

CS 432: Homework Number Seven

Due on April 8, 2019 at 4:20 PM

Alexander Nwala

Tim Bruce

Problem 1

Create a blog-term matrix. Start by grabbing 100 blogs hosted on <https://www.blogger.com>. Include: <http://f-measure.blogspot.com/> <http://ws-dl.blogspot.com/>

The method described in class no longer works. So use your discretion to grab 98 more blogs. Describe how you generated the blog data - manually or automatically. Note that 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 (if you used one) 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 (chapter 3 PCI book) (slide 8 - Week 11) has been satisfied. Remember that blogs are paginated (slide 46).

Solution

My main discovery for this question was that if you type "blogspot" into google with another term following it, you'll get a related blog from blogspot. With this knowlege, I was able to get enough unique blogs for this assignment. The biggest problem was when google would ban me for a couple of hours when they realized that I was being uncool. Because of this, I limited my calls to one per 60 seconds.

Listing 1: Getting random blogs.

```
1 finalResults = []
2     for query in getNewWords():
3         print(query)
4         queryResults = []
5         query = query + " .blogspot.com"
6         time.sleep(60)
7         for j in search(query, num=10, stop=20, pause=2):
8             if ".blogspot.com" in j:
9                 queryResults.append(j)
10                f=open("blogs.txt", "a")
11                f.write(j)
12                f.write('\n')
13                f.close()
14            finalResults.append(queryResults)
15            print(queryResults[0])
16        print(finalResults)
```

Unfortunately, this returned a lot of odd blogs, so characterization at the end may be a challenge. I also suspect that this will lead to fairly unrelated blogs as well.

I was then able to get the rss feed from the pages using a handy algorithm from user alexmill on github in their project 'python3feedfinder', which basically takes the uri as input and outputs a link to the rss feed. From this point on, a lot of the work is from the book "Programming Collective Intelligence" book by Toby Segaran. I was able to modify the code from the book for use with Python 3.5 which I am using. I was able to take the most popular 1000 terms and organize them as a blog-term matrix.

Problem 2

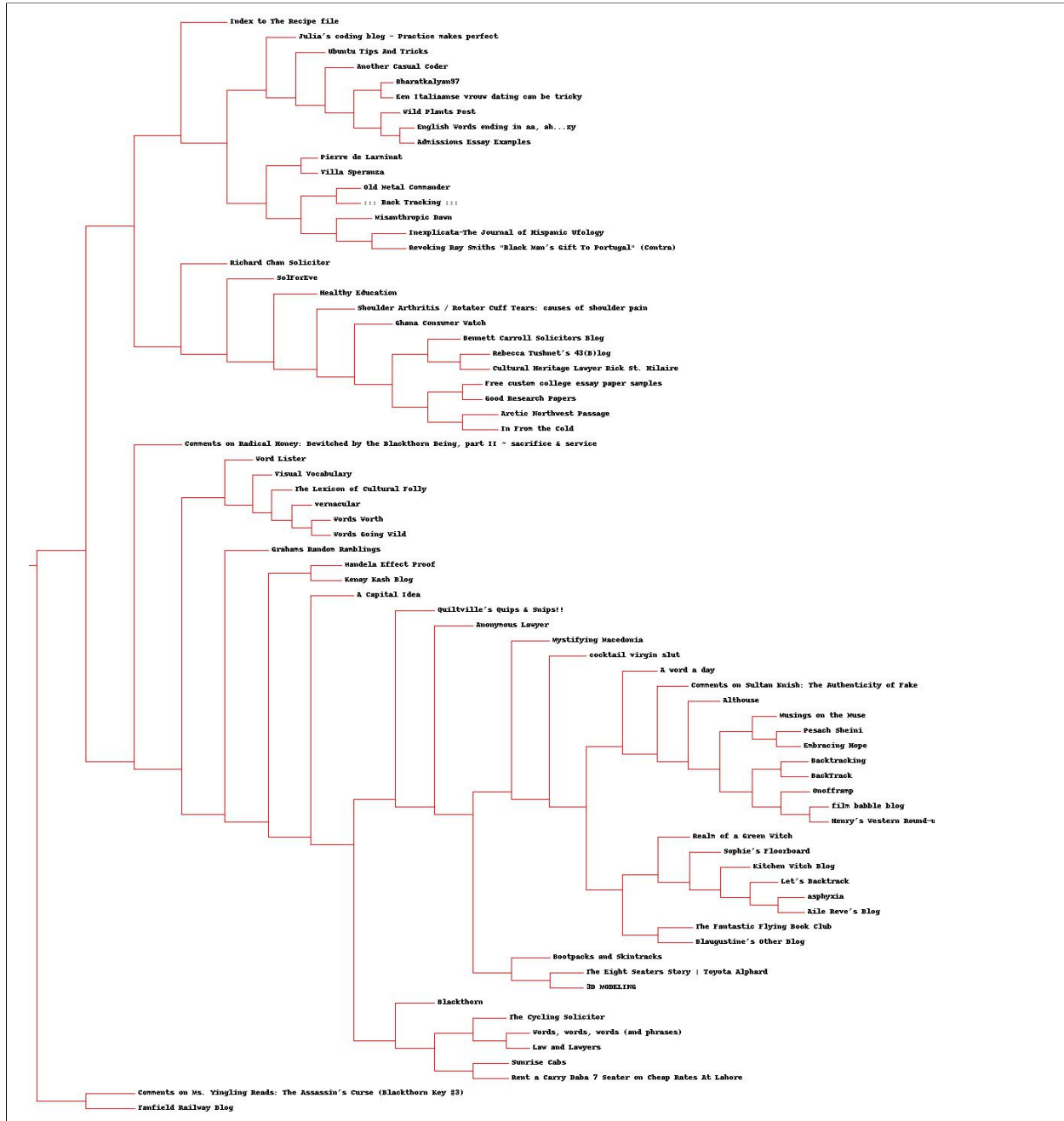
Create an ASCII and JPEG dendrogram that clusters (i.e., HAC) the most similar blogs (see slides 13 & 14). Include the JPEG in your report and upload the ascii file to github (it will be too unwieldy for inclusion

in the report).

Solution

Once again the PCI book has come in very handy. I was able to use its dendrogram method to create a very good ASCII dendrogram which is included in the code, as this is a bad format for the full ASCII dendrogram. However, for part two of this problem, the JPEG can be shown, but it is still unreadable.

Image 1: Dendrogram of similar blogs.



As you can see, the blogs have been clustered together.

Problem 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 ?

Solution

Once again, I have used the PCI book code to solve the problem. This time I converted their K-Means code into Python 3 and ran it three times, for $k=5$, $k=10$ and $k=20$. $k=5$ and $k=20$ took five iterations, and $k=10$ took four iterations, however as the k number increased, the computer took longer to process each iteration. The values in each centroid were too large to fit into this form factor, but can be found in the too-big-for-latex.txt file in this directory.

Problem 4

Use MDS to create a JPEG of the blogs similar to slide 29 of the week 11 lecture. How many iterations were required?

Solution

This is once again from the PCI book. It took 335 iterations to my understanding to complete the scale down, and produced a beautiful result.

Image 2: Image of similar blogs arranged by distance.

