Web Science: Assignment #6

Alexander Nwala

Mohd. Nauman Siddique

Sunday, March 31, 2019

Contents

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

To solve this problem I decided to write a shell script called **grabBlogs.py** to retrieve the 98 random blogs as shown in Listing 1. The script utilizes mainly utilizes the blogger url for profile search using a query parameter to solve the problem of retrieving unique blogs. The blogger url `http://www.blogger.com/profile-find.g?t=m&q=` called each time with search query parameter to retrieve 200 blogs. I used 200 executions because I happened to get duplicate blogs on some occasions and 100 calls wouldn't be enough to satisfy the requirements of this problem. For each new blog I got I saved the contents of the html page to a html document with the an id of the current iteration in the range of 200. I saved this file id and the URI found inside a file called **blogList.txt** as shown in Listing 2.
After 200 blogs were retrieved I used the sort command to find all unique blogs in the blog list file. Once this completed I had approximately 120 unique blogs. I then wrote a script in python 3.6 called **getFeed.py** as shown in Listing 3 which parsed each html document saved using the library Beautiful soup which allowed me to search the document by an HTML 'link' element to find the atom+xml feed of the blog. I saved these feeds to a file called **feedList.txt** which was later cleaned for any blogs that did not have atom+xml feeds. Usually these were blogs that no longer existed. Any extra blogs above 100 were simply discarded.
Finally I proceeded to to write another python script called **generateFeedVector.py** shown in Listing 4 which utilized the code provide by the Programming Collective Intelligence (PCI) book. This script was slightly modified to be usable for python 3.6 but also adding a limit to the amount of words the blog-term matrix allowed to a maximum of 1000 shown on line 65. I also didn't check for stop words when I retrieved the atom feeds, but I did check if words were stop words before creating the the matrix by using the nltk library's corpus of stop words shown on line 28. If a word was found to be a stop word according to their corpus it would not be added to the matrix. When this was completed it was saved to a file called **blogData.txt**.

```
import requests
from bs4 import BeautifulSoup
from bs4.element import Comment
import os
import sys
import clusters
#import generatefeedvector


'''
Function to fetch Blogger profiles and pass profiles to get their blogs
'''
```

