CS432 Web Science: Assignment 9

Finished on April 22, 2020

Dr. Michele C. Weigle

Hannah Holloway
hhollowa@odu.edu

Contents

List of Figures

Listings

Problem 1

Question

Q1. Using your data from HW7 (Twitter account-term matrix):

Consider each row in the account-term matrix as a 1000 dimension vector, corresponding to a Twitter account.

Modify getdistances() to use the cosine distance metric instead of Euclidean distance (see Euclidean vs. Cosine Distance on why we use cosine for documents). You may want to check out SciPy's cosine distance function.

Use a modified version of knnestimate() from Ch 8 to compute the 5 nearest neighbors for 3 of the accounts in your list (pick any 3).

Hint 1: Just return the distances, not some computed average (there's nothing to compute the average of anyway) Hint 2: Since the account you pick will be in your training set, that account should always be reported as the top nearest neighbor with a distance of 0. When reporting the 5 nearest neighbors, omit the account that is being tested (but still report 5 other neighbors).

Answer

In order to compute the 5 nearest neighbors for each of my chosen witter accounts, I needed to retrieve my userdata matrix from assignment 7 and process it with my script in Listing 1. The script creates a vector for each account which can be given as input to the following script (Listing 2). I got the processed user matrix as output in the terminal after running the script. Sample userdata matrix computed from the script can be found in Figure 1.

```
import json

input=open('userdata.txt','r')
output=open('rowassign','w')
flag=0
row=[]
for i in input:
    flag=flag+1
    if flag >1:
        dictionary={}
        i=i.strip()
        drow=i.split('\t')
        name=drow[0]
        drow.pop(0)
        rowassign=drow
        print name
        print rowassign
        dictionary[name]=rowassign #assigning each row vector to a user
        row.append(dictionary)
output.write(json.dumps(row))
```

