Assignment 1

CS 734: Introduction to Information Retrieval

Fall 2017

Grant Atkins

Finished on September 21, 2017

1

Question

```
1.2  Site Search is another common application of search engines.
In this case, search is restricted to the web pages at a given website.
Compare site search to web search, vertical search, and enterprise search.
```

Answer

To answer this question I decided to use the search query "cs834-f17" with modifications as necessary across each of these search types. For site search I used github.com as our course repository is hosted on github and will likely return results and as the name site search implies it only searches on the designated website. Google offers the option to site search with the command "site:website" which for my entry it is "cs834-f17 site:github.com" as shown in Figure 1 [3]. It only returned two results which is minimal and as expected because it is a unique term and the term "cs834-f17" isn't expected to be located in any other repository or description hosted on github.

For web search I continued to use google. Using the query term "cs834-f17" this time without specifying anything else returned 57 results as shown in Figure 2. Both web search and site search both return the same top result, which is the repository. Something that is interesting in this search is that it didn't show the second result from the site search. Instead the web search went to different domains to try and find "cs834-f17", for example *github.io*. This should be noted as the precision in site search is seems to be more effective in the context of terms.

Continuing with vertical search I also made use of google images. The goal of vertical search is to search on a specific content type. The results were somewhat expected as there was in fact a picture of the author of the cs834-f17 repository, Dr. Nelson as shown in Figure 3, however prior to that image it seems of had lot of patent schemes for protein formulas. In this context it probably wasn't the best to use a vertical search.

Finally when completing Enterprise search it should be noted that I don't have access to an intranet but I do have access to my own personal machine to conduct searches on my own file system. I used my operating system's implemented search in it file system and I also use the terminal command *grep* to show a customized search on a subset of directories. Using my operating systems *finder* program it searches across my entire computer of files with the words "cs834-f17" as shown in Figure 4. This type of seems more relatable to web search as it seem to generalize the results showing files

that mention this term, but not every single file in my filesystem. When I used the *grep* command searching any directories starting with "cs" it showed every occurrence where "cs834-f17" was used. This is more relatable to site search because it listed all locations of occurrence in these directories.

