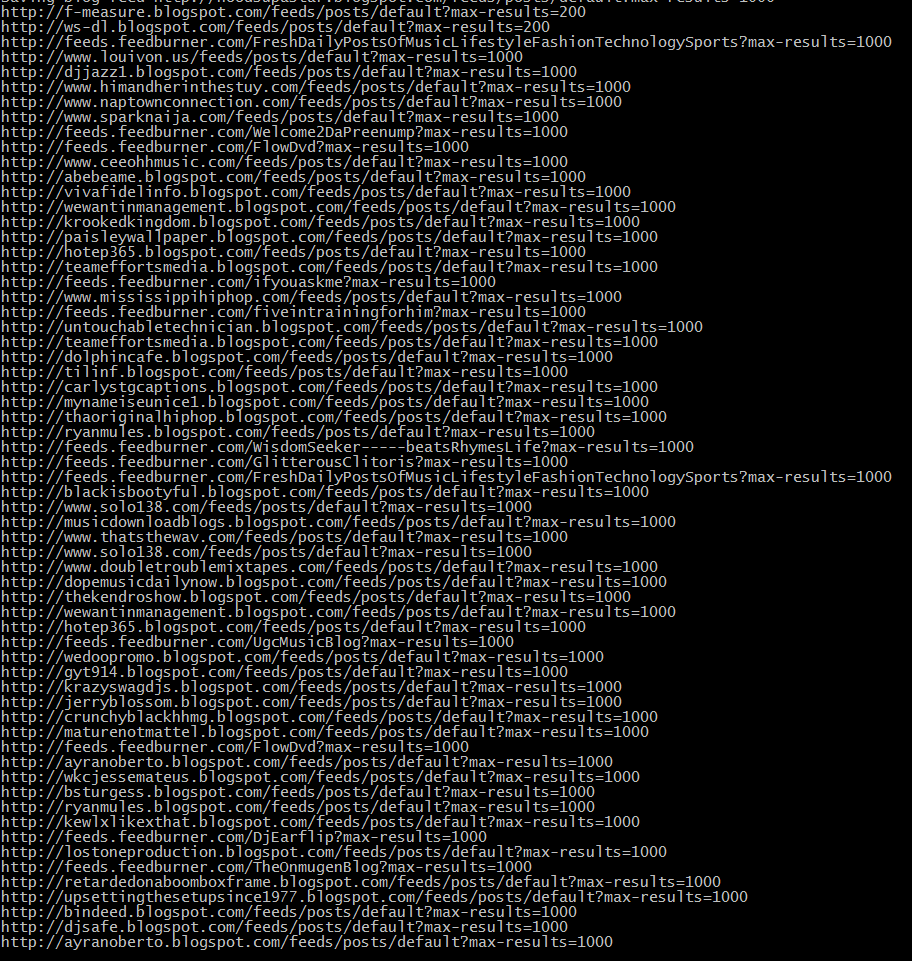
and grab 98 more as per the method shown in class. Note that this method randomly chooses blogs and each student will separately do this process, so it is unlikely that these 98 blogs will be shared among students. In other words, no sharing of blog data. Upload to github your code for grabbing the blogs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