Listing 1: Account Vector Script

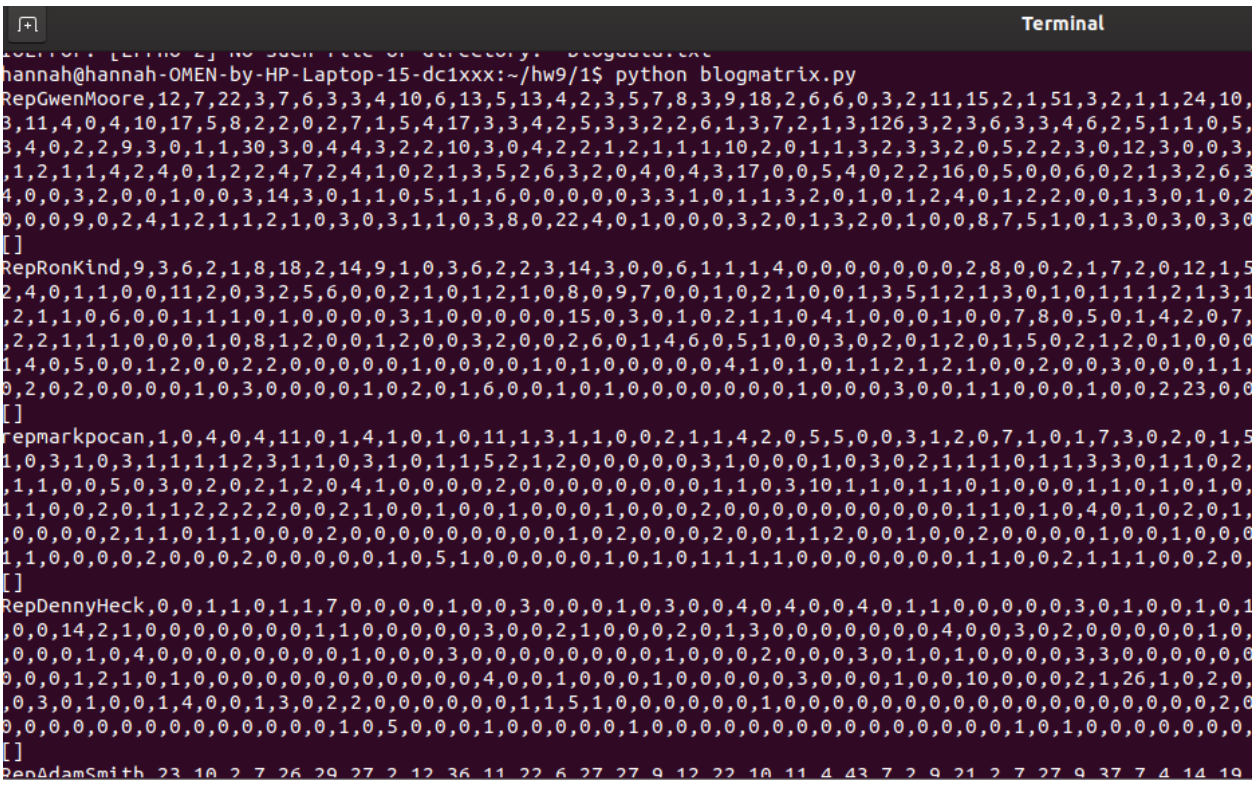

Figure 1: UserData Matrix Sample

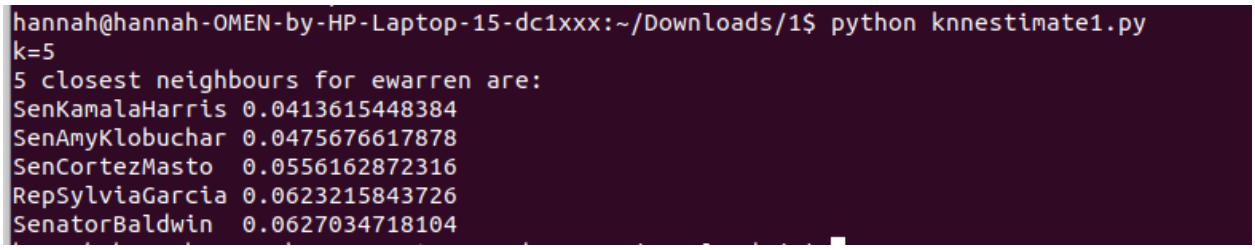

Figure 2: EWarren 5 closest Neighbors

Next, I removed the Euclidean function and replaced it with the cosine function as a distance metric. This was used to find the cosine between 1000 dimensions. The knnestimate() function's purpose was to find the neighbors for a particular account which takes input as k=1, 2, 3, 4 or 5. So, for each time it is given a k value the script returns the respective k number of neighbors for that particular twitter account.

I chose Senator Warren, Representative Sean Maloney and Representative John Lewis as my 3 twitter account choices. The 5 nearest neighbors for "ewarren" can be found in Figure 2. The 5 nearest neighbors for "RepSeanMaloney" can be found in Figure 3. The 5 nearest neighbors for "repjohnlewis" can be found in Figure 4.

```
hannah@hannah-OMEN-by-HP-Laptop-15-dc1xxx:~/Downloads/1$ python knnestimate1.py
k=5
5 closest neighbours for RepSeanMaloney are:
RepTedDeutch    0.0464412143282
SenBobCasey     0.048032034582
ChuckGrassley   0.0582832614182
SenatorTomUdall 0.0585226187351
SenRubioPress   0.0587551983649
```

Figure 3: RepSeanMaloney 5 closest Neighbors

```
hannah@hannah-OMEN-by-HP-Laptop-15-dc1xxx:~/Downloads/1$ python knnestimate1.py
k=5
5 closest neighbours for repjohnlewis are:
CoryBooker      0.0233584685339
RepGwenMoore    0.0394837818105
RepAnthonyBrown 0.0591862348825
RepAndreCarson  0.061898446059
RepBobbyRush    0.0782960292686
```

Figure 4: Repjohnlewis 5 closest Neighbors

```
from random import random,randint
import math
import json


def main():
    input= open('rowassign','r')
    userdata = json.load(input)
    for line in userdata:
        for nline in line:
            if nline == 'ewarren':
                vec1= line[nline]
                knnestimate(userdata,vec1)



def getdistances(userdata,vec1):
  distancelist=[]

  for i in userdata:
    for nline in i:
      if nline != 'F-Measure':
        vec2= i[nline]

    distancelist.append((cosineDistance(vec1,vec2),i))

  distancelist.sort()

  return distancelist


def cosineDistance(v1,v2):
  sumxx, sumxy, sumyy = 0, 0, 0
  for i in range(0,len(v1)-1):
    x = int(v1[i]); y = int(v2[i])
    sumxx += x*x
    sumyy += y*y
    sumxy += x*y
  return sumxy/math.sqrt(sumxx*sumyy)
```

```
40
def knnestimate(data,vec1,k=5):
  print 'k=5'
45  print "5 closest neighbours for ewarren are:"
  dlist=getdistances(data,vec1)
  avg=0.0
  # Take the average of the top k results
  for i in range(k):
50    idx=dlist[i]
    value = idx[0]
    for item in idx[1]:
        username= item
    print username +'\t'+ str(value)
55

main()
```

Listing 2: Python Code for finding neighbors

Problem 2

Question

For each of the accounts you chose, do the 5 nearest neighbors chosen by the algorithm make sense? What name (classification) would you give that group of accounts?

You will not get full credit on this assignment if you just report the list of neighbors generated. You must describe the changes you made to the code, how the process works, and your assessment of how well the kNN algorithm performed.

Answer

Yes, all of the closest neighbors emded up making sense. After retrieving ewarren's 5 nearest neighbor accounts, I looked up each account name on twitter to quickly scan over each of their timelines. I kept in mind the smaller the decimal the closer the neighbor. It made perfect sense as to why the 5 users were the closest to ewarren. All the users were female and associated themselves with the Democratic Party. They all recently tweeted about similar topics such as healthcare and financial crisis caused by COVID19. I would classify the group "female democratic politicians". Next, I looked up RepSeanMaloney's 5 nearest neighbor accounts, and it was no surprise again, that the accounts all seemed similar to Sean with their views, similar age and gender. They were all middle aged white males associated with the Democratic Party, which makes sense as to why they would share common tweet terms with him. I would classify this group as "middle-aged white democratic male politicians". Lastly, I looked up RepJohnLewis's nearest neighbor accounts, and they all seemed similar to him as well. They were all black, democratic politicians that seemed like they had similar personalities and recently tweeted about similar topics. I would classify this group as "black male democratic politicans".

As far as changing the code, I changed the if statement nline to the accountname (ewarren, RepJohnLewis and RepSeanMaloney) for each time running the script. I would also change the label printed out to the specific account's user name each time printing the output. I believe the kNN algorithm's predicition was pretty accurate. All of the nearest neighbors were similar to the account specified after checking, and I could definitly see manually the similar terms the corresponding neighbors tweeted about recently.

Extra Credit: Problem 3

Question

Answer

Extra Credit: Problem 4

Question

Answer

References

[1] https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/ .

[2] http://jurgens.people.si.umich.edu/tutorials .

[3] https://towardsdatascience.com/another-twitter-sentiment-analysis-with-python-part-5-50b4e87d9bdd .