CS896 Introduction to Web Science
Fall 2013
Report for Assignment 4

Corren G. McCoy

October 10, 2013

Contents

List of Figures

1 Question 1

1.1 Problem

From your list of 1000 links, choose 100 and extract all of the links from those 100 pages to other pages. For each URI, create a text file of all of the outbound links from that page to other URIs.

1.2 Response

We used Python to traverse our list of 1000 URIs and Beautiful Soup to retrieve the HTML source code for the URI. We then searched the source for the URI and title for any outbound links. If anchor tags with the href attribute were found, the referring site and list of outbound links were written to a text file as shown in Appendix B. For each referring URI, internal page links (i.e., navigation) were ignored along with duplicate references to the same external URI. We also excluded anchor tags that referenced JavaScript. The Python source code for *extractHREF* is shown in Appendix A. The 100 sites of interest were chosen based on a manual review of the original list of 1000 links. To compile our list of 100, we grouped the URIs into the following categories:

- blogs from a common domain (e.g., www.wordpress.com, www.blogspot.com);
- political sites (e.g., www.usa.gov, www.breitbart.com);
- print or television media (e.g., www.latimes.com, www.nytimes.com);
- sites with a large web presence (e.g., www.yahoo.com, www.gatesfoundation.org); and
- other miscellaneous entities (e.g., www.myitworks.com).

2 Question 2

2.1 Problem

Using these 100 files, create a single GraphViz 'dot' file of the resulting graph.

2.2 Response

The *extractHREF* code was also used to create the 'dot' file. Each URI, outbound link, and title attribute were combined to generate the node pairs and label required for the graph. After processing the 100 files, approximately 7200 node pairs were extracted. A portion of the GraphViz file output is shown below.

```
digraph twitter {
size="6,6";
node [color=lightblue2, style=filled];
"http://www.usa.gov/"->"http://business.usa.gov"
[label="Business"];
"http://www.usa.gov/"->"http://publications.usa.gov/"
[label="Consumer Publications"];
"http://www.usa.gov/"->"http://travel.state.gov/passport/passport_1738.html"
[label="Apply for a Passport"];
```

3 Question 3

3.1 Problem

Load the 'dot' file and use Gephi to:

- visualize the list
- calculate HITS and PageRank
- avg degree
- network diameter
- connected components

Put the resulting graphs in your report.

3.2 Response

As suggested in the assignment description, we reviewed our list of 1000 URIs and attempted to choose 100 which might display a high-level of connectivity. We then created a project in Gephi to visualize the network mapping. After importing the 'dot' file, we performed the following steps using the insight obtained from an online Gephi tutorial[1].

- First, we eliminated any noise in the model by applying a filter to reduce the network to its largest, connected component. This step eliminated about 100 node pairs from the graph. These nodes could not be reached from any other nodes.
- Second, we experimented with the node size by varying the size based on the degree.
- Third, for the initial layout, we started with the Fruchterman Reingold algorithm, which uses attraction-repulsion to separate the nodes. Applying this algorithm allowed us to see the initial formation of any communities.
- Fourth, we used the ForceAtlas 2 algorithm to further disperse the groups and add space around the larger nodes.
- Fifth, since our 'dot' file does not actually contain any categories as part of the data, we used node color to distinguish nodes based on the degree.
- Finally, to further detect any communities, we used the Gephi statistics to calculate modularity classes (i.e., subnetworks). Node colors were then applied based on the density of the class. Because our network contains thousands of nodes, the option to display edge labels was turned off to maintain the readability of the graph.

As shown in Figure 1, we observed one very dominant community which contains a cluster of over 21 percent of the network's nodes. This community represents the outgoing links from the Yahoo! web portal `http://www.yahoo.com`. No other community is comparable in size with most of the smaller subnetworks consisting of 4 percent or less. To assist with identification, Figure 2 shows a portion of the legend for the top 10 communities. We used the Gephi statistical functions to produce the remaining graphs. However, because we started with only 100 URIs, the additional graphs are sparse and do not convey a signicant amount of information about our network.