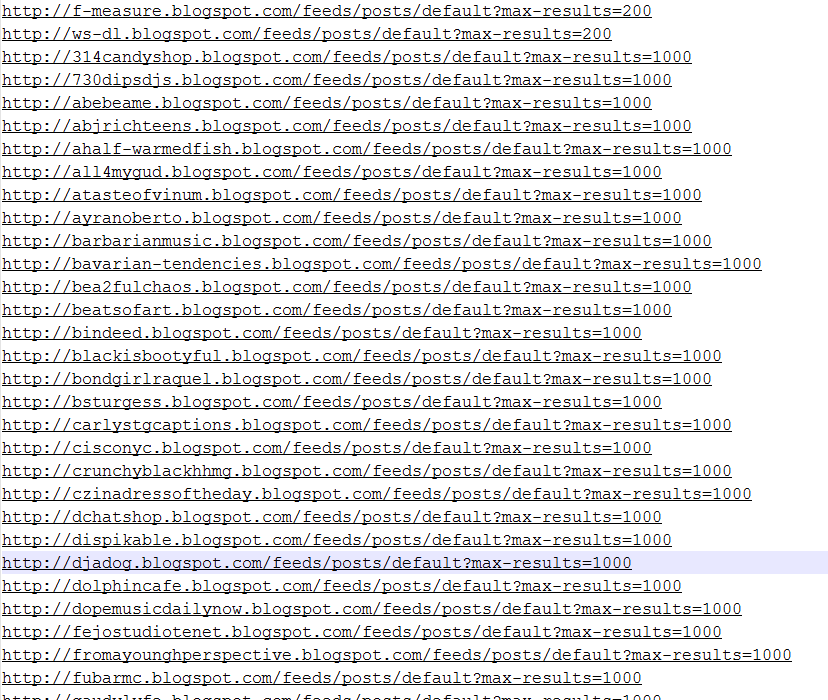


To start this problem, I used the file ‘fetchFeeds.py’ made by Shawn M. Jones which would take a particular blog ID (found in the html of the blog page) and would use the URI “http://www.blogger.com/next-blog?navBar=true&blogID=4244952099341869447” to list a random blog. This was achieved by using both the Urllib and BeautifulSoup libraries and with a small edit on my side posted all of the received blog’s URIs to ‘feedNames.txt’. I used two different starting blog IDs to get my list of 98 other blogs; ‘3395148574627097365’ and ‘4244952099341869447.’



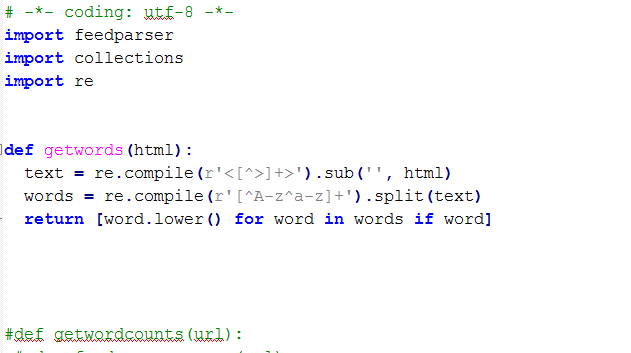


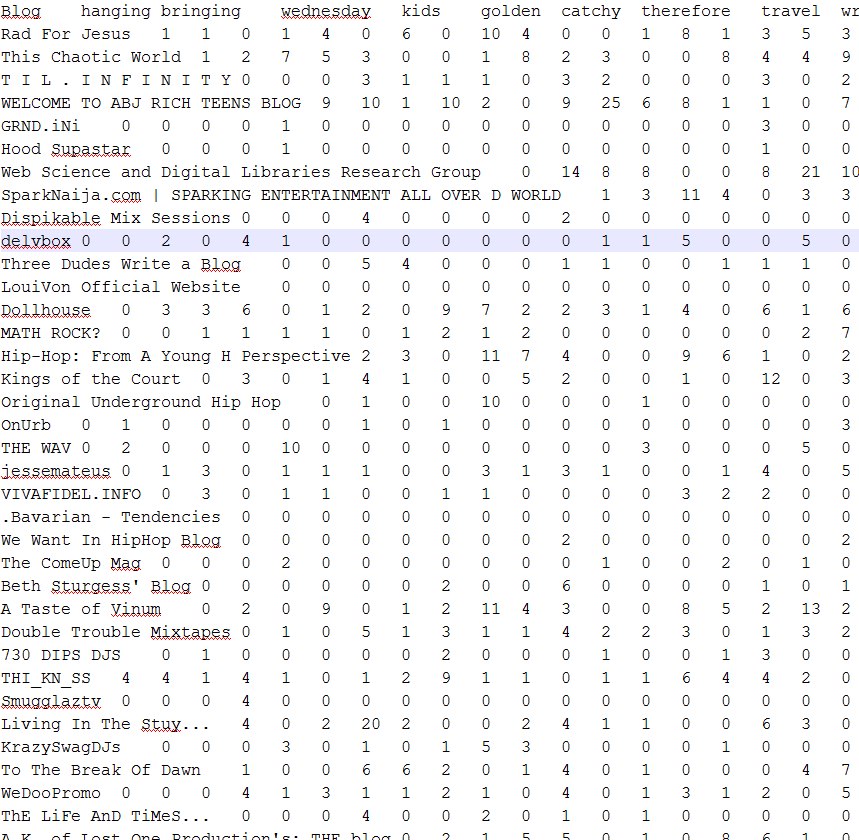
When running the program through the command prompt, I received my initial list of blogs that contained both duplicated and graphic blogs. I removed these by using the UNIX command ‘sort –u feedNames.txt’ and by manually going through to delete the provocative blogs. Each URI in this case has the max limit of pages set to 1,000 due to me not finding any file that might have more than that.



