CS 595: Assignment 2

Mallika Kogatam

Fall 2014

Contents

1 Problem 1 2
1.1 Answer 3
1.2 Code Listing 3
2 Problem 2 5
2.1 Answer 6
2.2 Code Listing 6
3 Problem 3 7
3.1 Answer 7

1 Problem 1

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

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 Answer

1. To get the 100 links, I did not pick them randomly from the 1000 links which we got from the Assignment 2. I just checked for the first 100 links
2. I gave the unique links file as a command line argument and then read the lines only till I get 100 URIs which are working.
3. I converted the URI to md5 hash so that I can save the files with the hash instead of URI which might have special characters
4. By using beautiful soup I got the content which is there in anchor tag and which are only href.
5. I ignore the line which I got from the previous step if it does not start with http:// or https://.
6. I save all the URIs that are collected by the above process in a file which is named with md5 hash value which we get by converting the parent URI to md5.
7. All these 100 files are saved in a folder which has a name as the creation time of that folder.

1.2 Code Listing

```
#! /usr/bin/env python

# Mallika Kogatam
# Fall 2014
# Assignment 4 part 1

import os
import sys
import urllib2
import time
import md5

from datetime import datetime

from bs4 import BeautifulSoup

LIMIT = 100

def main():

    if len( sys.argv ) != 2:
        print "Usage Error"
        sys.exit(1)

    url_filename = sys.argv[1]

    try:
        url_file = open(url_filename,"r")
    except IOError:
        print "File {} does not exist".format( url_filename )
        sys.exit(1)

    counter = 1



    folder = datetime.now().strftime("%Y-%m-%d.%H.%M.%S")

    os.mkdir( folder )

    log_file = open( folder + ".log", "w" )

    print folder

    for url in url_file:

        url = url.strip()

```

```
49          print "%"*40
50          print url
51
52          url_hash = md5.md5(url).hexdigest()
53
54          log_file.write( url_hash + " " + url + "\n" )
55          log_file.flush()
56
57          try:
58              html = urllib2.urlopen( url ).read()
59              soup = BeautifulSoup(html)
60
61              links = soup.find_all("a")
62
63              name = url_hash + ".links"
64              link_file = open( "{}/{}".format( folder, name ) ,'w')
65              print "{}/{}".format( folder, name )
66
67              for link in links:
68                  try:
69
70                      href = link["href"]
71
72                      if "http://" in href or "https://" in href:
73                          try:
74                              link_file.write( href + "\n" )
75                              link_file.flush()
76                          except UnicodeEncodeError:
77                              pass
78
79                  except KeyError:
80                      pass
81              counter += 1
82              link_file.close()
83
84          except urllib2.HTTPError, urllib2.URLError:
85              pass
86
87          if counter >= LIMIT:
88              break
89
90      log_file.close()
91
92  if __name__ == "__main__":
93      try:
94          main()
95      except KeyboardInterrupt:
96          sys.exit(1)
```

Listing 1: Python Program for printing out links from 100 pages

2 Problem 2

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.)
```

2.1 Answer

1. To generate a DOT file I wrote a small Python program which reads all the files that are created in the first question.

2. I have generated a log file in the question 1 which contains a md5 hash value for the respective URI and I am gonna make use of that file now to get the URI for respective file name

3. I read all the files which are generated in the question 1 and map those links with the respective parent URI.

4. Write all the links in a DOT file(linksgraph.dot) along with labels for each link as required.

2.2 Code Listing

```
1  #! /usr/bin/env python
2
3  import os
4  import sys
5  def main():
6
7      if len( sys.argv ) != 2:
8          print "Usage Error"
9          sys.exit(1)
10
11     directory_name = sys.argv[1]
12     log_filename    = directory_name + ".log"
13     print ("digraph dotgraph {")
14     #print directory_name
15     #print log_filename
16
17     log_file = open(log_filename, "r")
18
19     table_url_hash = { hash: url for hash, url in [
20             line.strip().split() for line in log_file
21         ]
22     }
23
24     #print table_url_hash
25
26     for root, _ , files in os.walk(directory_name):
27         for filename in files:
28             links_file = open(os.path.join(root , filename))
29             content = links_file.readlines()
30             links_file.close()
31
32             hash = filename[0:filename.find(".links")]
33
34             url = table_url_hash[hash]
35
36             for link in content:
37                 link = link.strip()
38                 print  '"' +url + '" -> "' + link + '"' + '[label = "' + url + ' links to '
                       + link + '"];'
39
40     print "}"
41
42 if __name__ == "__main__":
43     try:
44         main()
45     except KeyboardInterrupt:
46         sys.exit(1)
```

Listing 2: Python Program for generating a DOT file

3 Problem 3

Question

