INTRODUCTION TO WEB SCIENCES:
Assignment 4

Babitha Bokka

10 october 2014

Contents

1 Question 1

Choose 100 links from 1000 unique links and extract the outbound links from that page to other URIs.

1.1 Approach

To extract the outgoing links from the 100 URIs extractHyperLinks.py is loaded with the 1000 unique URIs. The LIMIT value in the program selects only 100 URIs, these are URIs which can establish the connection and has no exceptions.

1.2 Description of extractHyperLinks.py

1. Open the A2FinalOutput.txt.
2. Read each URL and extract the outgoing links.
3. Out of 1000 Unique URIs the Program will extract 100 working URIs and get the outgoing links from each URI.
4. Create a md5 file and write all the links to the ".link" file at the same time a log file (md5Links.txt) is generated which has the md5 and URL.
5. All the file are saved in A4/Q1/links.

1.3 Source Code

1.3.1 extractHyperLinks.py

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

import os
import md5
import sys
import time
import socket
import urllib2
import unicodedata
from bs4 import BeautifulSoup

LIMIT = 100

#Main Function
def main():
    # open the 100 links file
    uniqueLinks = open('A2FinalOutput.txt', 'r')
    writeFile   = open('md5Links.txt','w')

    counter = 0
    for url in uniqueLinks.readlines() :
        hashmd5    = md5.new(url).hexdigest()
        filename    = 'links/' + hashmd5 + '.link'
        try :
            request     = urllib2.Request(url)
            response    = urllib2.urlopen(url,timeout=30)
            html_content= response.read()

            # get the html content using beautiful soup
            soup        = BeautifulSoup(html_content)
            links       = soup.find_all('a')

            writeFile.write("{:<20} {}" .format(hashmd5,url))
            counter += 1

            saveFile= open(filename,'w')

            for tag in links:
                try :
                    link  = tag.get('href',None)
                    if link != None and link.startswith( "http" ):

                        saveFile.write(link)
                        saveFile.write('\n')

                except UnicodeEncodeError:
                    pass
            saveFile.close()

        except urllib2.HTTPError :
            pass
        except urllib2.URLError :
            pass
```

```
54          except socket.timeout :
55              pass
56          print filename , counter
57          print "-" * 50
58          # get only the 100 urls data
59          if counter >= LIMIT:
60              break
61
62
63  if __name__ == "__main__":
64      try:
65          main()
66      except KeyboardInterrupt:
67          sys.exit(1)
```

1.4 Input Files

1.4.1 A2FinalOutput.txt

```
Unique URIs

https://www.facebook.com/MyChickenRun
http://www.youtube.com/watch?v=Eubi9YI2dKE
http://geladoesntgiveadamn.tumblr.com/
http://youtu.be/1BKO2V9EaZ0?a
http://www.instagram.com/the_sanging_rebel
http://instagram.com/teustimao/
http://blogmylunch.com
http://Facebook.com/Robbiewhaylez
http://ifoodi.blogspot.com/
http://www.talkinggoodfood.co.uk
http://facebook.com/dimano.sterling
http://www.yellowkorner.com
http://Emerald.com
http://attackontiphan.tumblr.com
http://instagram.com/xxxmn88
http://linggez-network.blogspot.com
http://Instagram.com/hellocalm_
http://www.wagamama.com
http://pennyroyaltea.co.vu
http://www.oufancyphones.com
http://www.facebook.com/rappstartailgate
http://thedaintypig.com
http://instagram.com/victorparrini
http://instagram.com/_lafamiliaonly
http://rad-acid.tumblr.com
https://www.Youtube.com/iFarLiez
http://twitter.com
http://melonpatchtv.com
http://linkd.in/1qVvTtZ
http://www.huffingtonpost.com/alex-palombo/
http://youtube.com/user/IvanAlvir
http://www.musicsumo.com
http://www.KhanaPakana.com
```

1.5 Output Files

Aim : To generate the 100 ".link" files which have the outbound links from that page to other URIs.The files are saved in A4/Q1/links.

1.5.1 md5Links.txt

The file acts as a logfile to keep track of URI and corresponding md5 file.

```
md5 file name                    URI

5414920b1ad98e5a5c2fe91a48c78071 https://www.facebook.com/MyChickenRun
8a7aaf3d96587b6edd9381207370a03d http://www.youtube.com/watch?v=Eubi9YI2dKE
a524329028e8dcfc4879b4b453c1040a http://geladoesntgiveadamn.tumblr.com/
96dec514297f45ff7a40f0ec5b527dfc http://youtu.be/1BKO2V9EaZ0?a
10d84f5c5226fe47690138352748aa07 http://www.instagram.com/the_sanging_rebel
75c522f8af14203e53d45345fffa79ea http://instagram.com/teustimao/
c7a1e9146029a56d3fa0acab5d66688b http://blogmylunch.com
42305950be948963a249dc2e7c82e581 http://Facebook.com/Robbiewhaylez
aec16fa2999f0e1ed985ba24fc9793d0 http://ifoodi.blogspot.com/
db4488dcef3ed8da9fc7bb6b459c9cde http://www.talkinggoodfood.co.uk
```

1.5.2 a524329028e8dcfc4879b4b453c1040a.link

A sample ".link" file having the outgoing links.

