Web Science Intro

Sofia Huang

due 09/27/2022

1.

Draw the resulting directed graph (either sketch on paper or use another tool) showing how the nodes are connected to each other and include an image in your report. This does not need to fit into the bow-tie type diagram, but should look more similar to the graph on slide 24 from Module-01 Web-Science-Architecture.

```
A --> B
B --> C
C --> D
C --> A
C --> G
E --> F
G --> C
G --> H
I --> H
I --> K
L --> D
M --> A
M --> N
N --> D
O --> A
P --> G
```

For the graph, list the nodes (in alphabetical order) that are each of the following categories:

SCC - A, B, C, G
IN - M, O, P
OUT - D, H
Tendrils - I, K, L (these can reach OUT)
Tubes - N (connects IN node, M, to OUT node, D)
Disconnected - E, F

Figure 1 shows the resulting directed graph from the links given.

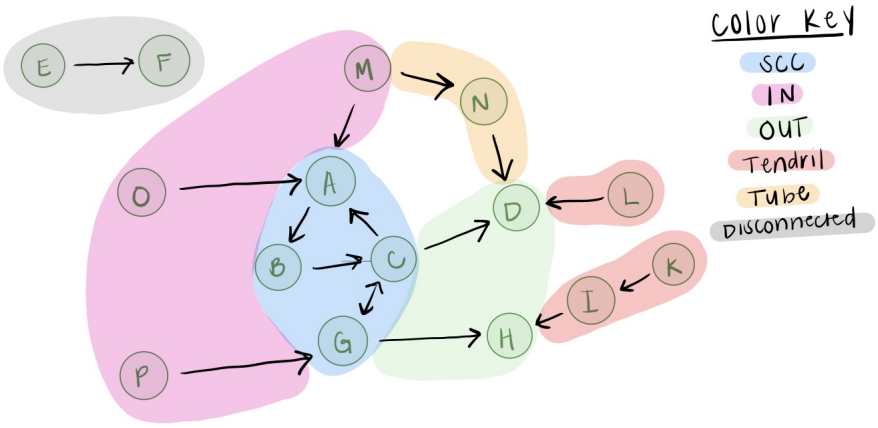

Figure 1: Bow-tie structure graph

To figure out the components of this graph I started with trying to figure out what the SCC was and quickly saw it was comprised of nodes A, B, C, and G. Then, I focused on finding which nodes went into the SCC and had no in-links, those were part of IN. Then, found which nodes came out of the SCC and had no out-links, those were part of OUT. From there, I noticed a node, N, that connected an IN node, M, to an OUT node, D. This is a tube, as it can only be reached from IN, only reach OUT and is not connected to the SCC. Next, I saw that there were some nodes that can only reach OUT nodes: I, L, and K, which means they are tendrils. Lastly, there were two nodes, E and F, that were not connected to any other nodes which made them disconnected.

2.

2.1

First, load this URI https://httpbin.org/user-agent directly in your browser and take a screenshot. The resulting webpage should show the "User-Agent" HTTP request header that your web browser sends to the web server.

Figure 2 shows the resulting web page from the URI: https://httpbin.org/user-agent


```
{
  "user-agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.36"
}
```


Figure 2: "User-Agent" HTTP request header from my web browser.

According to the Mozilla Developer's definition, a User-Agent is a "computer program acting

as a person". It identifies the browser, operating system, and version of the browser sending the HTTP request. httpbin.org is an HTTP Request and Response service that allows you to test various HTTP methods, inspect request and response data and much more. So, the specific URI given allows you to see the incoming HTTP request's User-Agent header from my computer.

2.2

In a single `curl` command, issue a `HEAD` HTTP request for the URI, https://t.co/EYgdZgrm2W. Show the HTTP response headers, follow any redirects, and change the User-Agent HTTP request field to "DATA 440" Show the command you issued and the result of your execution on the command line. (Either take a screenshot of your terminal or copy/paste into a code segment.)

```
(base) sofiahuang@Sofias-MacBook-Pro ~ % curl -A "DATA 440" https://
   httpbin.org/user-agent

{
  "user-agent": "DATA 440"
}
```

