Old Dominion Univeristy

CS 495: Introduction to Web Science
Instructor: Michael L. Nelson, Ph.D
Fall 2014 4:20pm - 7:10pm R, ECSB 2120

Assignment # 4

GeorgeC. Micros UIN: 00757376

Honor Pledge

I pledge to support the Honor System of Old Dominion University. I will refrain from any form of academic dishonesty or deception, such as cheating or plagTFIDF TF IDF URIiarism. I am aware that as a member of the academic community it is my responsibility to turn in all suspected violations of the Honor Code. I will report to a hearing if summoned.

Signed _____

October 10, 2014

George C. Micros

Written Assignment 4

Fall 2014
CS 495: Introduction to Web Science
Dr. Michael Nelson

October 10, 2014

Contents

1 Written Assignment 4 5
1.1 Question 1 6
1.1.1 The Question 6
1.1.2 The Answer 6
1.2 Question 2 8
1.2.1 The Question 8
1.2.2 The Answer 8
1.3 Question 3 10
1.3.1 The Question 10
1.3.2 The Answer 11

References 17

Chapter 1
Written Assignment 4

1.1 Question 1

1.1.1 The Question

From your list of 1000 links, choose 100 and extract all of the links from those 100 pages to other pages. We're looking for user navigable links, that is in the form of:

<A href="foo">bar

We're not looking for embedded images, scripts, ¡link¿ elements, etc. You'll probably want to use BeautifulSoup for this.
For each URI, create a text file of all of the outbound links from that page to other URIs (use any syntax that is easy for you). For example:
site:
`http://www.cs.odu.edu/~mln/`
links:
`http://www.cs.odu.edu/`
`http://www.odu.edu/`
`http://www.cs.odu.edu/~mln/research/`
`http://www.cs.odu.edu/~mln/pubs/`
`http://ws-dl.blogspot.com/`
`http://ws-dl.blogspot.com/2013/09/2013-09-09-ms-thesis-http-mailbox.html`
etc.
Upload these 100 files to github (they don't have to be in your report).

1.1.2 The Answer

```
#! /usr/bin/python

import urllib2 as ul
from bs4 import BeautifulSoup as bs
import re

def getLinks(url):
 try:
  req = ul.Request(url);
  res = ul.urlopen(req);
  html = res.read();
  soup = bs(html);
  links = []
  for lks in soup.find_all('a'):
   temp = lks.get('href');
   #print temp[0:4]
   if ("http" == (temp[0:4])):
    g.write(temp+"\n");
    #h.write(line+" -> "+temp+"\n");
 except:
  print "ERROR: "+url+"\n"
  pass;

f = open("100","r");
#h = open("graph", "w");

#h.write("diagraph test {\n");
cnt = 0;
for line in f:
 cnt = cnt +1
 print cnt;

 g = open("./lks/"+"{0:0>3}".format(str(cnt)),"w+");
 g.write("site: \n"+str(line)+"links: \n");
 #print line;
```

```
36   getLinks(line);
37
38 #h.write("}");
```

Listing 1: Bash script that creates a lookup tables of URLs and their hashes

1.2 Question 2

1.2.1 The Question

Using these 100 files, create a single GraphViz "dot" file of the resulting graph. Learn about dot at:
Examples:
http://www.graphviz.org/content/unix
http://www.graphviz.org/Gallery/directed/unix.gv.txt

Manual:
http://www.graphviz.org/Documentation/dotguide.pdf

Reference:
http://www.graphviz.org/content/dot-language
http://www.graphviz.org/Documentation.php

Note: you'll have to put explicit labels on the graph, see:
https://gephi.org/users/supported-graph-formats/graphviz-dot-format/

(note: actually, I'll allow any of the formats listed here:

https://gephi.org/users/supported-graph-formats/

but "dot" is probably the simplest.)