```
URI
http://geladoesntgiveadamn.tumblr.com/
Links
http://www.cherrybam.com/blackandwhite-tumblr-themes.php
http://www.cherrybam.com
https://www.tumblr.com/reblog/81587066245/kEKZcUHw
http://geladoesntgiveadamn.tumblr.com/post/81586972499/toxicgaskarth-all-time-low-so-long-soldier
https://www.tumblr.com/reblog/81586972499/CUjrIucX
http://geladoesntgiveadamn.tumblr.com/post/81586832033/jackbarakatofficial-all-time-low-insprired-by
https://www.tumblr.com/reblog/81586832033/4BCaP7rD
http://geladoesntgiveadamn.tumblr.com/post/81586733974/q-whats-the-worst-excuse-you-have-ever-used-for
https://www.tumblr.com/reblog/81586733974/KrMsNi8q
http://geladoesntgiveadamn.tumblr.com/post/81586630503
https://www.tumblr.com/reblog/81586630503/2ZXuDBxA
http://www.flickr.com/photos/thenmaysuhsaid/5744956806/
http://geladoesntgiveadamn.tumblr.com/post/81585758862/yeahbarakat-the-maine-by-maysa-askar-on-flickr
https://www.tumblr.com/reblog/81585758862/jBh5DfiU
http://geladoesntgiveadamn.tumblr.com/post/81585751637/john-who-the-fuck-even-sent-you-ocallaghan-x
https://www.tumblr.com/reblog/81585751637/zBgMQ9uU
http://geladoesntgiveadamn.tumblr.com/post/80128811883
https://www.tumblr.com/reblog/80128811883/8GQakRrF
http://foreverhalloween.us/
http://geladoesntgiveadamn.tumblr.com/post/80128520734/austincxrliles-original-credit
https://www.tumblr.com/reblog/80128520734/Y6WamLQe
http://geladoesntgiveadamn.tumblr.com/post/77538402738
https://www.tumblr.com/reblog/77538402738/2LVznsYP
```

2 Question 2

Generate a single DOT file for 100 links.

2.1 Description of createDotFile.py

1. Open the md5Links.txt, extract the url and md5.
2. Using glob to find the respective folder.
3. Check whether md5 of the current URL and the file name in the folder are matching if, they match then open the file and append the links inside each file to the URL
4. Write all the URL and respective links to “mapping.dot” file.

2.2 Source Code

2.2.1 createDotFile.py

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

import os
import sys
import glob


#Main Function
def main():

    path        = "/home/bbokka/cs594/A4/Q1/links/*.link"
    md5_Url     = open('md5Links.txt','r')
    mapping     = open('mapping.dot','w')
    mapping.write('digraph A4_question3 { size = "6,6"; node [color = lightblue2 , style = filled]; ')
    for line in md5_Url.readlines():
        line = line.split(" ")
        md5  = line[0]
        url  = line[1]

        first_spit          = url.split("://")
        store_first_split   = first_spit[1]
        second_split        = store_first_split.split("/")
        store_second_split  = second_split[0]
        url_label_string        = store_second_split
        #print label_string

        md5_file = glob.glob(path)
        for each_md5_file in md5_file:
            #print each_md5_file
            filename      = each_md5_file.split("/links/")
            md5_file_name = filename[1]
            md5_code      = md5_file_name.split(".link")
            md5_code_file = md5_code[0]
            #print md5_code_file
            if (md5 == md5_code_file):
                open_md5 = open(each_md5_file,'r')
                for link in open_md5.readlines():
                    first_spit_1        = link.strip().split("://")
                    store_first_split_1  = first_spit_1[1]
                    second_split_1       = store_first_split_1.split("/")
                    store_second_split_1  = second_split_1[0]
                    link_label_string        = store_second_split_1
                    link = link.strip()
                    url  = url.strip()
                    #string       = ' "'+url.strip()+'"' +'->' +'" ' +links.strip()+'"[' + 'label =' +'"'+ label_string + '"' +'];'
                    string         = '"'+ url.strip() + '"'+' '+'->'+' '+'"'+ link.strip() + '"'+ ' \n '+'"'+ url.strip()   + '"'+' '+ '[label =' + '"'+ url_label_string  + '"' +'];'+' \n ' +'"'+ links + '"'+' '+ '[label =' + '"'+ link_label_string + '"' +']' +';'
                    mapping.write(string)
                    mapping.write('\n')
```

```
49      mapping.write ('}')
50
51
52
53  if __name__ == "__main__":
54      try:
55          main()
56      except KeyboardInterrupt:
57          sys.exit(1)
```

2.3 Input Files

2.3.1 md5Links.txt

```
1  md5 file name                    URI
2
3  5414920b1ad98e5a5c2fe91a48c78071 https://www.facebook.com/MyChickenRun
4  8a7aaf3d96587b6edd9381207370a03d http://www.youtube.com/watch?v=Eubi9YI2dKE
5  a524329028e8dcfc4879b4b453c1040a http://geladoesntgiveadamn.tumblr.com/
6  96dec514297f45ff7a40f0ec5b527dfc http://youtu.be/1BKO2V9EaZ0?a
7  10d84f5c5226fe47690138352748aa07 http://www.instagram.com/the_sanging_rebel
8  75c522f8af14203e53d45345fffa79ea http://instagram.com/teustimao/
9  c7a1e9146029a56d3fa0acab5d66688b http://blogmylunch.com
10 42305950be948963a249dc2e7c82e581 http://Facebook.com/Robbiewhaylez
11 aec16fa2999f0e1ed985ba24fc9793d0 http://ifoodi.blogspot.com/
12 db4488dcef3ed8da9fc7bb6b459c9cde http://www.talkinggoodfood.co.uk
```

2.4 Output Files

2.4.1 mapping.txt

```
digraph A4_question3 {
  size = "6,6";
  node [color = lightblue2 ,style = filled];
  "https://www.facebook.com/MyChickenRun" -> "https://www.facebook.com/recover/initiate"
  "https://www.facebook.com/MyChickenRun" [label ="www.facebook.com"];
  "https://www.facebook.com/recover/initiate" [label ="www.facebook.com"];
  "http://www.youtube.com/watch?v=Eubi9YI2dKE" -> "https://plus.google.com/+youtube"
  "http://www.youtube.com/watch?v=Eubi9YI2dKE" [label ="www.youtube.com"];
  "https://plus.google.com/+youtube" [label ="plus.google.com"];
  "http://www.youtube.com/watch?v=Eubi9YI2dKE" -> "https://www.google.com/intl/en/policies/privacy/"
  "http://www.youtube.com/watch?v=Eubi9YI2dKE" [label ="www.youtube.com"];
  "https://www.google.com/intl/en/policies/privacy/" [label ="www.google.com"];
  "http://www.youtube.com/watch?v=Eubi9YI2dKE" -> "https://accounts.google.com/ServiceLogin?continue=https%3A%2F%2Fwww.youtube.com%2Fsignin%3Fhl%3Den%26app%3Ddesktop%26next%3D%252Fwatch%253Fv%253DEubi9YI2dKE%26feature%3Dplaylist%26action_handle_signin%3Dtrue&hl=en&service=youtube&passive=true&uilel=3"
  "http://www.youtube.com/watch?v=Eubi9YI2dKE" [label ="www.youtube.com"];  "https://accounts.google.com/ServiceLogin?continue=https%3A%2F%2Fwww.youtube.com%2Fsignin%3Fhl%3Den%26app%3Ddesktop%26next%3D%252Fwatch%253Fv%253DEubi9YI2dKE%26feature%3Dplaylist%26action_handle_signin%3Dtrue&hl=en&service=youtube&passive=true&uilel=3" [label ="accounts.google.com"];
}
```

3 Question 3

Download and install Gephi then load the dot file to visualize the graph.

3.1 Approach

Downloaded gephi and loaded the mapping.dot file into the software. In order to get more insight of the visualization layouts Fruchterman, Reingold and Yifan Hu and Yifan Hu Proportional are used. Run the layouts couple of time to get more insight and export them to a pdf or png format.

3.2 Input Files

The input is the output of the Question 2.

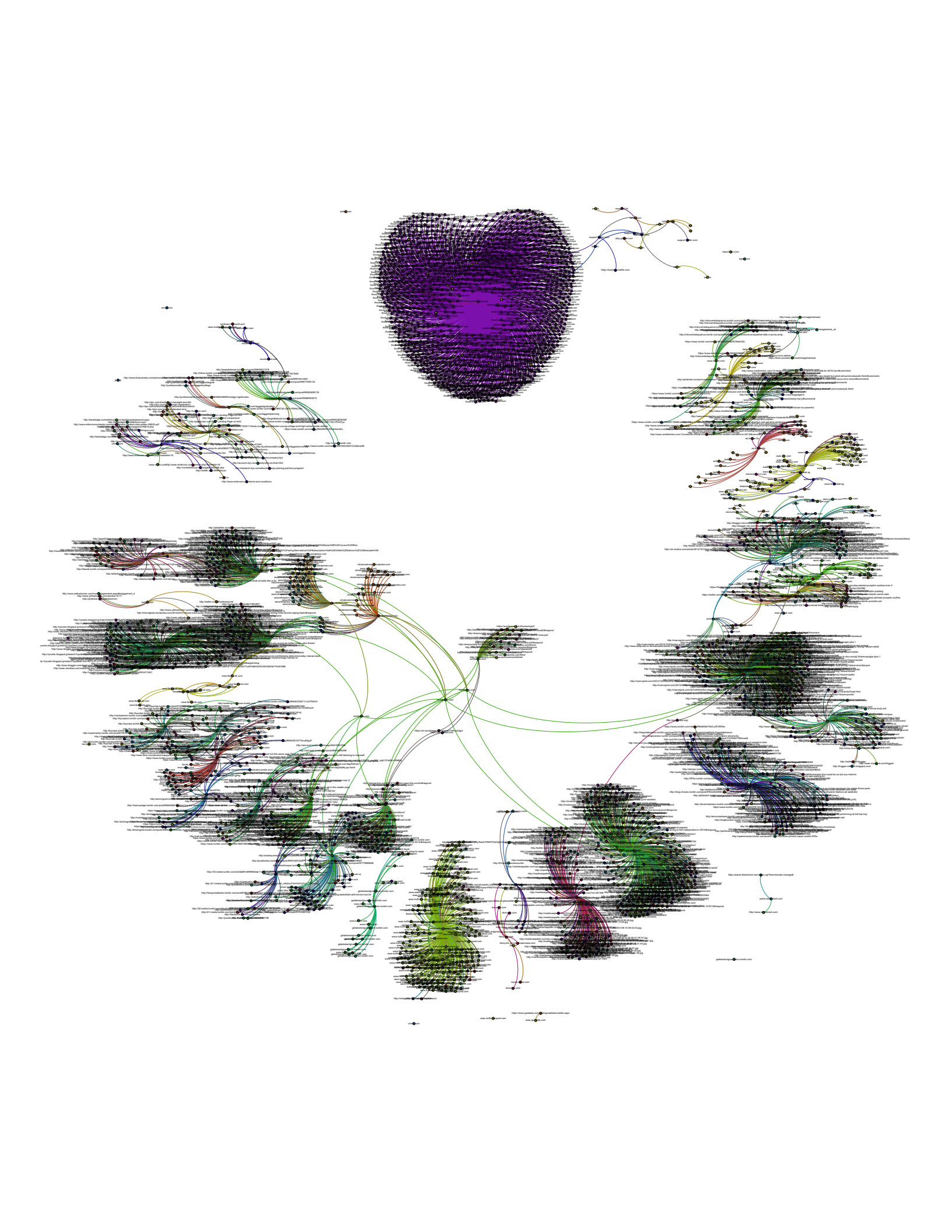
3.2.1 mapping.txt

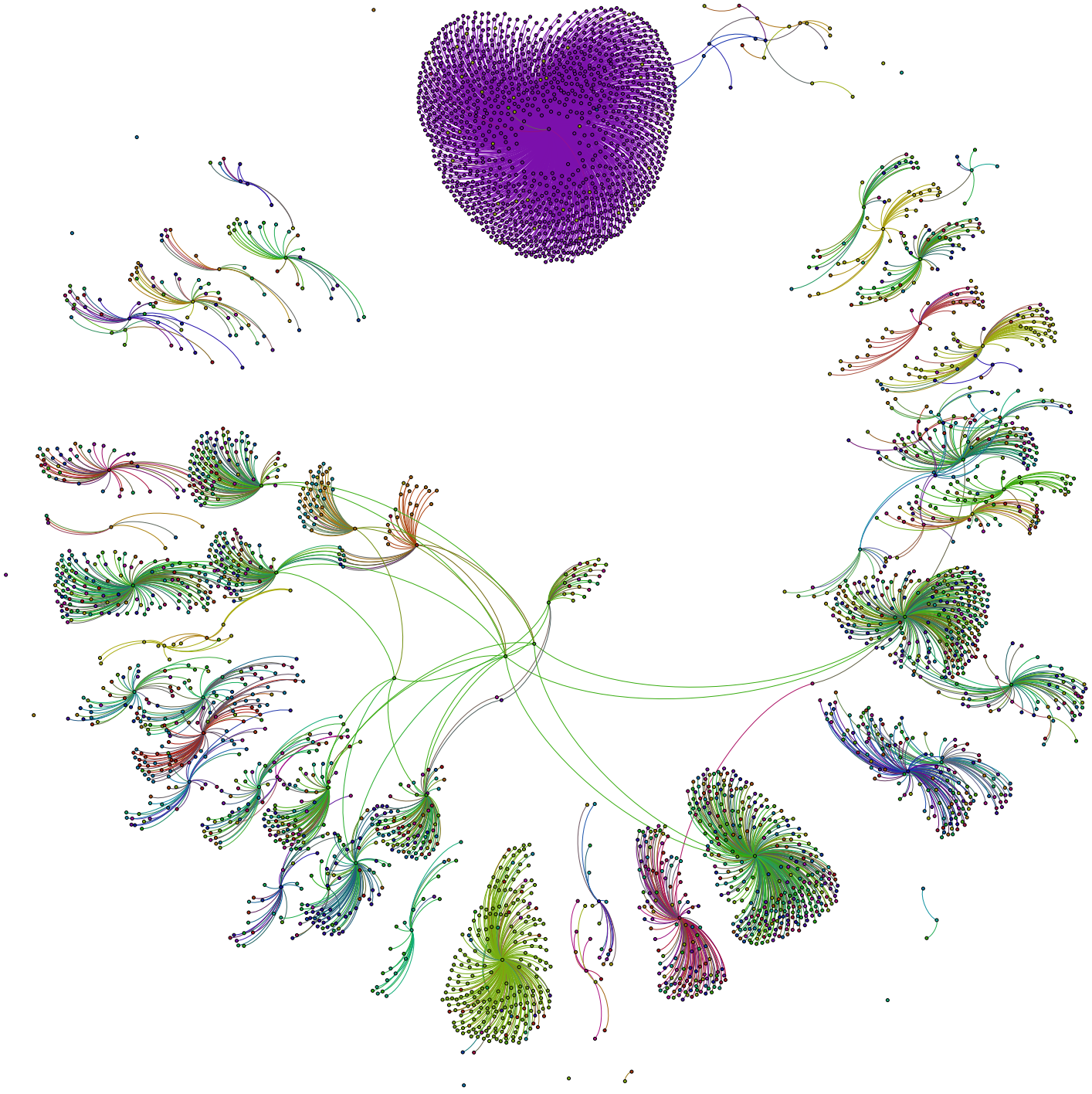
```
digraph A4_question3 {
  size = "6,6";
  node [color = lightblue2 ,style = filled];
 "https://www.facebook.com/MyChickenRun" -> "https://www.facebook.com/recover/
  initiate"
 "https://www.facebook.com/MyChickenRun" [label ="www.facebook.com"];
 "https://www.facebook.com/recover/initiate" [label ="www.facebook.com"];
 "http://www.youtube.com/watch?v=Eubi9YI2dKE" -> "https://plus.google.com/+youtube"
 "http://www.youtube.com/watch?v=Eubi9YI2dKE" [label ="www.youtube.com"];
 "https://plus.google.com/+youtube" [label ="plus.google.com"];
 "http://www.youtube.com/watch?v=Eubi9YI2dKE" -> "https://www.google.com/intl/en/
  policies/privacy/"
 "http://www.youtube.com/watch?v=Eubi9YI2dKE" [label ="www.youtube.com"];
 "https://www.google.com/intl/en/policies/privacy/" [label ="www.google.com"];
 "http://www.youtube.com/watch?v=Eubi9YI2dKE" -> "https://accounts.google.com/
  ServiceLogin?continue=https%3A%2F%2Fwww.youtube.com%2Fsignin%3Fhl%3Den%26app%3
  Ddesktop%26next%3D%252Fwatch%253Fv%253DEubi9YI2dKE%26feature%3Dplaylist%26
  action_handle_signin%3Dtrue&hl=en&service=youtube&passive=true&uilel=3"
 "http://www.youtube.com/watch?v=Eubi9YI2dKE" [label ="www.youtube.com"]; "https://
  accounts.google.com/ServiceLogin?continue=https%3A%2F%2Fwww.youtube.com%2Fsignin%3
  Fhl%3Den%26app%3Ddesktop%26next%3D%252Fwatch%253Fv%253DEubi9YI2dKE%26feature%3
  Dplaylist%26action_handle_signin%3Dtrue&hl=en&service=youtube&passive=true&uilel
  =3" [label ="accounts.google.com"];
}
```

3.3 Visualization with labels

The below graph depicts that there are not much connected components very few of the clusters are connected and few of them does not have outgoing links.

The second figure shows you how the nodes and edges are connected and it clearly depicts whether the clusters are connected or not connected when compared to the visualization with labels. A clear insight can be obtained from connected.pdf by enlarging.





3.3.1 Hits

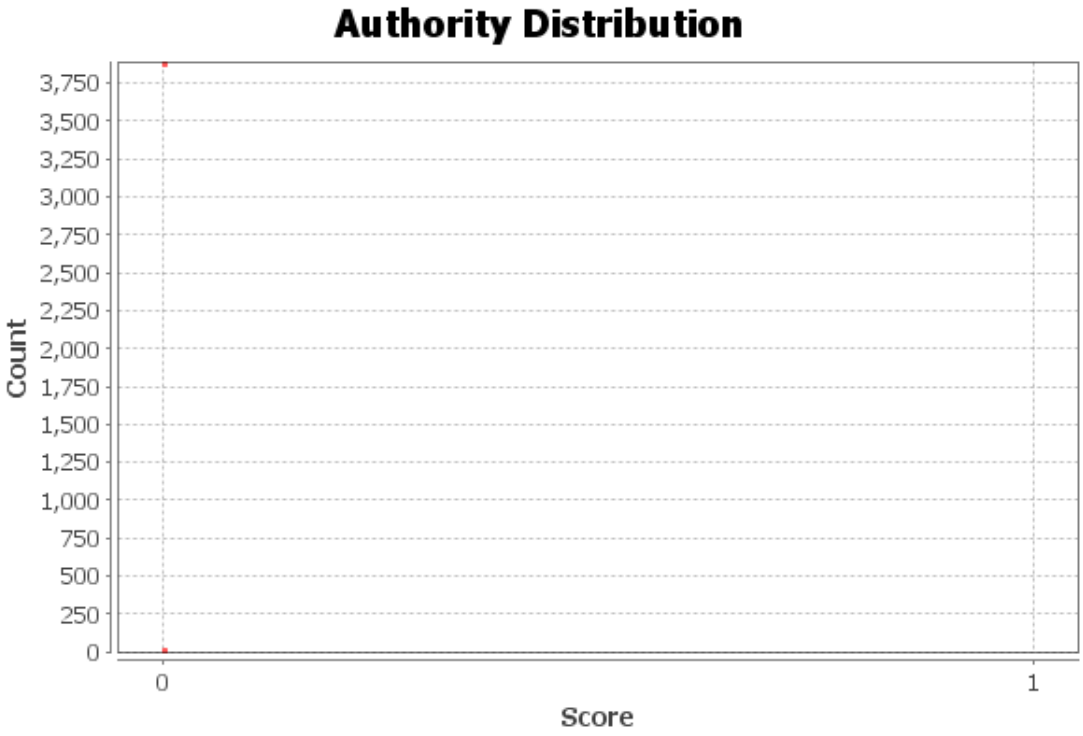


Figure 1: Authorities

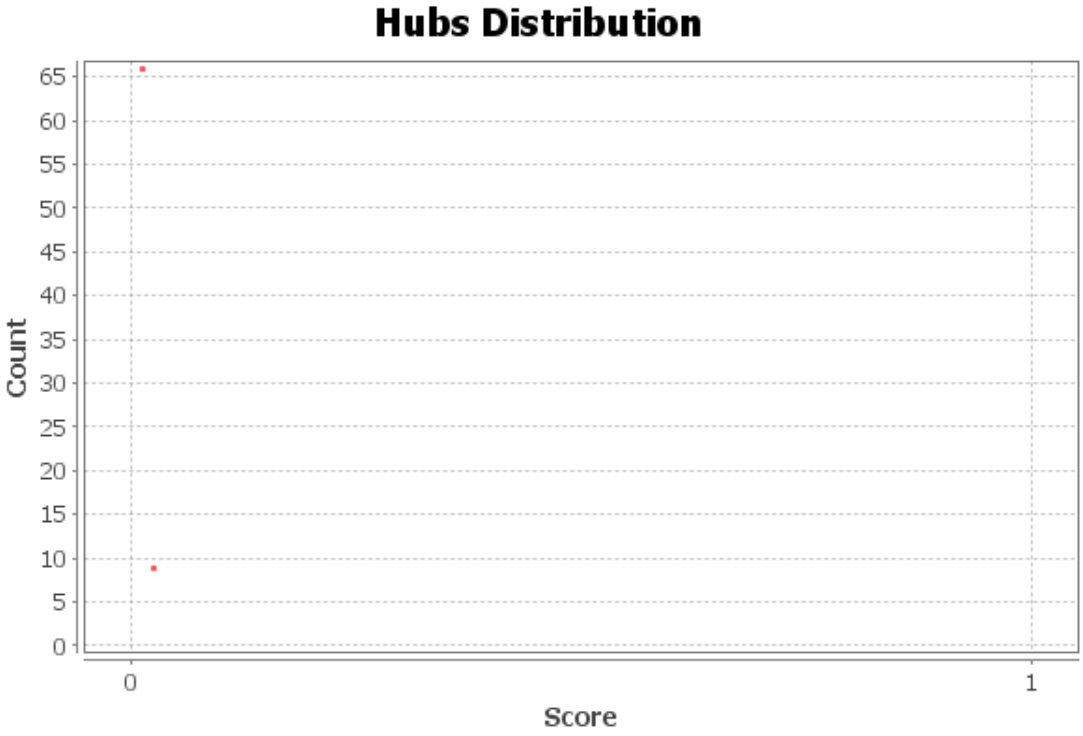


Figure 2: Hubs

3.3.2 PageRank

Epsilon = 0.001
Probability = 0.85

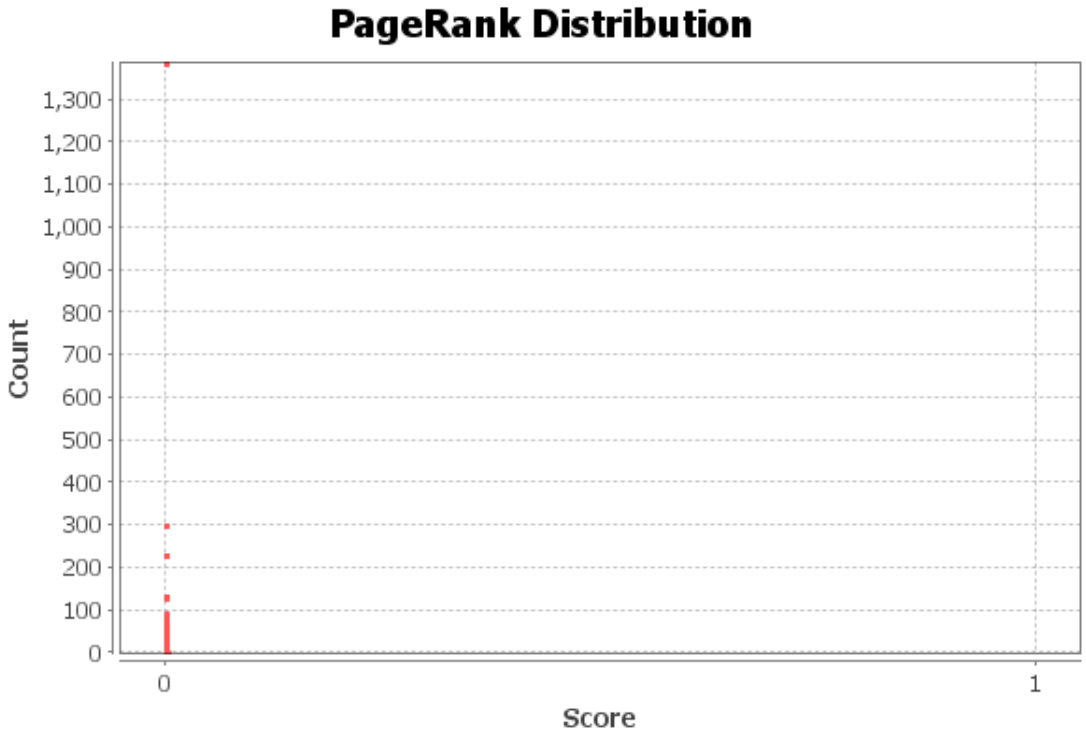


Figure 3: pagerank of links

3.3.3 Average

Average Degree: 0.997

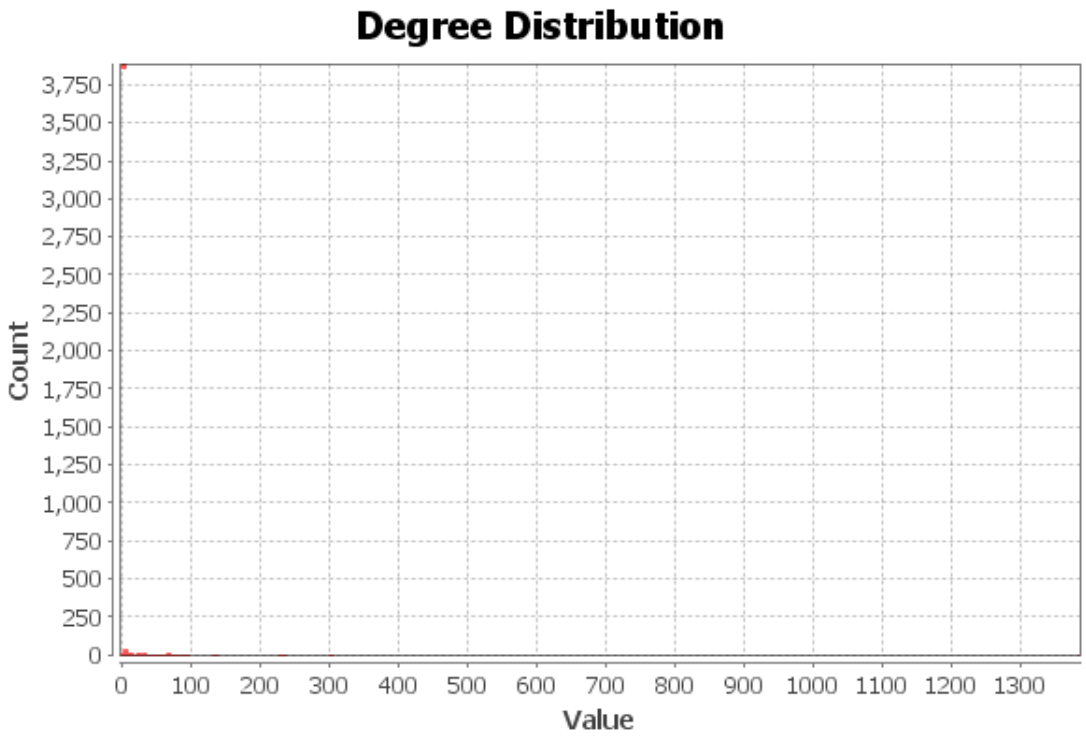


Figure 4: Average degree-distribution

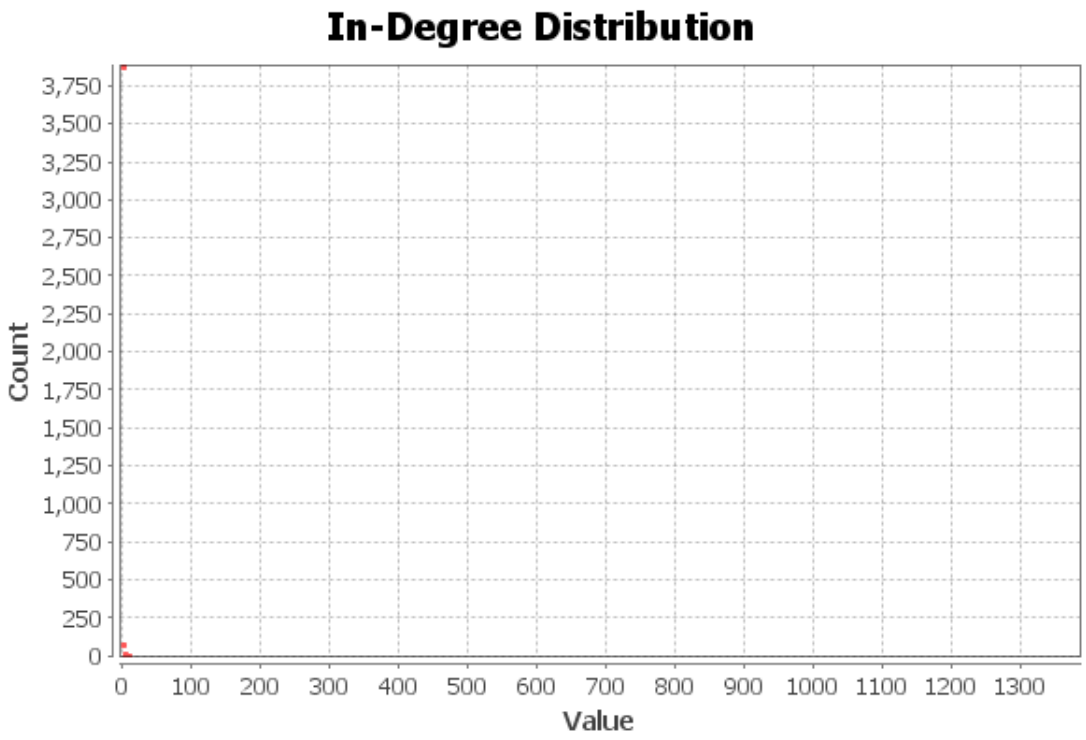


Figure 5: Average indegree-distribution

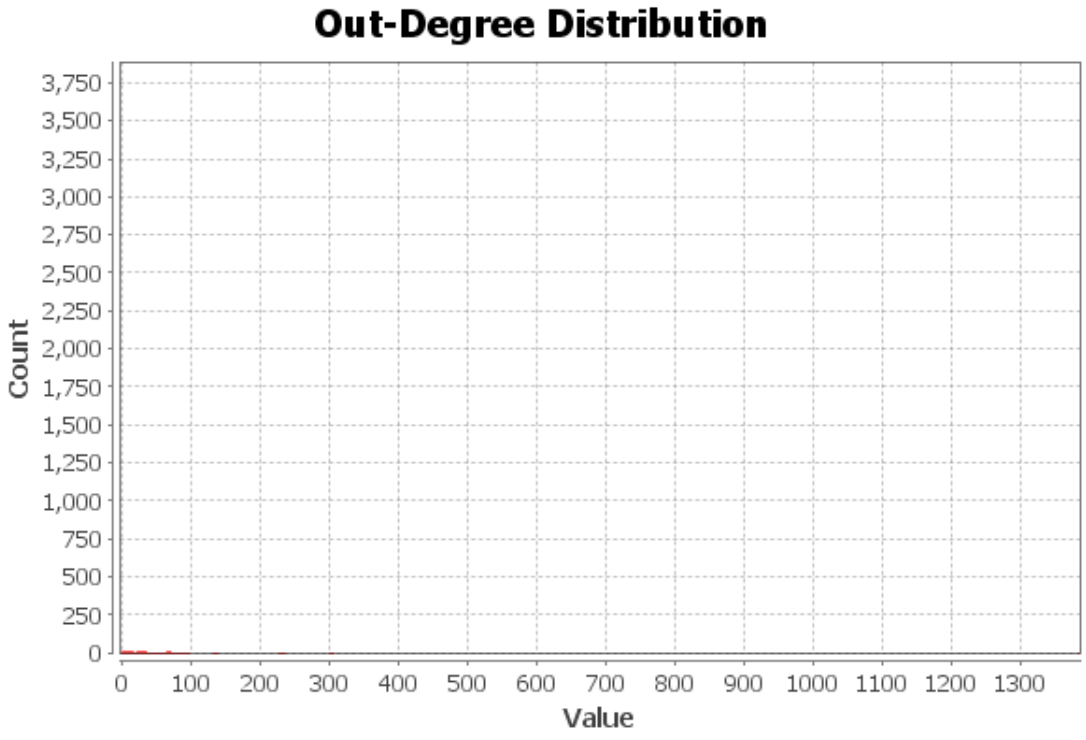


Figure 6: average outdegree-distribution

3.3.4 Network Diameter

Diameter: 1
Radius: 0
Average Path length: 1.0
Number of shortest paths: 3969

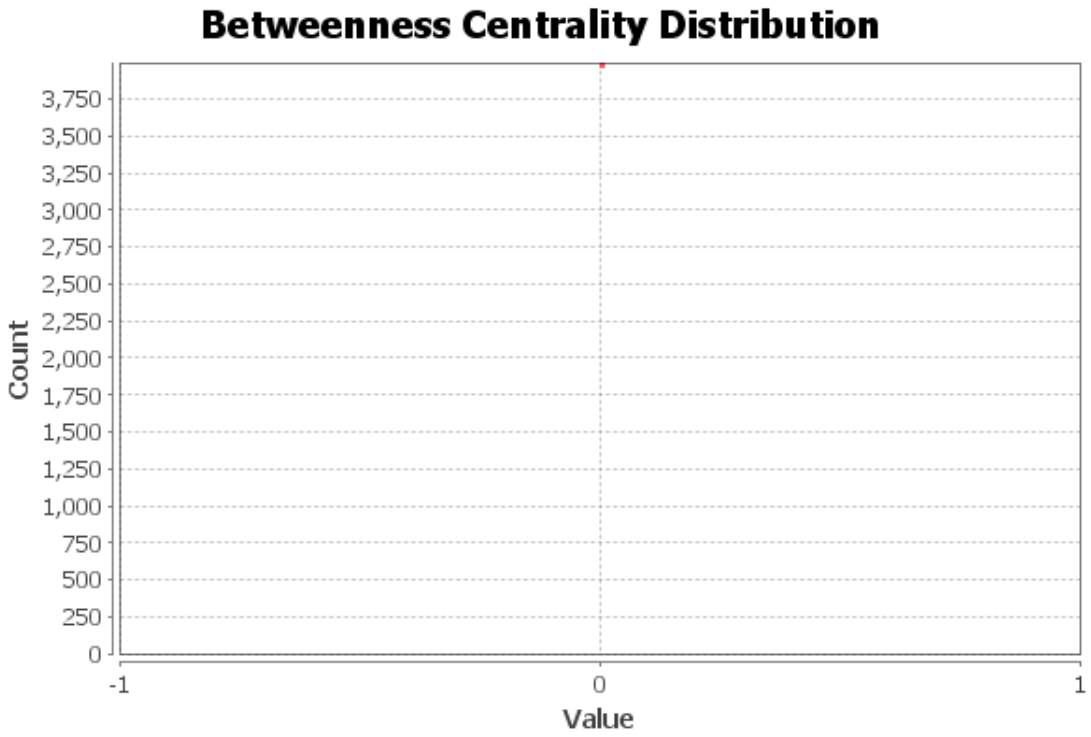


Figure 7: Betweeness

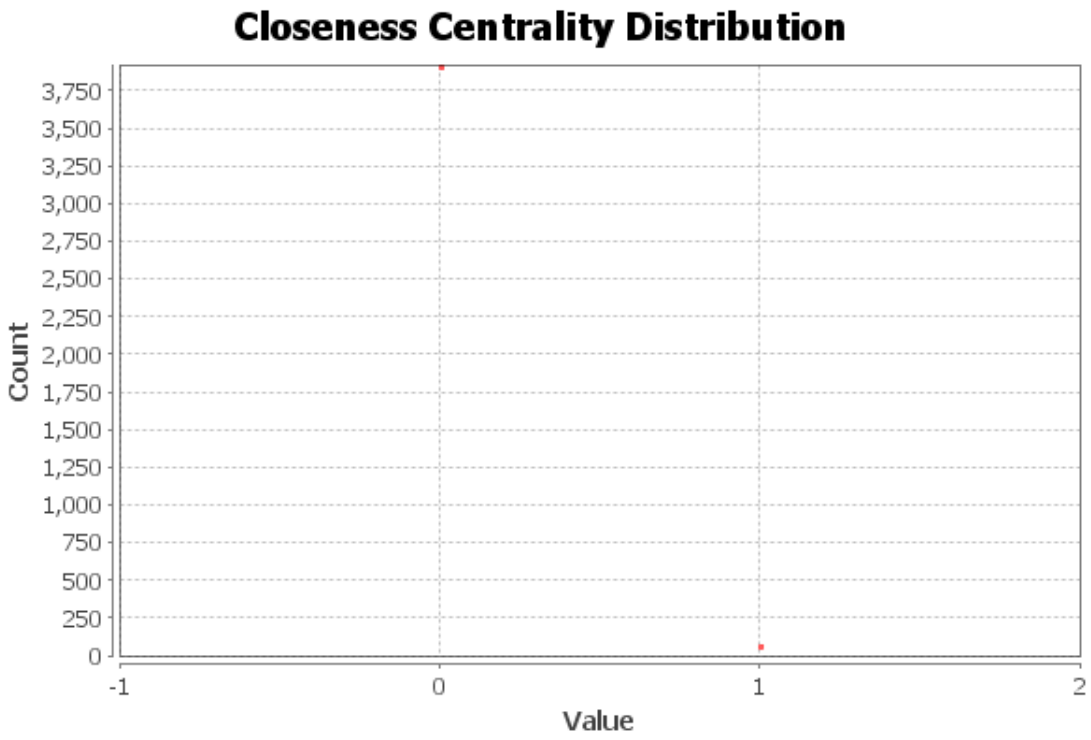


