Assignment 8 Report

Professor Nelson

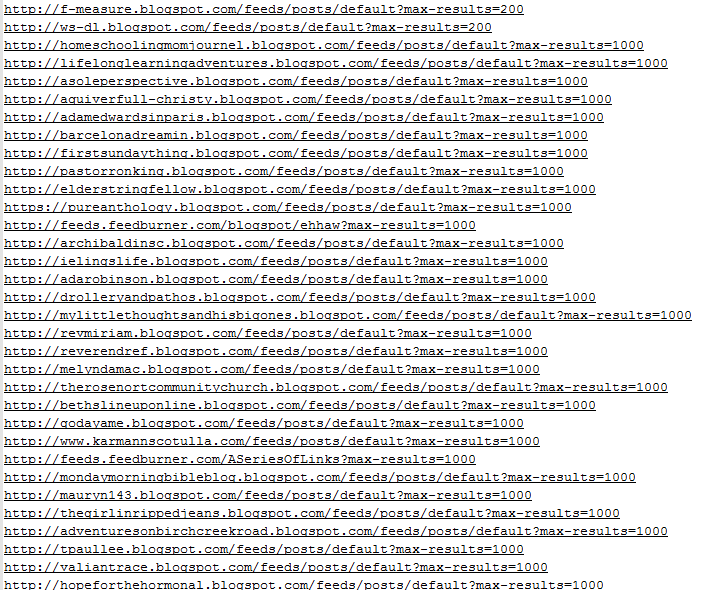
By Tyler Medina

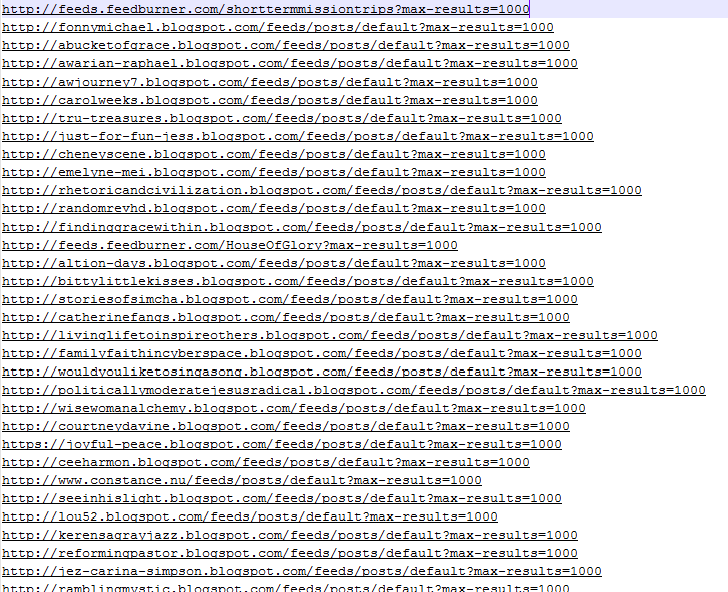
4/13/17

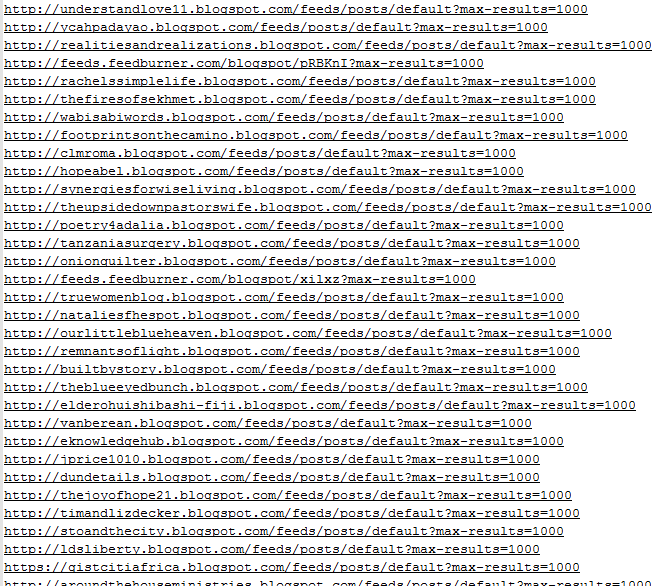
**Part 1**

**1.1 Getting 100 Blogs**

In getBlogs.py, I used BeautifulSoup to parse the blogs and get the uris.

