Assignment 8

CS 432

Miranda SMITH

2017

**Question 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*

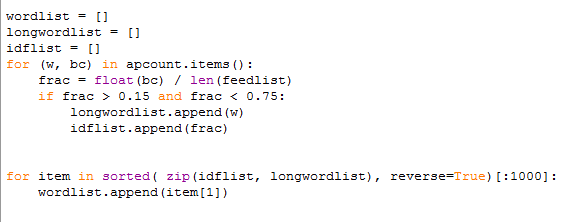
*matrix). Use the terms from every item/title (RSS) or entry/title (Atom) for the columns of the matrix. The values are the frequency of occurrence. Essentially you are replicating the format of the "blogdata.txt" file included with the PCI book code. Limit the*

*number of terms to the most "popular" (i.e., frequent) 1000 terms, this is \*after\* the criteria on p. 32 (slide 7) has been satisfied.*

I created a program ‘nextBlog.py’ to collect 100 blogs from blogspot.com, including the 2 required ones. It uses the URL base for the Next Blog button and attaches the current blog ID. If something is wrong, either the next blog doesn’t exist and gives you a Go Daddy page, or the url won’t open at all, it restarts from one of the beginning URLs. That works because that links gives a random blog, there is no order. All Blogs are listed below and in ‘100blogs.txt’. There are slightly more blogs than 100 because some are in another language and will not be parsed by the next program and is unable to be located by the blog title. However, that server does not return a language header, so the most convenient method was to edit it to append to the file and add a few more.

[Intentionally left blank]



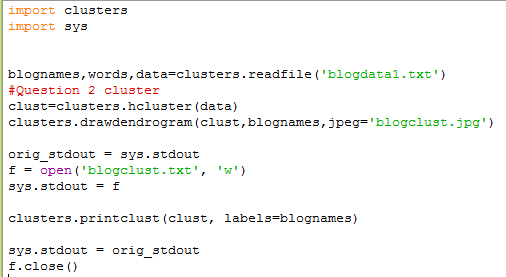
I used the program ‘generatefeedvector.py’ from the Programming Collective Intelligence book to create the matrix. I will not put the whole code into the report because of its length, however it will be in the same directory on GitHub. I included the title and the summary to increase the amount of words to choose from because with only the title it wasn’t anywhere close to 1000 words. The question wanted me to use TFIDF to choose the terms however, I found it prudent to not calculate the TF because that is a different score for each page and it didn’t make sense to use that to pick words across the whole collection. I stuck with what they did in the book, with some small modifications to pick the best 1000.

I choose those fractions based on a quick histogram I did in excel of all the fractions. The max was 0.87 and the lowest was 0. After running the program a few times to see the words chosen with the top limit, I decided on 0.75 because it removed most of the stop words without cutting into important words. From there I found that 0.15 was the largest number I could use that would include just over 1000 words.

The matrix is found in ‘blogdata1.txt’.

**Question 2**

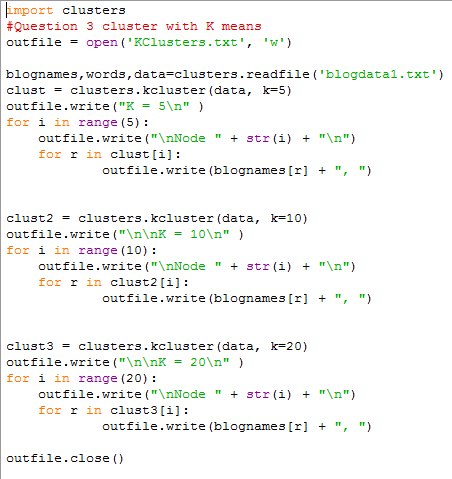
*Create an ASCII and JPEG dendrogram that clusters (i.e., HAC) the most similar blogs (see slides 12 & 13). Include the JPEG in your report and upload the ascii file to github (it will be too unwieldy for inclusion in the report).*

I used the program ‘createdendrogram.py’ to call the functions from ‘clusters.py’ which is a program found in the Programming Collective Intelligence book. Clusters had small modifications to make it compatible with python 3 but nothing to change the functionality. I set the standard output to a file because doing so in the function would be problematic since it is recursive. The created dendrogram is found in ‘blogclust.jpg’ and ‘blogclust.txt’. The picture is included on the next page.



**Question 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?*

I created the program ‘createKcluster.py’ to call the function kcluster() in the ‘cluster.py’ program and properly present the output. Since the output is long is will not be included in this report but it is in ‘KClusters.txt’

Iteration count for each k value per the program system output:

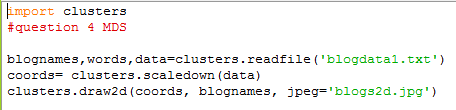
K=5 took 11 iterations

K= 10 took 9 iteration

K=20 took 6 iterations

**Question 4**

*Use MDS to create a JPEG of the blogs like slide 29 of the week 12 lecture. How many iterations were required?*

I created the program ‘MDS.py’ to call the functions scaledown() and draw2d() in the ‘cluster.py’ program.

The error went from a max of 4357.900447473182 to a minimum of 2501.438879265835 in 190 iterations, including the 1 it took to backtrack when the error started increasing.

The output is in ‘blogs2d.jpg’ which is included on the following page.