1.2.2 The Answer

```
#! /bin/bash



echo "digraph G {" > graph.dot
#echo "center=true;" >> graph.dot
#echo "size=\"6,6\";" >> graph.dot
echo "ranksep=8;" >> graph.dot
echo "ratio=auto;" >> graph.dot
echo "overlap=false;" >> graph.dot
for file in lks/*
do
 #cnt=1
 #file="lks/001"
 echo $file
 a=$(head -n 2 $file | tail -n 1)
 alab=$(echo $a | sed -e 's|^[^/]*//||' -e 's|^www\.||' -e 's|/.*$||')
 echo "\"$a\" [label=\"$alab\"]" >> graph.dot

 tail -n+4 $file |
 {
 while read  line
 do
  cnt=$((cnt + 1))
  #echo $cnt
  line=$line
  llab=$(echo $line| sed -e 's|^[^/]*//||' -e 's|^www\.||' -e 's|/.*$||')
  echo "\"$line\" [label=\"$llab\"]" >> graph.dot
  echo "\"$a\" -> \"$line\";" >> graph.dot

 done
 }
done

echo "}" >> graph.dot
```

Listing 2: R script to generate a histogram

1.3 Question 3

1.3.1 The Question

Download and install Gephi:

`https://gephi.org/`

Load the dot file created in #2 and use Gephi to:

- visualize the graph (you'll have to turn on labels)
- calculate HITS and PageRank
- avg degree
- network diameter
- connected components

Put the resulting graphs in your report.

You might need to choose the 100 sites with an eye toward creating a graph with at least one component that is nicely connected. You can probably do this by selecting some portion of your links (e.g., 25, 50) from the same site.