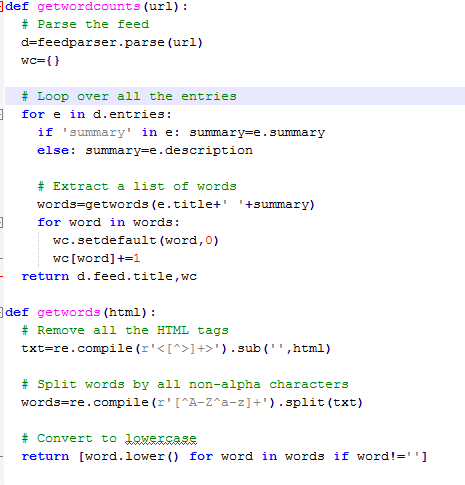




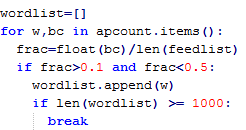


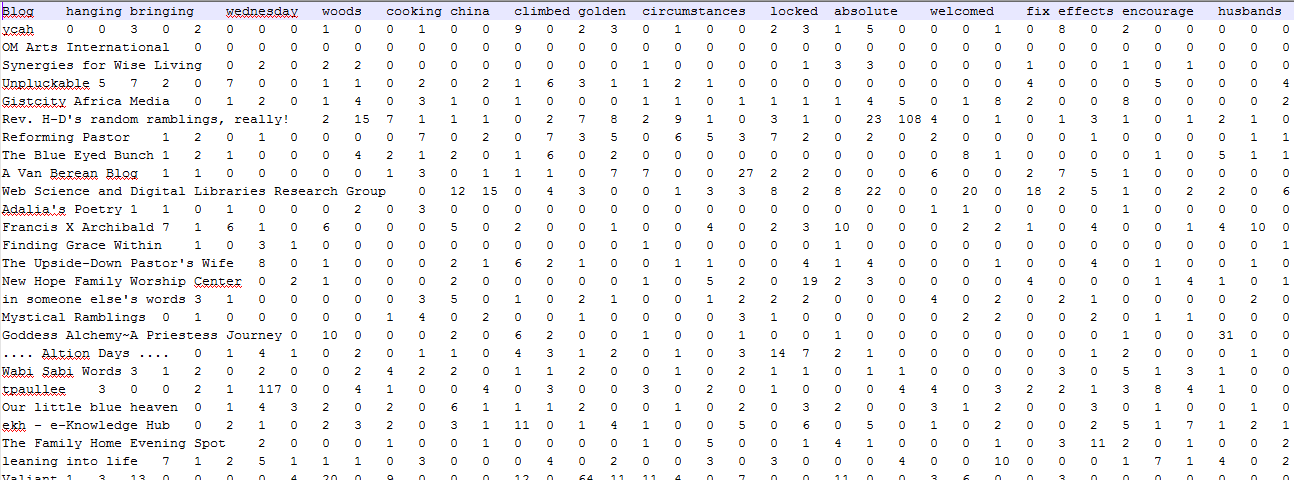
**1.2 Making a Blog-Term Matrix**

To be able to make a blog-term matrix, I used the getwordcounts() and getwords() functions in the collective intelligence textbook. Getwordcounts() gets the title and word counts for the RSS feed. The getwords() function strips the html and returns the words as a list.



To get a list of words we limit the amount of times the word occurs between 10 percent and fifty percent. This ensures that we throw out very specific words that’ll have title value and stop words such as “the”. With these bounds, we can craft a blog-term matrix.