Listing 1: changing User-Agent

To change the user agent, I used the curl argument `-A`. Then, using HTTP Bin, the URI given in the previous part, I was able to see that the User-Agent had indeed been changed to what I had set it to, "DATA 440".

```
(base) sofiahuang@Sofias-MacBook-Pro ~ % curl --head -L https://t.co/
   EYgdZgrm2W

HTTP/2 301
date: Wed, 21 Sep 2022 00:23:29 GMT
vary: Origin
server: tsa_b
expires: Wed, 21 Sep 2022 00:28:29 GMT
location: https://www.nytimes.com/interactive/2022/05/13/us/covid-
   deaths-us-one-million.html
set-cookie: muc=2600ecf3-30a6-460d-b675-ec7937c4510c; Max-Age=63072000;
    Expires=Fri, 20 Sep 2024 00:23:29 GMT; Domain=t.co; Secure;
   SameSite=None
set-cookie: muc_ads=2600ecf3-30a6-460d-b675-ec7937c4510c; Max-Age
   =63072000; Expires=Fri, 20 Sep 2024 00:23:29 GMT; Path=/; Domain=t.
   co; Secure; SameSite=None
cache-control: private,max-age=300
strict-transport-security: max-age=0
x-response-time: 7
x-connection-hash: 308
   b4ffb5d90668ab3f0181710c5c06a9d3b1979cfaa4091d553435a8a40e132
```

```
15
16 HTTP/2 200
17 server: nginx
18 content-type: text/html; charset=utf-8
19 x-cloud-trace-context: abacf9e208917890daaf39ca9ef955f1
   /2148248459137043822
20 x-b3-traceid: 18ea0a5f3ee84b2dad6725ad1a9de4bd
21 x-nyt-data-last-modified: Wed, 21 Sep 2022 00:23:29 GMT
22 last-modified: Wed, 21 Sep 2022 00:23:29 GMT
23 x-scoop-last-modified: 2022-06-02T20:37:29.713Z
24 x-pagetype: vi-interactive-minimal
25 x-xss-protection: 1; mode=block
26 x-content-type-options: nosniff
27 cache-control: s-maxage=5,no-cache
28 x-nyt-route: vi-interactive
29 x-origin-time: 2022-09-21 00:23:29 UTC
30 accept-ranges: bytes
31 date: Wed, 21 Sep 2022 00:23:29 GMT
32 age: 0
33 x-served-by: cache-lga21975-LGA, cache-iad-kiad7000045-IAD
34 x-cache: MISS, MISS
35 x-cache-hits: 0, 0
36 x-timer: S1663719809.487903,VS0,VE376
37 vary: Accept-Encoding, Fastly-SSL
38 set-cookie: nyt-a=2T5hJtzFuIUuoJMSNjZhi0; Expires=Thu, 21 Sep 2023
   00:23:29 GMT; Path=/; Domain=.nytimes.com; SameSite=none; Secure
39 x-nyt-app-webview: 0
40 set-cookie: nyt-gdpr=0; Expires=Wed, 21 Sep 2022 06:23:29 GMT; Path=/;
   Domain=.nytimes.com
41 x-gdpr: 0
42 set-cookie: nyt-purr=cfhhcfhhhckfh; Expires=Thu, 21 Sep 2023 00:23:29
   GMT; Path=/; Domain=.nytimes.com; SameSite=Lax; Secure
43 x-frame-options: DENY
44 onion-location: https://www.
   nytimesn7cgmftshazwhfgzm37qxb44r64ytbb2dj3x62d2lljsciiyd.onion/
   interactive/2022/05/13/us/covid-deaths-us-one-million.html
45 x-api-version: F-F-VI
46 content-security-policy: upgrade-insecure-requests; default-src data: '
   unsafe-inline' 'unsafe-eval' https:; script-src data: 'unsafe-inline
   ' 'unsafe-eval' https: blob:; style-src data: 'unsafe-inline' https
   :; img-src data: https: blob: android-webview-video-poster:; font-
   src data: https:; connect-src https: wss: blob:; media-src data:
   https: blob:; object-src https:; child-src https: data: blob:; form-
   action https:; report-uri https://csp.nytimes.com/report;
47 strict-transport-security: max-age=63072000; preload; includeSubdomains
48 set-cookie: nyt-b3-traceid=18ea0a5f3ee84b2dad6725ad1a9de4bd; Path=/;
   Domain=.nytimes.com; SameSite=none; Secure
```

```
x-nyt-edge-cache: MISS-MISS
content-length: 338478
```

Listing 2: HEAD HTTP request

When I used the command, `curl --head https://t.co/EYgdZgrm2W`, I was able to see that the page had a HTTP response code of 301, meaning the page had been moved permanently. To find the final URI, I used the `-L` option to follow the redirects and `--head` to show the headers from all requested pages.

3.

