Introduction to Web Science/595: Assignment #4

Dr. Nelson

Francis Pruter

Thursday, 9 October, 2014

Contents

Problem 1

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).

SOLUTION In order to get all the links from 100 URIs, I used a python script. BeautifulSoup was used to extract all the links and ensure they were valid links (returned a 200 when followed. Additionally in order to try to make sure I had a good spread, I used grep to extract all the nytimes.com and add them to the front of the list. "egrep nytimes.com uniqueURIs ¿ uris" then "cat uniqueURIs ¿ uris"

Listing 1: downloadLinks.py

```
import urllib2
import requests
from bs4 import BeautifulSoup

def testURI(link):
 try:
  if (len(link) == 0):
    return [False, ""]
  if (link[0].isalpha()):
    try:
      r = requests.head(link, allow_redirects=True)
      if r.status_code == 200:
          #print r.url
          return [True, r.url]
      return [False, ""]
    except requests.ConnectionError:
      return [False, ""]
  else:
    return [False, ""]
 except:
  return [False, ""]

with open('uris', 'r') as f:
    uriList=f.readlines()

fileNumber = 1


for uri in uriList:
  if (fileNumber == 101):
    break
  try:
      uri = testURI(uri)
      if (uri[0]):
         uri = uri[1]
      else:
         continue
      uriFile = urllib2.urlopen(uri)
      uriHTML = uriFile.read()
      uriFile.close()
  except Exception as e:
      print e
      continue

  soup = BeautifulSoup(uriHTML)
  linksList = []
  try:
      for links in soup.find_all('a'):
          uriLink = links.get('href')
          l = testURI(uriLink)
          #print l
          if (l[0]):
              if (l[1] not in linksList):
                  linksList.append(l[1])
          else:
              continue
  except Exception as e:
      print e
      continue


  if (len(linksList) == 0):
     continue
  with open('links/'+str(fileNumber), 'w') as fout:
      fout.write('Site:\n'+uri+'\nLinks:\n')
      for links in linksList:
       try:
         fout.write(links+'\n')
       except:
         print "error"
  print fileNumber
  fileNumber = fileNumber + 1
```

Downloads all the links from 100 URIS:

Problem 2

```
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.)
```

SOLUTION

In order to solve this problem, I used a python script to make a graphViz file that gephi

Listing 2: makeGraphViz.py

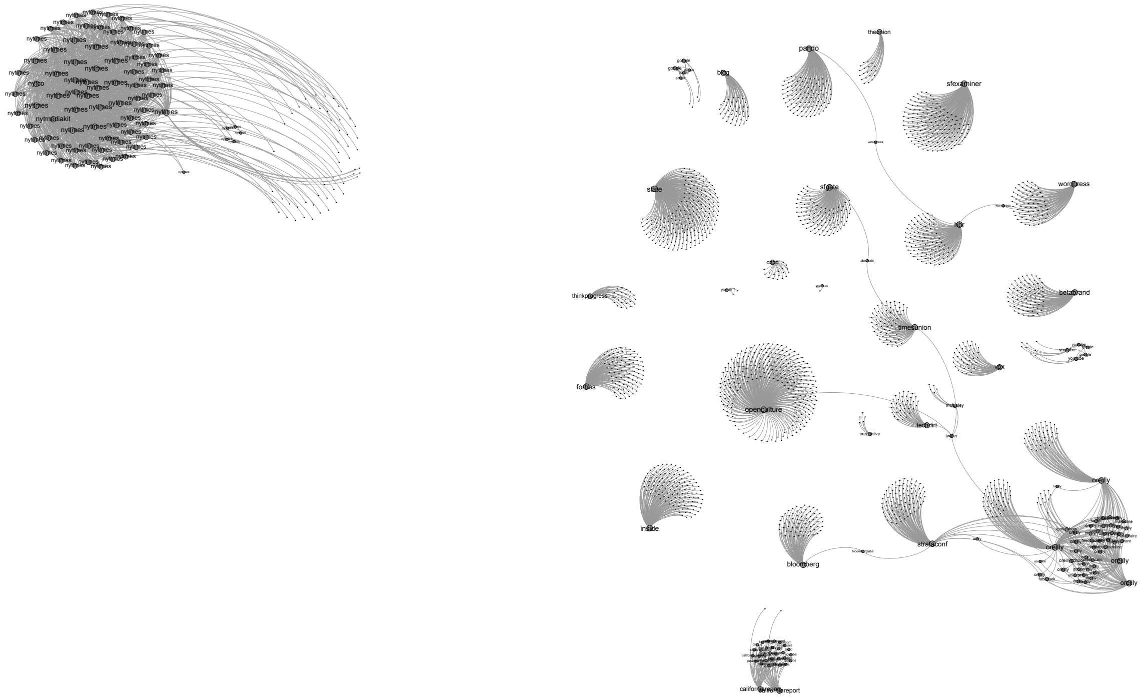
```python
import tld #used to get the domain from a uri.  used as a label

def getLabel(link):
  try:
    return tld.get_tld(link, as_object=True).domain
  except:
    return link

with open("graph.gv", "w+") as fout:
  fout.write("digraph graphName {\n")

  for x in range(1,101):
      print x
      links = []
      with open('links/'+str(x), "r") as f:
          links = f.readlines()

      numNodes = len(links) - 3

      head = links[1].rstrip('\n')
      ctr = 3
      while (ctr < len(links)):
        link = links[ctr].rstrip(’\n’)
        fout.write("\t\"" + head + "\" -> \"" + link + "\"\n")
        fout.write("\t\"" + head + "\" [label=\"" + getLabel(head).rstrip(’\n’) + "\"];\n")
        fout.write("\t\"" + link + "\" [label=\"" + getLabel(links[ctr]).rstrip(’\n’) + "\"];\n")
        ctr = ctr+1


  fout.write("}")
```