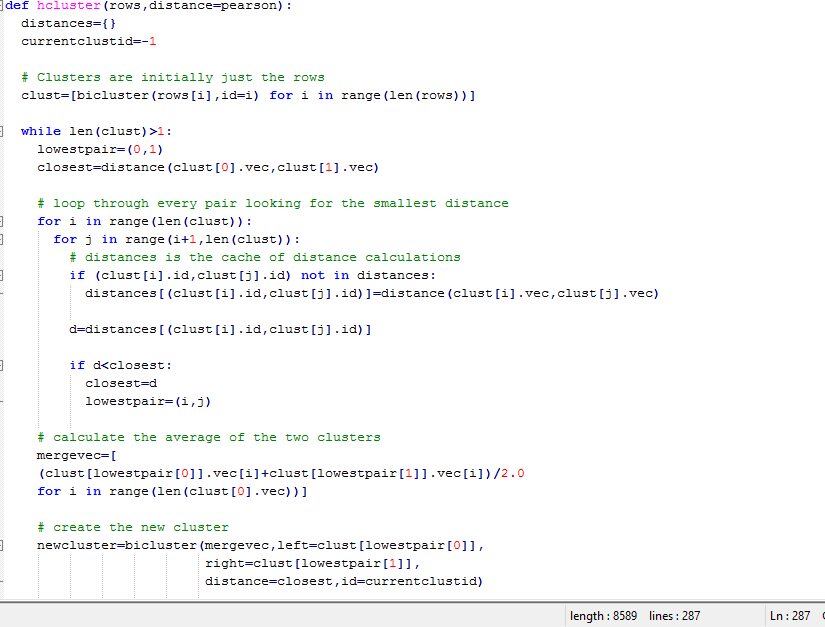


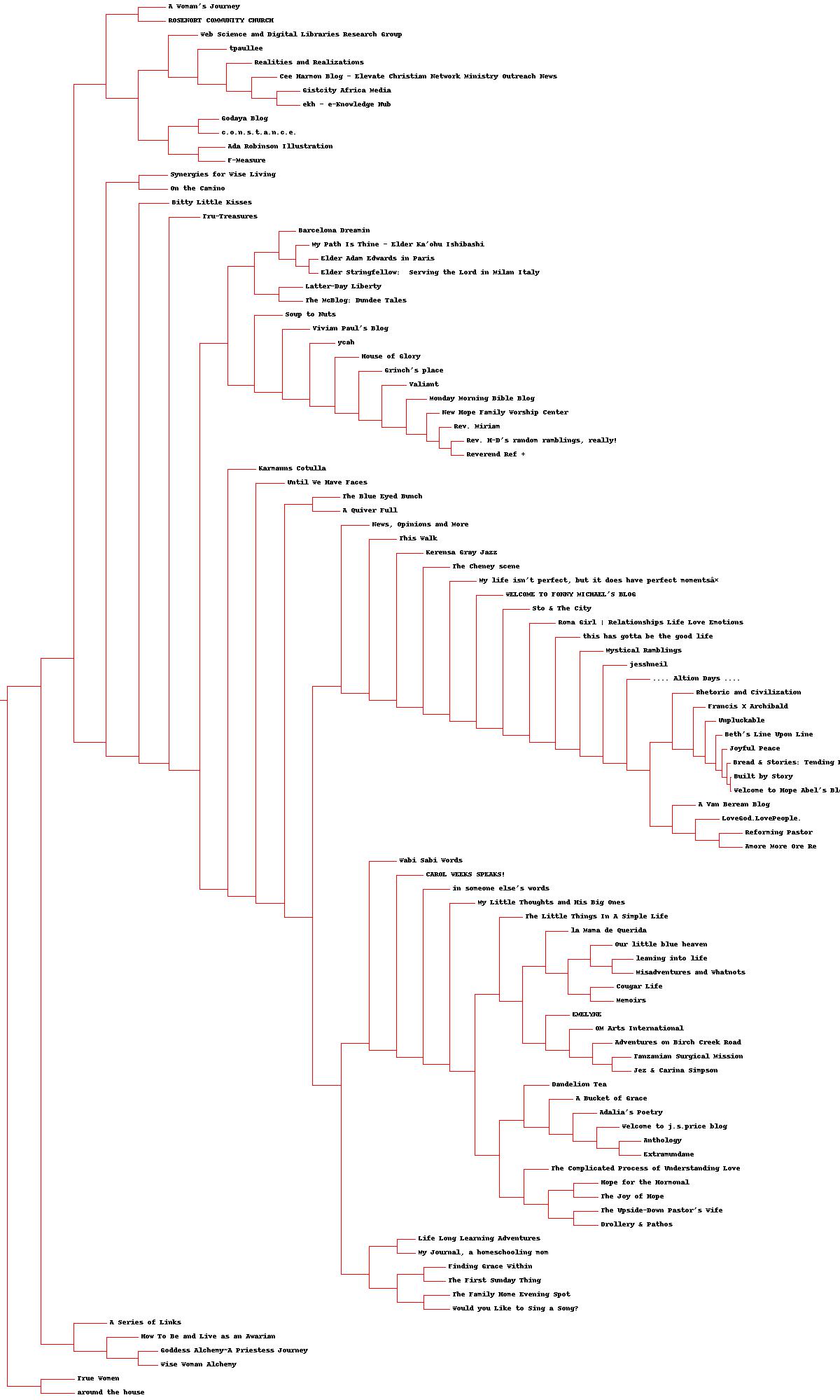


**Part 2**

**2.1 Creating a Dendrogram**

The Collective Intelligence textbook includes the functions to create the dendrogram. By using the hcluster() function, this ensures that the clusters recursively compare themselves