RACHEL MCCREARY
CS 432 ASSIGNMENT 10 REPORT

PART 1

Using the data from Assignment 8 (under `A8/data/blogdata.txt` on GitHub), I considered each row in the blog-term matrix as a 500 dimensional vector, corresponding to a blog. From `numpredict.py` in chapter 8 of Programming Collective Intelligence, I replaced `numpredict.euclidean()` under `getdistances()` with cosine as the distance metric (`scipy.spatial.distance.cosine`) to compute the cosine between vectors of 500 dimensions. I also added a way to print out the titles of the closest blogs:

```
46  def getdistances(data, blognames, vec1):
47    distancelist=[]
48
49    # Loop over every item in the dataset
50    for i in range(len(data)):
51      vec2=data[i]
52
53      # Add the distance and the index
54      #distancelist.append((euclidean(vec1,vec2),i))
55      distancelist.append((scipy.spatial.distance.cosine(vec1,vec2),i))
56
57    # Sort by distance
58    distancelist.sort()
59    return distancelist
60
61  def knnestimate(data, blognames, vec1,k=5):
62    # Get sorted distances
63    dlist=getdistances(data, blognames, vec1)
64    titles=[]
65    avg=0.0
66    #print dlist
67
68    # Take the average of the top k results
69    for i in range(k+1):
70      idx=dlist[i][1]
71      if (data[idx] == vec1):
72          pass
73      else:
74        titles.append(blognames[idx])
75        for j in data[idx]:
76          avg += j
77        avg=avg/k
78    return 'average: ' + str(avg) + ' ' + str(titles)
```

I used `knnestimate()` to compute the nearest neighbors for both:

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

Printing out the average and titles for k = { 1, 2, 5, 10, 20 }:

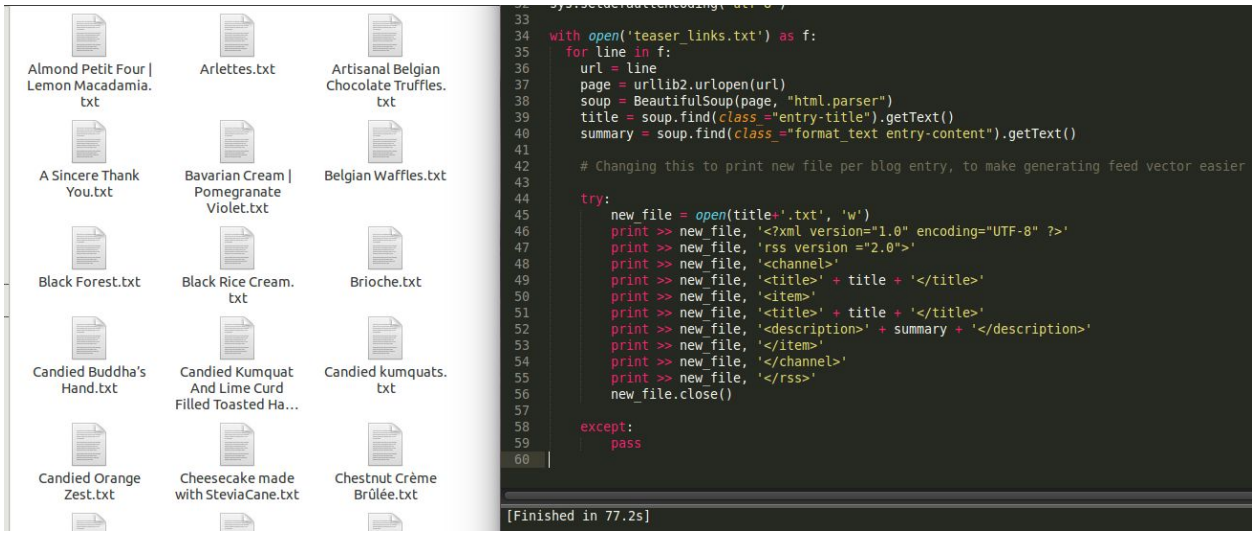
numpredict.py | part_1.py

```
import numpredict
import clusters

if __name__ == "__main__":

  blognames,words,data=clusters.readfile('blogdata.txt')

  print 'F-Measure' # in data[75]
  #print 'distance list: ' + str(numpredict.getdistances(data, blognames, data[75]))
  print 'k = 1 ' + str(numpredict.knnestimate(data, blognames, data[75], k = 1))
  print 'k = 2 ' + str(numpredict.knnestimate(data, blognames, data[75], k = 2))
  print 'k = 5 ' + str(numpredict.knnestimate(data, blognames, data[75], k = 5))
  print 'k = 10 ' + str(numpredict.knnestimate(data, blognames, data[75], k = 10))
  print 'k = 20 ' + str(numpredict.knnestimate(data, blognames, data[75], k = 20))

  print '\nWeb Science and Digital Libraries Research Group' # in data[7]
  #print 'distance list: ' + str(numpredict.getdistances(data, blognames, data[7]))
  print 'k = 1 ' + str(numpredict.knnestimate(data, blognames, data[7], k = 1))
  print 'k = 2 ' + str(numpredict.knnestimate(data, blognames, data[7], k = 2))
  print 'k = 5 ' + str(numpredict.knnestimate(data, blognames, data[7], k = 5))
  print 'k = 10 ' + str(numpredict.knnestimate(data, blognames, data[7], k = 10))
  print 'k = 20 ' + str(numpredict.knnestimate(data, blognames, data[7], k = 20))