- visualize the list, Figure 1;

- calculate HITS and PageRank, Figures 3, 4 and 5;
- avg degree, Figures 6, 7 and 8;
- network diameter, Figures 9, 10, and 11; and
- connected components, Figure 12.

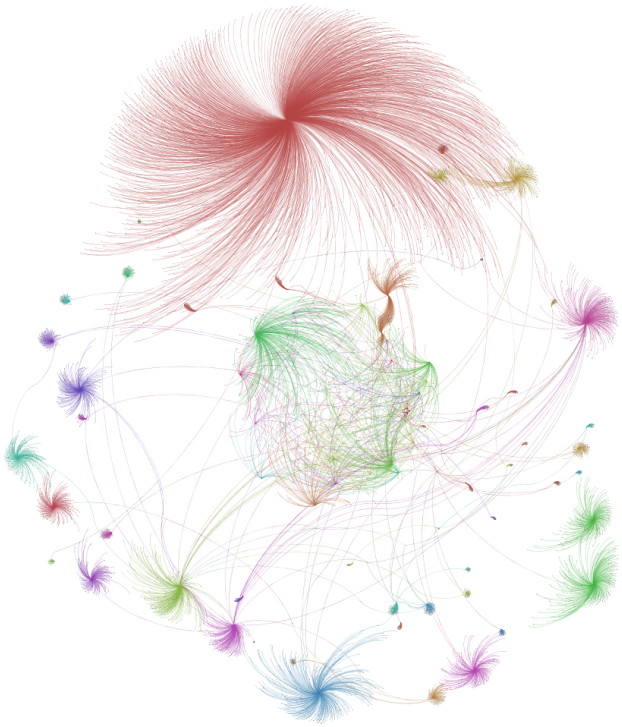

Figure 1: Twitter Network Grouped by Modularity Class

[1] http://pegasusdata.com/2013/01/10/facebook-friends-network-mapping-a-gephi-tutorial/