with each other before merging. This process will continue until only one cluster is rest.

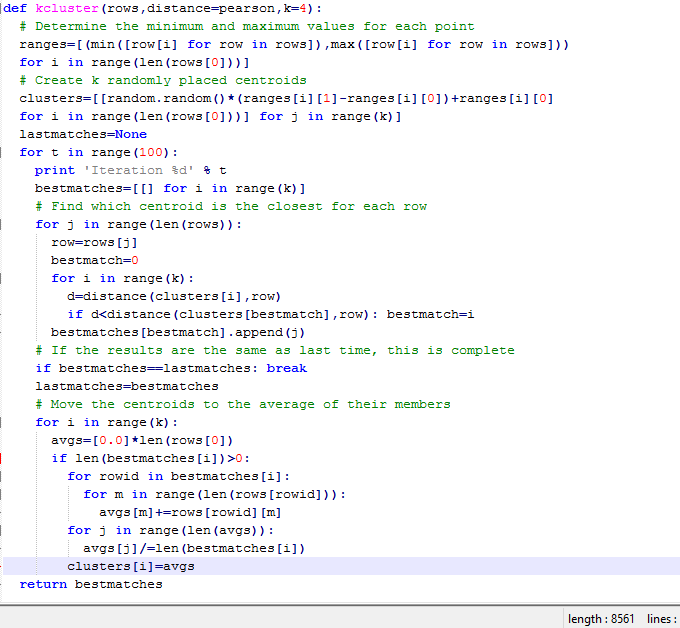




**Part 3**

**3.1 Kcluster**

Using k-means instead of hierarchical is similar into part 2, but we use the kcluster() function instead of the hcluster() function.

