CS532 - Homework 9

Himarsha Jayanetti

Spring 2020

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

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

I have created a function named "create_vector()" to handle the creation of a vector considering each row in the account-term matrix as a 1000 dimension vector which corresponds to a Twitter account. The output file from HW7, "account_term_matrix" is used to create this data vector.

```
1  import numpy as np
2
3  def create_vector():
4      allrows = []
5      vector_list = []
6      filename = ("account_term_matrix")
7      with open(filename, "r") as f:
8          rows = f.readlines()
9          list = []
10         rows.pop(0) #Delete the first row - word list
11         for row in rows: #For each row in the matrix
12             eachrow = []
13             list = row.split("\t")
14             list.pop(0)
15             for item in list: #Each item/value in a row
16                 item = item.strip("\n")
17                 eachrow.append(item)
18             allrows.append(eachrow)
19         vector_rows = np.array(allrows, dtype='float64')
20     return vector_rows
```

1.2 Modify getdistances() to use the cosine distance metric instead of Euclidean distance.

I have modified the "getdistances()" function to use the cosine distance metric instead of Euclidean distance (reference 1). I used the scipy distance function to compute cosine distance between each pair of the two vectors. Below piece of code shows how it was done.

```
from scipy.spatial import distance

def cosine_d(v1,v2):
  d = distance.cosine(v1, v2) #Cosine distance
  return d

def getdistances(data,vec1):
  distancelist=[]
  # Loop over every item in the dataset
  for i in range(len(data)):
    vec2=data[i]
    # Add the distance and the index
    distancelist.append((cosine_d(vec1,vec2),i))
  # Sort by distance
  distancelist.sort()
  return distancelist
```

The "getdistances()" function will take the data vector and the target vector as parameters. It will loop over each item in the data vector, compute the distance between each item and the target vector, and append those distances and the index as a tuple into a list. Finally, this distances list is sorted in the increasing order of distances and returned.

Note: I had to make sure that the data type was set to "float" when creating the vector by setting "dtype='float64' ". I encountered with an error when computing cosine distance, if this type conversion was not done (reference 2).

1.3 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).

I have modified the "knnestimate()" function to compute the 5 nearest neighbours for a particular account. The steps are as below:

Step 01: Get the sorted distances list using the "getdistances()" function.

Step 02: Get the top 6 tuples with distance index and print the values (top 6 as I am including the account being tested).

Step 03: Remove the top most tuple from the list (to remove the account being tested and get the top 5 nearest neighbors) and return it.

I have modified the "knnestimate()" by removing the steps related to computing the average and adding the above steps to get the top 5 nearest neighbours.

As mentioned in one of the hints, the function now returns the distances, not some computed average since there's nothing to compute the average of anyway.

The part of code handling the above is as below:

```
def knnestimate(data,vec1,k=5):
  # Get sorted distances
  dlist=getdistances(data,vec1)
  top = dlist[:6]
  print("The top five tuples with distance & index (including the account being tested):")
  print(top)
  print("")
  print("Top 5 nearest neighbors:")
  top_five =[]
  for each in top:)
    top_five.append(each[1])
  top_five.pop(0) #Remove the account being tested
  return top_five

def get(X, I):
    screen_name_list = []
    with open("popular_users", "r") as f1:
        content = f1.readlines()
    for item in content:
        screen_name_list.append(item.strip(" \n"))
    items_as_dict = dict(zip(screen_name_list,range(0,len(screen_name_list)))) #Dictionary
    inv_dict = dict(zip(items_as_dict.values(),items_as_dict.keys())) #Inverse Dictionary
    if I == True:
        Y = items_as_dict[X] # Y = index
    else:
        Y = inv_dict[X] # Y = screen name
    return Y


if __name__ == "__main__":
    data = create_vector()
    account = "HillaryClinton" #Enter the account screen name for which you need to find
        nearest neighbors for
    print("Computing top 5 nearest neighbors for the account:")
    print(account)
    print("")
    index = get(account, I = True)  #To get the index when the account screen name is given
    #print(index)
    vec1 = data[index]
    top_five = knnestimate(data,vec1,k=5)
    for index in top_five:
        user = get(index, I = False) #To get the account screen name when the index name is
            given
        print(index, user)
```

I have created a function "get()" with the intention of mapping the account index to the screen name. This function is created in a way that if we provide the index of the account, it will return the screen name and vice versa.