After the sorting and compilation I came out with the revised ‘feedNames.txt’ shown partially above.

For the second part of this question I used the file ‘generateFeed.py’ made by the creators of PCI, which took all of the blogs that I received earlier and created an index based off of the most popular words for each. This was limited to 500 words at most and ordered everything as required. The output was placed into ‘blogdata.txt’



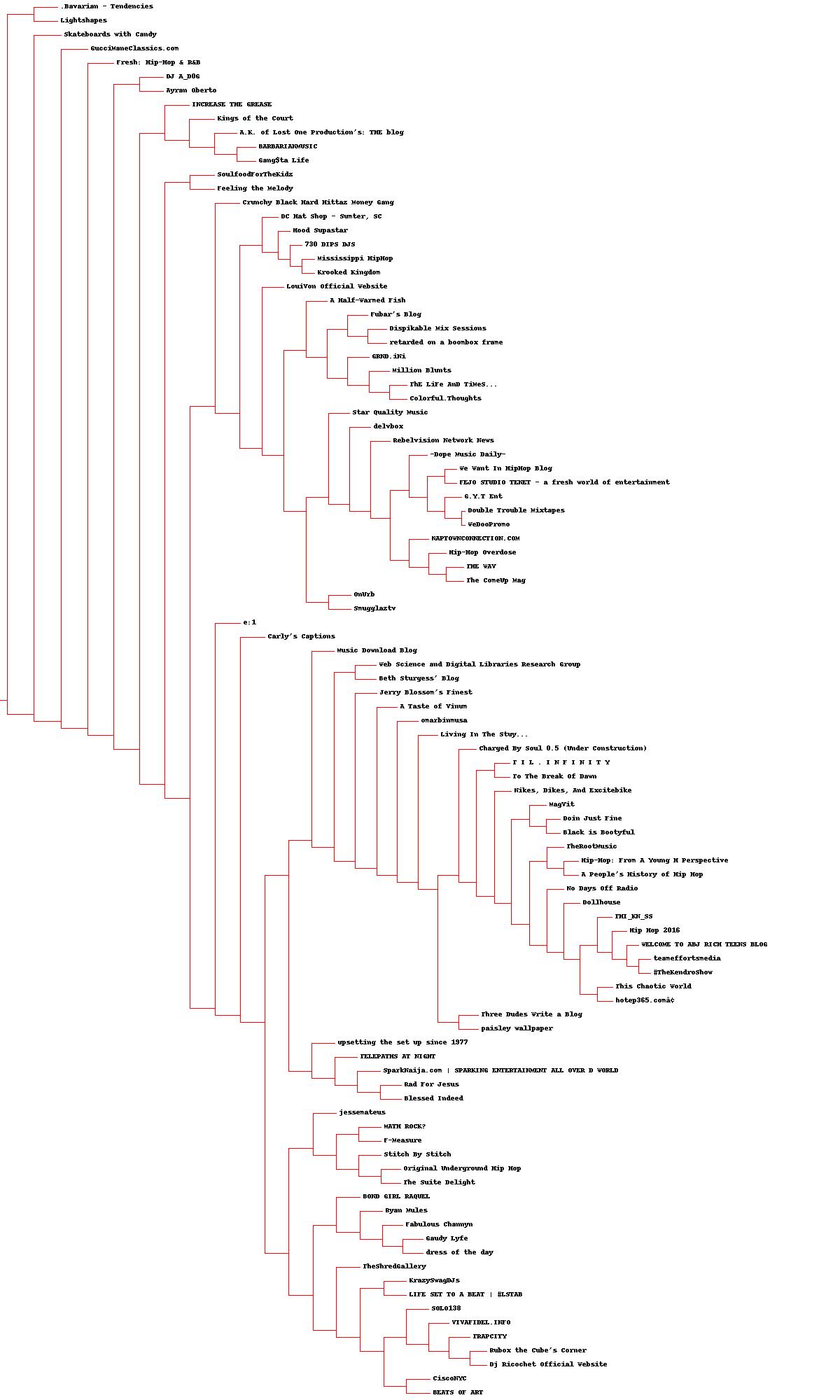


Shown above is a part of the data received from compiling ‘generateFeed.py’

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).