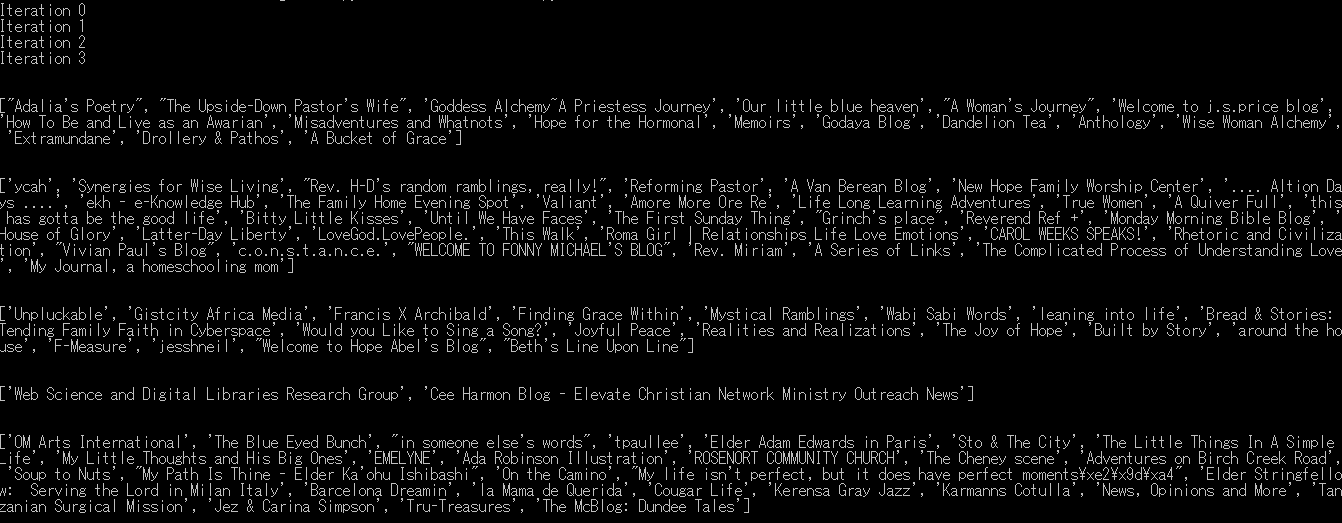


**3.2 Getting the centroids and iterations**

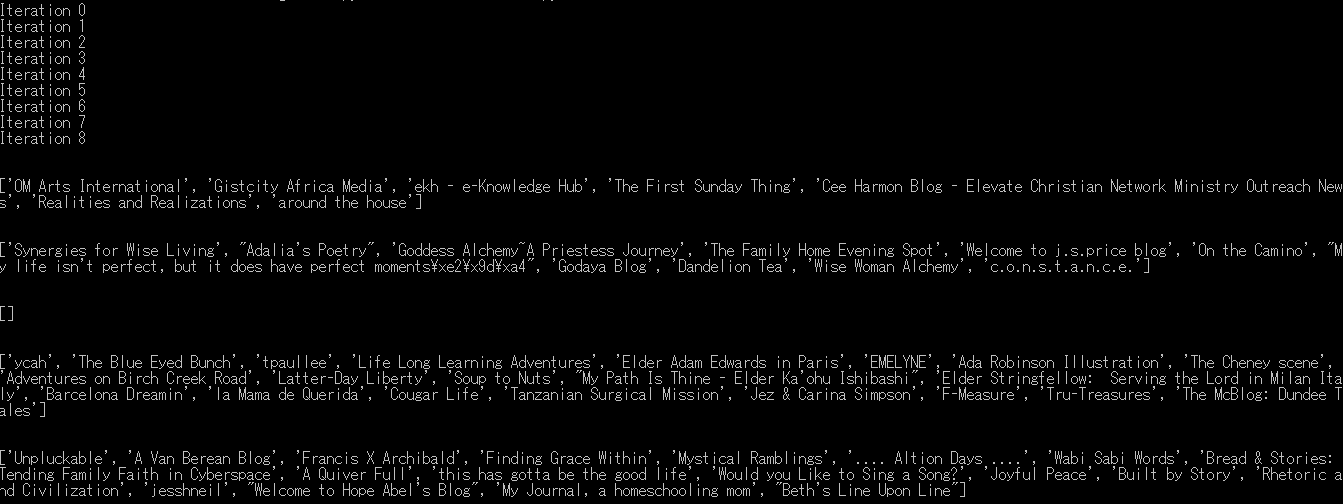
I pass k=5 as a parameter to get the number of iterations to get 5 clusters.