The top 5 nearest neighbors for the 3 accounts I picked are as follows. I have also printed out the top 6 tuples with distances and their index. You may notice that for each of the accounts I picked, that particular account is reported as the top nearest neighbor with a distance of 0 since the account selected is in the training set.

Screenshot 1: Harvard kNN.py

```
himarsha@marsh:~/Documents/CS532 Web Science/HW9$ python3 kNN.py
Computing top 5 nearest neighbors for the account:
Harvard

The top five tuples with distance & index (incuding the account being tested):
[(0.0, 16), (0.4714746531726911, 17), (0.5170000900643892, 92), (0.5231624644765748, 18), (0.5643878373211904, 6),
 (0.5733812207588661, 64)]

Top 5 nearest neighbors:
17 Stanford
92 ODU
18 USouthFlorida
6 nytimes
64 NEHgov
```

Screenshot 2: GooglePlay kNN.py

```
himarsha@marsh:~/Documents/CS532 Web Science/HW9$ python3 kNN.py
Computing top 5 nearest neighbors for the account:
GooglePlay

The top five tuples with distance & index (incuding the account being tested):
[(0.0, 1), (0.3958478770001993, 2), (0.6556965484784911, 39), (0.6741136175464686, 3), (0.6861607940540531, 93),
 (0.7294397696402646, 65)]

Top 5 nearest neighbors:
2 googlechrome
39 igrigorik
3 Android
93 googledevs
65 KHayhoe
```

Screenshot 3: HillaryClinton kNN.py

```
himarsha@marsh:~/Documents/CS532 Web Science/HW9$ python3 kNN.py
Computing top 5 nearest neighbors for the account:
HillaryClinton

The top five tuples with distance & index (incuding the account being tested):
[(0.0, 84), (0.3855552554881543, 60), (0.4038604355702877, 13), (0.4286898309370596, 81), (0.4364088034630611, 61),
 (0.43929542718934544, 73)]

Top 5 nearest neighbors:
60 timkaine
13 realDonaldTrump
81 ewarren
61 MarkWarner
73 KevinMKruse
```

2 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?

For each of the three accounts, we got the nearest five neighbours. They are similar to each other & can be classified as one group. By simply looking at the results, it is clear that the algorithm has done a good job.However, not all 5 accounts can be classified under one name although they have their similarities. But, I have tried my best to give the most appropriate classification name.

2.1 Harvard

The nearest five neighbours:

1. Stanford
2. ODU
3. USouthFlorida
4. nytimes - The New York Times
5. NEHgov - National Endowment For The Humanities Federal Government Office.

Classification: Educational, Government & News Institutes.

2.2 GooglePlay

The nearest five neighbours:

1. googleChrome
2. igrigorik - Ilya Grigorik - Developer advocate and web performance engineer at Google
3. Android
4. googledevs
5. Khayhoe - Prof. Katharine Hayhoe - An atmospheric scientist.

Classification: Technology & Science

2.3 HillaryClinton

The nearest five neighbours:

1. timkaine - Tim Kaine - U.S Senator
2. realDonaldTrump - Donald Trump - President
3. ewarren - Elizabeth Warren - U.S Senator
4. MarkWarner - Mark Warner - U.S Senator
5. KevinMKruse - Prof.Kevin M. Kruse - Political, social, & urban/suburban American history.

Classification: Political

3 References

1. https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.cosine.html
2. https://stackoverflow.com/questions/42013903/typeerror-ufunc-multiply-did-not-contain-a-loop-with-signature-matching-types
3. https://stackoverflow.com/questions/35640780/python-fastest-way-to-find-indexes-of-item-in-list
4. https://stackoverflow.com/questions/24914412/python-how-to-get-first-5-or-last-5-from-list-of-10