Problem 1 continued on next page...

```
def fetch_blogs(query, limit):
    list_blogger_profiles = []
    list_blogger_urls = []
    output_directory = "Outputs/"
    blogger_url = "http://www.blogger.com/profile-find.g?t=m&q=" + query
    if not os.path.isdir(output_directory):
        os.mkdir(output_directory)
    while len(list_blogger_urls) < limit:
        response = requests.get(blogger_url)
        print(blogger_url)
        if response.status_code == 200:
            soup = BeautifulSoup(response.text, 'html.parser')
            profiles_tag_selector = soup.find_all('dl')
            for tags in profiles_tag_selector:
                profile_url_selector = tags.find('a', href=True)
                blogger_profile_url = "http://www.blogger.com/" + profile_url_selector['href
                    '][1:]
                list_blogger_profiles.append(blogger_profile_url)
            list_blogger_urls.extend(fetch_blog_urls(list_blogger_profiles, limit))
            print(list_blogger_urls)
            list_blogger_profiles = []
            blogger_url_selector = soup.find('a', {'id': 'next-btn'},
                                             href=True)
            if blogger_url_selector:
                blogger_url = blogger_url_selector['href']
            else:
                break
            print(len(list_blogger_urls))
    print(len(list_blogger_urls))
    if os.path.isfile(output_directory + "BlogUrls1.txt"):
        file_blog_urls = open(output_directory + "BlogUrls.txt", "a+")
    else:
        file_blog_urls = open(output_directory + "BlogUrls.txt", "w")
    for urls in list_blogger_urls:
        file_blog_urls.write(urls + "\n")
    file_blog_urls.close()


'''
Function to fetch blog urls from Blogger profiles
'''


def fetch_blog_urls(list_blogger_profiles, limit):
    count = 0
    list_blogger_urls = []
    for profile in list_blogger_profiles:
        if count < limit:
            response = requests.get(profile)
            if response.status_code == 200:
                soup = BeautifulSoup(response.text, 'html.parser')
                profile_blog_link_selector = soup.find_all('span', dir=True)
                for profile_blog_links in profile_blog_link_selector:
                    # print(profile_blog_links)
                    if profile_blog_links['dir'] == 'ltr':
                        blog_links_selector = profile_blog_links.find_all('a', href=True)
                        # print(blog_links_selector)
                        for blog_links in blog_links_selector:
                            print(blog_links['href'])
                            try:
```

```
                                response = requests.head(blog_links['href'], timeout=60)
                                print(response.headers)
                                if response.encoding and (response.encoding == 'ISO-8859-1'
                                    or response.encoding == 'UTF-8'):
                                    list_blogger_urls.append(blog_links['href'])
                                    print()
                                    count += 1
                            except Exception as e:
                                print(e)
    print(list_blogger_urls)
    return list_blogger_urls


def write_blogs():
    if not os.path.isdir("Outputs/blogs/"):
        os.mkdir("Outputs/blogs/")
    file_urls = open("Outputs/BlogUrls.txt", "r")
    file_list = open("Outputs/BlogList.txt", "w")
    count = 0
    for line in file_urls:
        response = requests.get(line.rstrip())
        if response.status_code == 200:
            filename = "blog_" + str(count) + ".txt"
            file_blog = open("Outputs/blogs/" + filename, "w")
            file_blog.write(response.text)
            file_blog.close()
            count += 1
            file_list.write(filename + " " + line.rstrip() + "\n")
    file_list.close()
    file_urls.close()
'''
Create Blog Matrix for blogger urls
'''

'''
def create_blog_matrix():
    apcount = {}
    word_counts = {}
    feed_list = [line.rstrip() for line in open('Outputs/BlogUrls.txt')]
    for feed_url in feed_list:
        print(feed_url)
        (title, wc) = generatefeedvector.getwordcounts(feed_url)
        if title and wc:
            word_counts[title] = wc
            for (word, count) in wc.items():
                apcount.setdefault(word, 0)
                if count > 1:
                    apcount[word] += 1
    print(len(apcount))
    wordlist = []
    for (w, bc) in apcount.items():
        frac = float(bc) / len(feed_list)
        if frac > 0.1 and frac < 0.5:
            wordlist.append(w)

    out = open('Outputs/blogdata.txt', 'w')
    out.write('Blog')
    for word in wordlist:
        out.write('\t%s' % word)
    out.write('\n')
    print(len(word_counts))
    for (blog, wc) in word_counts.items():
```

```
        print(blog)
        out.write(blog)
        for word in wordlist:
            if word in wc:
                out.write('\t%d' % wc[word])
            else:
                out.write('\t0')
        out.write('\n')


def tag_visible(element):
    if element.parent.name in ['style', 'script', 'head', 'meta', '[document]']:
        return False
    if isinstance(element, Comment):
        return False
    return True
'''

'''
Replicate code from Programming-Collective-Intelligence/blob/master/chapter3/
    generatefeedvector.py
'''

'''
def get_word_count(url):

    # Parse the feed
    print("Word Count")
    try:
        response = requests.get(url, timeout=60)
        wc = {}
        if response.status_code == 200:
            # Loop over all the entries
            soup = BeautifulSoup(response.text, 'html.parser')
            texts = soup.findAll(text=True)
            visible_texts = filter(tag_visible, texts)
            title = soup.title.string
            summary = " ".join(t.strip() for t in visible_texts)
            # Extract a list of words
            words = get_words(title + ' ' + summary)
            for word in words:
                wc.setdefault(word, 0)
                wc[word] += 1
            print(title.rstrip())
            print(wc)
            return title.rstrip(), wc
    except Exception as e:
        print(e)
        print(url)
        return None, None
    return None, None


def get_words(html):
    # Remove all the HTML tags
    txt = re.compile(r'<[^>]+>').sub('', html)

    # Split words by all non-alpha characters
    words = re.compile(r'[^A-Z^a-z]+').split(txt)

    # Convert to lowercase
    return [word.lower() for word in words if word != '']
```

```
200
def draw_acii_dendogram():
    blognames, words, data = clusters.readfile('Outputs/blogdata.txt')
    clust = clusters.hcluster(data)
    clusters.printclust(clust, labels=blognames)
205

def draw_dendogram():
    blognames, words, data = clusters.readfile('Outputs/blogdata.txt')
    clust = clusters.hcluster(data)
210     clusters.drawdendrogram(clust, blognames, jpeg='blogclust.jpg')

def text():
    blognames, words, data = clusters.readfile('Outputs/blogdata.txt')
    coords = clusters.scaledown(data)
215     kclust = clusters.kcluster(data, k=4)


# fetch_blogs("Twitter", 100)
# fetch_blogs("Cricket", 56)
220 # fetch_blogs("Patna", 16)
# fetch_blogs("Manchester", 13)
# fetch_blogs("India", 6)
'''
'''
225 fetch_blogs("Music", 100)
file_open = open("Outputs/BlogUrls.txt", "a+")
file_open.write("http://f-measure.blogspot.com/" + "\n")
file_open.write("http://ws-dl.blogspot.com/" + "\n")
file_open.close()
230 '''

def createDendrogram():
    blogs, colnames, data = clusters.readfile('Outputs/blogdata.txt')
    cluster = clusters.hcluster(data)
235     clusters.drawdendrogram(cluster, blogs, jpeg='../docs/q2Dendrogram.jpg')
    f = open("Outputs/q2ASCII.txt", 'w')
    sys.stdout = f
    clusters.printclust(cluster, labels=blogs)
    f.close()
240     sys.stderr.close()


def kmeans():
    karr = [5, 10, 20]
245     blogs, colnames, data = clusters.readfile('Outputs/blogdata.txt')
    for i in karr:

        kclust, itercount = clusters.kcluster(data, k=i)
        print(kclust)
250         f = open("Outputs/kclust_%d.txt" % i, 'w')
        f.write("Iteration count: %d \n" % itercount)
        print(len(kclust))
        for cluster in kclust:
            f.write("**************************\n")
255             f.write("[")
            for blogid in cluster:
                f.write(blogs[blogid] + ", ")
```

```
            f.write("]\n")


def mds():
    blognames, words, data = clusters.readfile('Outputs/blogdata.txt')
    coords, itercount = clusters.scaledown(data)
    clusters.draw2d(coords, labels=blognames, jpeg='../docs/q4Mds.jpg')
    print ('Iteration count: %d' % itercount)

createDendrogram()
# write_blogs()
# create_blog_matrix()
# draw_acii_dendogram()
# draw_dendogram()
# text()
```