```

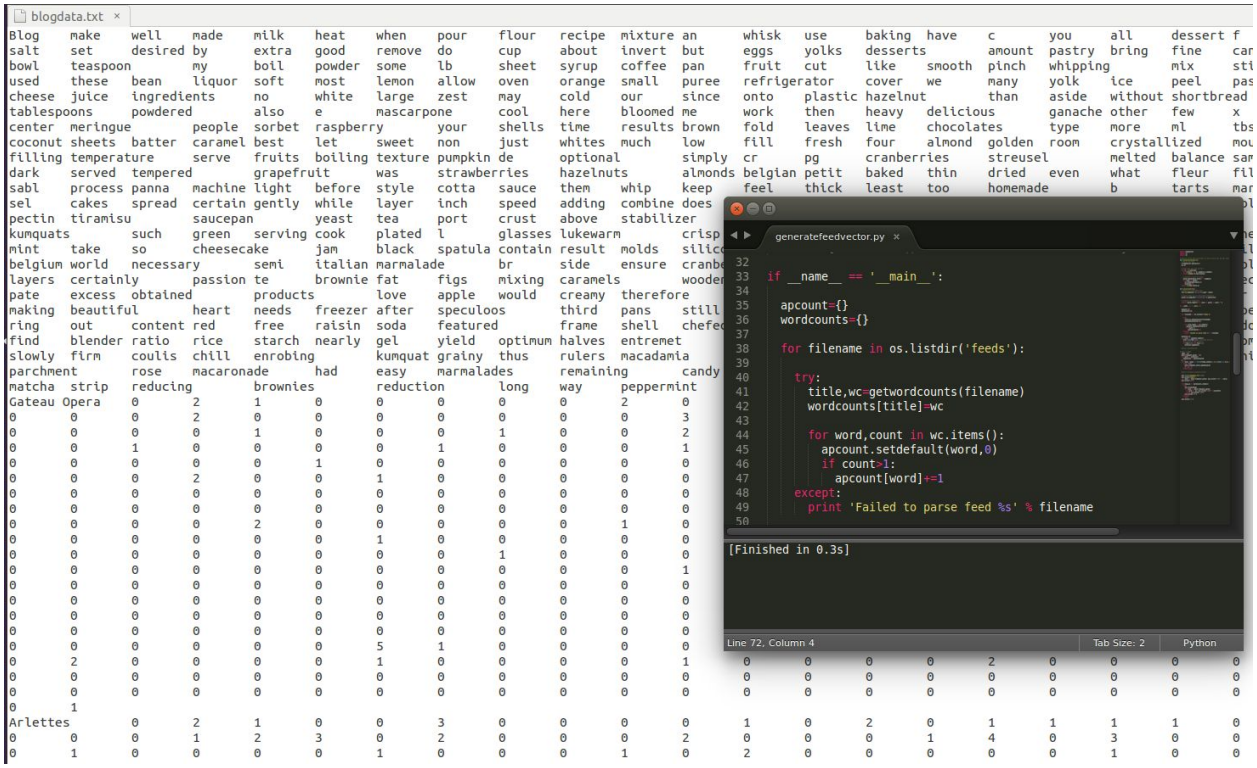
```
F-Measure

k = 1 average: 6921.0 ['notes from caroline', 'Gingey Bites']
k = 2 average: 1698.75 ['notes from caroline', 'Gingey Bites', 'The KM Projects']
k = 5 average: 915.986368 ['notes from caroline', 'Gingey Bites', 'The KM Projects', 'Web Science and
Digital Libraries Research Group', 'Every good and perfect gift is from above.', 'Just Belle']
k = 10 average: 49.1275767618 ['notes from caroline', 'Gingey Bites', 'The KM Projects', 'Web Science and
Digital Libraries Research Group', 'Every good and perfect gift is from above.', 'Just Belle', 'Madh Mama',
'Did Not Chart', 'Cold Knowledge', 'OXV234', "Toxicbreed's Funhouse"]
k = 20 average: 8.73731386995 ['notes from caroline', 'Gingey Bites', 'The KM Projects', 'Web Science and
Digital Libraries Research Group', 'Every good and perfect gift is from above.', 'Just Belle', 'Madh Mama',
'Did Not Chart', 'Cold Knowledge', 'OXV234', "Toxicbreed's Funhouse", 'Floorshime Zipper Boots',
'Adventures of a London Kiwi', 'a blog', 'jaaackie.', 'The Songbird', 'audriestorme', 'Green Eggs and Ham
Mondays 8-10am', 'whenyousaidtulips', 'rilphly.com', 'heal me donald']

Web Science and Digital Libraries Research Group
k = 1 average: 2442.0 ['Every good and perfect gift is from above.']
k = 2 average: 2625.0 ['Every good and perfect gift is from above.', 'Gingey Bites']
k = 5 average: 315.52384 ['Every good and perfect gift is from above.', 'Gingey Bites', 'notes from