Using the file ‘clusters.py’ provided by PCI, I was able to use both the drawdendrogram and printclust functions to generate both a .jpg and ascii file based off of the data that I generated in the first question.

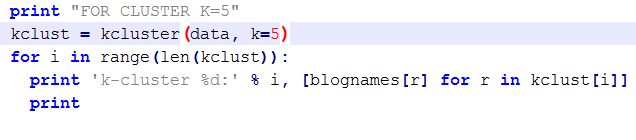
The ascii file is called ‘asciiDiagram.txt’ in github and the .jpg is called ‘blogclust.jpg’ (shown on the next page).

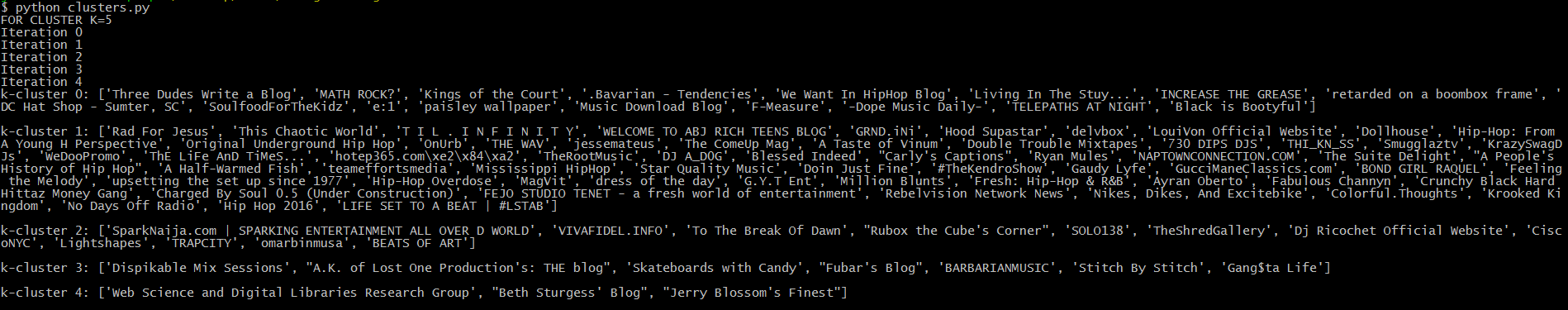


3. Cluster the blogs using K-Means, using k=5,10,20. (see slide 18). Print the values in each centroid, for each value of k. How many interations were required for each value of k?

For this question I used the data received in question 1 and the ‘clusters.py’ file (from PCI).

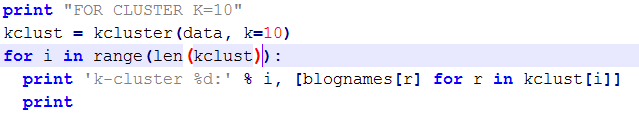
For k = 5: Computed in 5 total iterations.





Output shown in ‘ClusterFive.txt’

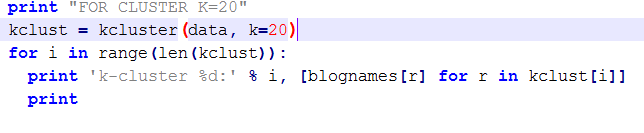
For k = 10: Completed in 5 total iterations.





Output shown in ‘kclustten.txt’

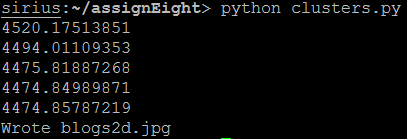
For k = 20: Completed in 7 total iterations.