Problem 3

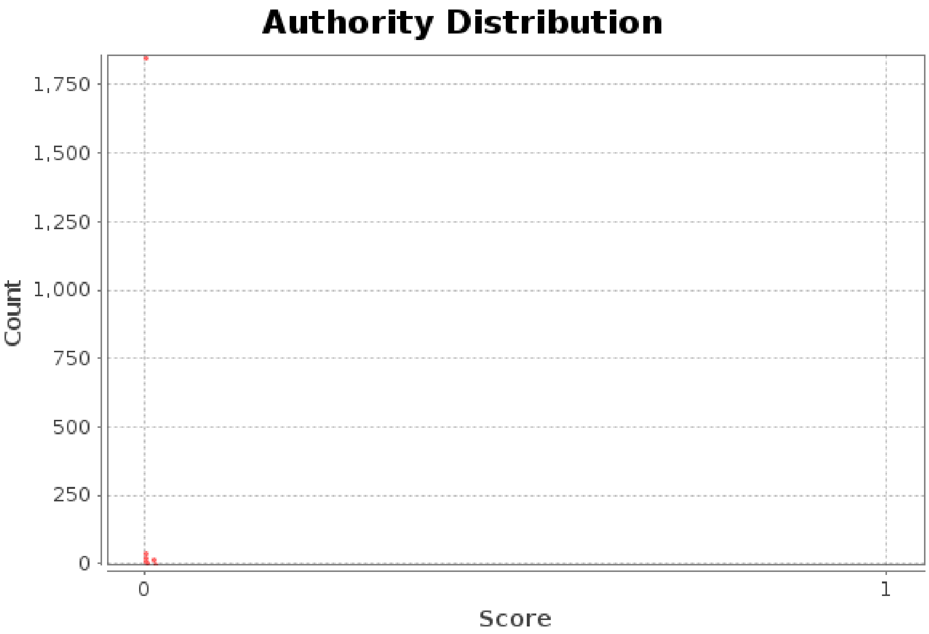
```
3.  Download and install Gephi:

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.
```

SOLUTION

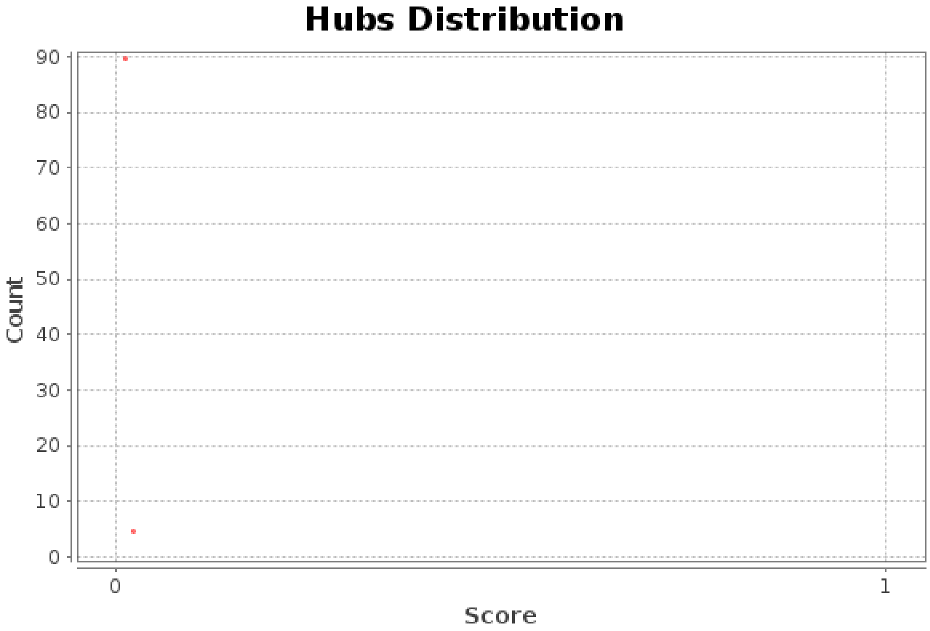
Problem 3 continued on next page. . .

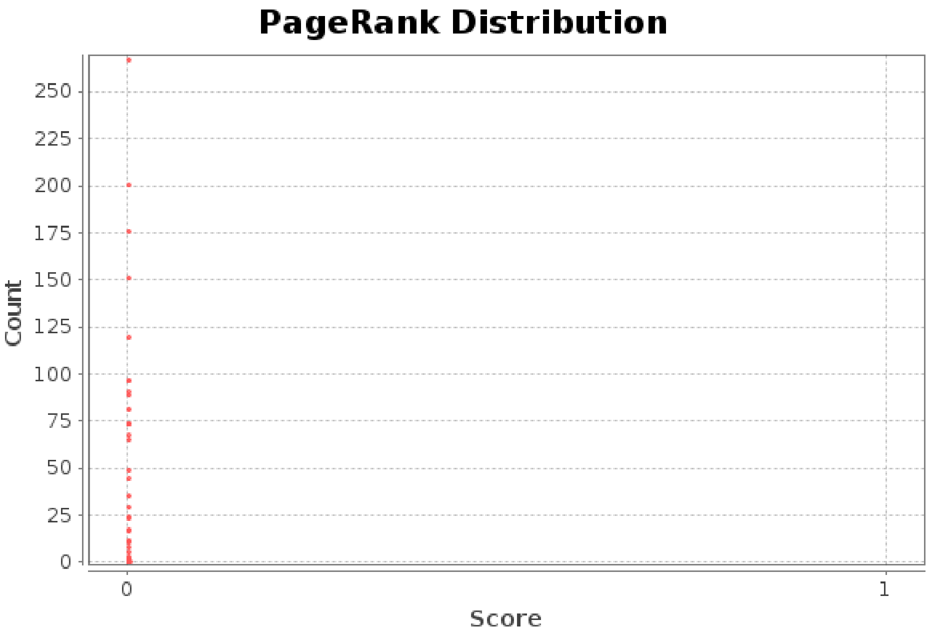

nytimes
nytmediakit
pando
theonion
sfexaminer
blog
slate
sfgate
wordpress
thinkprogress
betabrand
timesunion
forbes
openculture
techdirt
oreilly
inside
strataconf
bloomberg
californiareport