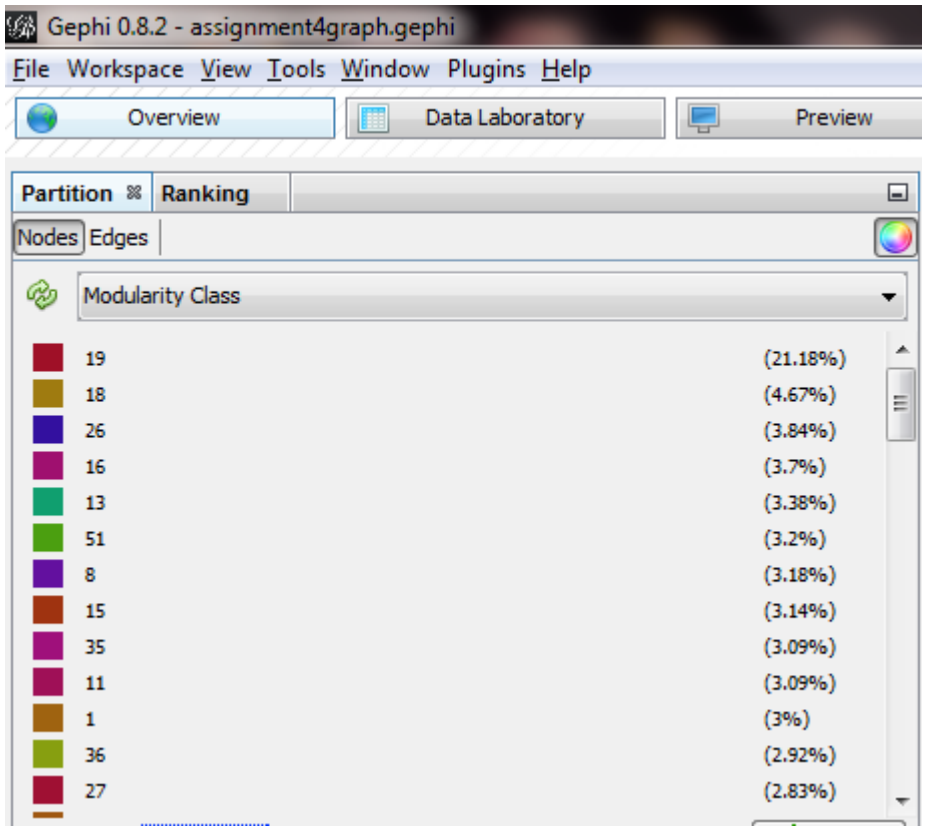

Figure 2: Modularity Legend

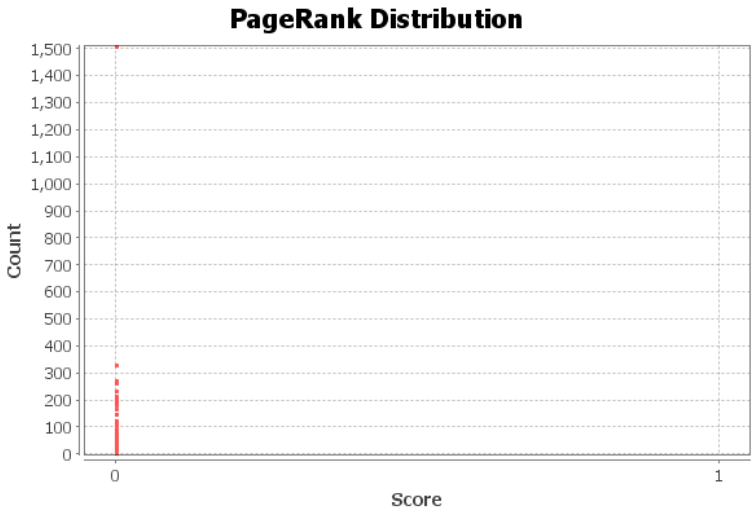

Figure 3: PageRank

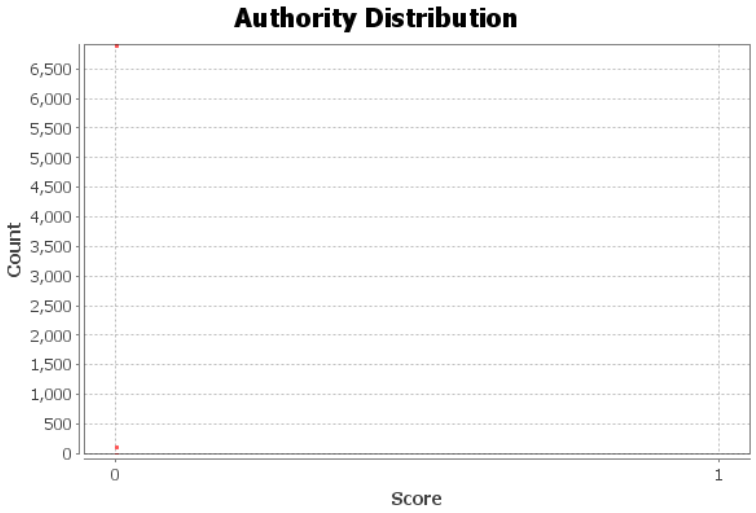

Figure 4: HITS - Authority Distribution

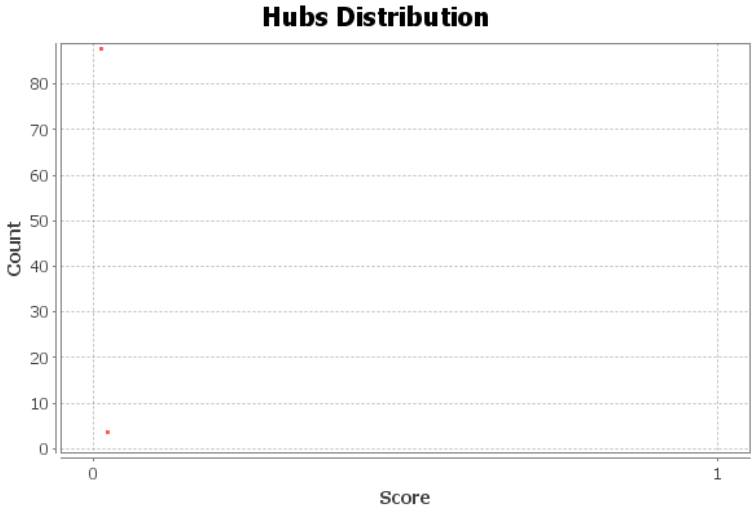

Figure 5: HITS - Hubs Distribution

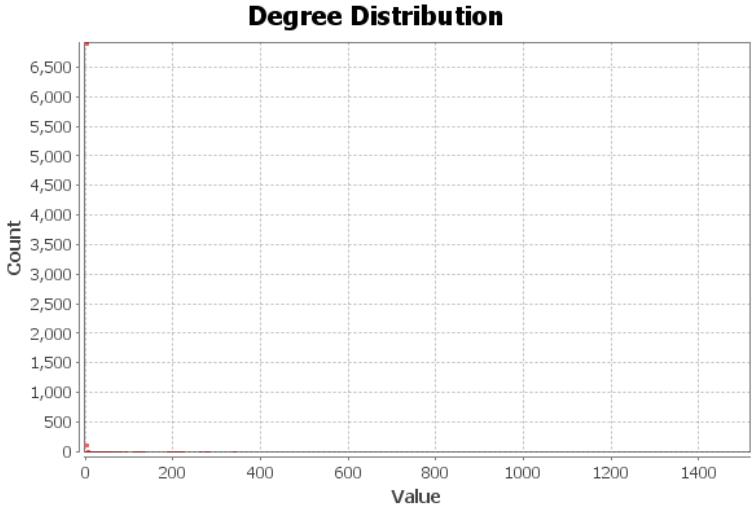

Figure 6: Avg Degree - Degree Distribution

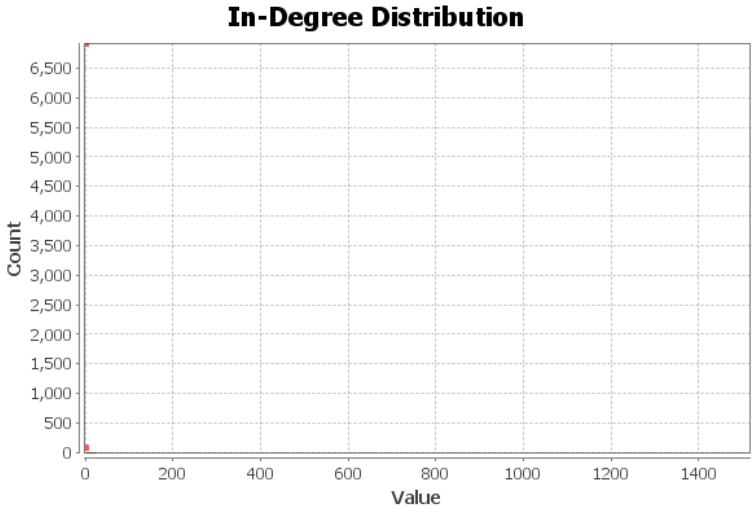

Figure 7: Avg Degree - Indegree Distribution

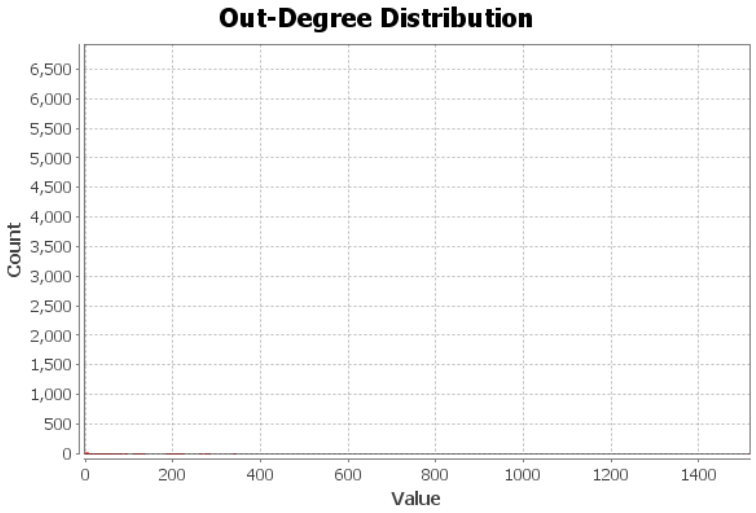

Figure 8: Avg Degree - Out Degree Distribution

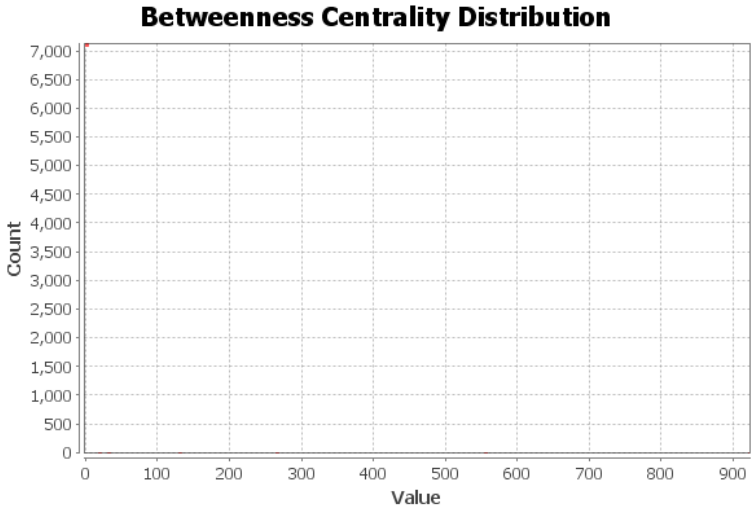

Figure 9: Network Diameter - Betweenness

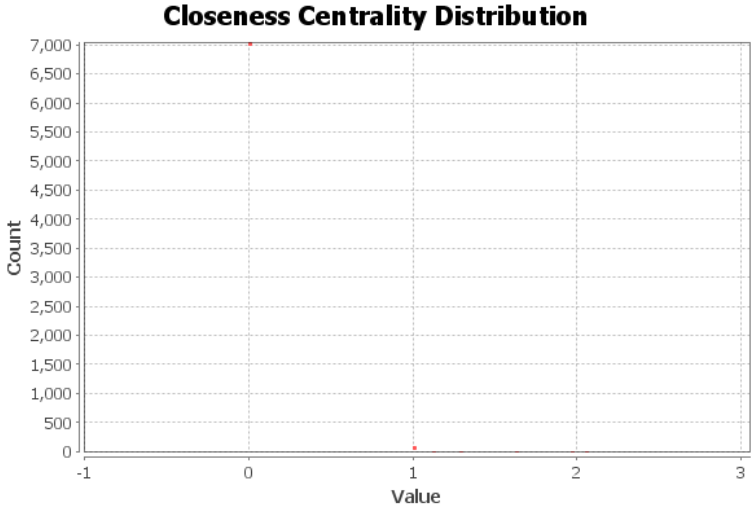

Figure 10: Network Diameter - Closeness

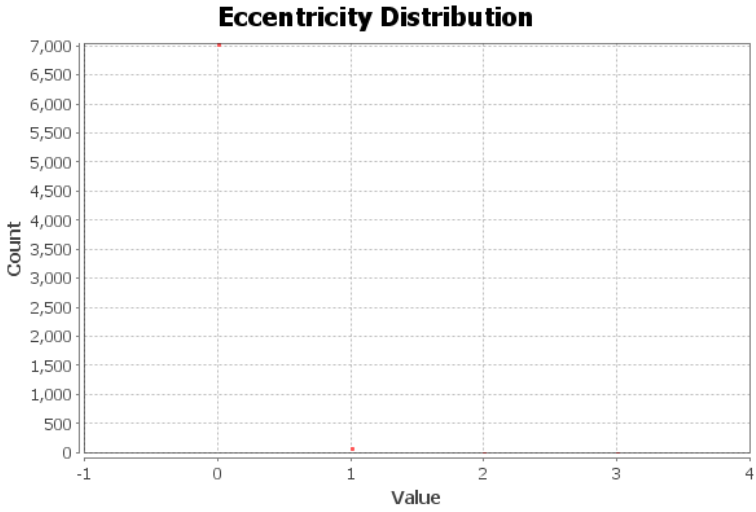

Figure 11: Network Diameter - Eccentricity

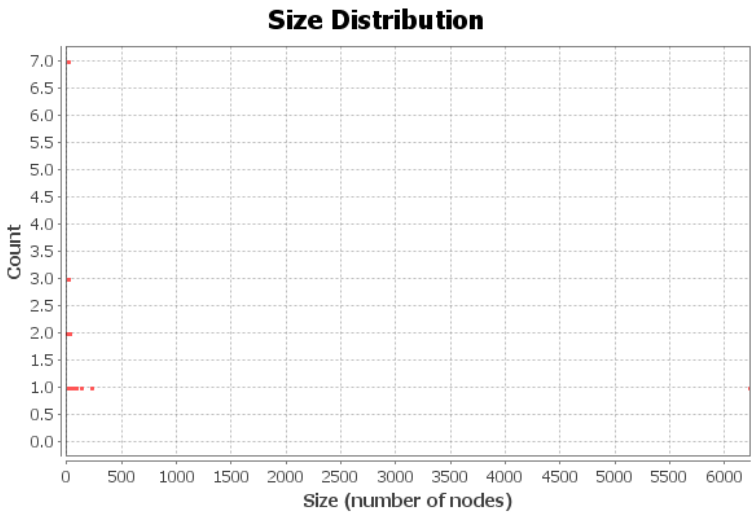

Figure 12: Connected Components

Appendix A

Python Source - extractHREF.py

```
#!/usr/bin/python
from BeautifulSoup import BeautifulSoup
import codecs
import os
import sys
import time
import urllib2
import urllib
import urlparse
import unicodedata

'''
Given a file of URIs, extract all user-navigable links from the web page.
Create a "dot" file in the GraphViz format using
the URI and the title associated with the "<a href=. . .>" tag.
Author:  Corren McCoy, 2013