```
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.
```

3.1 Answer

1. The graph described by the dot file in Question 2 contain 3021 nodes and 3176 edges.
2. From Gephi, visualizing the graph produced an unclear graph shown in Figure 1
3. So I used Yifan Hu Proportional layout, the graph became much better as shown in Figure 2. This graph is with out labels.
4. After that I enabled the labels for the Figure 2 which resulted as Figure 3
5. I saved the Figure 1 , 2 and 3 in pdf format from Gephi so that we can see the labels clearly if image is zoomed.
6. Figure 4 shows low authority scores on all pages. Figure 5 shows what look to be two hubs.
7. PageRank for all pages approaches 0 as shown in Figure 6 because so many pages link to one another.
8. Figure 8 shows that very few nodes have in-degree values. Surprisingly Figure 9 shows that a lot of pages do seem to link out but in an almost horizontal linear pattern.
9. The graph has a network diameter of 1 , with 3173 shortest paths. The average path length is 1.0.
10. Figure 11 shows most of the nodes are so connected that their closeness is quite low.
11. Figure 13 shows that 3201 components are strongly connected in this graph. And 58 components are weakly connected.

Graph with Yifan Hu layout with labels

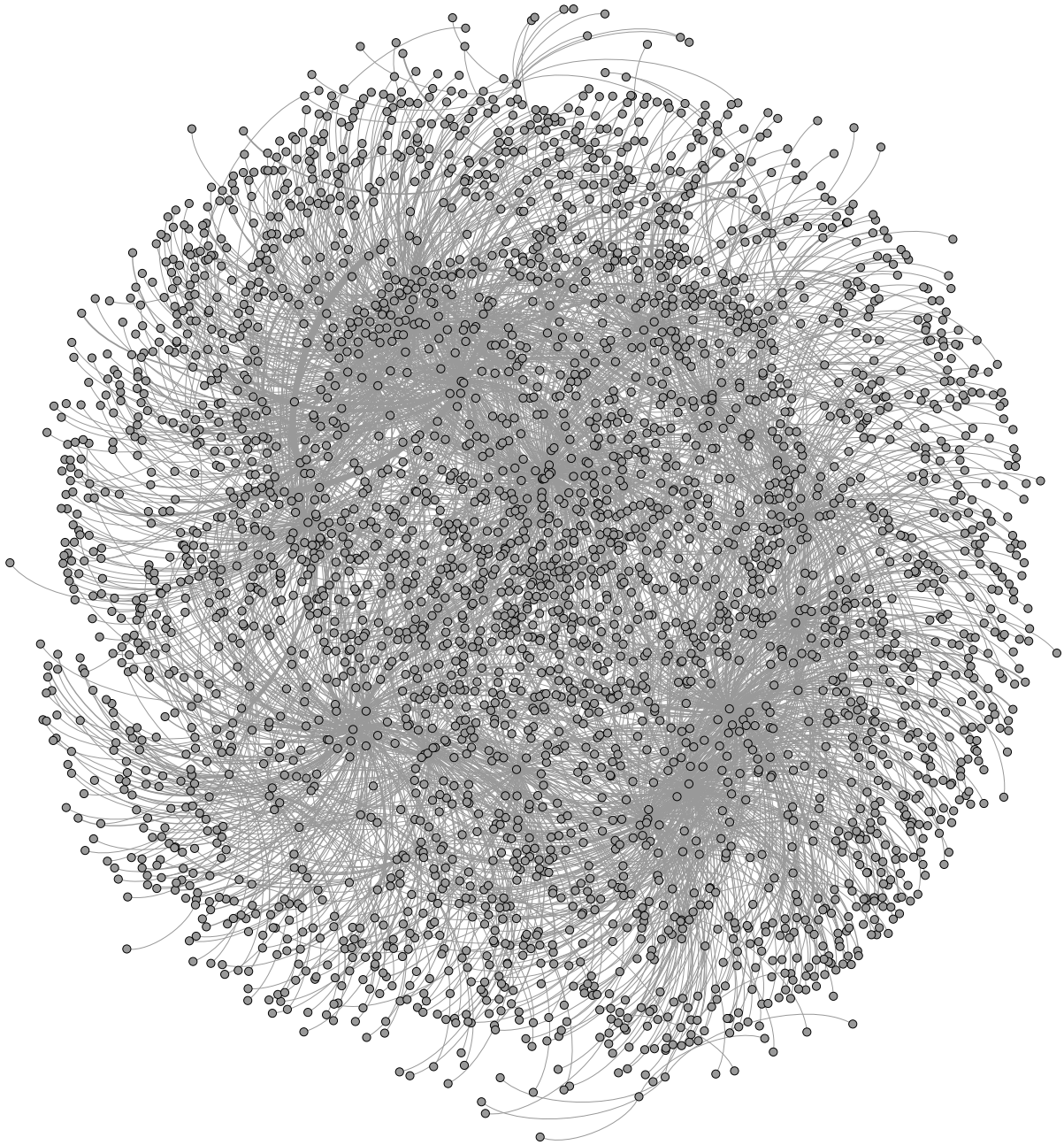

Figure 1: Initial Visualization of the graph for DOT file from Question 2

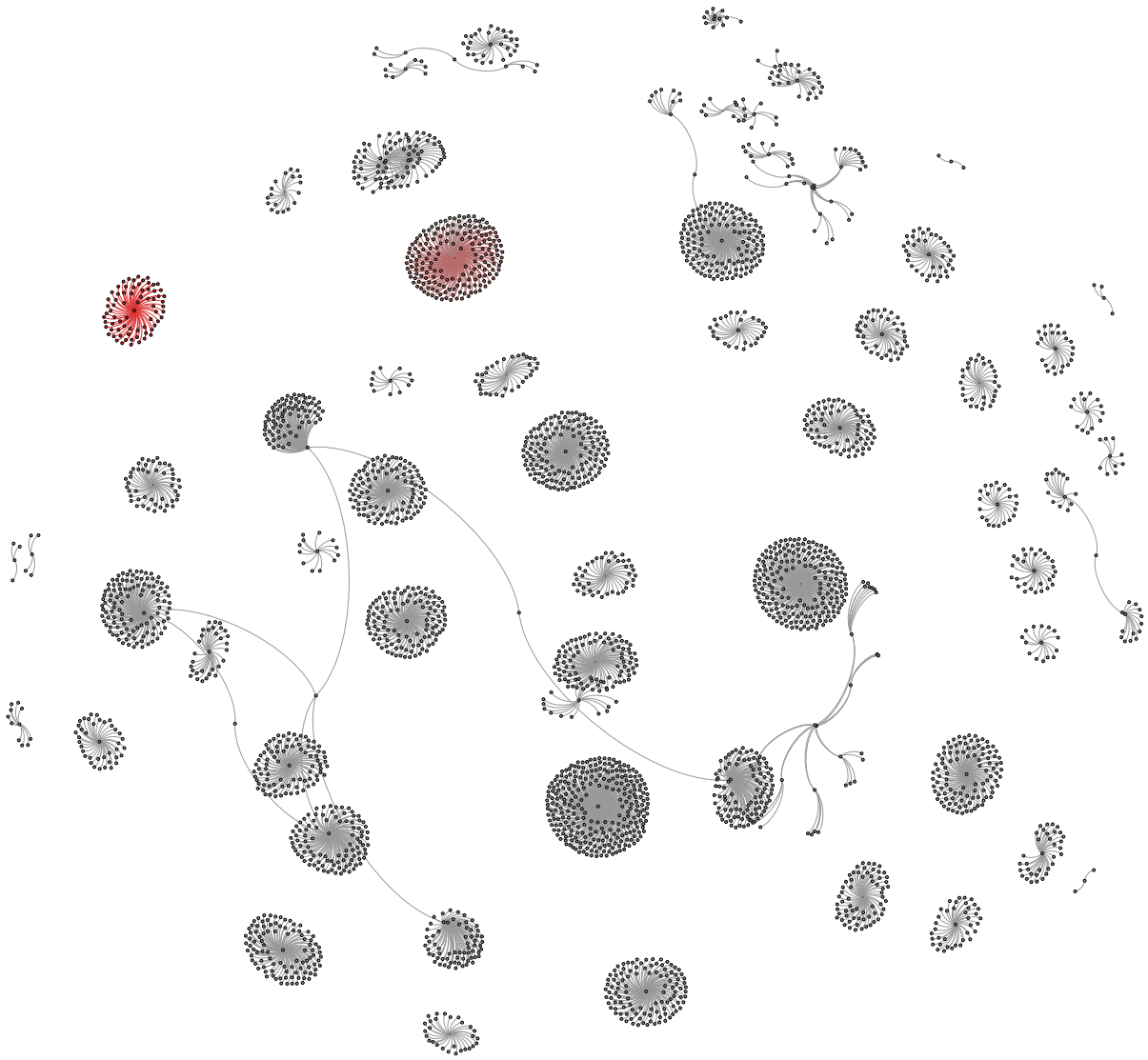

Figure 2: Graph with Yifan Hu layout

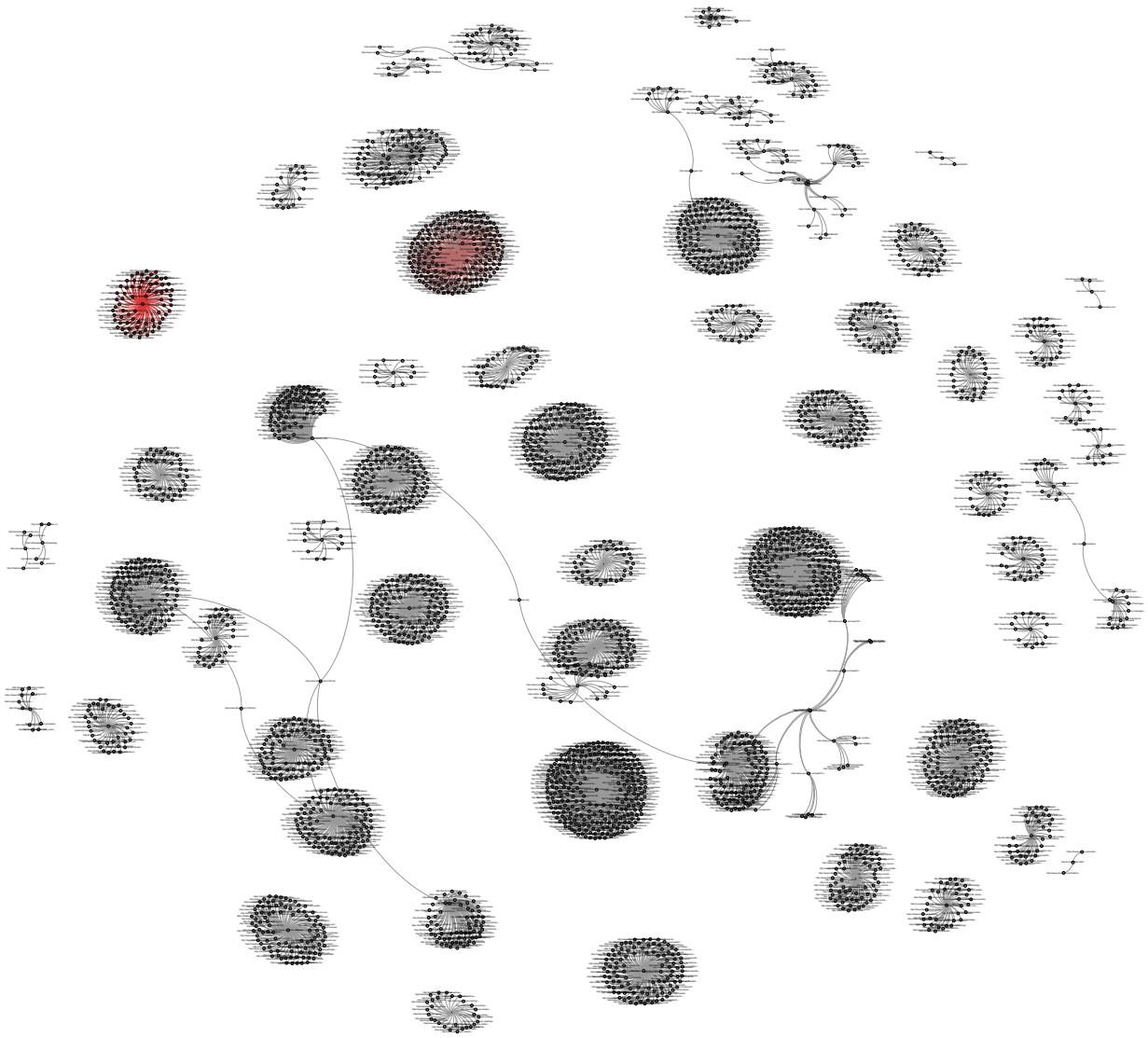

Figure 3: Graph with Yifan Hu layout with labels

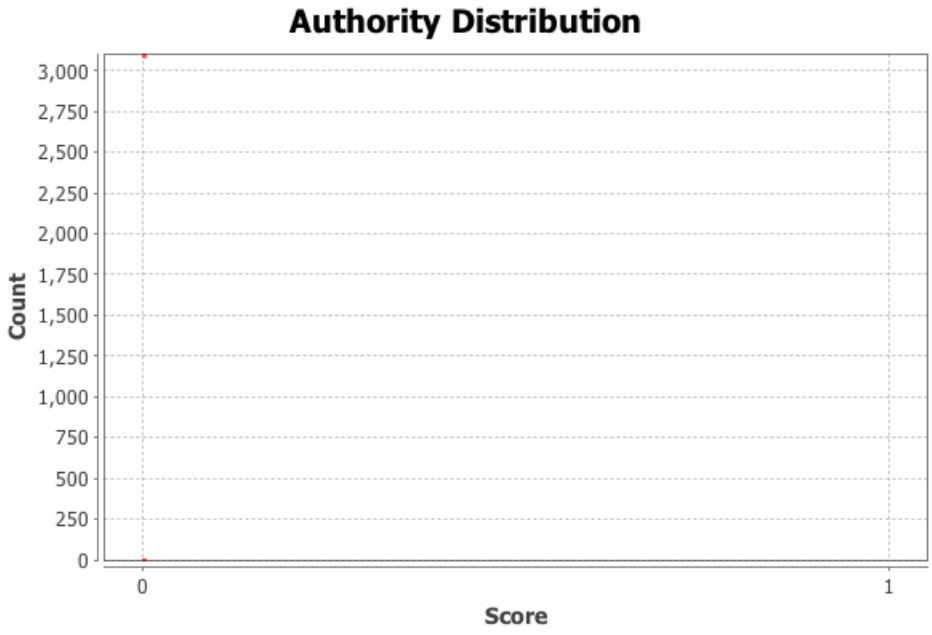

Figure 4: Authority Distribution for HITS

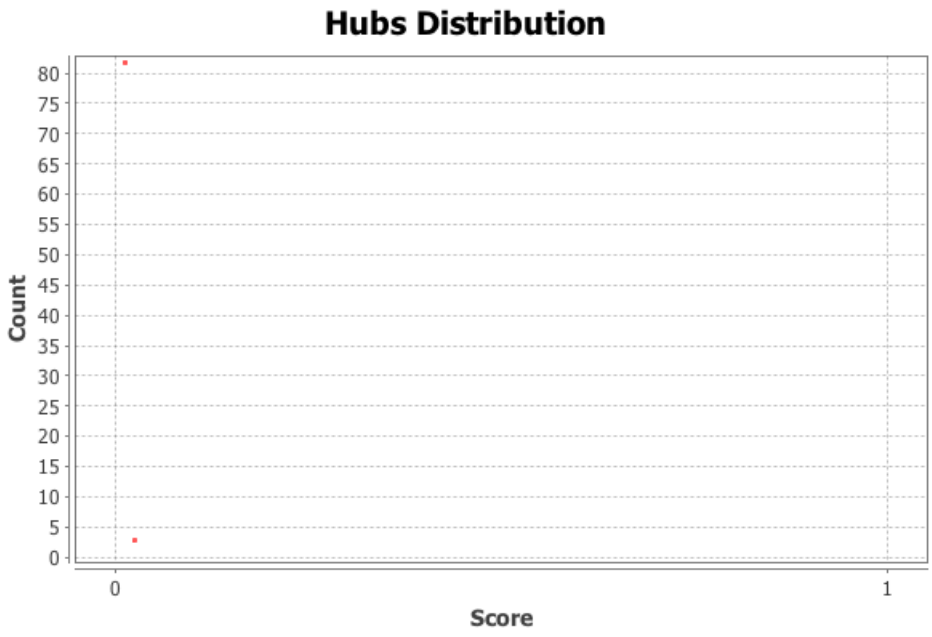

Figure 5: Authority Distribution for HITS

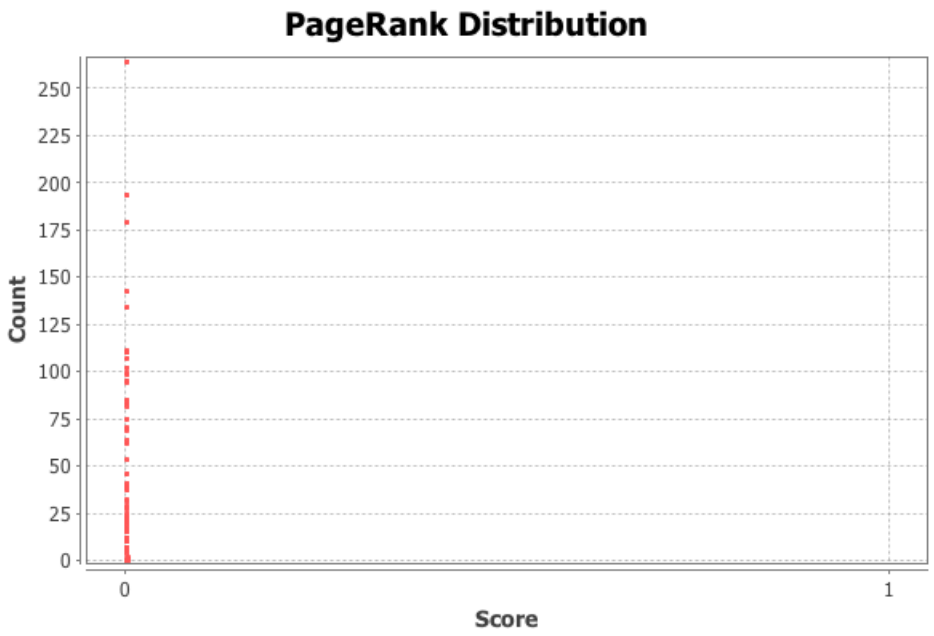

Figure 6: PageRank Distribution for PageRanks

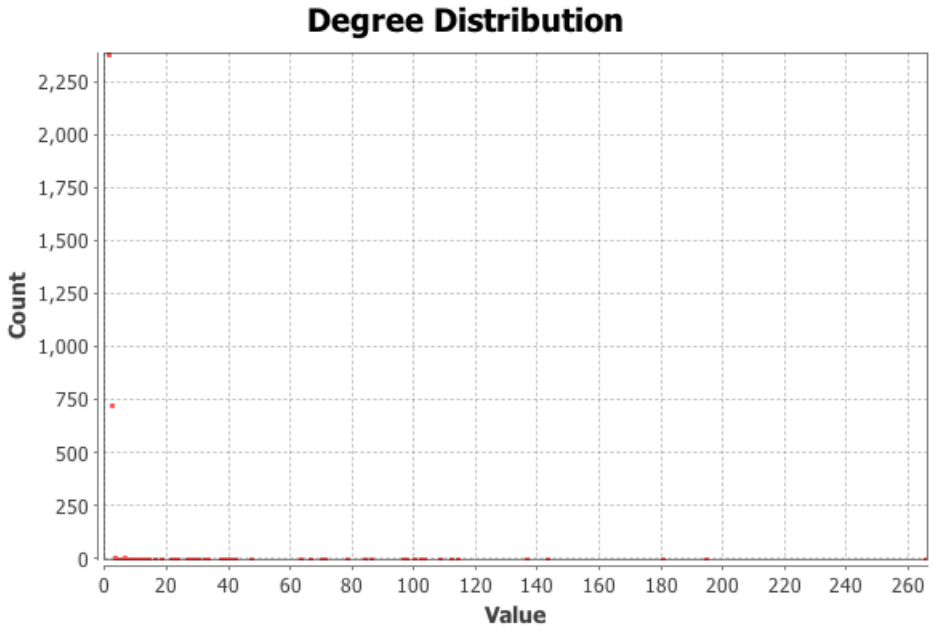

Figure 7: Degree Distribution for Average degree

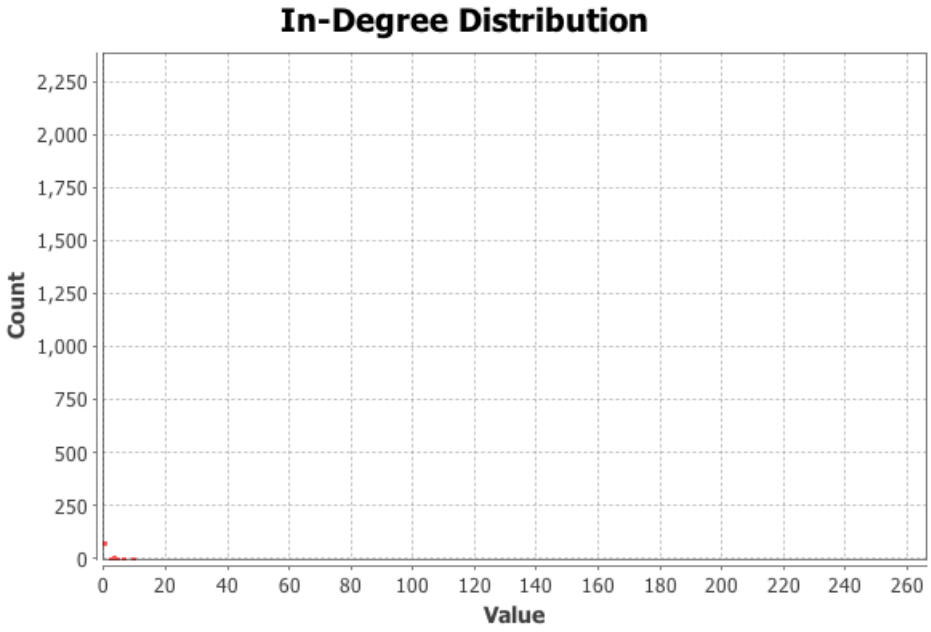

Figure 8: In-Degree Distribution for Average degree

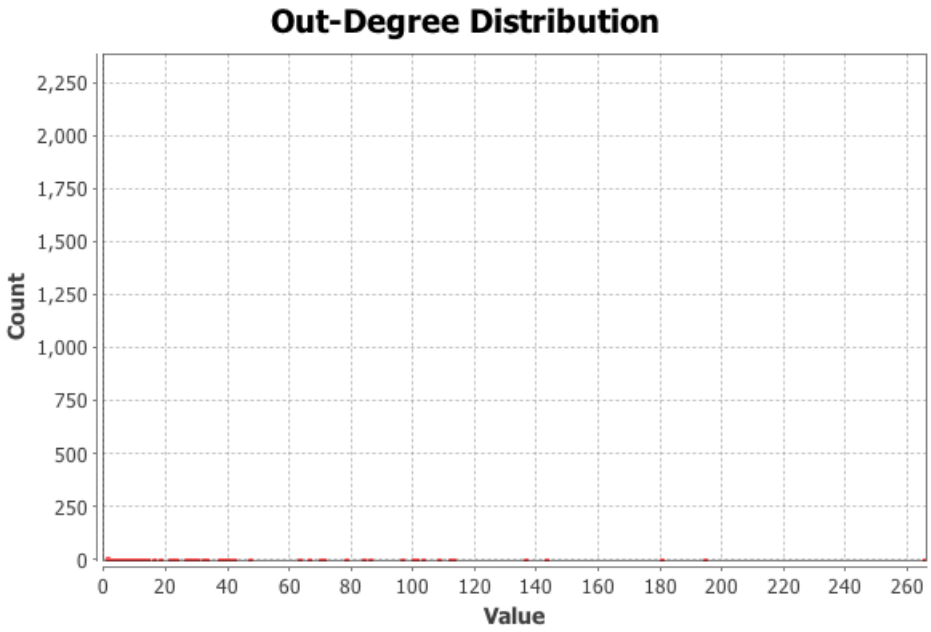

Figure 9: Out-Degree Distribution for Average degree

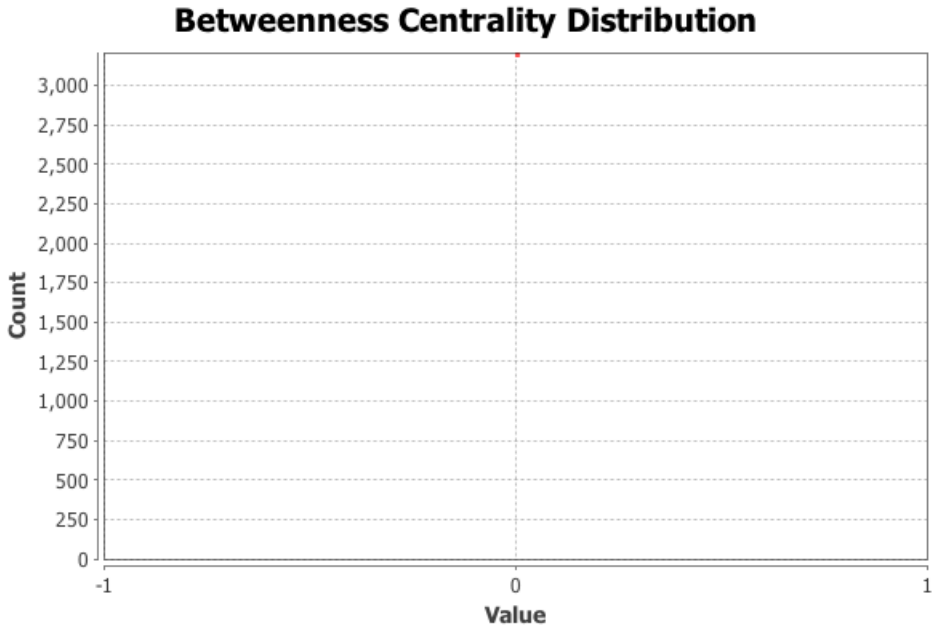

Figure 10: Betweenness Centrality Distribution for Network Diameter

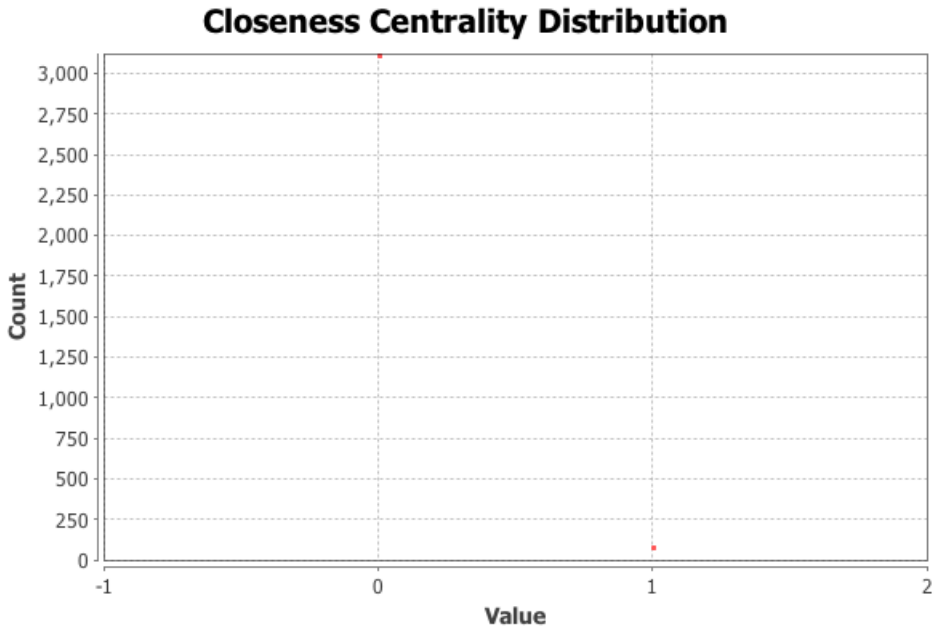

Figure 11: Closeness Centrality Distribution for Network Diameter

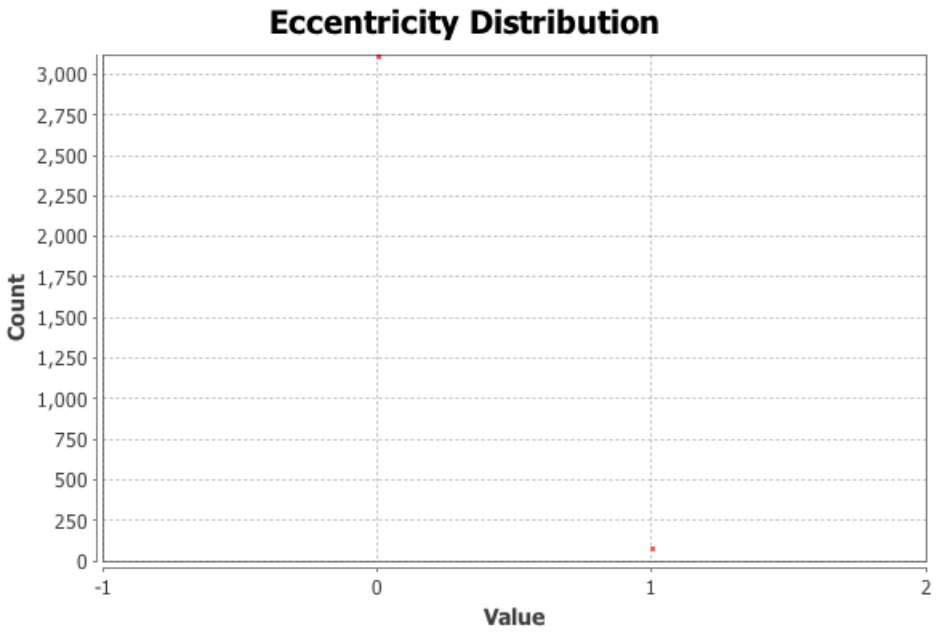

Figure 12: Eccentricity Distribution for Network Diameter

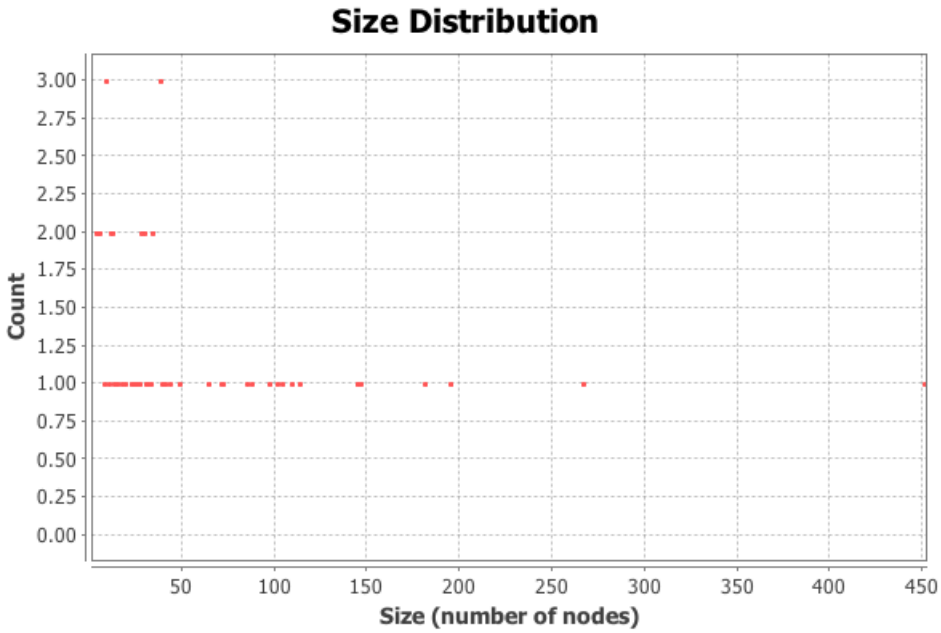

Figure 13: Size Distribution for Connected Components

Bibliography

[1] Basic date and time types. https://docs.python.org/2/library/datetime.html.

[2] Graph visualization software. http://www.graphviz.org/content/dot-language.

[3] Graph visualization software. http://www.graphviz.org/content/unix.

[4] Graphviz dot format. https://gephi.github.io/users/supported-graph-formats/graphviz-dot-format/.

[5] Inserting a pdf file in latex. http://stackoverflow.com/questions/2739159/inserting-apdf-file-in-latex.

[6] Inserting and labelling figures. http://www.stat.berkeley.edu/ paciorek/computingTips/Inserting_labelling_figures.html.