Listing 1: Pythonl script to retrieve unique blogs

```
http://f-measure.blogspot.com/
http://ws-dl.blogspot.com/
http://macthemost.blogspot.com/?expref=next-blog
http://itll-glow-on-you.blogspot.com/?expref=next-blog
http://themusicbinge.blogspot.com/?expref=next-blog
http://tmcbryrone.blogspot.com/?expref=next-blog
http://mccookerybook.blogspot.com/?expref=next-blog
http://revolverusa.blogspot.com/?expref=next-blog
http://ahtapotunbahcesi.blogspot.com/?expref=next-blog
http://cherryarea.blogspot.com/?expref=next-blog
http://indietop20.blogspot.com/?expref=next-blog
http://stonehillsketchbook.blogspot.com/?expref=next-blog
http://cuzmusicrocks.blogspot.com/?expref=next-blog
http://glipress.blogspot.com/?expref=next-blog
http://truthfulmood.blogspot.com/?expref=next-blog
http://www.sunstockmusic.com/?expref=next-blog
http://markfishers-musicreview.blogspot.com/?expref=next-blog
http://ohyesjonsi.blogspot.com/?expref=next-blog
http://blog.spinitron.com/?expref=next-blog
http://lost-places-hamburg.blogspot.com/?expref=next-blog
http://antonellagiugliano.blogspot.com/?expref=next-blog
http://storiesfromthecityradiovalencia.blogspot.com/?expref=next-blog
http://onwarmermusic.blogspot.com/?expref=next-blog
http://floorshimezipperboots.blogspot.com/?expref=next-blog
http://noradiorecs.blogspot.com/?expref=next-blog
http://www.chrisanne-grise.com/?expref=next-blog
http://momentarilymusical.blogspot.com/?expref=next-blog
http://maggotcaviar.blogspot.com/?expref=next-blog
http://stephanieveto.blogspot.com/?expref=next-blog
http://mondaywakeup.blogspot.com/?expref=next-blog
http://www.mpcfilmco.com/?expref=next-blog
http://justwordsnomeaning.blogspot.com/?expref=next-blog
http://blog.spin-itrecords.ca/?expref=next-blog
http://davecromwellwrites.blogspot.com/?expref=next-blog
http://mobbie2.blogspot.com/?expref=next-blog
http://intheframefilmreviews.blogspot.com/?expref=next-blog
http://mileinmine.blogspot.com/?expref=next-blog
http://insearchoftrueromantics.blogspot.com/?expref=next-blog
http://bonjourgirl.blogspot.com/?expref=next-blog
http://didnotchart.blogspot.com/?expref=next-blog
http://wildnightstranger.blogspot.com/?expref=next-blog
http://mesastivromia.blogspot.com/?expref=next-blog
http://bartkings.blogspot.com/?expref=next-blog
http://jlmdlhlcm1516.blogspot.com/?expref=next-blog
```