'''

# Initialize global variables
baseFileName = 'C:/Python27/myFiles/Assignment 4/output/file'
dotFileName  = 'C:/Python27/myFiles/Assignment 4/gephi/twitter.dot'
fileLimit = 100
fileCounter = 0
# Initialize the dot file
graphviz = codecs.open(dotFileName, 'w', 'utf-8')


def extractHREF(url):
    global baseFileName
    # Prepare the output file for this URL
    currentFileName = baseFileName + str(fileCounter) + '.txt'
    currentFile = codecs.open(currentFileName,'w','utf-8')


    # Write the header for the URI file
    currentFile.write('site:\n')
```

```
    currentFile.write(url + '\n')
    currentFile.write('links:')

    # package the request
    try:
        request=urllib2.Request(url)
        request.add_header('User-agent','Mozilla 5.10')
        response=urllib2.urlopen(request)
        if response.code == 200:
            html=response.read()
            # decode byte stream to unicode
            html = html.decode("utf-8")
            # encode to ASCII byte stream, removing characters with codes >127
            html = html.encode("ascii", "ignore")
            soup = BeautifulSoup(html)
        response.close()

        # create an empty dictionary. We want to build a dictionary {uri: {link:[title]}}
        site={}
        links={}
        numLinks=0
        # Extract information from the anchor tag