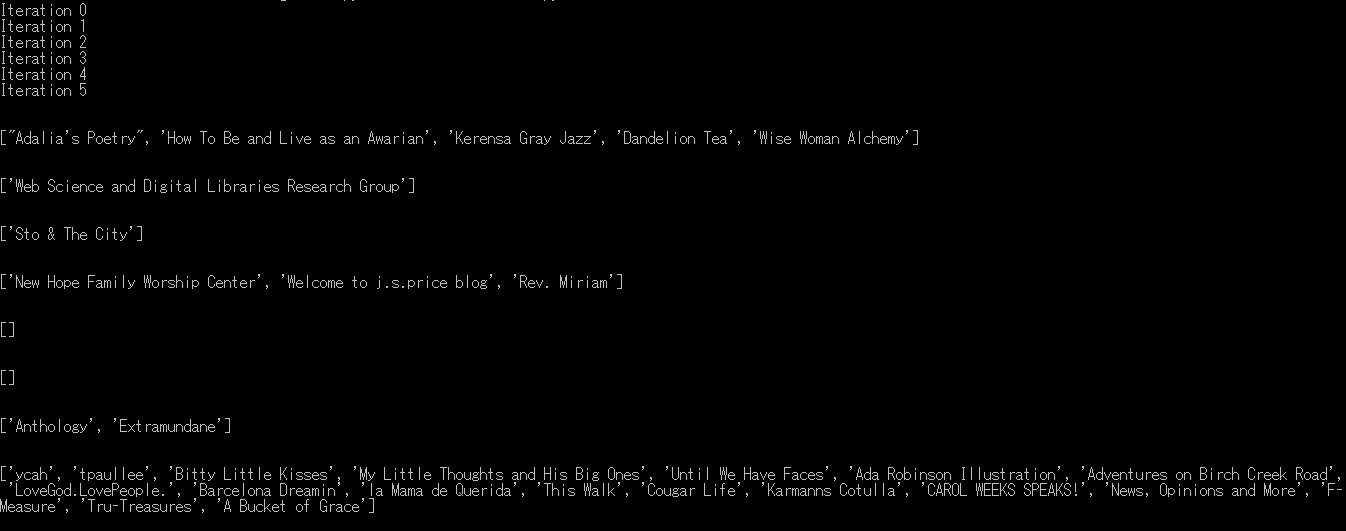
When k = 5, it takes 4 iterations.



When k=10, it takes 9 iterations.



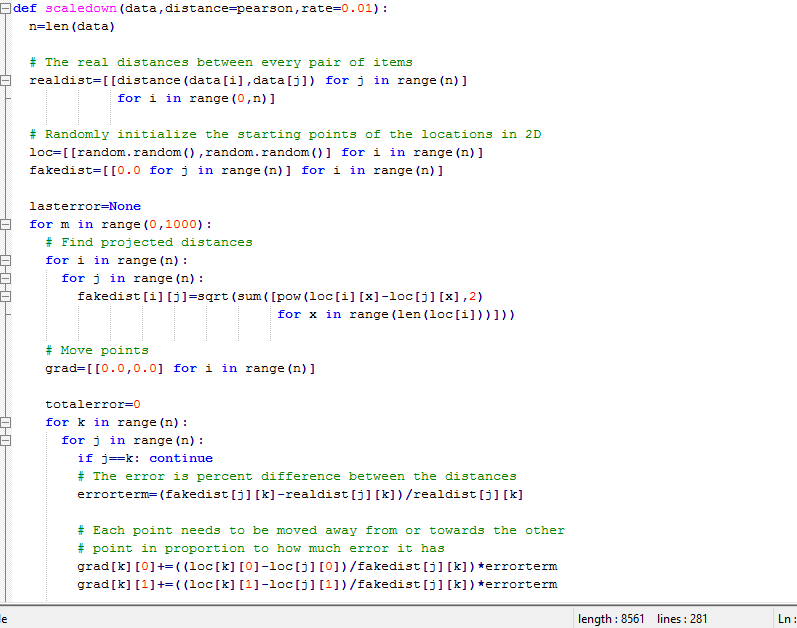
When k=20, it takes 6 iterations.

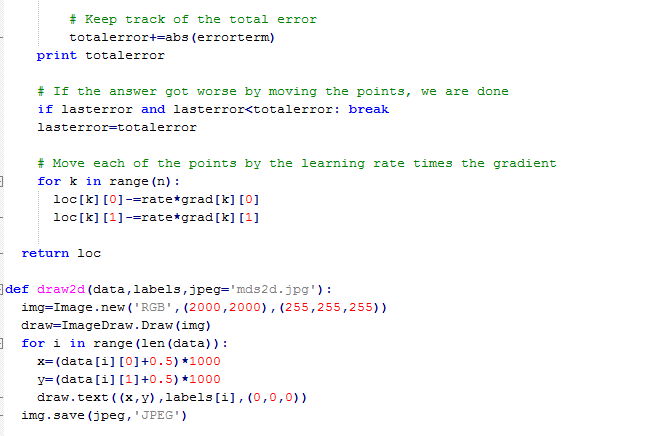


**Part 4**

**4.1 Creating MDS**

I used the scaledown() draw2d() function in the collective intelligence book to create the MDS style display. This creates a 2D representation of the data set.





**2.2 Results**

There were 129 iterations and the graphic looks like this:

