

# CS595 Intro to Web Science, Assignment #4

Valentina Neblitt-Jones

October 10, 2013

## Question 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 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/2-13/09/2013-09-09-ms-thesis-http-mailbox.html etc.
```

Upload these 100 files to github (they don't have to be in your report).

## Answer to Question 1

I selected 100 links from the finding aid in Assignment #2 from the following agencies:

- TMZ.com
- Sesame Workshop
- VT News
- Washington City Paper
- Library of Congress

Figure 1 showed a sample of the input file. It has the md5 and the URI for each link.

```

248e690ea8146c52ea527b3c3667f28f http://www.vtnews.vt.edu/articles/2013/05/051313-vpa-bicyclebronze
79d599e3d88824b7f52801409c093cff http://www.vtnews.vt.edu/articles/2013/01/013013-caus-acsadunay.ht
eae903e9adddd703367acedc4f45ee6a http://www.vtnews.vt.edu/articles/2013/07/071713-research-sandy.ht
43941ed8750c9b72cf531beb9bf27f5 http://www.vtnews.vt.edu/articles/2013/08/082113-vetmed-animalbloo
c1f4fef83f273feb1d0aceaa92be339b http://www.vtnews.vt.edu/articles/2013/09/090413-vtc-servicelearn
011ba09cbde51e236f1fe671bd612954 http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-
2aa6fc33eeb9fbf5736b45998fb65511 http://www.sesameworkshop.org/our-blog/2013/05/15/encouraging-fina
20c580dfa0c4b80d9f408c2728f07946 http://www.sesameworkshop.org/our-blog/2013/02/19/got-questions-ab
59f332ec57929f074826b3f591ba04c4 http://www.sesameworkshop.org/our-blog/2013/05/24/herry-me/
aadae41a49249926616fb452b31d79c3 http://www.sesameworkshop.org/press-room/divorce/
376151118e11995481449fbc86481391 http://www.sesameworkshop.org/our-blog/2013/04/19/more-tragic-time
0e18a533564a2b85272ba2cbe618a350 http://www.sesameworkshop.org/our-blog/2013/05/21/teaching-strateg
af471244cbca8fbd1752c2bc685b9709 http://www.washingtoncitypaper.com/blogs/youngandhungry/2013/09/09
f431a0e5044e6b790c872fe5cc25987e http://www.washingtoncitypaper.com/blogs/citydesk/2013/09/06/the-c
13a3ecd4881796bb5c753c2ffc7800a4 http://www.washingtoncitypaper.com/blogs/housingcomplex/2013/09/10
8a2aa84907e2ef65039c22ccd9cd94eb http://www.washingtoncitypaper.com/articles/44800/cat-tales-twenty
4b57c6c131fac8184fb2c23b3602b338 http://www.washingtoncitypaper.com/articles/44734/shadow-of-a-doub
f9508ec34443ce5ae8dea378d4d03ee0 http://www.washingtoncitypaper.com/blogs/housingcomplex/2013/09/12
1c60de75830236d0664a0f261d58d13d http://www.washingtoncitypaper.com/blogs/artsdesk/music/2013/09/03
0bb6e775e10d83313f9bce6ef8711b0 http://www.washingtoncitypaper.com/blogs/citydesk/2013/09/06/photo
a7053150d5d8e8ae2a69089ff848da0a http://www.washingtoncitypaper.com/blogs/citydesk/2013/09/10/capti
07da4407624074b7373e9b795764717a http://www.washingtoncitypaper.com/blogs/citydesk/2013/09/09/the-

```

Figure 1: Sample of Input File

My Python code (Figure 2) split each line of the input file into two variables called rawfile and uri. As suggested, I used BeautifulSoup to find all the links in the file. Outputting the links was a little tricky since using the simple example below at [www.pythonforbeginners.com/python-on-the-web/beautifulsoup-4-python](http://www.pythonforbeginners.com/python-on-the-web/beautifulsoup-4-python), represented relative links as they are in the file and I needed full links.

```

from bs4 import BeautifulSoup
import urllib2

redditFile = urllib2.urlopen("http://www.reddit.com")
redditHtml = redditFile.read()
redditFile.close()

soup = BeautifulSoup(redditHtml)
redditAll = soup.find_all("a")
for links in soup.find_all('a'):
    print (links.get('href'))

```

So I needed to take care of a few cases and urlparse was needed to break the uri into its key components as explained in Python documentation <http://docs.python.org/2/library/urlparse.html> - scheme, netloc and path. It checks if the link is relative. If it is relative then it should add the scheme from the parent page and the netloc from the parent page. There was also the matter of extra slashes. Some links already had a slash and others did not. I need it to add the slash if it was not already there and there was a path. I also needed it to throw out any javascript that was picked up by BeautifulSoup. I formatted the output files as suggested in the assignment description (Figure 3).

```

from bs4 import BeautifulSoup
import urllib.request
from urllib.parse import urlparse

f = open('inputfile.txt', 'r')

for line in f:

    (rawfile, uri) = line.split()
    print("Working on " + rawfile)
    g = open('/Users/vneblitt/Documents/cs595-f13/assignment03/raw/' + rawfile)
    piece = urlparse(uri)

    first = g.read()
    soup = BeautifulSoup(first)
    s = open('/Users/vneblitt/Documents/cs595-f13/assignment04/q1/' + rawfile, 'w')
    s.write('site:' + '\n')
    s.write(uri + '\n')
    s.write('links:' + '\n')

    for link in soup.find_all('a'):
        a = link.get('href') #full link or link fragment
        b = urlparse(a) #parsed link

        if not b.scheme: #is it relative?
            if not b.path:
                c = (str(piece.scheme) + '://' + str(piece.netloc) + '/' + '\n')
                s.write(c)
            else:
                c = (str(piece.scheme) + '://' + str(piece.netloc) + str(b.path) + '\n')
                s.write(c)
        elif b.scheme == 'javascript':
            pass
        else:
            s.write(a + '\n')

    s.close()

    g.close()

f.close()

```

Figure 2: ExtractLinks.py

```

site:
http://www.washingtoncitypaper.com/blogs/housingcomplex/2013/09/12/gray-said-the-living-wage-bill-would-kill-m
links:
http://www.washingtoncitypaper.com/
http://ad.doubleclick.net/N622/jump/washingtondc.creativeloafing/edit/housingcomplex_blog;pg=housingcomplex;sz:
http://www.washingtoncitypaper.com/calendar/
http://www.washingtoncitypaper.com/slideshows/photos/
http://www.washingtoncitypaper.com/craftybastards/
http://www.washingtoncitypaper.com/bestofdc/
http://www.washingtoncitypaper.com/thisweek/
http://www.washingtoncitypaper.com/
http://www.washingtoncitypaper.com/pages/masthead/
http://www.washingtoncitypaper.com/corrections/
http://www.washingtoncitypaper.com/pages/workhere/
http://www.washingtoncitypaper.com/pages/internships/
http://www.washingtoncitypaper.com/pages/freelancersguide/
http://www.washingtoncitypaper.com/pages/contact/
http://www.washingtoncitypaper.com/pages/advertising/
http://www.washingtoncitypaper.com/pages/backissues/
http://www.washingtoncitypaper.com/pages/findapaper/
http://www.washingtoncitypaper.com/news/
http://www.washingtoncitypaper.com/articles/44879/sole-collective-the-lamont-street-collective/
http://www.washingtoncitypaper.com/articles/44879/sole-collective-the-lamont-street-collective/
http://www.washingtoncitypaper.com/blogs/looselips/2013/09/30/morning-links-85/
http://www.washingtoncitypaper.com/blogs/youngandhungry/2013/09/30/last-nights-leftovers-breakfast-edition-2/
http://www.washingtoncitypaper.com/blogs/housingcomplex/2013/09/30/morning-links-274/
http://www.washingtoncitypaper.com/blogs/artsdesk/general/2013/09/30/arts-roundup-big-ol-fancy-library-edition
http://www.washingtoncitypaper.com/blogs/artsdesk/music/2013/09/27/how-to-spend-rock-the-bells-weekend-without
http://www.washingtoncitypaper.com/blogs/citydesk/
http://www.washingtoncitypaper.com/blogs/artsdesk/
http://www.washingtoncitypaper.com/blogs/looselips/

```

**Figure 3:** Sample of an Output File

## Question 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.

## Answer to Question 2

Since I needed to iterate through a file directory I needed to use `listdir` from `os`. I found guidance for using this on a tutorial page [http://www.tutorialspoint.com/python/os\\_listdir.htm](http://www.tutorialspoint.com/python/os_listdir.htm).

```
import os, sys

# Open a file
path = "/var/www/html/"
dirs = os.listdir( path )

# This would print all the files and directories
for file in dirs:
    print file
```

I tested this in the IDLE Python shell and found out that the `.DS_Store` file would be included. Rather than find out what my code would do with that file I chose to remove it. So in my code (Figure 4), I started with a modification of the code above to create the list that the rest of my code would iterate through. Basically I needed to skip the first line of the file which just had the word “site:”. Then I wanted to make the second line in the file a variable called `site`. Next I needed to skip the third line in the file which just had the word “link”. Finally, I needed the fourth line until the end of the file to be stored as links. The code output the site and relates it to the link and adds the label. As it ran through the files, I needed everything to end up in a single document “`mydotfile.dot`” and named the digraph “spaghetti”. Figure 5 shows a sample of the dot file produced.

```

from os import listdir

# Access input file directory
mypath = '/Users/vneblitt/Documents/cs595-f13/assignment04/q1/linkingfiles/'
files = listdir(mypath)
files.remove('.DS_Store')

# Open the output file
s = open('/Users/vneblitt/Documents/cs595-f13/assignment04/q2/mydotfile.dot', 'w')
s.write('digraph spaghetti {\n')

# Iterate through input file directory
for file in files:
    f = open(mypath + file)
    mylines = f.readlines()

    site = mylines[1].strip('\n')
    links = mylines[3:]

    for link in links:
        link = link.strip()
        s.write('"' + site + '" -> "' + link + '" [ label = "' + site + " to " + link + " ]; \n')

    f.close()

s.write('}')

s.close()

```

Figure 4: CreateDotFile.py

```

digraph spaghetti {
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://www.sesameworkshop.org/" [ label = "http://www.sesameworkshop."
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://www.sesameworkshop.org/contact-us" [ label = "http://www.sesam
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://www.sesameworkshop.org/partners/donate" [ label = "http://www.
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://store.sesamestreet.org" [ label = "http://www.sesameworkshop.o
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://sesamestreet.org" [ label = "http://www.sesameworkshop.org/our
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://www.sesameworkshop.org/what-we-do/our-initiatives/" [ label = "
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://www.sesameworkshop.org/what-we-do/our-initiatives/" [ label = "
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://www.sesameworkshop.org/what-we-do/our-results/" [ label = "htt
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://www.sesameworkshop.org/what-we-do/our-research-model/" [ label
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://www.sesameworkshop.org/what-we-do/our-approach-in-action/" [ l
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://www.sesameworkshop.org/where-we-work/" [ label = "http://www.s
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://www.sesameworkshop.org/partners/our-partners-stories/" [ label
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://www.sesameworkshop.org/partners/donate/" [ label = "http://www
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://www.sesameworkshop.org/partners/partners/" [ label = "http://w
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://www.sesameworkshop.org/partners/annual-gala/" [ label = "http:
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://www.sesameworkshop.org/partners/annual-gala/" [ label = "http:
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://www.sesameworkshop.org/our-blog/" [ label = "http://www.sesame
"http://www.sesameworkshop.org/our-blog/2013/04/11/jackie-robinson-on-sesame-street/" -> "http://www.sesameworkshop.org/our-blog/" [ label = "http://www.sesame

```

Figure 5: Sample of the Dot File Output

## Question 3

Download and install Gephi:

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.

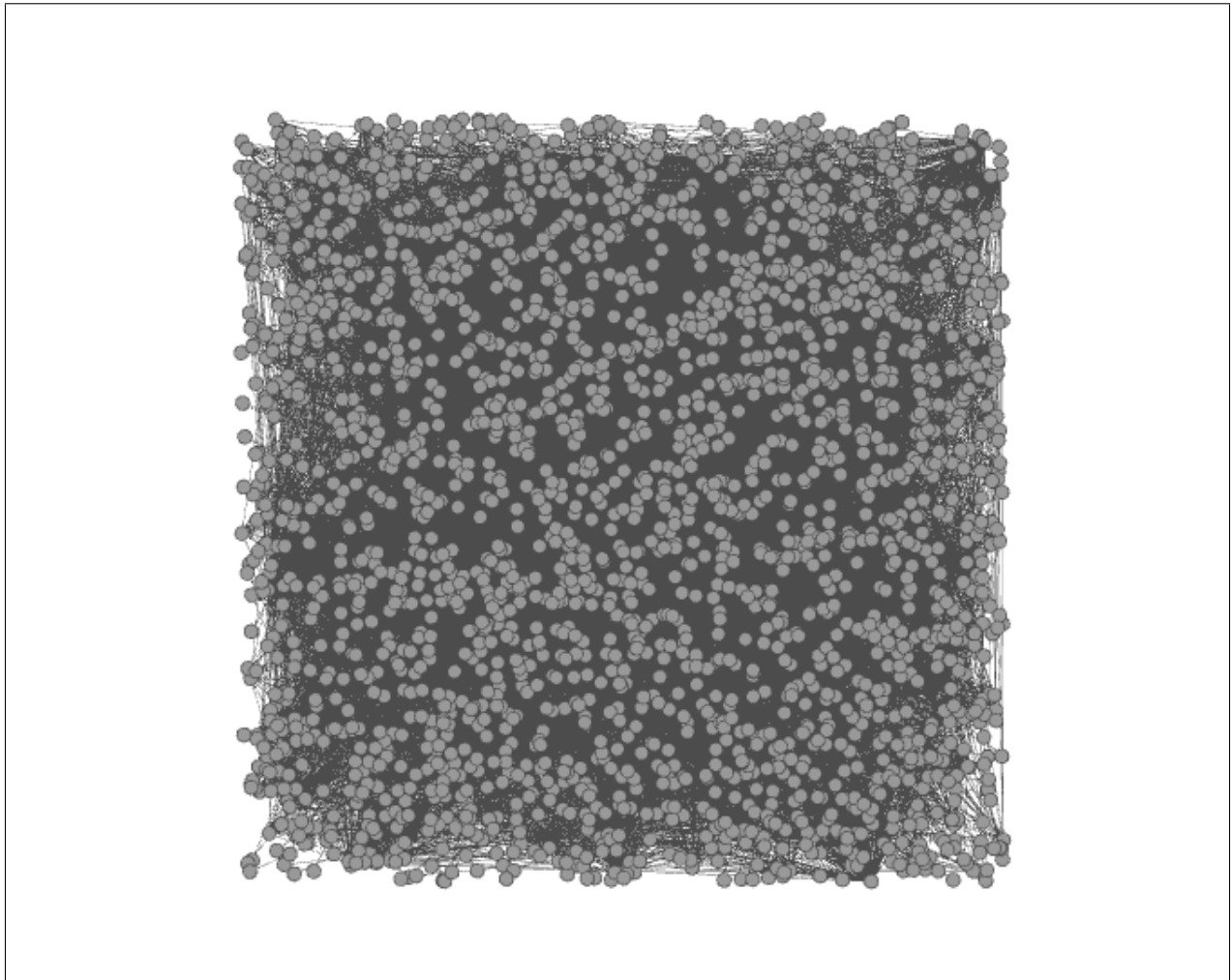
You might need to choose the 100 sites with an eye toward creating a graph with at least one component that is nicely connected. You can probably do this by selecting some portion of your links (e.g., 25, 50) from the same site.

## Answer to Question 3

I downloaded Gephi and imported my dot file. The first result was called in the email list a hairball, but I thought dirty pillow was another good description (Figure 6). I attempted several layouts, but the results did not illustrate much. I used Force Atlas and the graph started spreading out too far to still be represented in a report. The nodes for VT News and Washington City Paper managed to be totally separated from other groupings. However there was some connection between Library of Congress, TMZ, and Sesame Workshop. In Figure 7, the grouping in the top right represents VT News. The grouping in the top left represents Washington City Paper. The center groupings are Library of Congress (left) and Sesame Workshop (right). The bottom grouping is TMZ.

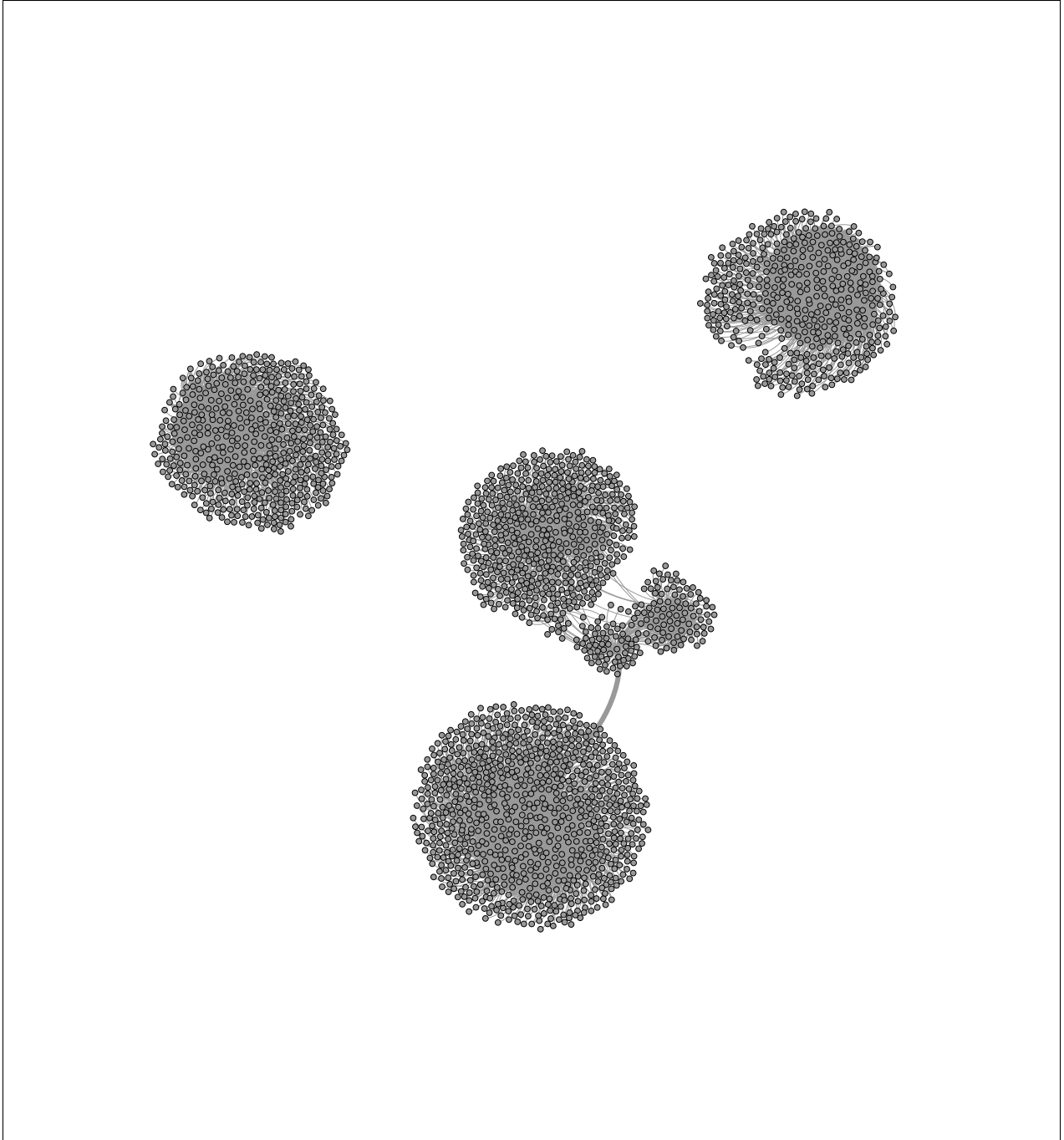
I ran the requested reports as well. The report names matched the titles in the assignment description with the exception of network diameter. The tool called it Network Diameter (Figure 8), but the reports were labeled Graph Distance. There was not much to see in the reports, but it seems that we did not have enough data to populate those charts.

## Visualizations



**Figure 6:** Dirty Pillow





**Figure 7:** Visualization with Force Atlas 2

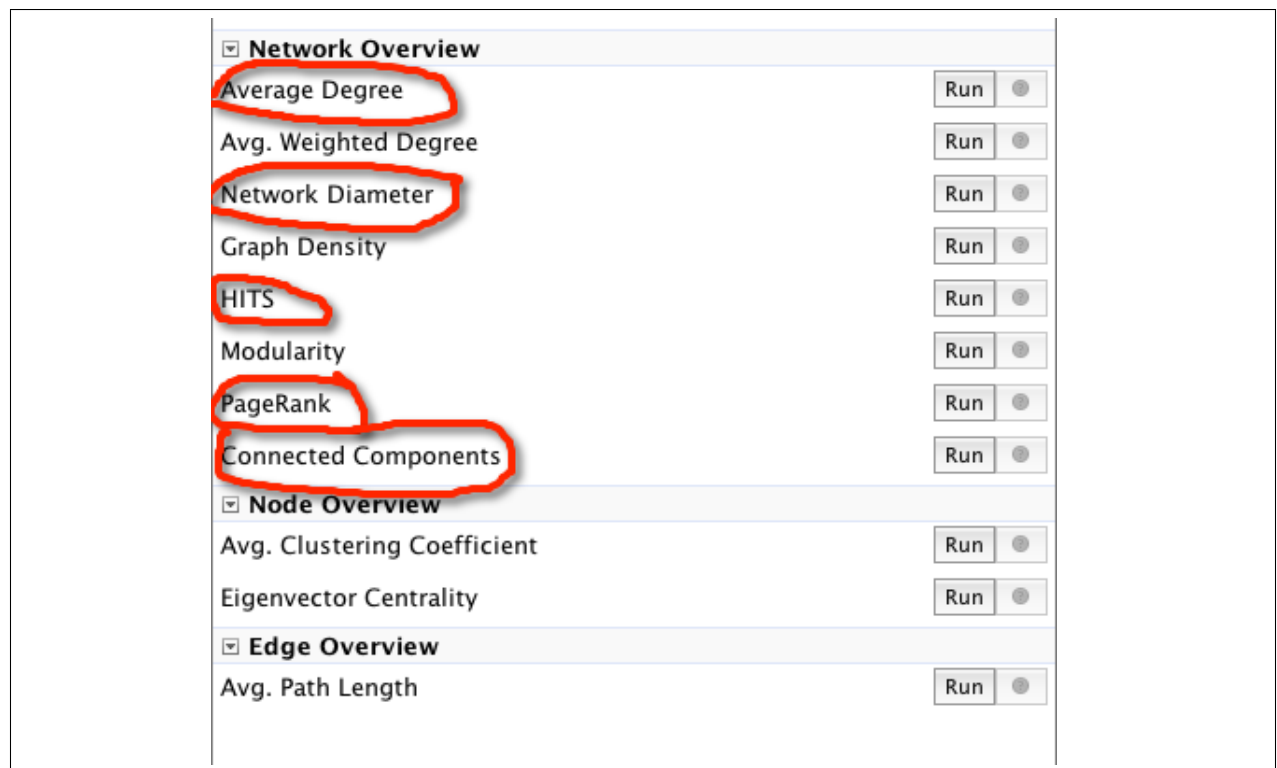


Figure 8: Gephi Reports Interface

## HITS Metric Report

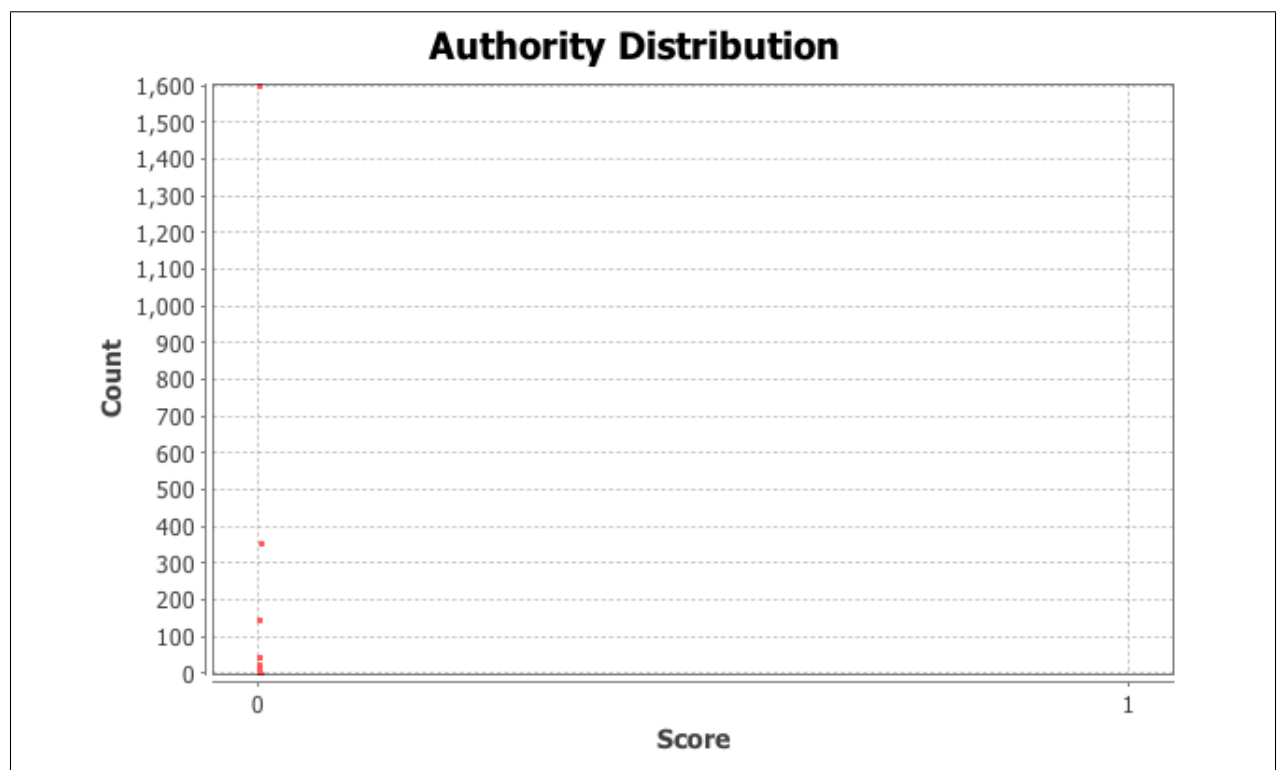


Figure 9: Authority Distribution