Problem 1 continued on next page...

```
http://abueveryday.blogspot.com/?expref=next-blog
http://doyouneedatv.blogspot.com/?expref=next-blog
http://srkikiblog.blogspot.com/?expref=next-blog
http://bleakbliss.blogspot.com/?expref=next-blog
http://dancingincirclesnow.blogspot.com/?expref=next-blog
http://ridingaborrowedbike.blogspot.com/?expref=next-blog
http://www.hipindetroit.com/?expref=next-blog
http://marshwiggle.blogspot.com/?expref=next-blog
http://sixeyes.blogspot.com/?expref=next-blog
http://encorenorthernireland.blogspot.com/?expref=next-blog
http://johnandmaureensanto.blogspot.com/?expref=next-blog
http://franbrighton.blogspot.com/?expref=next-blog
http://mtjrrantsravesonmusic.blogspot.com/?expref=next-blog
http://turnitupjack.blogspot.com/?expref=next-blog
http://outanddowninthecolonies.blogspot.com/?expref=next-blog
http://organmyth.blogspot.com/?expref=next-blog
http://jojobethkatiehannahlcm1516.blogspot.com/?expref=next-blog
http://tuqueresveristo.blogspot.com/?expref=next-blog
http://jbreitling.blogspot.com/?expref=next-blog
http://paulinag-mediaa2.blogspot.com/?expref=next-blog
http://www.thejeopardyofcontentment.com/?expref=next-blog
http://www.juanbook.com/?expref=next-blog
http://skinnyshoes.blogspot.com/?expref=next-blog
http://steel-city-rust.blogspot.com/?expref=next-blog
http://fridaynightdream.blogspot.com/?expref=next-blog
http://avidsblog.blogspot.com/?expref=next-blog
http://atozappa.blogspot.com/?expref=next-blog
http://ps-music.blogspot.com/?expref=next-blog
http://primitiveofferings.blogspot.com/?expref=next-blog
http://markeortega.blogspot.com/?expref=next-blog
http://londynsky.blogspot.com/?expref=next-blog
http://somecallitnoise.blogspot.com/?expref=next-blog
http://liquid-pop.blogspot.com/?expref=next-blog
http://elijace.blogspot.com/?expref=next-blog
http://theworldsfirstinternetbaby.blogspot.com/?expref=next-blog
http://dazeyrosie.blogspot.com/?expref=next-blog
http://earenjoy.blogspot.com/?expref=next-blog
http://my-name-is-blue-canary.blogspot.com/?expref=next-blog
http://www.thestarkonline.com/?expref=next-blog
http://dana9morgan.blogspot.com/?expref=next-blog
http://hellomynameisjustin.blogspot.com/?expref=next-blog
http://paradoxical-era.blogspot.com/?expref=next-blog
http://pithytitlehere.blogspot.com/?expref=next-blog
http://myopiamuse.blogspot.com/?expref=next-blog
http://jamiemclelland.blogspot.com/?expref=next-blog
http://ngaio1619.blogspot.com/?expref=next-blog
http://aubade1.blogspot.com/?expref=next-blog
http://hani-bittersweet.blogspot.com/?expref=next-blog
http://worldofpearljambootlegs.blogspot.com/?expref=next-blog
http://isyelili.blogspot.com/?expref=next-blog
http://adrianomarquesblog.blogspot.com/?expref=next-blog
http://naoponhomusica.blogspot.com/?expref=next-blog
http://dcresider.blogspot.com/?expref=next-blog
http://fractalpress.blogspot.com/?expref=next-blog
http://doginasweatershowreviews.blogspot.com/?expref=next-blog
http://flowradio.blogspot.com/?expref=next-blog
```

Listing 2: 100 unique blogs collected

```
import requests
from bs4 import BeautifulSoup
```

```
import os


def getFeed(filename):
    print("FILENAME:", filename)
    with open("data/blogs/" + filename) as f:
        f.seek(0)
        html = f.read()
        soup = BeautifulSoup(html, 'html.parser')
        feed = soup.find_all(
            'link', attrs={'type': 'application/atom+xml'})

        if(feed):
            return feed[0]['href']
        return None


def deleteNoFeedFiles():
    noneLines = []
    f = open("data/feedList.txt", "r+")
    d = f.readlines()
    f.seek(0)
    for i, line in enumerate(d):
        if 'None' in line:
            noneLines.append(i)
        else:
            f.write(line)
    f.truncate()
    f.close()

    print(noneLines)
    f = open("data/blogList.txt", "r+")
    d = f.readlines()
    f.seek(0)
    for i, line in enumerate(d):
        if i in noneLines:
            # delete file
            fileToDelete = line.split(' ')[0]
            print("Deleting:", fileToDelete)
            os.remove("data/blogs/" + fileToDelete)
        else:
            f.write(line)
    f.truncate()
    f.close()


if __name__ == "__main__":

    with open("data/blogList.txt") as f, open("data/feedList.txt", 'w') as out:
        for line in f:
            filename = line.split(' ')[0]
            feed = getFeed(filename)
            print(feed, file=out)

    deleteNoFeedFiles()
```

Listing 3: Python script to retrieve atom feeds of blogs

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import feedparser
import re
```

```
from nltk.corpus import stopwords

stops = stopwords.words("english")

def getwordcounts(url):
    '''
    Returns title and dictionary of word counts for an RSS feed
    '''
    # Parse the feed
    d = feedparser.parse(url)
    wc = {}

    # Loop over all the entries
    for e in d.entries:
        if 'summary' in e:
            summary = e.summary

        else:
            summary = e.description

        # Extract a list of words
        words = getwords(e.title + ' ' + summary)
        for word in words:
            if word not in stops:
                wc.setdefault(word, 0)
                wc[word] += 1

    return (d.feed.title, wc)


def getwords(html):
    # Remove all the HTML tags
    txt = re.compile(r'<[^>]+>').sub('', html)

    # Split words by all non-alpha characters
    words = re.compile(r'[^A-Z^a-z]+').split(txt)

    # Convert to lowercase
    return [word.lower() for word in words if word != '']


apcount = {}
wordcounts = {}
feedlist = [line for line in open('data/feedList.txt')]
for feedurl in feedlist:
    try:
        (title, wc) = getwordcounts(feedurl)
        wordcounts[title] = wc
        for (word, count) in wc.items():
            apcount.setdefault(word, 0)
            if count > 1:
                apcount[word] += 1
    except:
        print('Failed to parse feed', feedurl)

wordlist = []
for (w, bc) in apcount.items():
    frac = float(bc) / len(feedlist)
    if frac > 0.1 and frac < 0.5:
        wordlist.append(w)
    if len(wordlist) >= 1000:
        break
```

```
out = open('data/blogdata.txt', 'w')
out.write('Blog')
for word in wordlist:
    out.write('\t%s' % word)
out.write('\n')
for (blog, wc) in wordcounts.items():
    print(blog)
    out.write(blog)
    for word in wordlist:
        if word in wc:
            out.write('\t%d' % wc[word])
        else:
            out.write('\t0')
    out.write('\n')
```

Listing 4: Python script to generate 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

To solve this question I used the code provided by the Programming Collective Intelligence book to write a script in python 2.7 called **createClusters.py** as shown in Listing 5. This script has a method called *createDendrogram* which utilizes the clusters.py file provided by the PCI book load the blog-term matrix created in question 1 to create a Hierarchical Clustering tree image, the dendrogram, as shown in Figure 1. I also created an ASCII file named **q2ASCII.txt** to represent this tree structure in text which is available on my Github page.

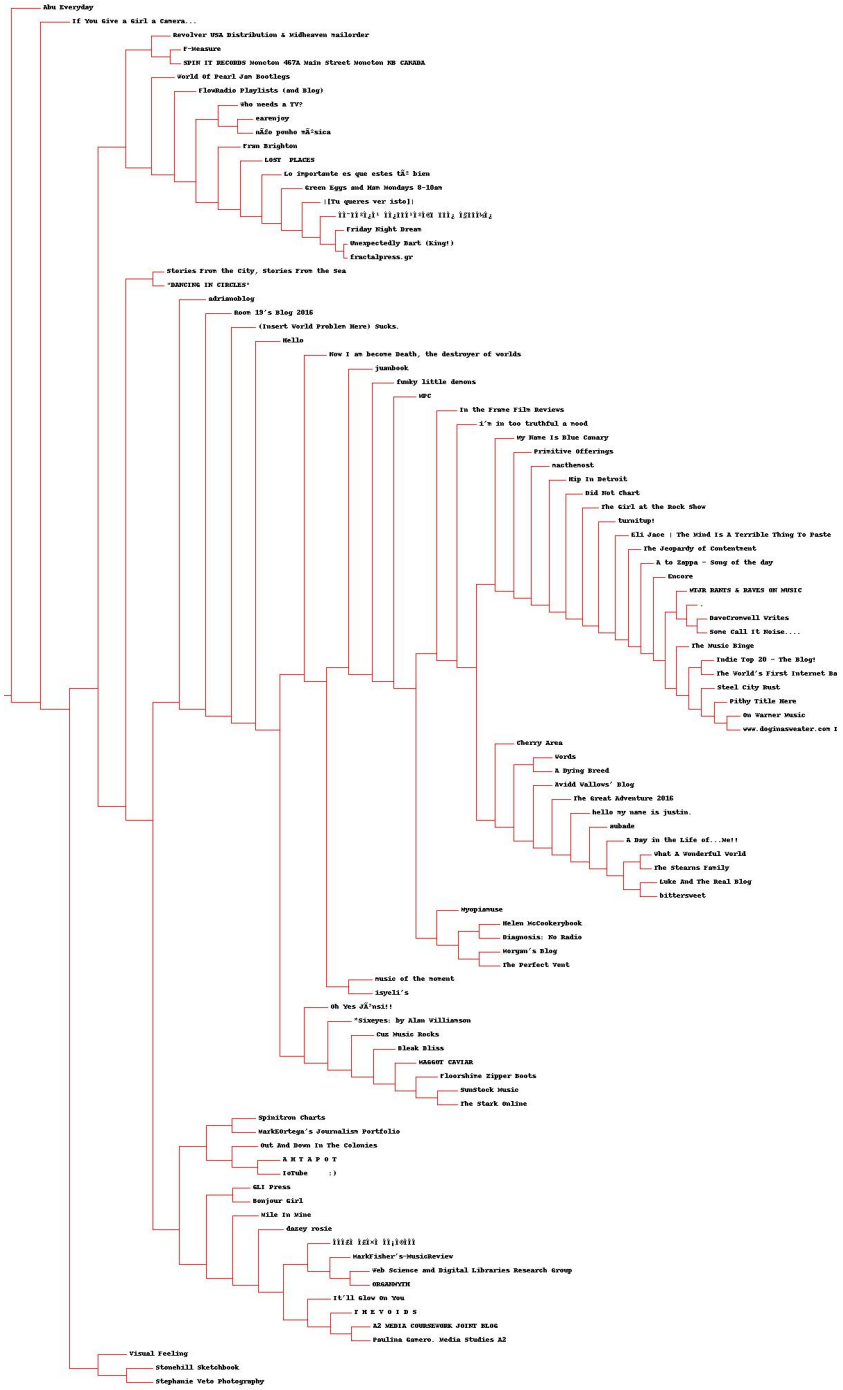

Figure 1: Dendrogram for blogs collected