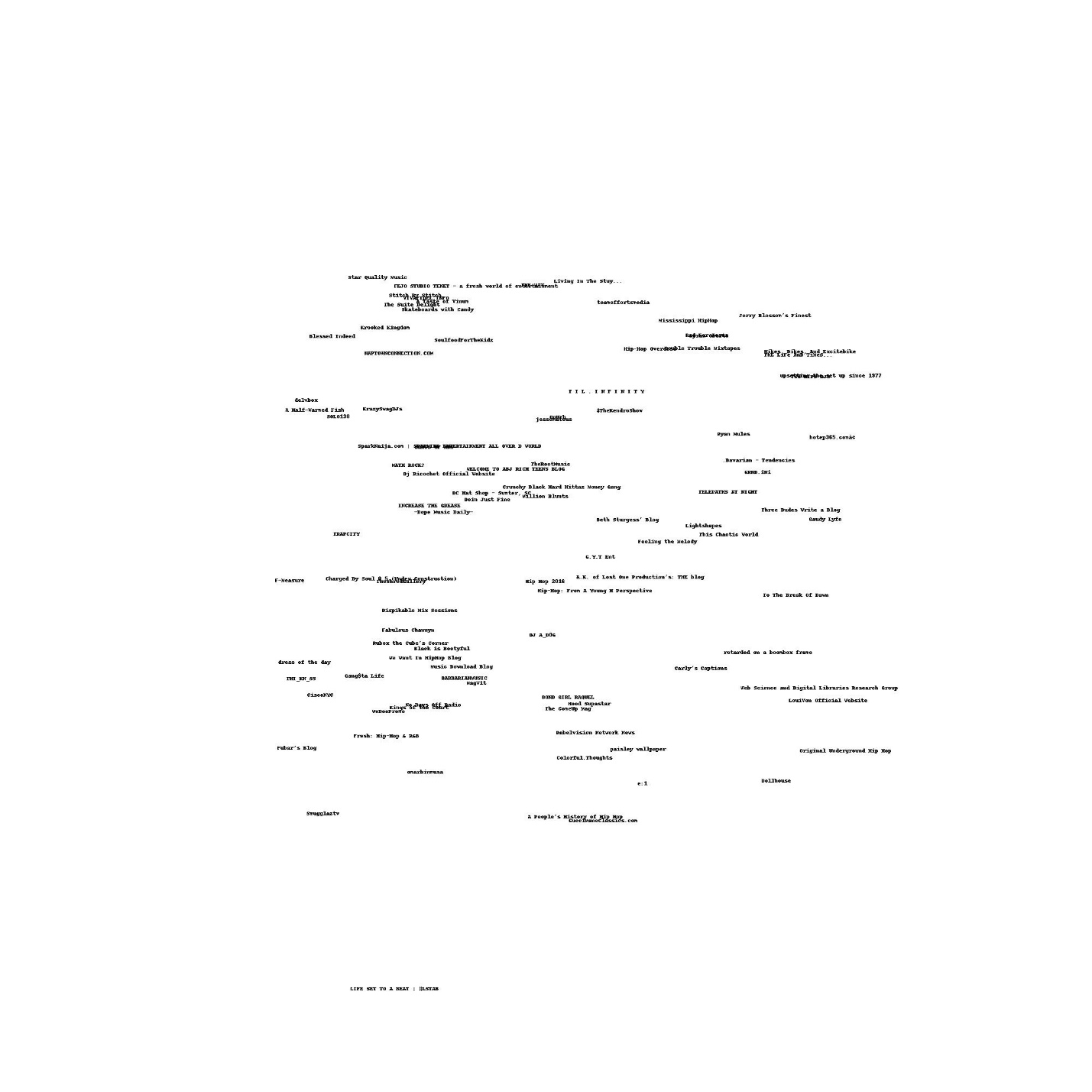


Output shown in ‘kclusttwenty.txt’

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



In ‘clusters.py’ (PCI) I used the ‘drawclust.draw2d’ function and the data from question 1 to create the 2d graph. Five total iterations were required in this compilation. The picture is called ‘blogs2d.jpg.’



References:

<https://github.com/shawnmjones/cs595-f13/blob/master/assignment9/q1/fetchFeeds.py>

https://github.com/nico/collectiveintelligence-book