1.3.2 The Answer

1.3.2.1 Network Graph

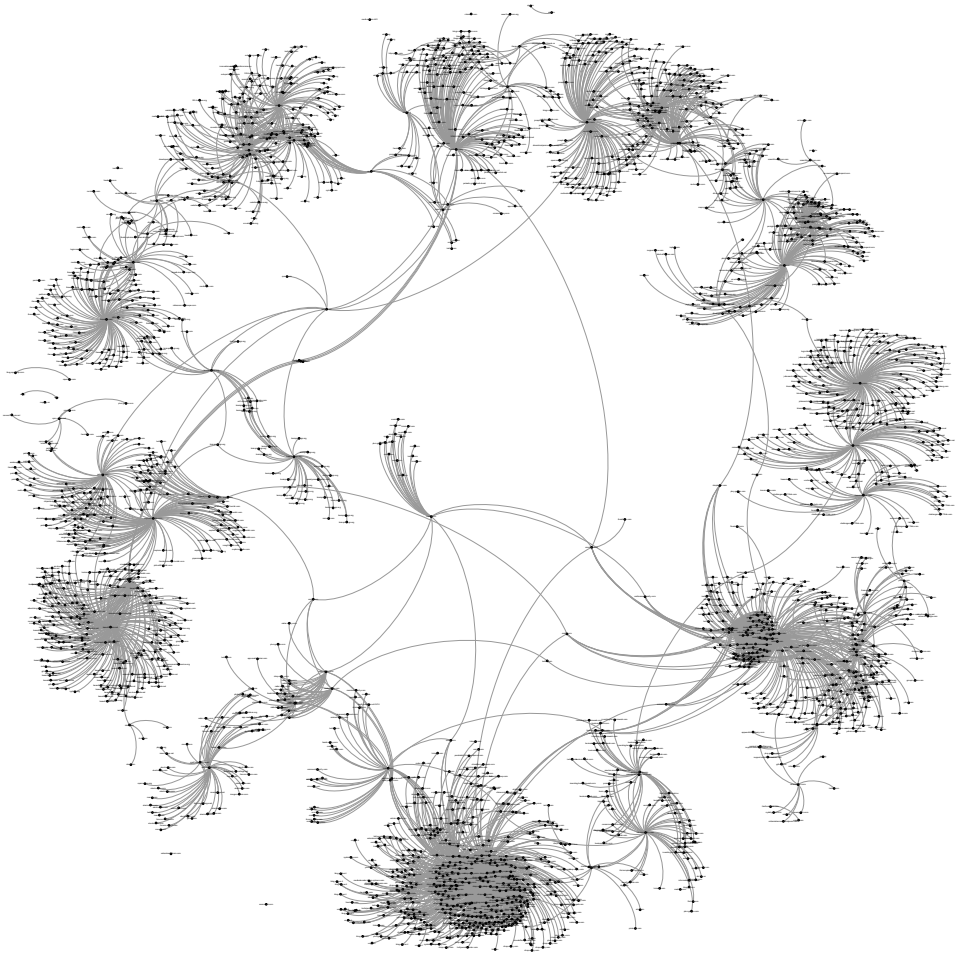


Fig. 1.1: Network Graph

1.3.2.2 HITS

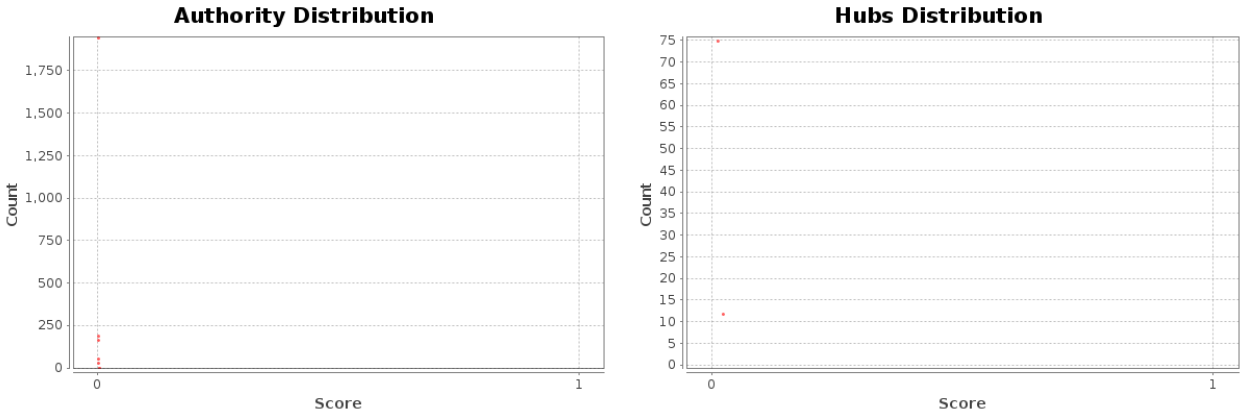


Fig. 1.2: HITS results