```
import clusters
import sys


def createDendrogram():
    blogs, colnames, data = clusters.readfile('data/blogdata.txt')
    cluster = clusters.hcluster(data)
    clusters.drawdendrogram(cluster, blogs, jpeg='../docs/q2Dendrogram.jpg')
    f = open("data/q2ASCII.txt", 'w')
    sys.stdout = f
    clusters.printclust(cluster, labels=blogs)
    f.close()
    sys.stderr.close()


def kmeans():
    karr = [5, 10, 20]
    blogs, colnames, data = clusters.readfile('data/blogdata.txt')
    for i in karr:

        kclust, itercount = clusters.kcluster(data, k=i)
        print(kclust)
        f = open("data/kclust_%d.txt" % i, 'w')
        f.write("Iteration count: %d \n" % itercount)
        print(len(kclust))
        for cluster in kclust:
            f.write("**************************\n")
            f.write("[")
            for blogid in cluster:
                f.write(blogs[blogid] + ", ")
            f.write("]\n")


def mds():
    blognames, words, data = clusters.readfile('data/blogdata.txt')
    coords, itercount = clusters.scaledown(data)
    clusters.draw2d(coords, labels=blognames, jpeg='../docs/q4Mds.jpg')
    print ('Iteration count: %d' % itercount)


if __name__ == "__main__":
    '''
    Written in python 2.7
    '''
    createDendrogram()
    kmeans()
    mds()
```

Listing 5: Python script to generate clusters with different approaches

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

To solve this question I again used the code provided by the Programming Collective Intelligence book as shown in Listing 5 to create a method called *kmeans*. The *kmeans* method again reads my blog-term matrix and using the method *kcluster* for the clusters.py library. I modified the *kcluster* method to also return the iteration count so it could be saved along with the blog title. For each value of k I created a separate file named **kclut_n.txt** where n is the value of k. For a k value of 5 it took 7 iterations with the values of each centroids shown in Listing 6. For a k value of 10 it took 5 iterations with the values of each centroids shown in Listing 7. For a k value of 20 it took 5 iterations with the values of each centroids shown in Listing 8.

