OLD DOMINION UNIVERSITY

CS 495: Introduction to Web Science
Instructor: Micheal L. Nelson, Ph.D
Fall 2014 Thursdays 4:20pm – 7:10pm ECSB 2120

Assignment # 4

Joseph Elder UIN: 00844802

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 plagiarism. 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

October 9, 2014

Contents

1 Assignment 4

1.1 Question 1

1.1.1 The Problem

1. 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</a>
```

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 Solution

Python version 2.7.6 is used to choose 100 links from the 1000 links supplied from a previous assignment. The file “A4.py” contains the source code for this assignmnet. The 1000 URIs are read in, their HTML is then downloaded to be accessed in python using the python module “requests.” The HTML from each URI is then scraped using the python module “BeautifulSoup” to collect all of the anchor tags on the page. Each anchor tag is assesed using string manipulations in python to ensure only links are to be saved. Once all links are collected a file is generated containing the URI and the links formatted similarly to the example file format in section 1.1.1. These files are generated until there are 100 files present in the previously empty directory “sitelinks.”

1.2 Question 2

1.2.1 The Problem

2. 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 Solution

The file “A4.py” is used to generate a GraphViz dot file to be used later for graphing. Each file in the sitelinks directory is opened to be processed in python. The URI and links from each file are extracted and using string manipulations they are put into the dot language syntax. The URI points to each link in the format URI -> Link;. Each line of the dot syntax is added to the file “href.dot.”

1.3 Question 3

1.3.1 The Problem

3. 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 Solution

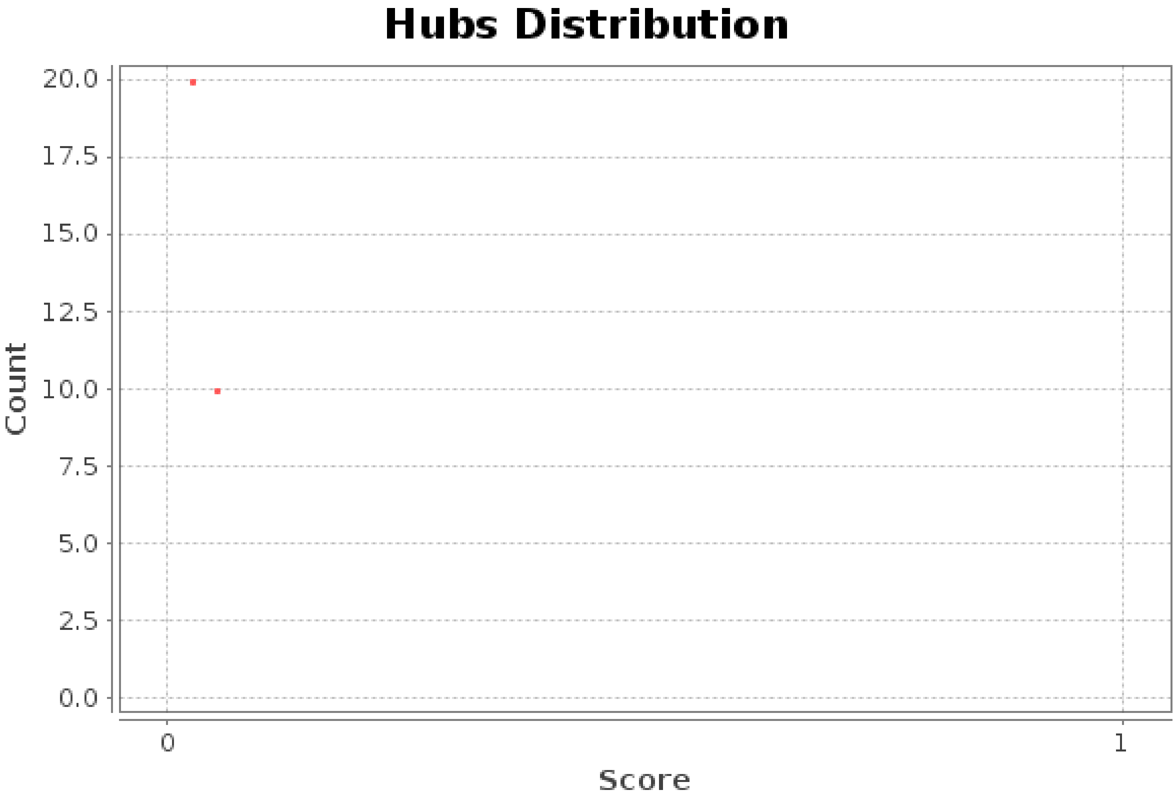
The dot file “href.dot” is then opened using gephi to visualize the connections of the links from each URI. The Following Graphs and statistics were generated using Gephi.