1.3.2.3 Page Rank

Epsilon = 0.001 Probability = 0.85

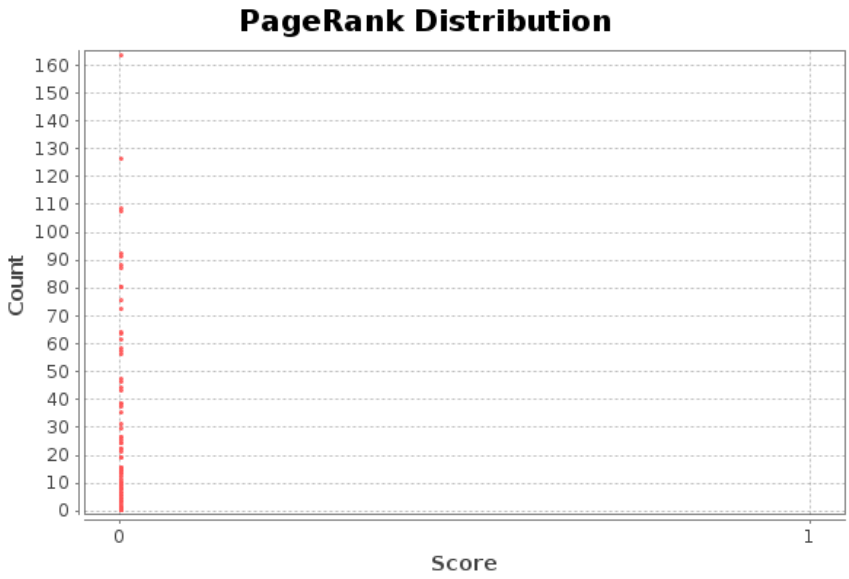
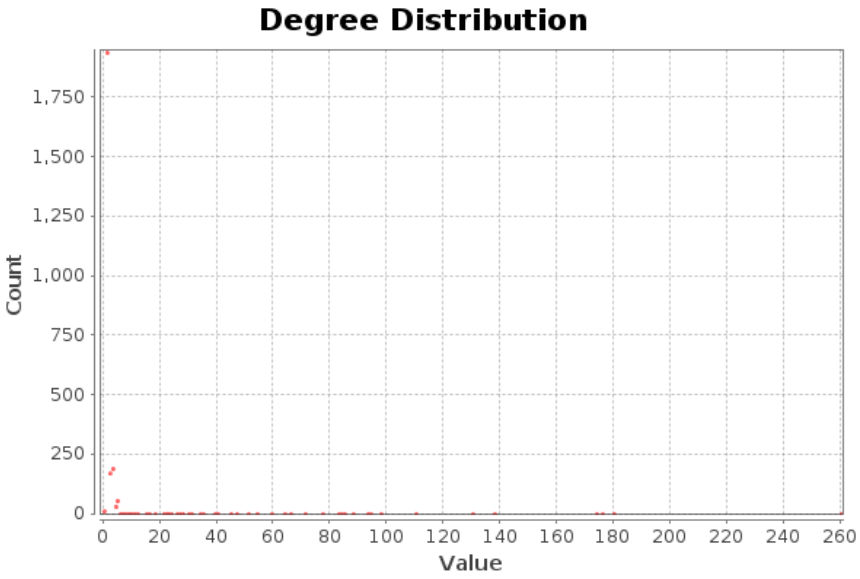
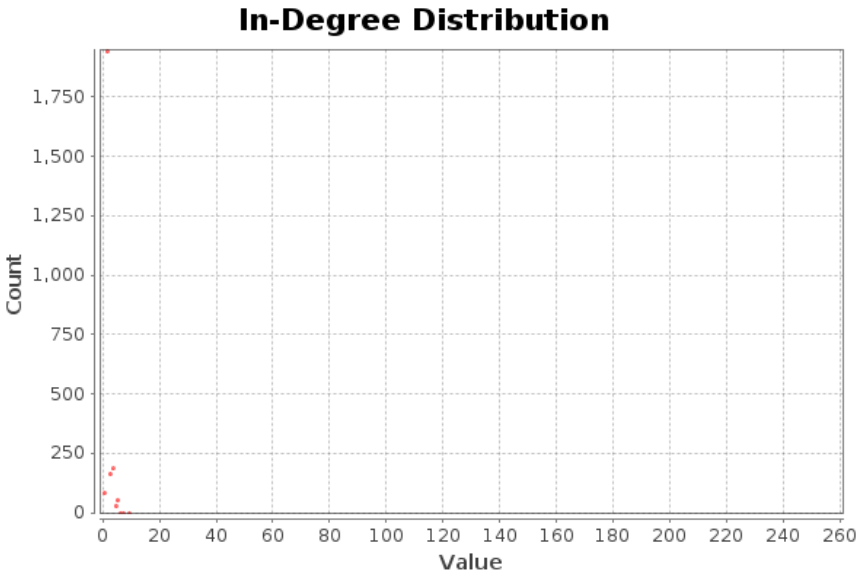