```
Iteration count: 7
**************************
[Web Science and Digital Libraries Research Group, macthemost, The Music Binge, Indie Top 20
    - The Blog!, Cuz Music Rocks, i'm in too truthful a mood, SunStock Music, MarkFisher's-
    MusicReview, Oh Yes J nsi!!, On Warmer Music, Floorshime Zipper Boots, The Girl at the
    Rock Show, MAGGOT CAVIAR, DaveCromwell Writes, In the Frame Film Reviews, Mile In Mine,
    Bonjour Girl, Did Not Chart, Bleak Bliss, Hip In Detroit, *Sixeyes: by Alan Williamson,
    Encore, MTJR RANTS & RAVES ON MUSIC, turnitup!, ., The Jeopardy of Contentment, Steel
    City Rust, A to Zappa - Song of the day, Primitive Offerings, MarkEOrtega's Journalism
    Portfolio, Some Call It Noise...., Eli Jace | The Mind Is A Terrible Thing To Paste, The
    World's First Internet Baby, The Stark Online, www.doginasweater.com Live Show Review
    Archive, ]
**************************
[A H T A P O T, Spinitron Charts, LOST  PLACES, Stories From the City, Stories From the Sea,
    Green Eggs and Ham Mondays 8-10am,                    , Unexpectedly Bart (
    King!), A2 MEDIA COURSEWORK JOINT BLOG, Lo importante es que estes t  bien, "DANCING IN
    CIRCLES", Out And Down In The Colonies, ORGANMYTH, T H E V O I D S, |[Tu queres ver
    isto]|, Paulina Gamero. Media Studies A2, If You Give a Girl a Camera..., Friday Night
    Dream,                                                , My Name Is Blue Canary,
    fractalpress.gr, ]
**************************
[F-Measure, Revolver USA Distribution & Midheaven mailorder, SPIN IT RECORDS Moncton 467A
    Main Street Moncton NB CANADA, Abu Everyday, ]
**************************
[It'll Glow On You, Stephanie Veto Photography, IoTube      :), Who needs a TV?, Fran
    Brighton, dazey rosie, earenjoy, World Of Pearl Jam Bootlegs, adrianoblog, n o ponho
    m sica, FlowRadio Playlists (and Blog), ]
**************************
[(Insert World Problem Here) Sucks., Helen McCookerybook, Cherry Area, Stonehill Sketchbook,
    GLI Press, Visual Feeling, Diagnosis: No Radio, music of the moment, MPC, Words, A
    Dying Breed, Hello, Luke And The Real Blog, funky little demons, The Great Adventure
    2016, juanbook, Avidd Wallows' Blog, What A Wonderful World, Now I am become Death, the
    destroyer of worlds, Morgan's Blog, hello my name is justin., The Perfect Vent, Pithy
    Title Here, Myopiamuse, The Stearns Family, Room 19's Blog 2016, aubade, bittersweet,
    isyeli's, A Day in the Life of...Me!!, ]
```

Listing 6: K-means clustering with a value of 5

```
Iteration count: 5
**************************
[SPIN IT RECORDS Moncton 467A Main Street Moncton NB CANADA, ]
**************************
[(Insert World Problem Here) Sucks., Helen McCookerybook, A H T A P O T, Cherry Area, i'm in
    too truthful a mood, LOST  PLACES, Diagnosis: No Radio, music of the moment, Green Eggs
```

```
 and Ham Mondays 8-10am, IoTube :), Bonjour Girl, Lo importante es que estes t
    bien, Luke And The Real Blog, funky little demons, Fran Brighton, Avidd Wallows' Blog,
    Morgan's Blog, bittersweet, isyeli's, FlowRadio Playlists (and Blog), ]
**************************
[Bleak Bliss, ]
**************************
[Stonehill Sketchbook, Stories From the City, Stories From the Sea, Stephanie Veto
    Photography, Hello, "DANCING IN CIRCLES", The Great Adventure 2016, If You Give a Girl a
     Camera..., What A Wonderful World, The Stearns Family, Room 19's Blog 2016, World Of
    Pearl Jam Bootlegs, adrianoblog, n o ponho m sica, ]
**************************
[macthemost, The Music Binge, Revolver USA Distribution & Midheaven mailorder, Cuz Music
    Rocks, SunStock Music, MarkFisher's-MusicReview, Oh Yes J nsi!!, Floorshime Zipper
    Boots, MAGGOT CAVIAR, DaveCromwell Writes, Did Not Chart, *Sixeyes: by Alan Williamson,
    Encore, MTJR RANTS & RAVES ON MUSIC, turnitup!, ., The Jeopardy of Contentment, Some
    Call It Noise...., Eli Jace | The Mind Is A Terrible Thing To Paste, The Stark Online, ]
**************************
[Words, A Dying Breed, juanbook, Now I am become Death, the destroyer of worlds, hello my
    name is justin., The Perfect Vent, A Day in the Life of...Me!!, ]
**************************
[Web Science and Digital Libraries Research Group, GLI Press,                    ,
    ORGANMYTH, dazey rosie, earenjoy, ]
**************************
[It'll Glow On You, Spinitron Charts, Unexpectedly Bart (King!), A2 MEDIA COURSEWORK JOINT
    BLOG, Who needs a TV?, Out And Down In The Colonies, |[Tu queres ver isto]|, Paulina
    Gamero. Media Studies A2, Friday Night Dream,
              , fractalpress.gr, ]
**************************
[F-Measure, Abu Everyday, T H E V O I D S, ]
**************************
[Indie Top 20 - The Blog!, Visual Feeling, On Warmer Music, The Girl at the Rock Show, MPC,
    In the Frame Film Reviews, Mile In Mine, Hip In Detroit, Steel City Rust, A to Zappa -
    Song of the day, Primitive Offerings, MarkEOrtega's Journalism Portfolio, The World's
    First Internet Baby, My Name Is Blue Canary, Pithy Title Here, Myopiamuse, aubade, www.
    doginasweater.com Live Show Review Archive, ]
```

Listing 7: K-means clustering with a value of 10