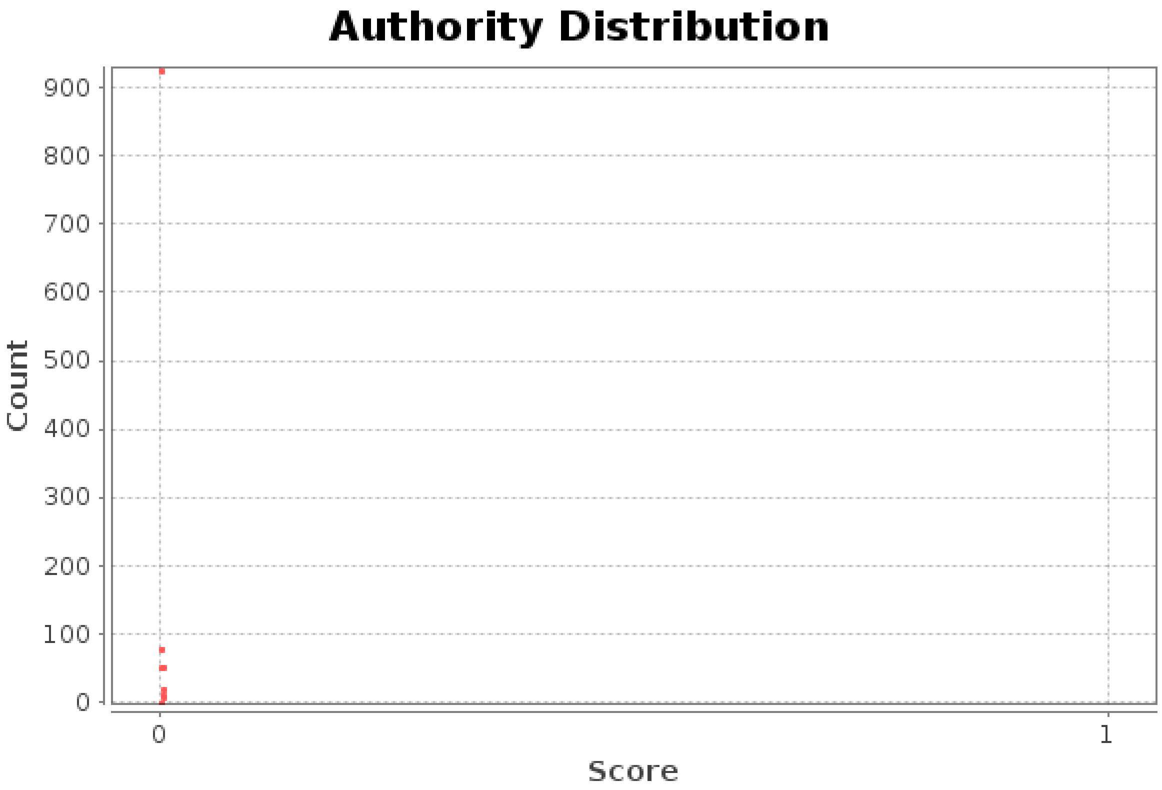
HITS Metric Report

Parameters:

E = 1.0E-4

Results:





Algorithm:

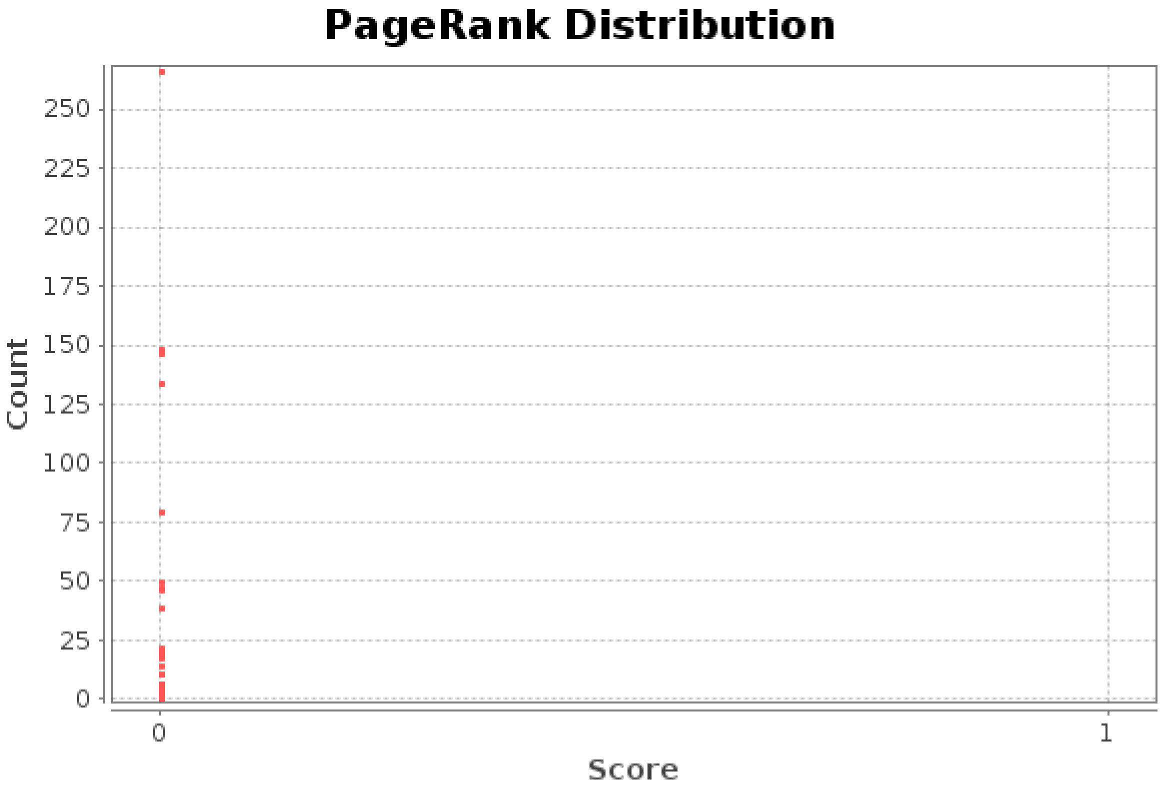
Jon M. Kleinberg, *Authoritative Sources in a Hyperlinked Environment*, in Journal of the ACM 46 (5): 604–632 (1999)

PageRank Report

Parameters:

Epsilon = 0.001
Probability = 0.85

Results:



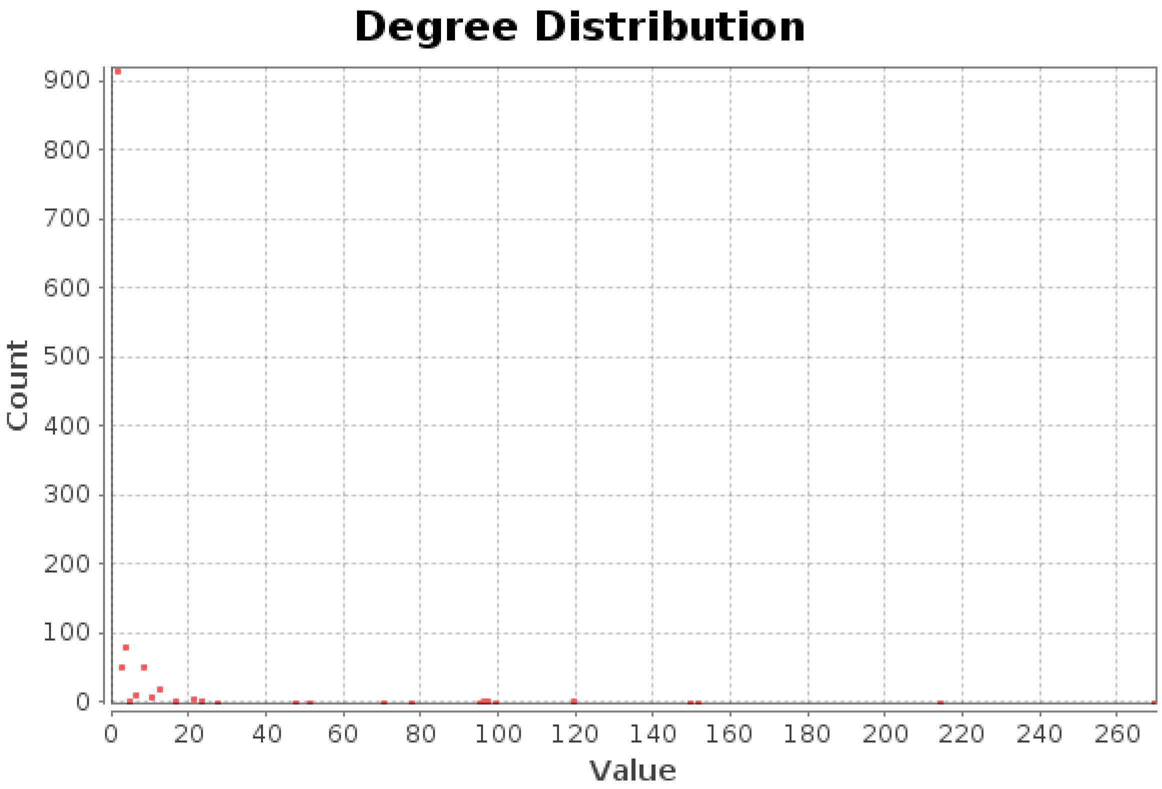
Algorithm:

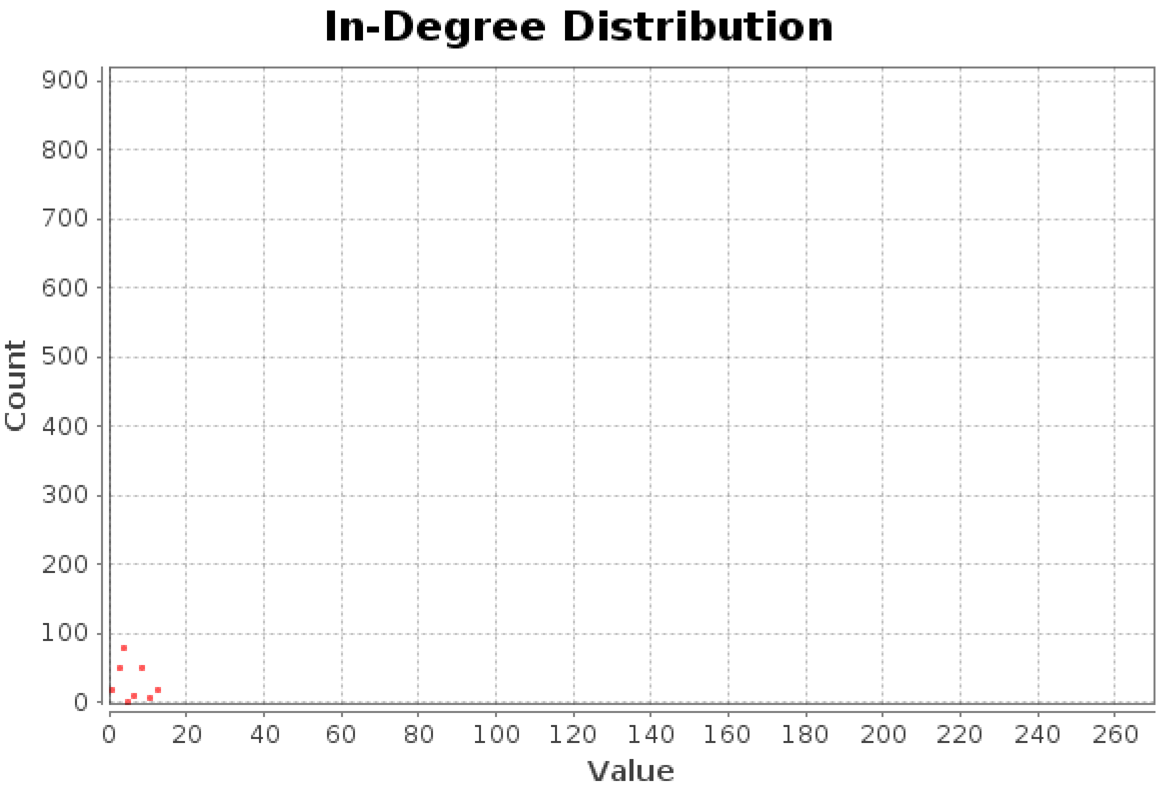
Sergey Brin, Lawrence Page, *The Anatomy of a Large-Scale Hypertextual Web Search Engine*, in Proceedings of the seventh International Conference on the World Wide Web (WWW1998):107-117

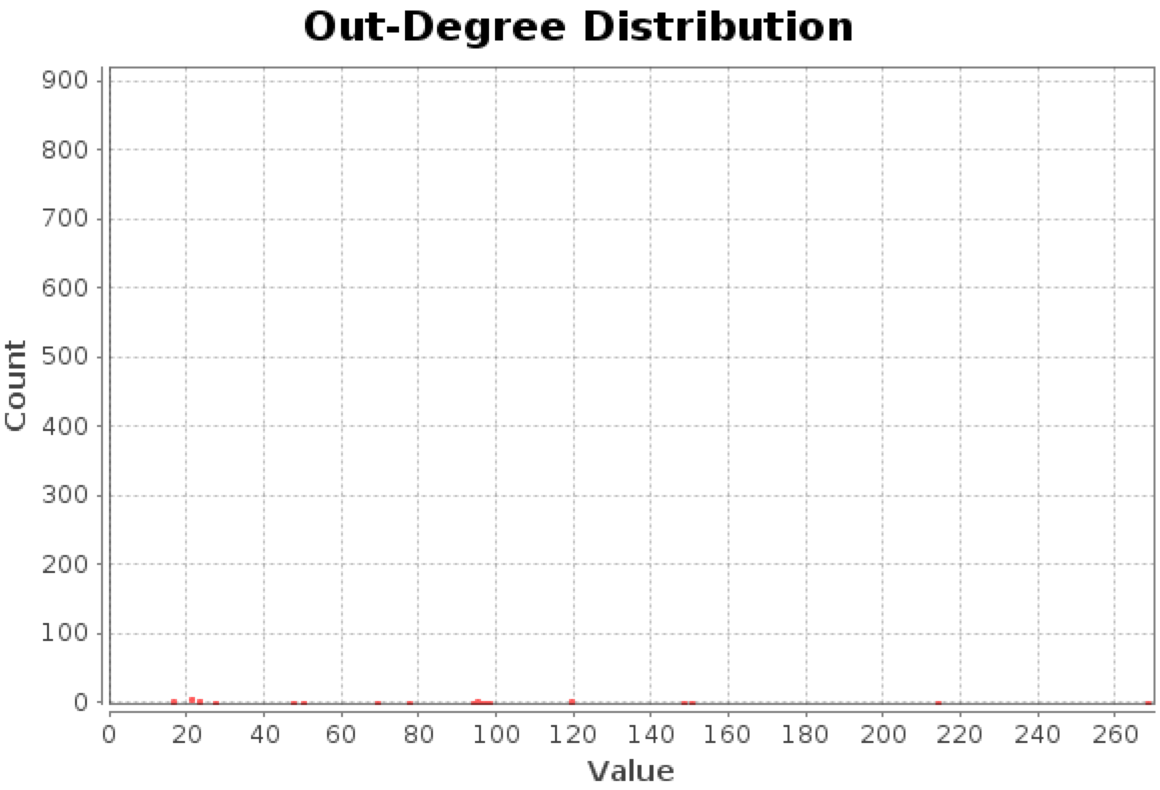
Degree Report

Results:

Average Degree: 1.810




In-Degree Distribution
Count
900
800
700
600
500
400
300
200
100
0
0
20
40
60
80
100
120
140
160
180
200
220
240
260
Value


Out-Degree Distribution
Count
900
800
700
600
500
400
300
200
100
0
0
20
40
60
80
100
120
140
160
180
200
220
240
260
Value

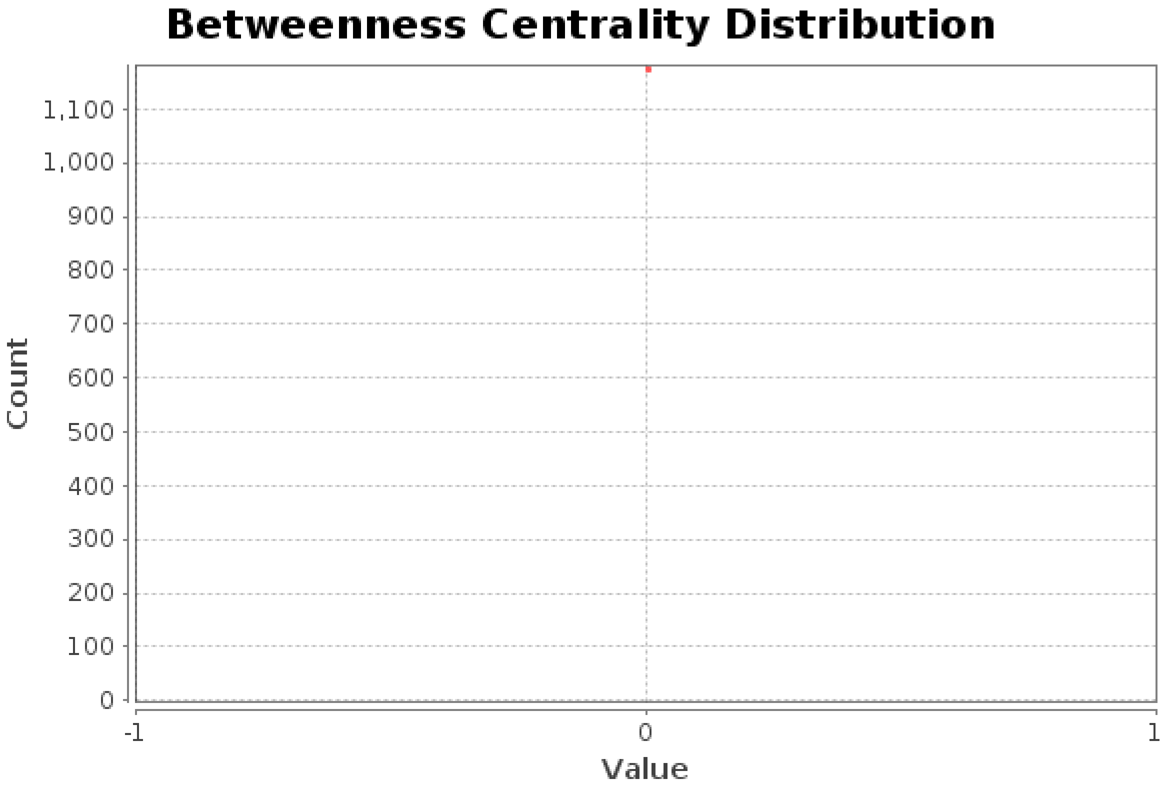
Graph Distance Report

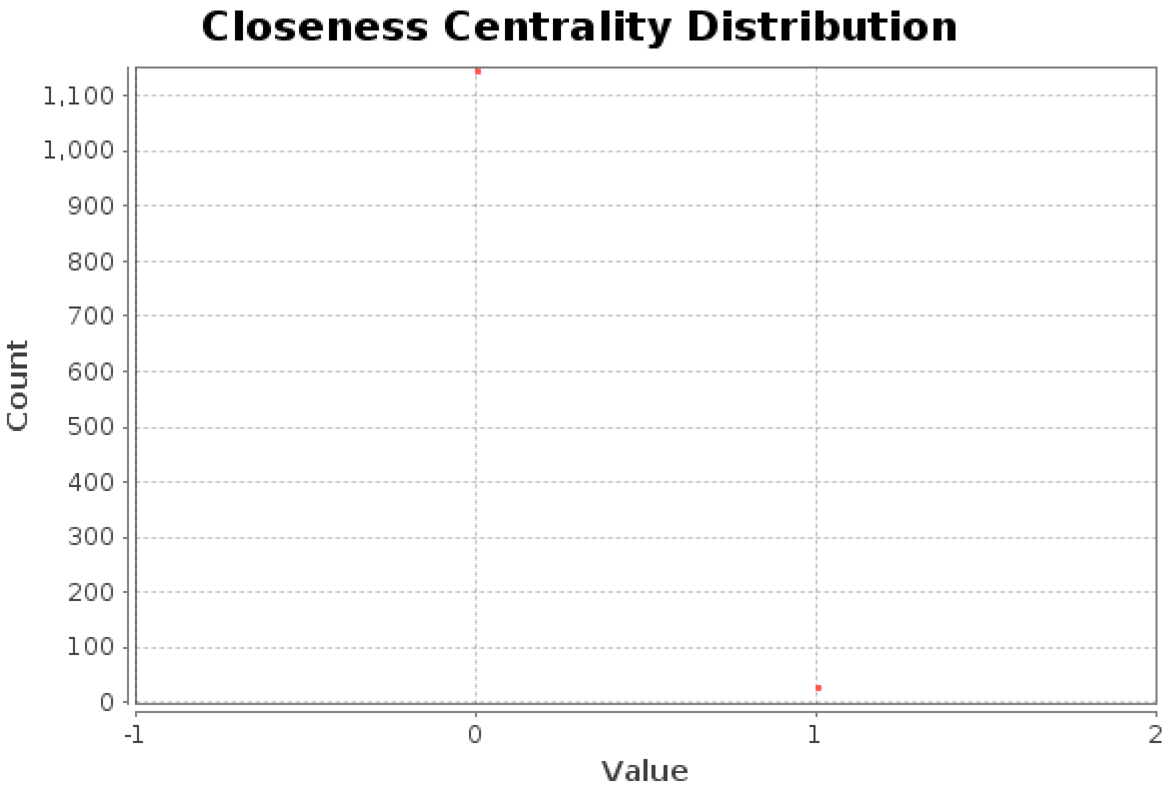
Parameters:

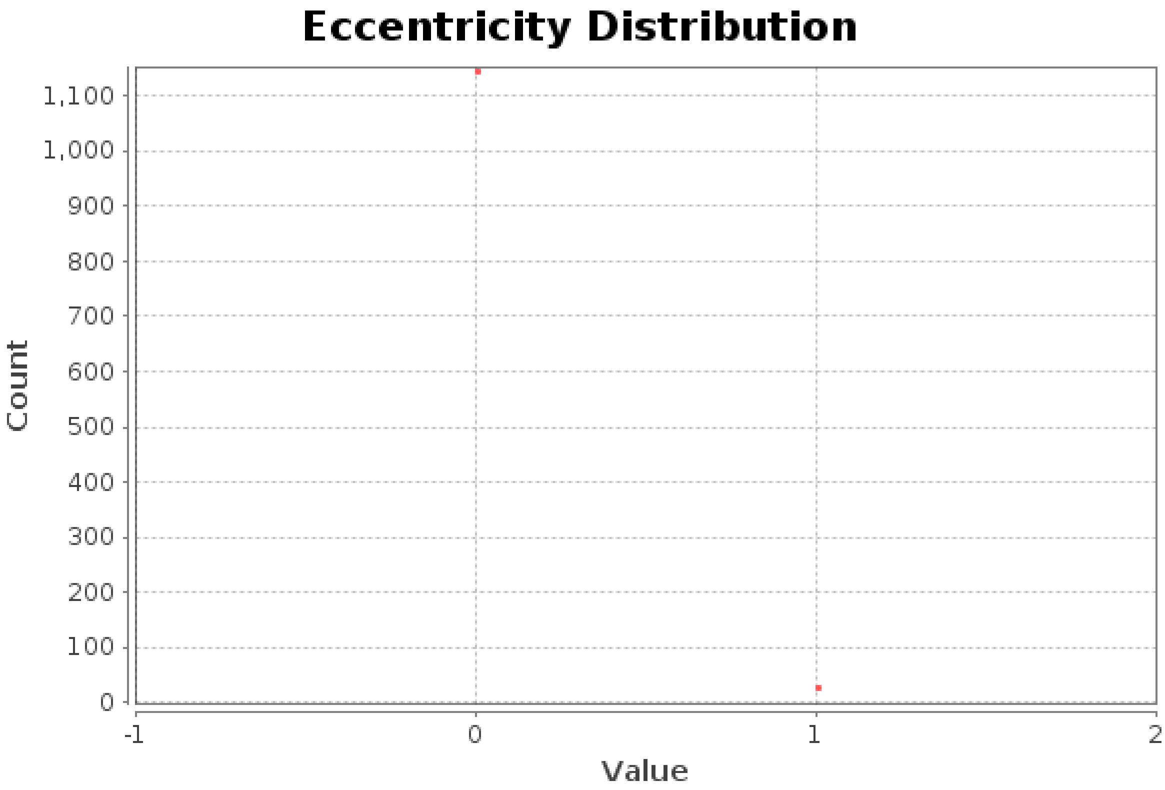