Figure 8: Closeness

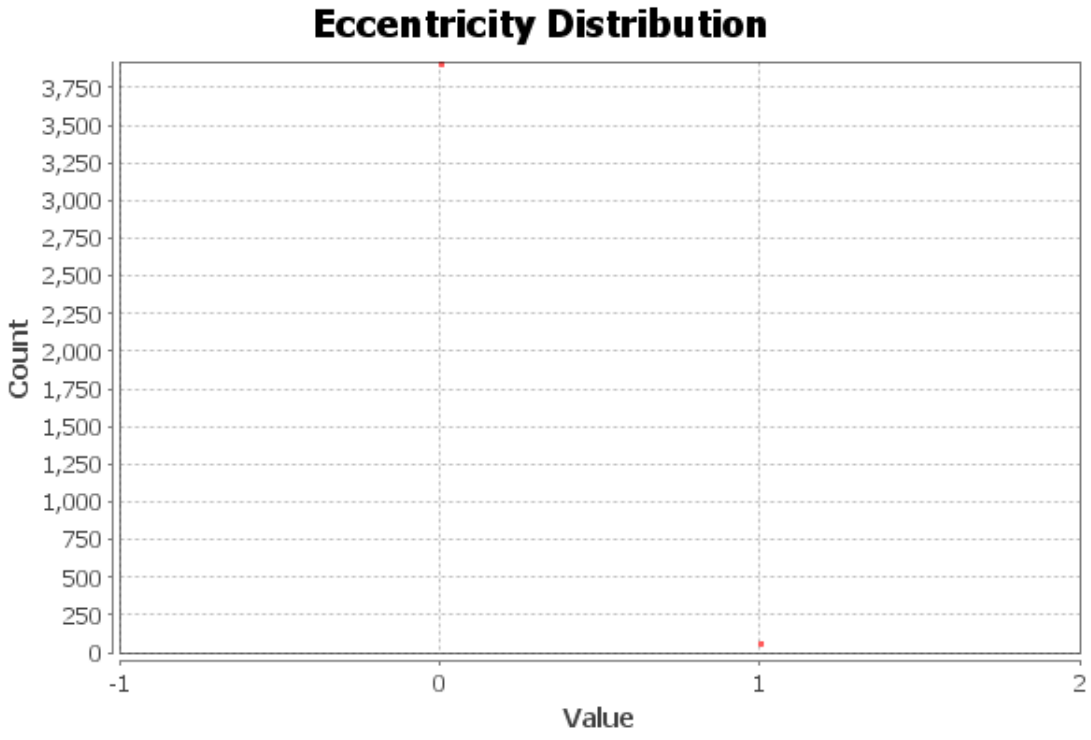


Figure 9: EccentricityDistribution

3.3.5 Connected Components

Number of Weakly Connected Components: 58
Number of Stronlgy Connected Components: 3990

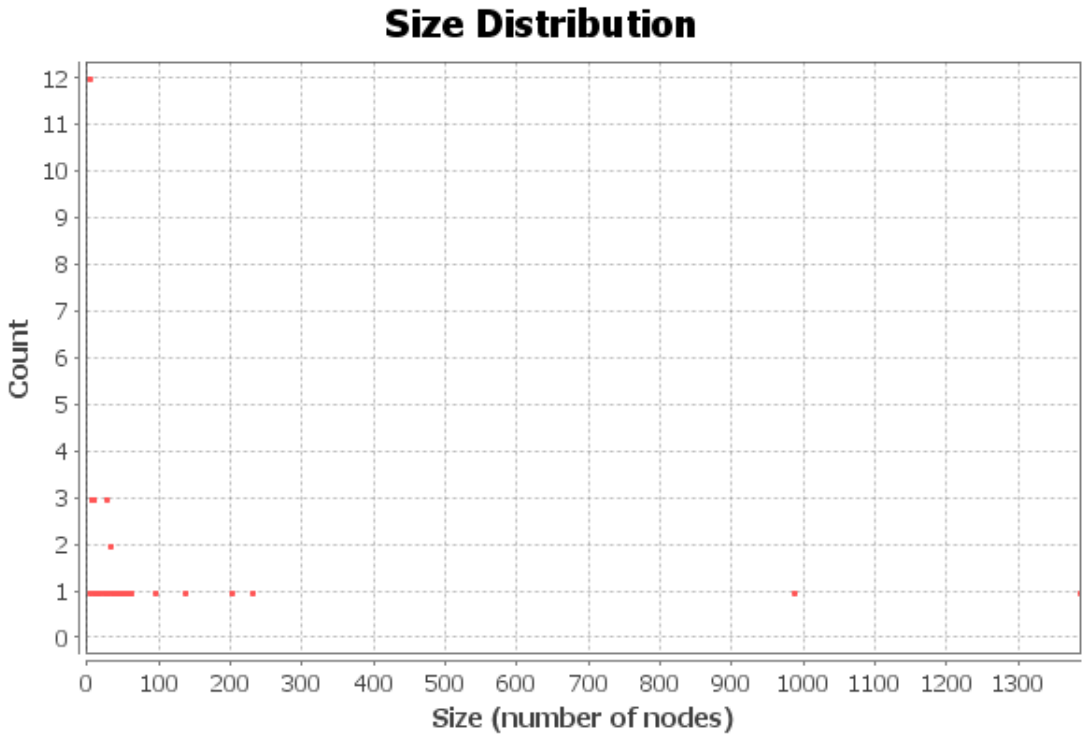


Figure 10: Graph Connected Components

References

[1] python. http://docs.python-requests.org/en/latest/user/quickstart/.

[2] python. http://askubuntu.com/questions/352198/reading-all-files-from-a-directory.

[3] python. http://stackoverflow.com/questions/5815747/beautifulsoup-getting-href.

[4] Urlunicode. https://docs.python.org/2/howto/unicode.html.

[5] Urlunicode. http://stackoverflow.com/questions/19212306/difference-between-ascii-and-unicode.

[]