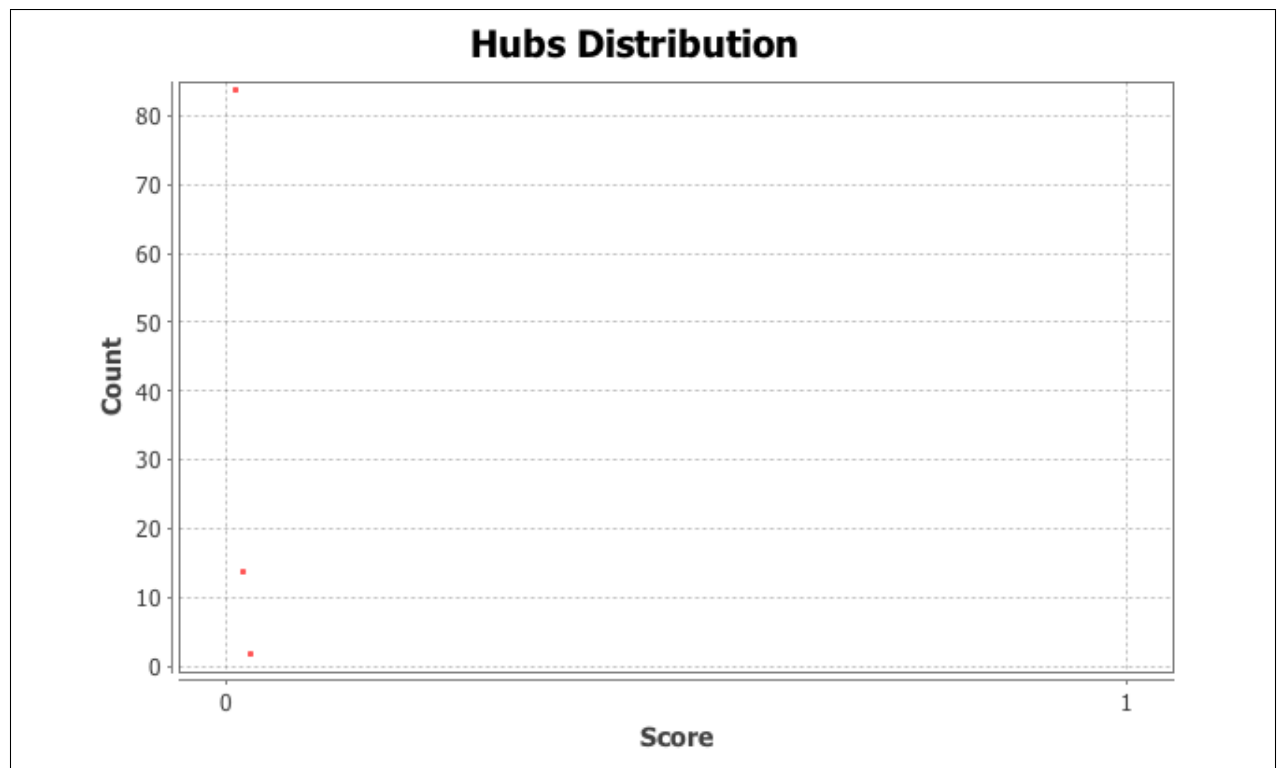


Figure 10: Hubs Distribution

## PageRank Report

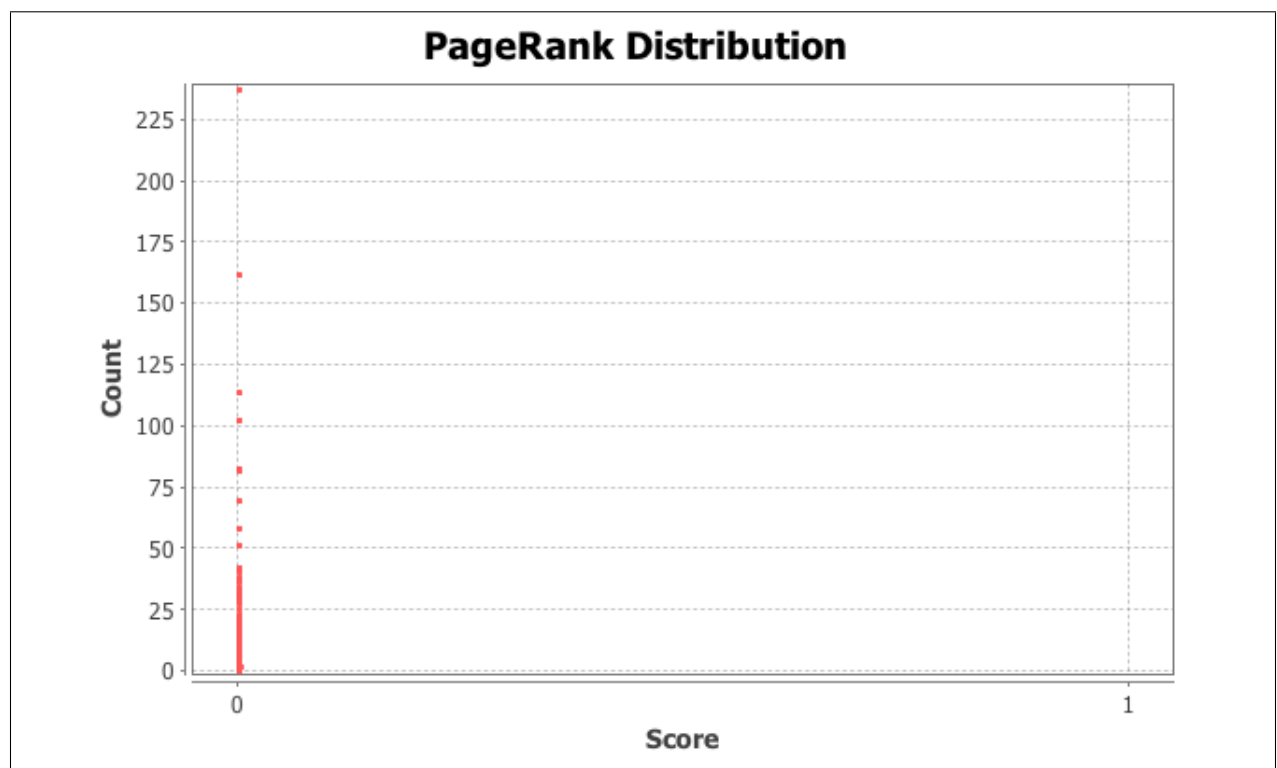


Figure 11: PageRank Distribution

## Average Degree Reports

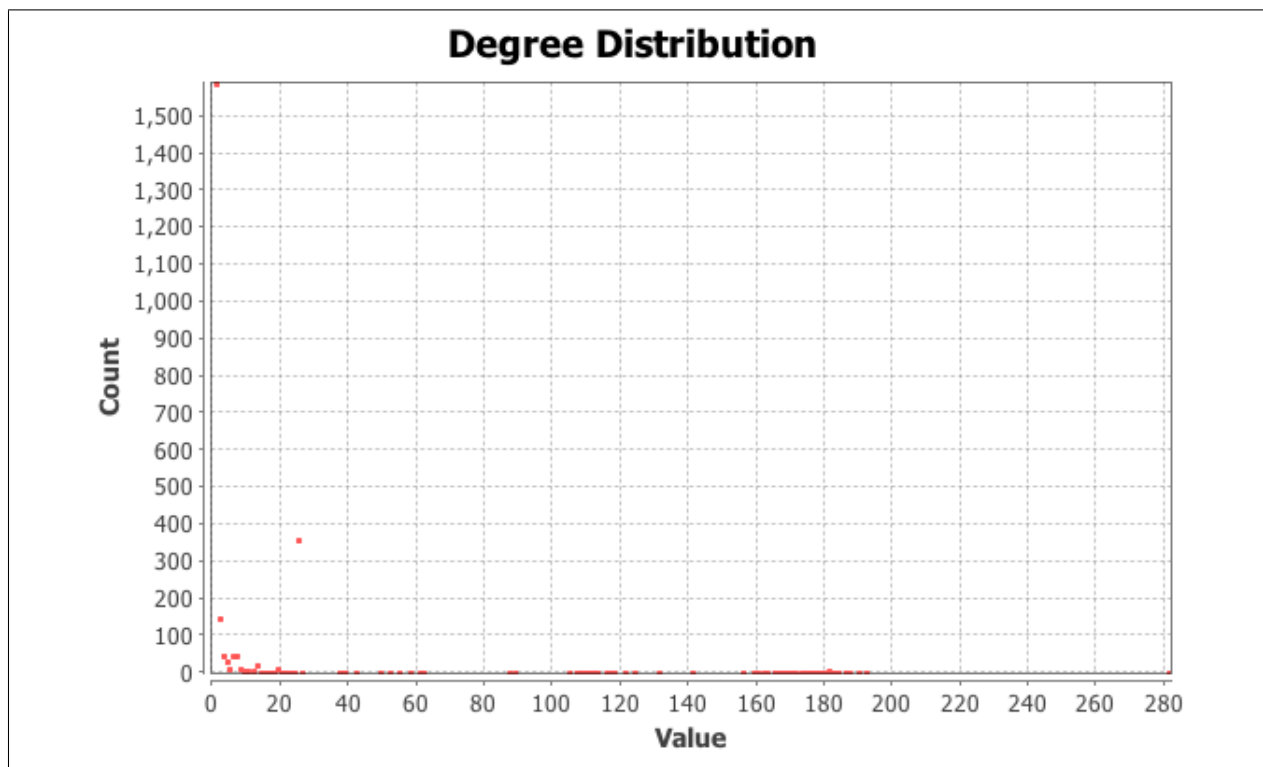


Figure 12: Degree Distribution

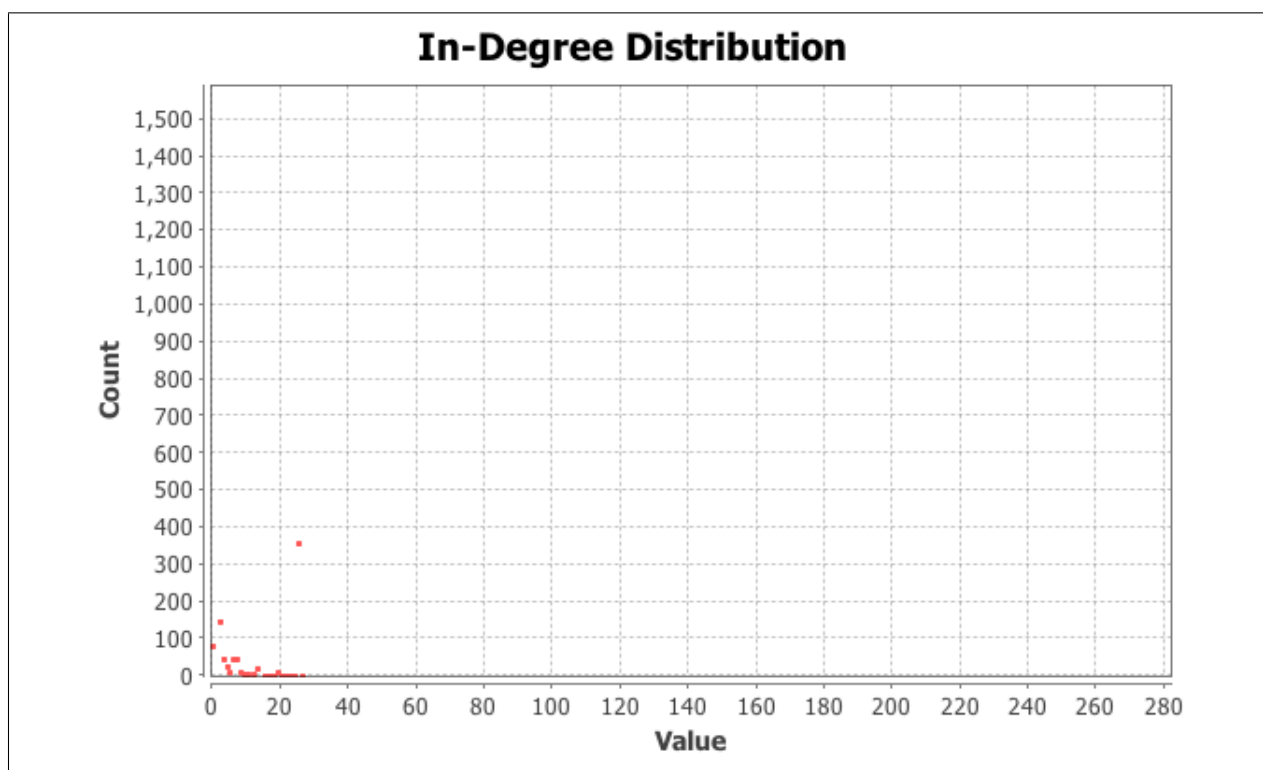


Figure 13: In-Degree Distribution

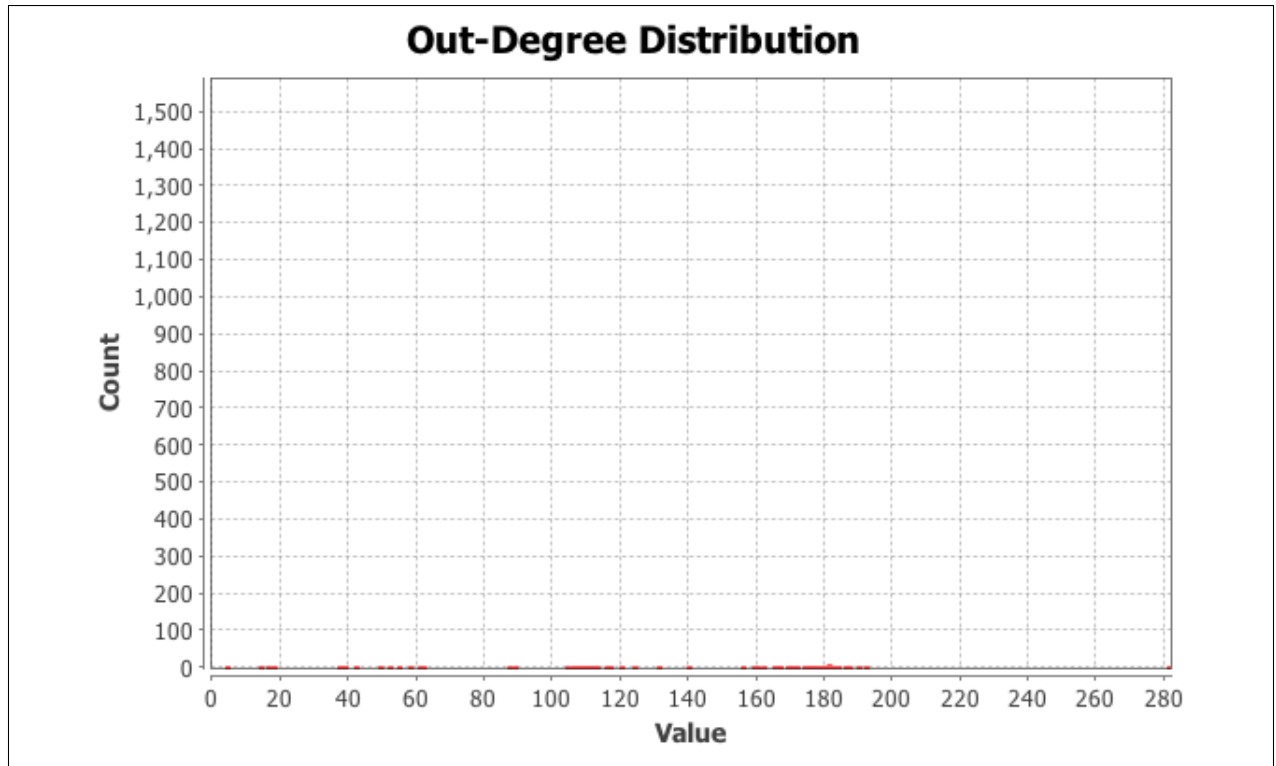


Figure 14: Out-Degree Distribution

#### Graph Distance Report

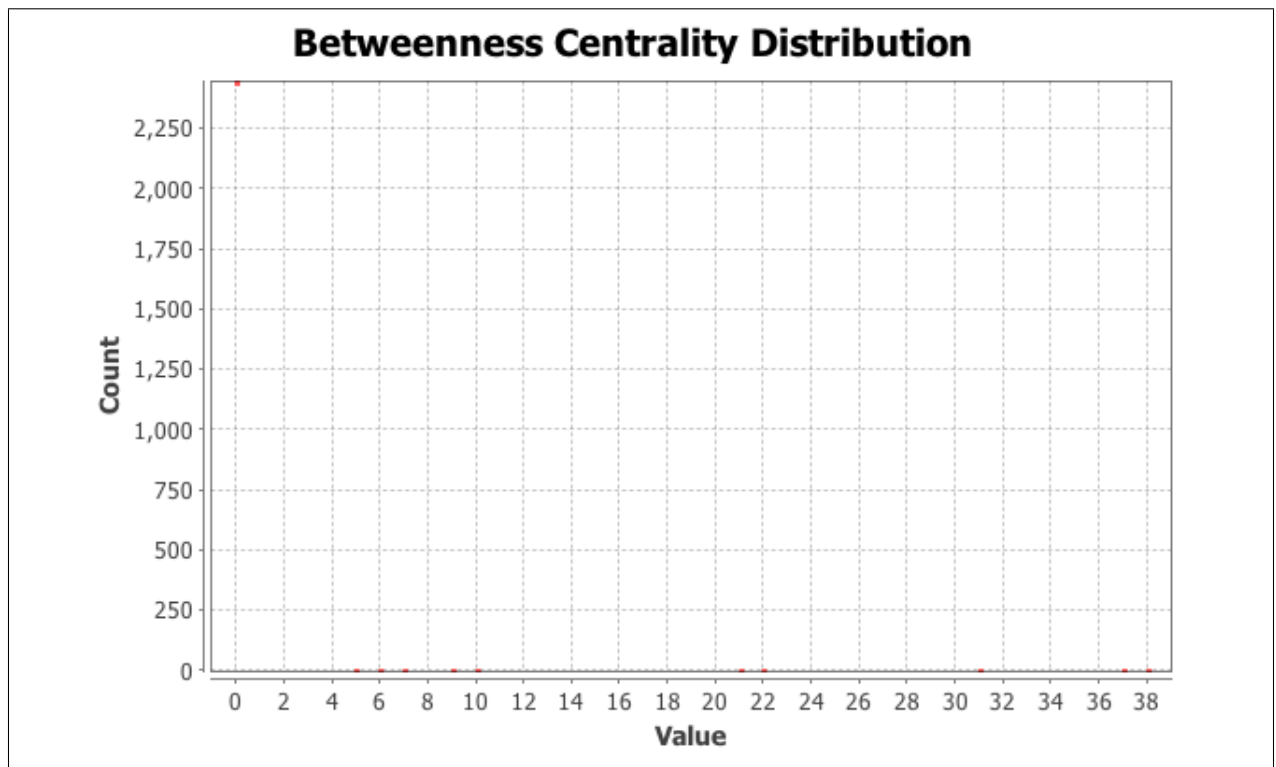


Figure 15: Betweenness Centrality Distribution

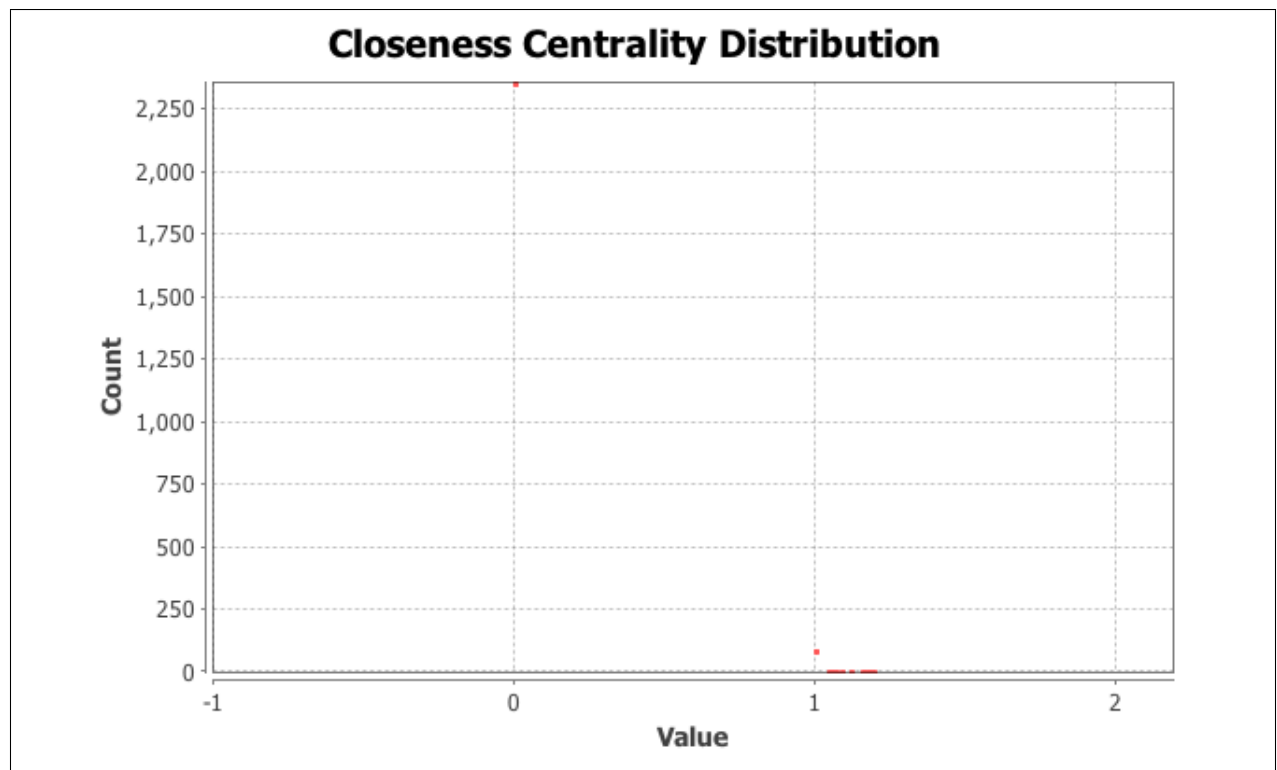


Figure 16: Closeness Centrality Distribution

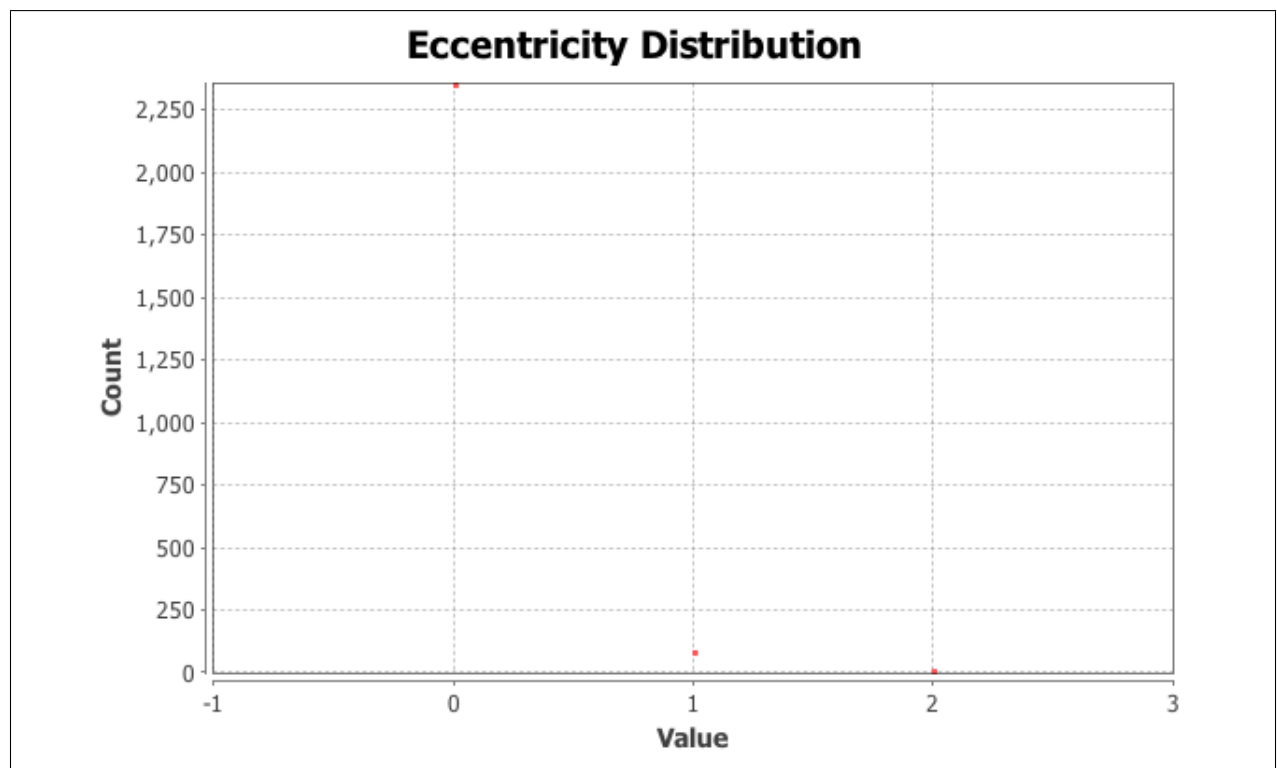


Figure 17: Eccentricity Distribution

## Connected Components Report

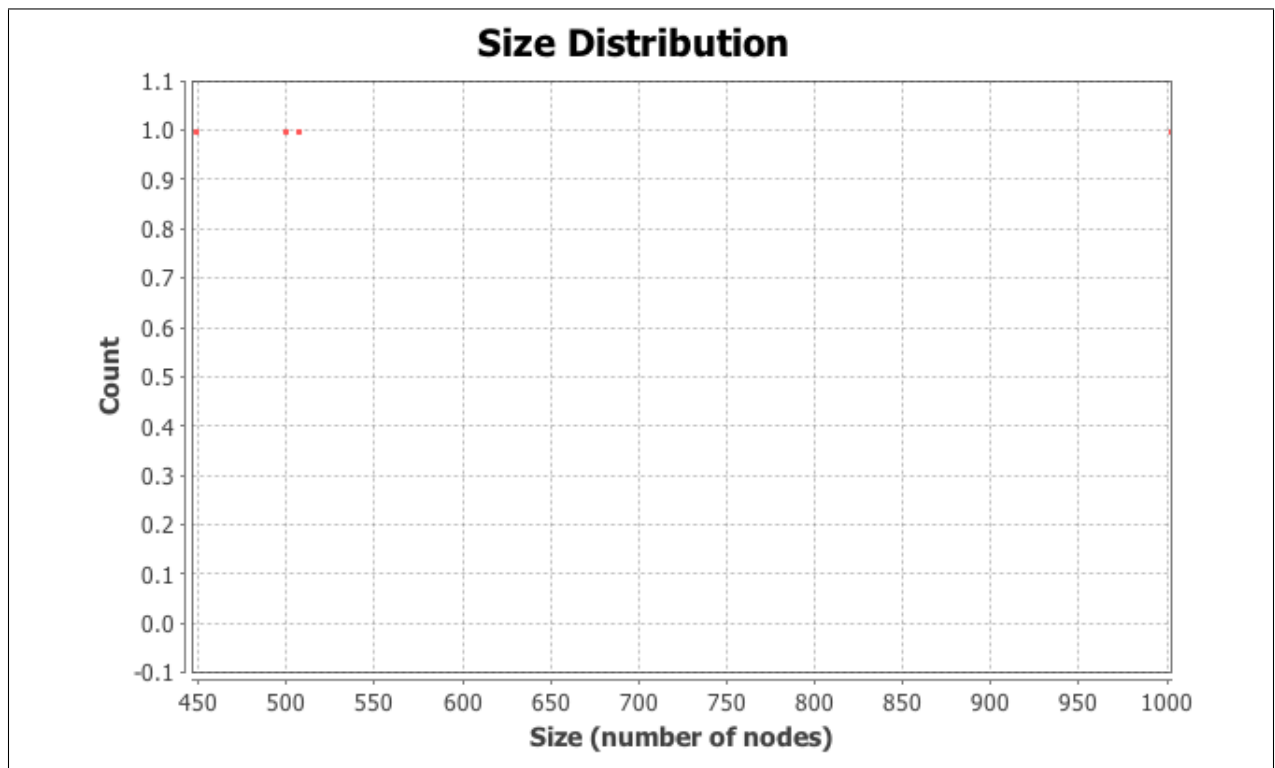


Figure 18: Size Distribution