graph.pdf

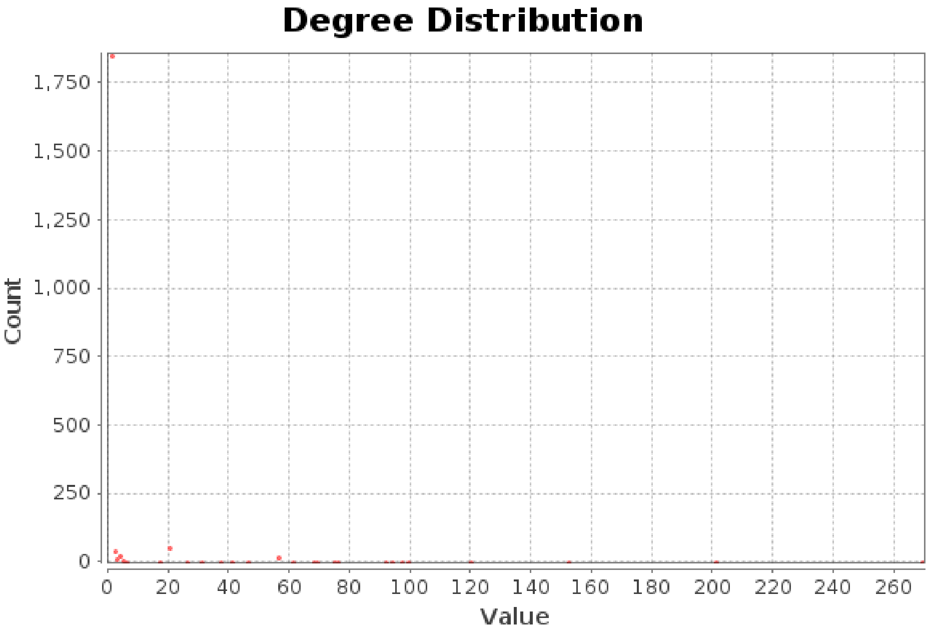


HITS Metrics

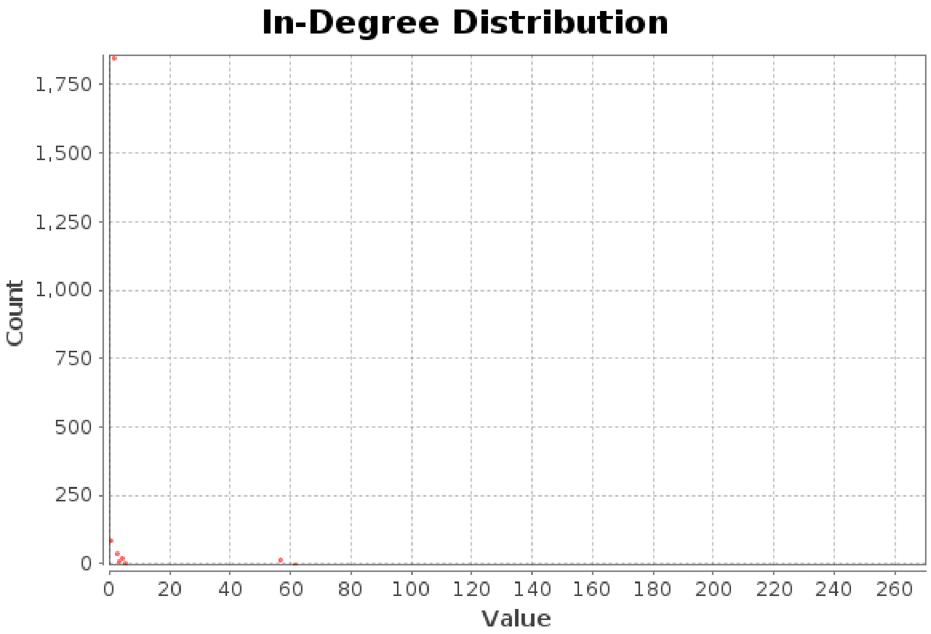


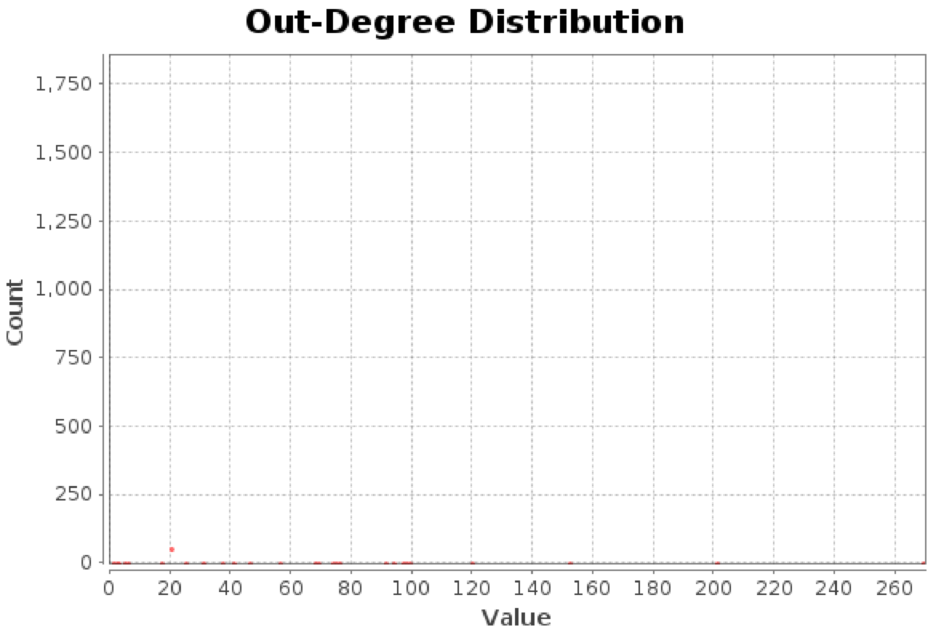


PageRank

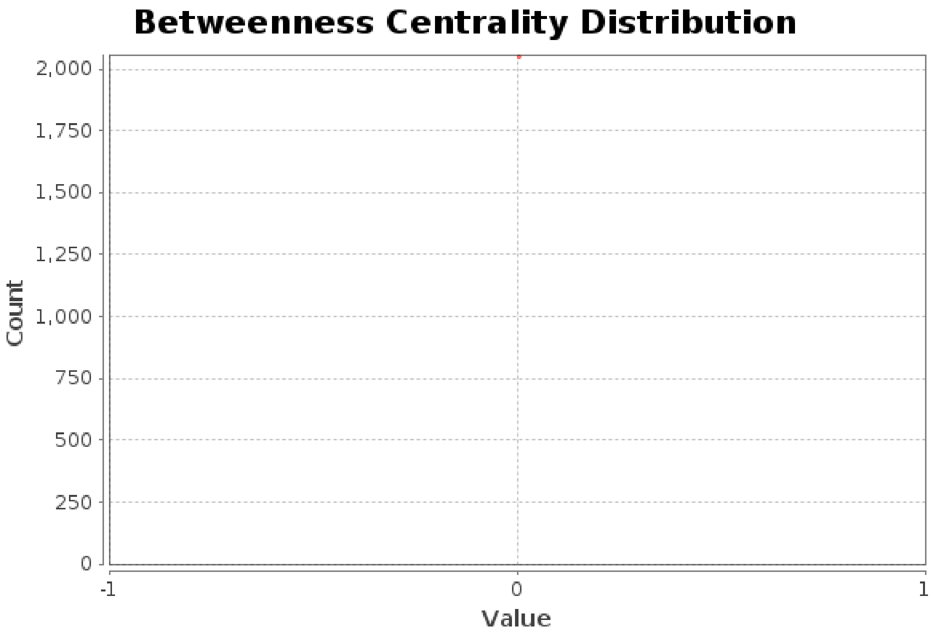


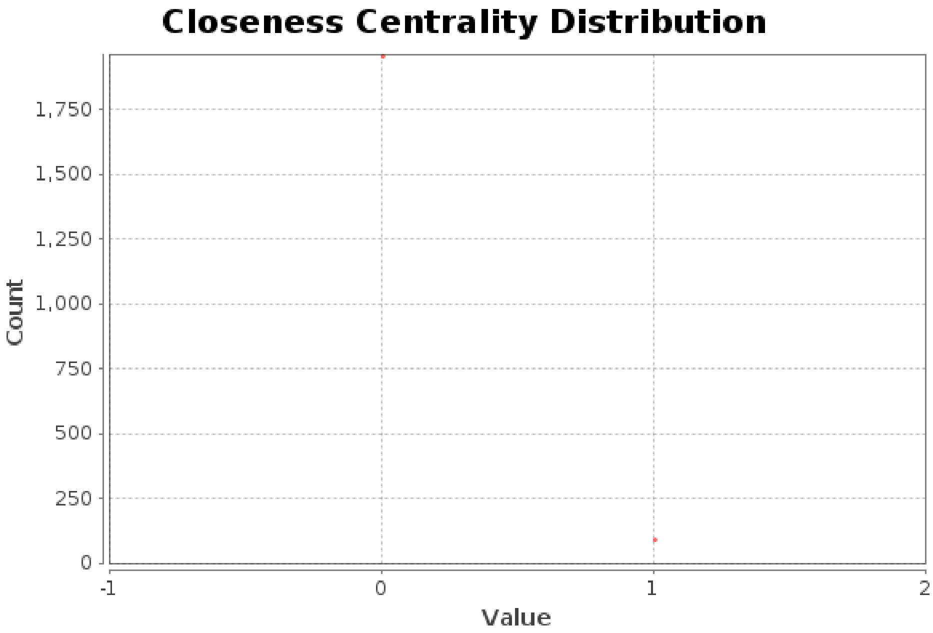
Avg Degree: 1.553