caroline', 'Just Belle', 'The KM Projects']
k = 10 average: 178.065516093 ['Every good and perfect gift is from above.', 'Gingey Bites', 'notes from
caroline', 'Just Belle', 'The KM Projects', 'Madh Mama', 'Did Not Chart', 'OXV234', 'Cold Knowledge', 'a
blog']
k = 20 average: 6.78139294219 ['Every good and perfect gift is from above.', 'Gingey Bites', 'notes from
caroline', 'Just Belle', 'The KM Projects', 'Madh Mama', 'Did Not Chart', 'OXV234', 'Cold Knowledge', 'a
blog', 'Adventures of a London Kiwi', "Toxicbreed's Funhouse", 'papersouldesign', 'Floorshime Zipper
Boots', 'The Songbird', 'jaaackie.', 'Danjjroberts', 'audriestorme', 'heal me donald', 'rilphly.com']
[Finished in 0.4s]
```

PART 2

For this part, to make a similar `blogdata.txt` like in A8, I decided to print out a separate “feed” for each of the blog entries from A9 (using the same links under `A9/data/teaser_links.txt` on GitHub, since I had to make a fake feed last time) so it would be easier to plug them into generatefeedvector.py:



Generating feed vectors (making it go through each rss in a directory rather than a list of URIs like in A8):



This step was only meant for getting the word counts and most frequent words for only one blog. For the actual categorizing, I planned on using the same fake rss I made in A9 (under `A9/data/full_rss.txt` on GitHub), but ended up having to stop here.