Network Interpretation: directed

Results:

Diameter: 1
Radius: 0
Average Path length: 1.0
Number of shortest paths: 2124




Closeness Centrality Distribution
Count
1,100
1,000
900
800
700
600
500
400
300
200
100
0
-1
0
1
2
Value



Algorithm:

Ulrik Brandes, *A Faster Algorithm for Betweenness Centrality*, in Journal of Mathematical Sociology 25(2):163-177, (2001)

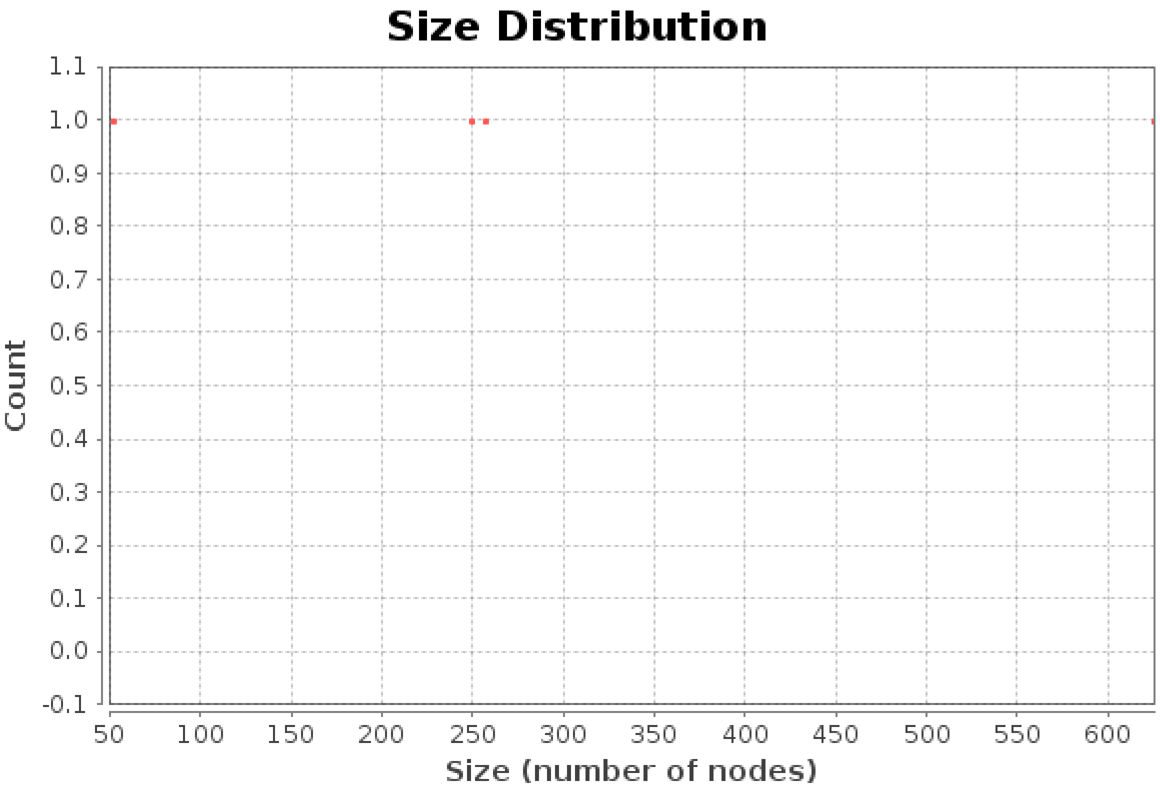
Connected Components Report

Parameters:

Network Interpretation: directed

Results:

Number of Weakly Connected Components: 4
Number of Stronlgy Connected Components: 1179



Algorithm:

Robert Tarjan, *Depth-First Search and Linear Graph Algorithms*, in SIAM Journal on Computing 1 (2): 146–160 (1972)

References

1. https://docs.python.org/2/
2. https://www.bing.com/
3. http://www.graphviz.org/
4. http://gephi.github.io/