```
Iteration count: 5
**************************
[(Insert World Problem Here) Sucks., Helen McCookerybook, Cherry Area, Stonehill Sketchbook,
     Diagnosis: No Radio, music of the moment, Words, A Dying Breed, Hello, Luke And The
    Real Blog, funky little demons, The Great Adventure 2016, juanbook, Avidd Wallows' Blog,
     What A Wonderful World, Now I am become Death, the destroyer of worlds, Morgan's Blog,
    hello my name is justin., The Perfect Vent, The Stearns Family, Room 19's Blog 2016,
    aubade, bittersweet, isyeli's, A Day in the Life of...Me!!, ]
**************************
[]
**************************
[Revolver USA Distribution & Midheaven mailorder, T H E V O I D S, ]
**************************
[SPIN IT RECORDS Moncton 467A Main Street Moncton NB CANADA, ]
**************************
[]
**************************
[It'll Glow On You, GLI Press, Bonjour Girl, Bleak Bliss, Paulina Gamero. Media Studies A2,
    The Jeopardy of Contentment, World Of Pearl Jam Bootlegs, ]
**************************
[]
**************************
[F-Measure, ]
**************************
```

```
[]
**************************
[Who needs a TV?, If You Give a Girl a Camera..., earenjoy, n o ponho m sica, FlowRadio
    Playlists (and Blog), ]
**************************
[Oh Yes J nsi!!, Visual Feeling, DaveCromwell Writes, A2 MEDIA COURSEWORK JOINT BLOG, *
    Sixeyes: by Alan Williamson, The World's First Internet Baby, ]
**************************
[Spinitron Charts, LOST  PLACES, Green Eggs and Ham Mondays 8-10am, IoTube      :),
    Unexpectedly Bart (King!), Lo importante es que estes t  bien, Out And Down In The
    Colonies, |[Tu queres ver isto]|, Friday Night Dream,
                    , adrianoblog, fractalpress.gr, ]
**************************
[MarkFisher's-MusicReview, Stories From the City, Stories From the Sea, "DANCING IN CIRCLES
    ", ]
**************************
[Abu Everyday, ]
**************************
[The Music Binge, Indie Top 20 - The Blog!, Cuz Music Rocks, i'm in too truthful a mood,
    SunStock Music, On Warmer Music, Floorshime Zipper Boots, The Girl at the Rock Show,
    MAGGOT CAVIAR, Stephanie Veto Photography, MPC, Mile In Mine, Encore, MTJR RANTS & RAVES
     ON MUSIC, turnitup!, Steel City Rust, A to Zappa - Song of the day, Eli Jace | The Mind
     Is A Terrible Thing To Paste, The Stark Online, Pithy Title Here, Myopiamuse, ]
**************************
[Web Science and Digital Libraries Research Group,                    , ORGANMYTH,
    dazey rosie, ]
**************************
[A H T A P O T, In the Frame Film Reviews, ]
**************************
[]
**************************
[macthemost, Did Not Chart, Hip In Detroit, ., Primitive Offerings, MarkEOrtega's Journalism
     Portfolio, Some Call It Noise...., My Name Is Blue Canary, www.doginasweater.com Live
    Show Review Archive, ]
**************************
[Fran Brighton, ]
```

Listing 8: K-means clustering with a value of 20

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

To solve this question I again used the code provided by the Programming Collective Intelligence book as shown in Listing 5 to create a method called *mds*. The *mds* method again reads my blog-term matrix and this time uses the method *scaledown* in the clusters.py library. I modified the *scaledown* method to also return the iteration count so it could be saved along with the blog title. The *mds* method then utilizes the data it takes from my blog-term matrix and utilizes multidimensional scaling (MDS) to create a visual representation of the distance matrix in two dimensions created from the *scaledown* method. The MDS visualization is shown in Figure 2. The iteration count was 209 as shown in Figure 3.

Figure 2: Two dimensional representation of blogs

```
3098.11685568
3097.82126325
3097.51767443
3097.27480397
3097.06770676
3096.86875752
3096.67643345
3096.46641474
3096.25083578
3096.02975133
3095.80500667
3095.5724262
3095.32195761
3095.12994981
3094.9459891
3094.78174481
3094.62723437
3094.48607808
3094.34287719
3094.24528186
3094.20318161
3094.15606151
3094.12051491
3094.06563715
3093.98029746
3093.90577805
3093.85371607
3093.81960369
3093.75527808
3093.6760618
3093.6560641
3093.61960957
3093.5834691
3093.56244968
3093.54188991
3093.52311775
3093.49468353
3093.50415866
Iteration count: 209
```

Figure 3: Command line view of the iteration count for MDS