        for tag in soup.findAll('a', href=True):
            tag['href'] = urlparse.urljoin(url, tag['href'])
            tld = tag['href']
            # Outbound links only. Must not have the same top-level domain
            # Ignore javascript in anchor tag (e.g. javascript:void(0)
            if tld.find(url) == -1 and tld.find("javascript") == -1:
                title=  str(tag.string).strip()
                # components for the dictionary
                links[tag['href']]= title
                site[url] = links
            # Verification: Keep track of the number of original links encountered
            numLinks = numLinks + 1
        # Iterate over the full URI dictionary.
        # The links will be unique, unduplicated
        for siteKey, linkValue in site.iteritems():
            for link, title in linkValue.iteritems():
                currentFile.write('\n')
                currentFile.write(link)
                # add the node to Graphviz file
                graphviz.write('\n')
                graphviz.write('"' +siteKey+'"' + '->' + '"' + link + '" '
+ ' [label="' + title + '"];')
            # Verification. Compare original to unduplicated number of key-value pairs.
            print(siteKey, "Unduplicated:", len(links), "Full URI Count", str(numLinks))
        # Done with this URI
        currentFile.close()
```

```
    except urllib2.HTTPError,e:
        print "Error", url, e
        return
    except urllib2.URLError, e:
        print "Error", url, e
        return
    except IOError, e:
        print "Error", url, e
        return
    except UnicodeDecodeError, e:
        print "Error", url, e
        return

#############################################################################
#  This is main procedure in this package
#############################################################################
def main():
    print "Press Control-C to exit"
    #continue until Control-C is entered from keyboard

    # The tweet file contains our 1000 URIs
    fileObject = open('C:/Python27/myFiles/Assignment 4/tweetFile1000.txt','r')

    # Extract links from 100 pages
    uriFile = open('C:/Python27/myFiles/Assignment 4/tweetFile1000.txt').readlines()
[1:fileLimit]
    # Write the header
    graphviz.write("digraph twitter { \n")
    graphviz.write('size="6,6"; \n')
    graphviz.write('node [color=lightblue2, style=filled];');

    for line in fileObject.readlines():
        try:
            global fileCounter
            fileCounter = fileCounter+1

            # Remove the final newline character
            url = line.rstrip('\n')
