Web Science Assignment 9 Report

Dr. Nelson

James Pindell

4/14/2018

Content

General Info

Support your answer: include all relevant discussion, assumptions,

examples, etc.

Problem 1

Using the data from A7:

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

- Use knnestimate() to compute the nearest neighbors for both:

a. http://f-measure.blogspot.com/

b. http://ws-dl.blogspot.com/

for k={1,2,5,10,20}.

Use cosine distance metric (chapter 8) not euclidean distance. So you have to implement numpredict.cosine() instead of using numpredict.euclidean() in:

https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter8/numpredict.py

Solution

The solution for this problem is outlined by the following steps:

1. **Modify Blog Data**: In order to use the data from A7, the data was copied from the A7 package and modified, where all of the words within the matrix were removed, creating the necessary numerical vectors. Unfortunatly, the matrix is too large to show as a snippet, but you can find the matrix, called Blog Data.txt, from within the package.

2. **Modify Code:** In order to use the knnestimate() to compute the nearest neighbors for both http://f-measure.blogspot.com/ and http://ws-dl.blogspot.com/, the code provided needed to be modified, where the cosine distance metric was used instead of the euclidean distance. In other words, numpredict.cosine() was implemented instead of numpredict.euclidean(). You can find the code file, called NumPredict.py, from within the package, but a snippet of the code is shown below:

```
import math
import re

'''
Original Code: https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter8/numpredict.py
'''

def cosine(vec1,vec2):
    'compute cosine similarity of vec1 to vec2: (vec1 dot vec2)/{||vec1||*||vec2||)'
    sumxx, sumxy, sumyy = 0, 0, 0
    for i in range(0, 958):
        x = vec1[i]; y = vec2[i]
        sumxx += x*x
        sumyy += y*y
        sumxy += x*y
    return sumxy/math.sqrt(sumxx*sumyy)


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(vec1,vec2)))

  # Sort by distance
  distancelist.sort()
```

3. **Run Code**: In order to actually run the modified code, a separate code file was created. You can find the code file, called KNN.py, from within the file, but a snippet of the code is shown below:

```
import NumPredict
import math
import re

file = open('Blog Data.txt')
data = []
for line in file:
  line = line.strip()
  data.append([int(x) for x in line.split('\t')])
vectorA = data[0]
vectorB = data[1]

print('Results for vectorA:')
result = NumPredict.knnestimate(data, vectorA, 1)
print('kNN estimate of k = 1: ' + str(result))

result = NumPredict.knnestimate(data, vectorA, 2)
print('kNN estimate of k = 2: ' + str(result))

result = NumPredict.knnestimate(data, vectorA, 5)
print('kNN estimate of k = 5: ' + str(result))

result = NumPredict.knnestimate(data, vectorA, 10)
print('kNN estimate of k = 10: ' + str(result))

result = NumPredict.knnestimate(data, vectorA, 20)
print('kNN estimate of k = 20: ' + str(result))

print(' ')
```

4. **Similarity Results**: Based upon the results that the code produces, it can be concluded that neither http://f-measure.blogspot.com/ nor http://ws-dl.blogspot.com/ are similar to any of the other blogs from within the data, for k = {1, 2, 5, 10, 20}, since all of the results are extremely close to 0. A snippet of the results is shown below:

```
C:\Users\James Pindell\Desktop\CS 432\Assignment 9>python KNN.py
Results for vectorA:
kNN estimate of k = 1: 0.004022785219121585
kNN estimate of k = 2: 0.006034177828682377
kNN estimate of k = 5: 0.03724638647412336
kNN estimate of k = 10: 0.0573013788409147
kNN estimate of k = 20: 0.08327266926448677

Results for vectorB:
kNN estimate of k = 1: 0.008467694960173217
kNN estimate of k = 2: 0.011493383662572996
kNN estimate of k = 5: 0.049190282333269955
kNN estimate of k = 10: 0.07410624906558902
kNN estimate of k = 20: 0.10005092537740934
```