Assignment 7

Joshua Graham

April 30, 2018

Abstract

All Scripts used in the assignment can be found in the A9 folder, if needed.

0.1 Problem 1 Process

I had a lot of trouble with this assignment. The first hard part I tackled was figuring out how I was going to link name and index. I cannot remember where I saw it, it was probably online but I cannot find the reference I used. I figured out I could do it with:

Linking by index

```
1 #print(blogs.index("F-Measure"))
```

The second tackle I had was fixing numpredict to stop throwing errors on strings. After reading the error dialog, I found I just had to get rid of the section argument in the list inside of numpredict.py. Below is the changed function:**I believe all that changed was vec2=data[i]**. This previously had a ["title"] argument beside it.

Linking by index

```
1 def getdistances(data,vec1):
2   distancelist=[]
3
4   # Loop over every item in the dataset
5   for i in range(len(data)):
6     vec2=data[i]
7
8     # Add the distance and the index
9     distancelist.append((cosine(vec1,vec2),i))
10
11  # Sort by distance
12  distancelist.sort()
13  return distancelist
```

I implemented the cosine function manually instead of using one of the many math pacakges because it was a short method and I'm running off a Virtual Machine after my laptop died. I am not happy with this method, this is where I thought I could get rid of the title matching itself as a neighbor. The function is shown below:

Linking by index

```
1
2 def cosine(v1,v2):
3     sumTop=0.0
4     for i in range(len(v1)):
5         sumTop += v1[i]*v2[i]
6     sumBottom1 = 0.0
7     sumBottom2 = 0.0
8     for i in range(len(v1)):
9         sumBottom1 += v1[i] ** 2
10        sumBottom2 += v2[i] ** 2
11    if(sumBottom1 * sumBottom2 == 0):
12        return 0
13    else:
14        return (sumTop)/((sumBottom1 * sumBottom2)**0.5)
```

I ended up being able to ignore the specific entry that matches itself by only viewing the second to last k values in the list. I relearned that negative indices are a wonderful tool, and it made my data correctly match the intent of the assignment.

knnestimate()

```
def knnestimate(data,vec1,k=5):
  # Get sorted distances
   dlist=getdistances(data,vec1)
   neighbors = dlist[-k-1:-1]
   return neighbors
```

Lastly, the code can be run with the run.py script, shown below. I went through several iterations of this trying to figure out what worked best. In the end I went for readability.

Linking by index

```
import clusters
import numpredict
def findNeigh(i, data, k):
    testing = data[i]
    neighbors = numpredict.knnestimate(data, testing, k)
    for i in neighbors:
        print(blogs[i[1]])


blogs, text, data = clusters.readfile("blogdata1.txt")
#print(blogs.index("F-Measure"))
#findNeigh(blogs.index("F-Measure"), data, 5)
for name in "F-Measure", "Web Science and Digital Libraries Research Group":
    for k in 1, 2, 5, 10, 20:
        print("Running", name, "against k =", k)
        findNeigh(blogs.index(name), data, k)
        print("\n\n")
```

0.2 Problems with the code

Both examples found the same nearest neighbor. I believe this is because the data is bad, not because the code is. If I had to do it again, I'd grab a new dataset. Viewing the data for that entry, I discovered that it had a 0 for everything except one word. This led me to believe I did not have the actual contents of the blog downloaded correctly, so I removed it from the dataset. This still left me with over the required amount of blogs.

0.3 Example Output

I've left the entire output in the file output.txt. However, I clipped F-Measure's neighbors with k=5 below.

Running F-Measure against k = 5
The World's First Internet Baby
CardrossManiac2
SPIN IT RECORDS Moncton 467A Main Street Moncton NB CANADA
F-Measure
Abu Everyday