Write a Python program to find links to PDFs in a webpage. Your program must do the following:

- **take the URI of a webpage as a command-line argument**
- **extract all the links from the page**
- **for each link, request the URI and use the `Content-Type` HTTP response header to determine if the link references a PDF file**
- **for all links that reference a PDF file, print the original URI (found in the parent HTML page), the final URI (after any redirects), and the number of bytes in the PDF file. (Hint: `Content-Length` HTTP response header)**

```
(base) sofiahuang@Sofias-MacBook-Pro ~ % python3 /Users/sofiahuang/
   Desktop/hw1.py http://www.math.wm.edu/~rrkinc/teaching.html
URI: http://www.math.wm.edu/~rrkinc/cs658.pdf
Final URI: http://www.math.wm.edu/~rrkinc/cs658.pdf
Content-Length: 33,524 bytes

URI: http://www.math.wm.edu/~rrkinc/cs520.pdf
Final URI: http://www.math.wm.edu/~rrkinc/cs520.pdf
Content-Length: 75,028 bytes

URI: http://www.math.wm.edu/~rrkinc/cs628.pdf
Final URI: http://www.math.wm.edu/~rrkinc/cs628.pdf
Content-Length: 21,434 bytes

Number of pdf links: 3
```

Listing 3: get_pdf.py output snippet

```
import requests
from bs4 import BeautifulSoup
import re
import sys
from urllib.parse import urljoin
from urllib3.exceptions import NewConnectionError


def request_url(url):
    # try to request url, catches exception if invalid url is input
    try:
        response = requests.get(url)
    except NewConnectionError:
        print(f'Invalid URL: "{url}"')
    except Exception as e:
        print('There was an exception that occurred when requesting the URL')
        print(e)
        return

    # get html text using BeautifulSoup
    soup = BeautifulSoup(response.text,features="html.parser")
    # counter to track how many pdf links on web page
    pdf_link_count = 0
    # loop through all links on web page
    for links in soup.find_all('a'):
        # make sure the full url is obtained from the href attribute
        href = links.get('href')
        full_url = urljoin(url, href)
        try:
            link_response = requests.get(full_url)
        except Exception as e:
            print('There was an exception that occurred when requesting the URL')
            print(e)
            continue
        # use regex to see if link is a pdf using its content type header
        pattern = re.compile('([a-z]+\/)(pdf)')
        m = pattern.match(link_response.headers['Content-Type'])
        # if link is pdf, print necessary info
        if (m != None):
            # print URI
            print ("URI: {}".format(link_response.url))
            # print final URI, following redirects if necessary
            if (str(link_response.status_code)[0] != '3'):
                print ("Final URI: {}".format(link_response.url))
```

```
45              else:
46                  redirect_response = requests.get(link_response.url)
47                  while (str(link_response.status_code)[0] == '3'):
48                      redirect_response = requests.get(link_response.url)
49                  print ("Final URI: {}".format(redirect_response.url))
50              # print content length
51              print ("Content-Length: {:,} bytes \n".format(int(
    link_response.headers['Content-Length'])))
52              pdf_link_count += 1
53
54     print('Number of pdf links: {}'.format(pdf_link_count))
55
56 if __name__ == "__main__":
57     # https://alexandernwala.com/files/teaching/fall-2022/week-2/2018
    _wsdl_publications.html
58     # https://haipeng-chen.github.io/
59     # http://www.math.wm.edu/~rrkinc/teaching.html
60     input_url = str(sys.argv[1])
61     request_url(input_url)
```

Listing 4: get_pdf.py code

First, I used a try-catch block to request the URL given as input through the command line to make sure the URL is valid. Then, I used BeautifulSoup to get the html and looped through all of the links on the page. For each link, I requested the page, and used regex to see if the link was a pdf by checking the content type. If it was, I printed the URI, followed any redirects, using the status code, to get the final URI, and printed the content length. Finally, I printed the number of pdf links found on the original webpage. I tested the script on Prof. Nwala's publications page, as well as other web pages I found that had pdf links.

While completing this question, I ran into the problem of not getting the full URL from the `href` attribute of the link tag in the html. To fix this, I used `urljoin` from the `urllib.parse` library so I could request the page.

References

- Regex - How to match an optional character `https://stackoverflow.com/questions/4007302/regex-how-to-match-an-optional-character`
- HTTP Bin `https://httpbin.org/user-agent`
- User-Agent `https://developer.mozilla.org/en-US/docs/Glossary/User_agent`
- BS4 - Get full URL in source code `https://stackoverflow.com/questions/17972496/using-beautiful-soup-to-get-the-full-url-in-source-code`

- Python Requests - Response Status Codes `https://www.geeksforgeeks.org/response-status_code-python-requests/`
- Q3 - Find PDF links - URI#1 `https://alexandernwala.com/files/teaching/fall-2022/week-2/2018_wsdl_publications.html`
- Q3 - Find PDF links - URI#2 `https://haipeng-chen.github.io/`
- Q3 - Find PDF links - URI#3 `http://www.math.wm.edu/~rrkinc/teaching.html`