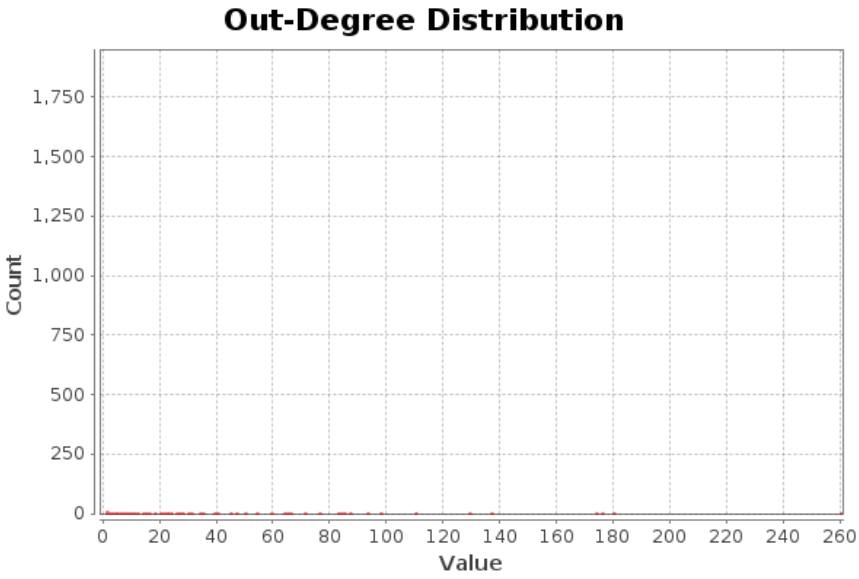
Fig. 1.3: Rage Pank Results

1.3.2.4 Average Degrees

Average Degree: 1.347

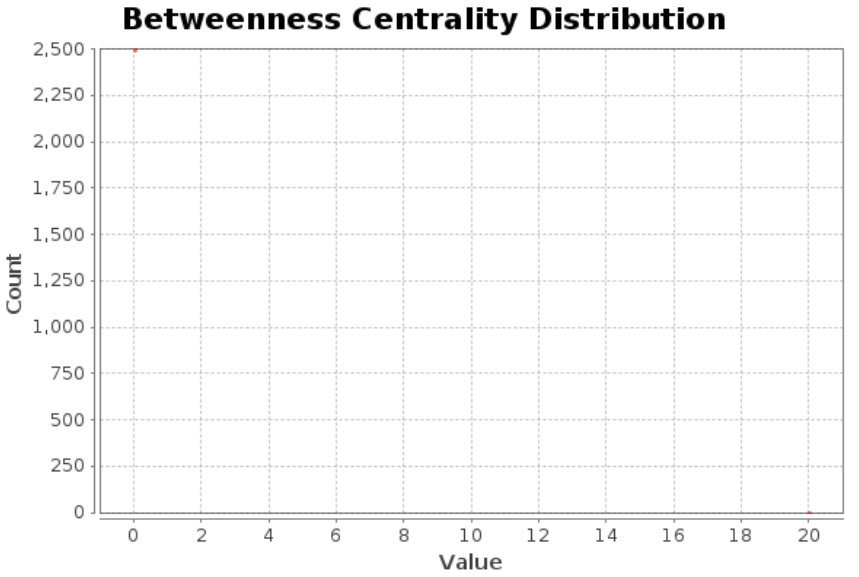

Degree Distribution
Count
1,750
1,500
1,250
1,000
750
500
250
0
0 20 40 60 80 100 120 140 160 180 200 220 240 260
Value

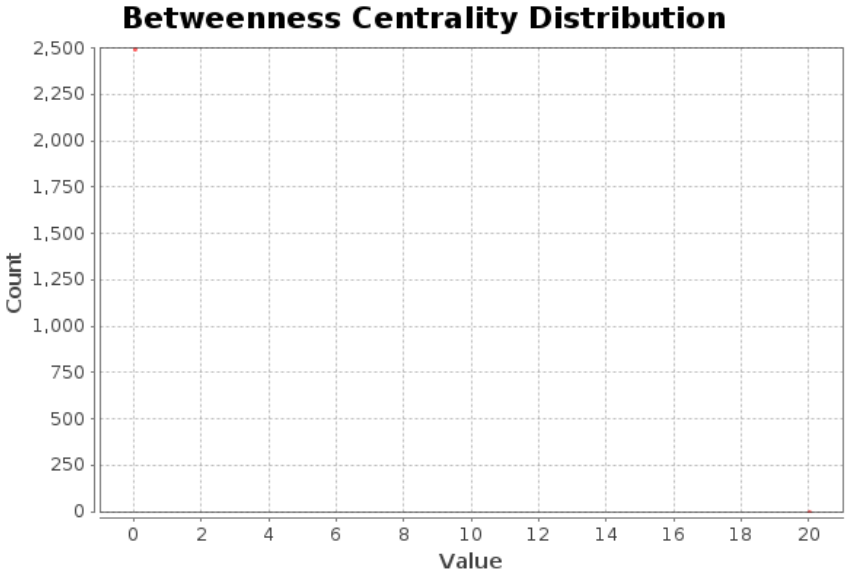

In-Degree Distribution
Count
1,750
1,500
1,250
1,000
750
500
250
0
0 20 40 60 80 100 120 140 160 180 200 220 240 260
Value