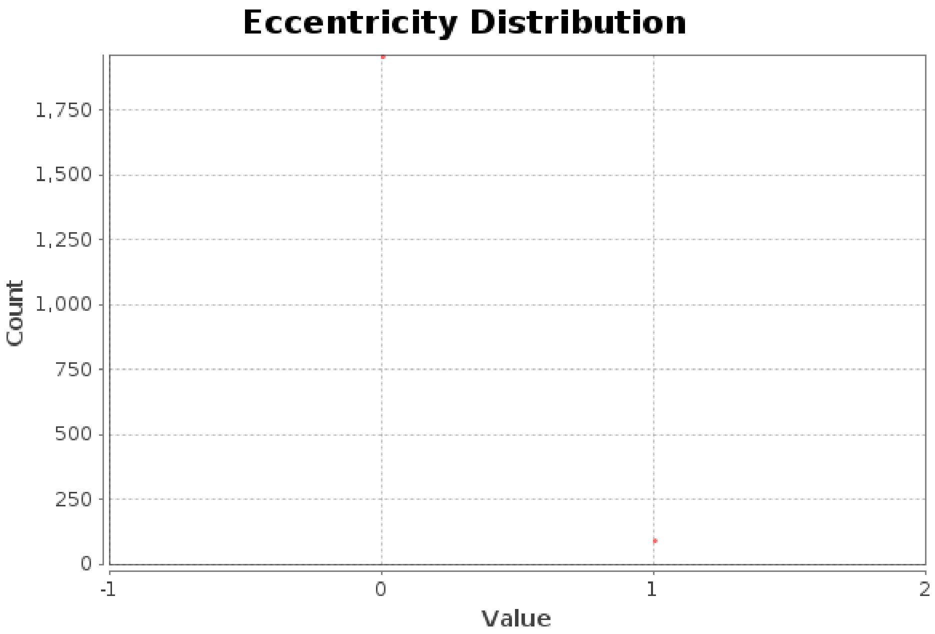




Network Diameter
Diameter: 1
Radius: 0
Average Path length: 1.0
Number of shortest paths: 3182


Betweenness Centrality Distribution
Count
2,000
1,750
1,500
1,250
1,000
750
500
250
0
-1
0
1
Value

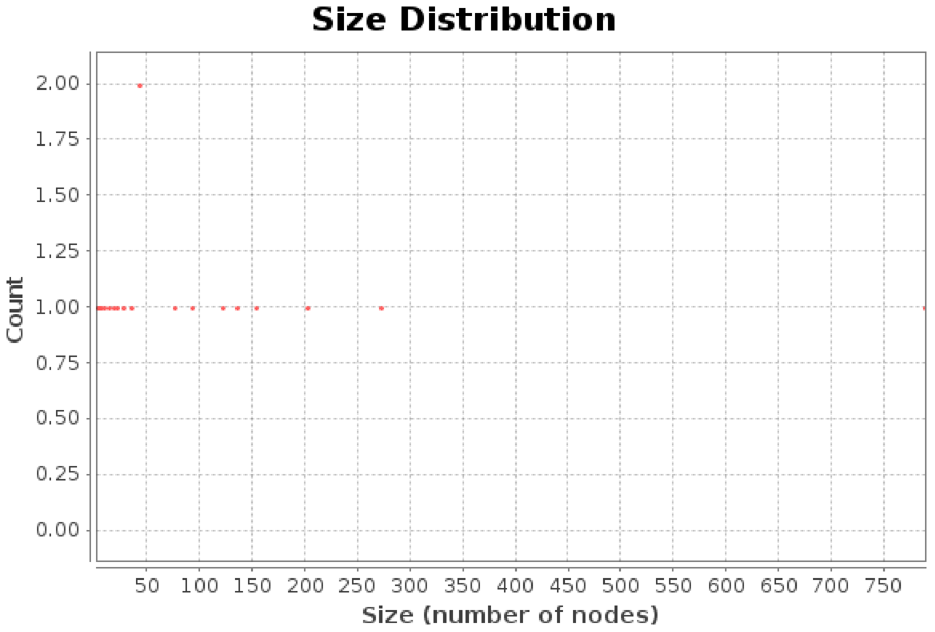

Closeness Centrality Distribution
Count
1,750
1,500
1,250
1,000
750
500
250
0
-1
0
1
2
Value



Connected Components

Number of Weakly Connected Components: 19

Number of Stronlgy Connected Components: 2052



References

[1] https://gephi.github.io/