            # Processing is limited to 100 files
            if fileCounter > fileLimit:
                break
            print "Processing file", fileCounter
            extractHREF(url)
            # Raised when the user hits the interrupt key (normally Control-C or Delete).

        except KeyboardInterrupt:
            print ""
            print "processing terminated."
            sys.exit(0)
```

```
# close the Tweek file
fileObject.close()
# close the Graphviz dot file
graphviz.write("\n}")
graphviz.close()
print ">>>>File processing complete"
```

Appendix B

Sample URI file

```
site:
http://www.usa.gov/
links:
http://business.usa.gov
http://publications.usa.gov/
http://travel.state.gov/passport/passport_1738.html
http://www.youtube.com/USGovernment
https://public.govdelivery.com/accounts/USAGOV/subscriber/new
http://www.facebook.com/USAgov
http://www.fueleconomy.gov/feg/gasprices/states/index.shtml
http://apps.usa.gov/
http://answers.usa.gov/system/templates/selfservice/USAGov/#portal/1012
http://1.usa.gov/qHoZ0j
http://answers.usa.gov
http://answers.usa.gov/system/templates/selfservice/USAGov/#portal/1012/chat
http://go.usa.gov/DAMF
http://apps.usa.gov/hurricane-by-american-red-cross.shtml
http://www.kids.gov/
http://blog.usa.gov/
http://twitter.com/USAgov
http://blog.usa.gov
http://www.youtube.com/USAGOV
```