CSE-3024 Web Mining

Lab Assignment 7

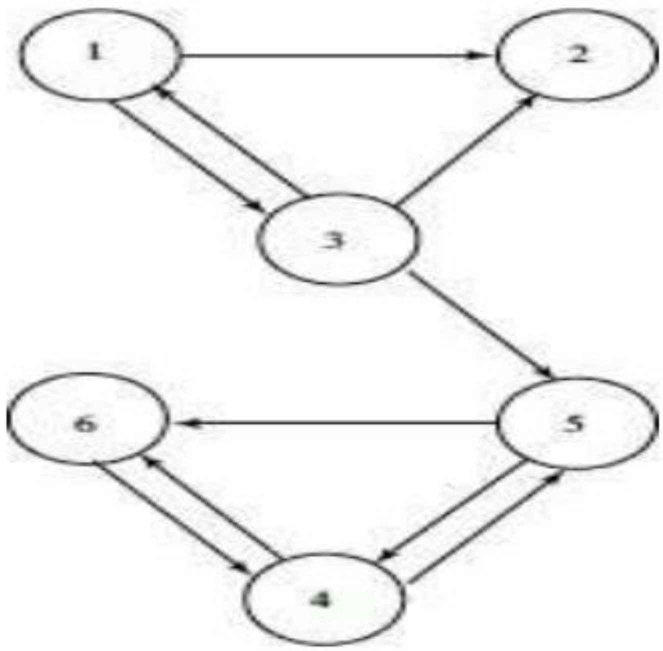
Alokam Nikhitha

19BCE2555

Page Rank

Question

Write a python program to find the ranks for the given graph.



Perform 7 iteration and print the final iteration value only.

Problem statement:

Python program to find the Page Rank of all the nodes for the given Graph after 7 iterations.

Procedure:

-Firstly, we import the necessary libraries of numpy, scipy and sparse.

-Then write a compute page rank function, which takes in three input parameters, namely, links, damping factor and number of iterations.

-We initialize the damping factor to a standard 0.85 value and number of iterations to 7 as mentioned in question.

-Then we use that function to compute page rank of each page in our network.

Code:

```
#Alokam Nikhitha 19BCE2555
import scipy
from scipy import sparse
import numpy
def ComputePageRank(links, c=0.85, iteration = 7):
    count = 0
    ones = numpy.ones(len(links))
    sources = [x[0] for x in links]
    targets = [x[1] for x in links]
    n = max(max(sources), max(targets)) + 1
    HT = sparse.coo_matrix((ones, (targets, sources)), shape=(n, n))
    num_outlinks = numpy.array(HT.sum(axis=0)).flatten()
    HT.data /= num_outlinks[sources]
    d_indices = numpy.where(num_outlinks == 0)[0]
    r = numpy.ones(n) / n
    while True:
        previous_r = r
        r = c * (HT * r + sum(r[d_indices])/n) + (1.0 - c)/n
        # r.sum() ≈ 1 but prevent errors from adding up.
        r /= r.sum()
        count = count+1
```

```
if(count >iteration):
    #     if scipy.absolute(r - previous_r).sum() < epsilon:
    return r



print(ComputePageRank([(0,1), (0,2), (2,0),(2, 1),(2, 4),(3, 4),(3,5),(4,3),(4,5),
(5,3) ]))
```

Code Snippets and Outputs:

```
In [17]: #Alokam Nikhitha 19BCE2555
import scipy
from scipy import sparse
import numpy
def ComputePageRank(links, c=0.85, iteration = 7):
    count = 0
    ones = numpy.ones(len(links))
    sources = [x[0] for x in links]
    targets = [x[1] for x in links]
    n = max(max(sources), max(targets)) + 1
    HT = sparse.coo_matrix((ones, (targets, sources)), shape=(n, n))
    num_outlinks = numpy.array(HT.sum(axis=0)).flatten()
    HT.data /= num_outlinks[sources]
    d_indices = numpy.where(num_outlinks == 0)[0]
    r = numpy.ones(n) / n
    while True:
        previous_r = r
        r = c * (HT * r + sum(r[d_indices])/n) + (1.0 - c)/n
        # r.sum() ≈ 1 but prevent errors from adding up.
        r /= r.sum()
        count = count+1
        if(count >iteration):
            #     if scipy.absolute(r - previous_r).sum() < epsilon:
            return r


print(ComputePageRank([(0,1), (0,2), (2,0),(2, 1),(2, 4),(3, 4),(3,5),(4,3),(4,5), (5,3) ]))

[0.05276657 0.07551812 0.05864201 0.34644978 0.19951605 0.26710748]
```

we are importing our libraries. We import scipy, numpy and sparse from scipy. The computePageRank function returns a list of page ranks for each page in our network. It accepts three parameters as input: the links between our network's nodes, the damping factor, and the number of iterations. As stated in our query, we set the damping factor to a standard of 0.85 and the number of iterations to 7.

The page rank of each page in our network has been calculated here.

We renamed each node in our question because we didn't have a 0 node and counting in Python starts at 0. They are each decremented by one.

1 → 0   2 → 1   3 → 2   4 → 3   5 → 4   6 → 5

Results:

```
[0.05276657 0.07551812 0.05864201 0.34644978 0.19951605 0.26710748]
```

The page rank of each node in our network is as follows:

Node 1 → 0.05276657

Node 2 → 0.07551812

Node 3 → 0.05864201

Node 4 → 0.34644978

Node 5 → 0.19951605

Node 6 → 0.26710748

Sum of page rank of each node in our network is 1.

The nodes in decreasing order of page ranks are: Node 4 > Node 6 > Node 5 > Node 2 > Node 3 > Node 1