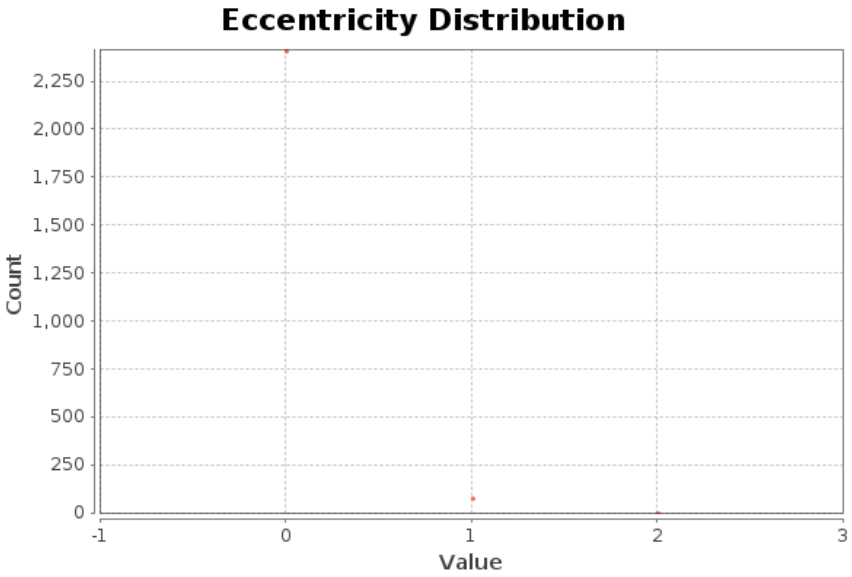


1.3.2.5 Network Diameter

Diameter: 2 Radius: 0 Average Path length: 1.0059294396679515 Number of shortest paths: 3373

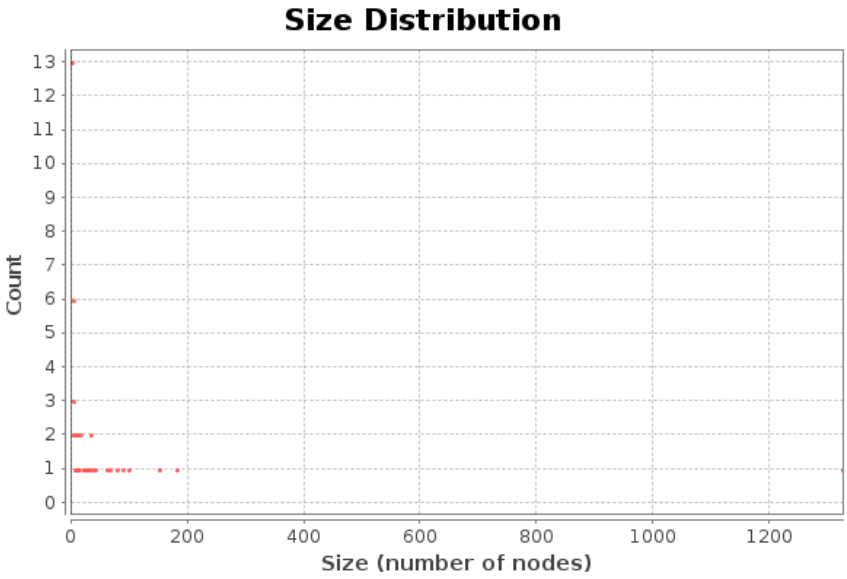






1.3.2.6 Connected Components

Number of Weakly Connected Components: 54 Number of Stronlgy Connected Components: 2498


Size Distribution
Count
Size (number of nodes)
0
1
2
3
4
5
6
7
8
9
10
11
12
13
0
200
400
600
800
1000
1200

References

1. https://stat.ethz.ch/pipermail/r-help//2012-August/333656.html
2. http://digitalpbk.com/perl/perl-script-check-google-pagerank
3. http://www.google.com
4. http://jakevdp.github.io/blog/2012/10/14/scipy-sparse-graph-module-word-ladders/
5. http://curl.haxx.se/docs/httpscripting.html
6. http://www.crummy.com/software/BeautifulSoup/bs4/doc/
7. http://www.rmi.net/~lutz/
8. http://www.cs.cornell.edu/home/kleinber/networks-book/
9. http://thomassileo.com/blog/2013/01/25/using-twitter-rest-api-v1-dot-1-with-python/
10. http://www.cs.odu.edu/~mklein/cs796/lecture/