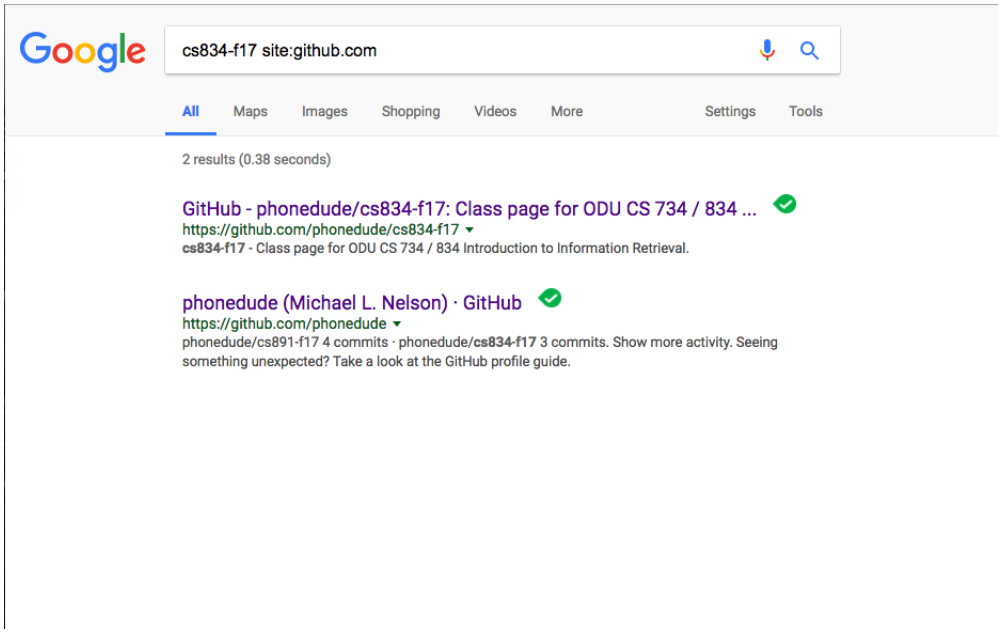

Figure 1: Site search example in google

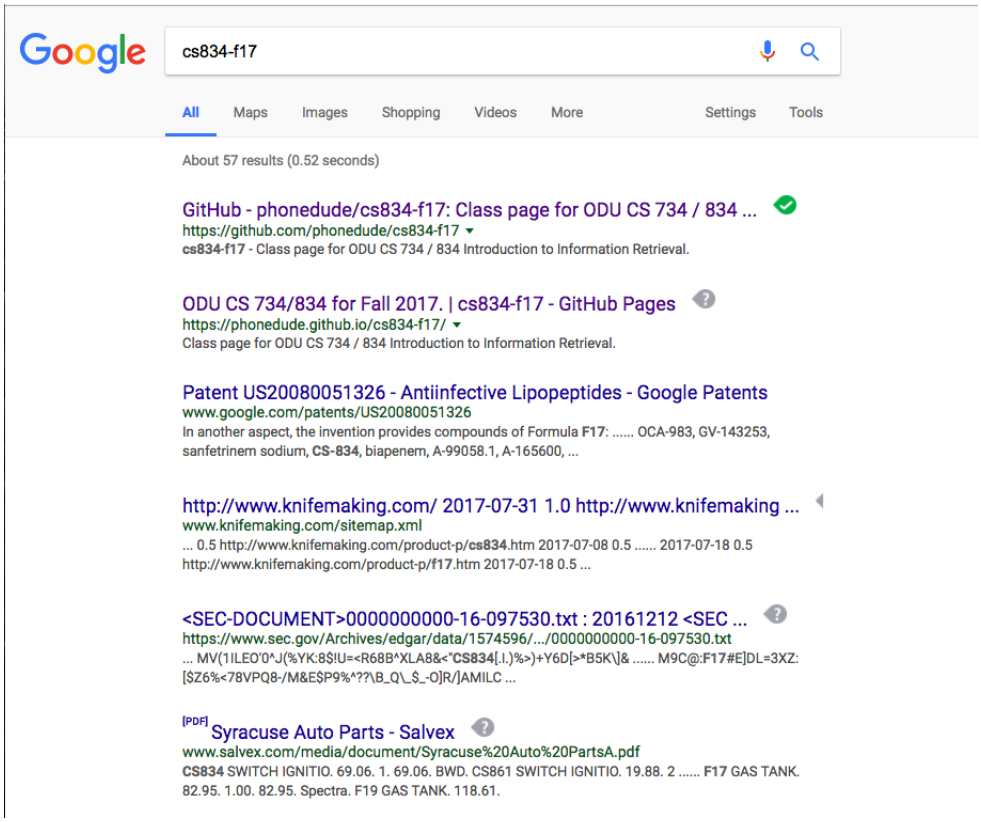

Figure 2: Web search example in google

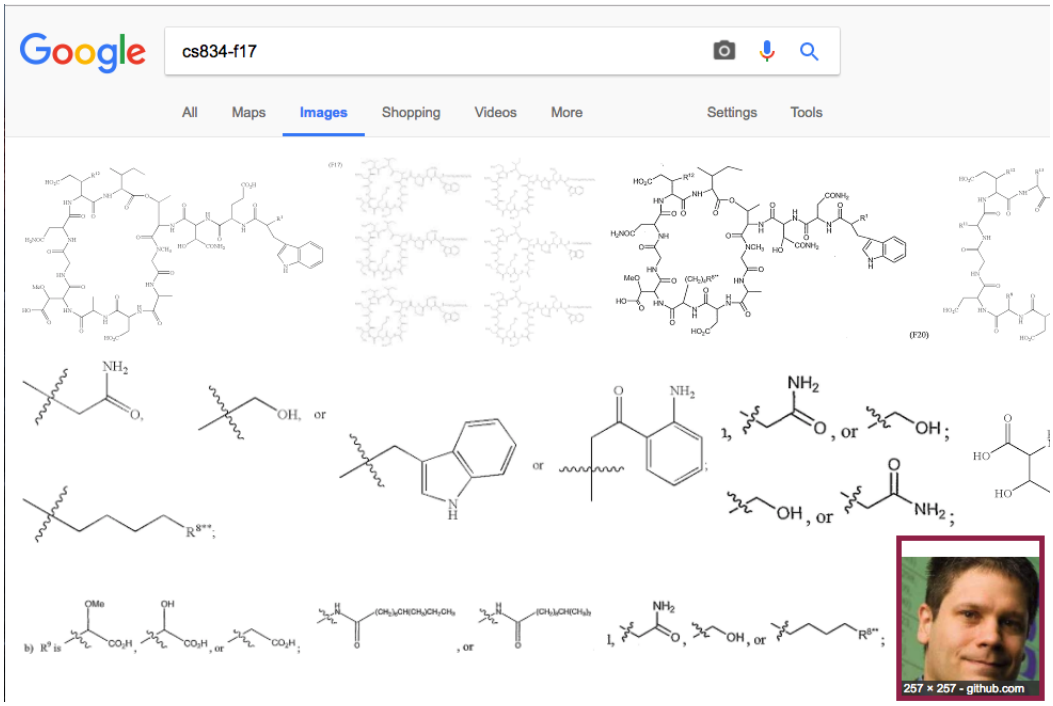

Figure 3: Vertical search example in google

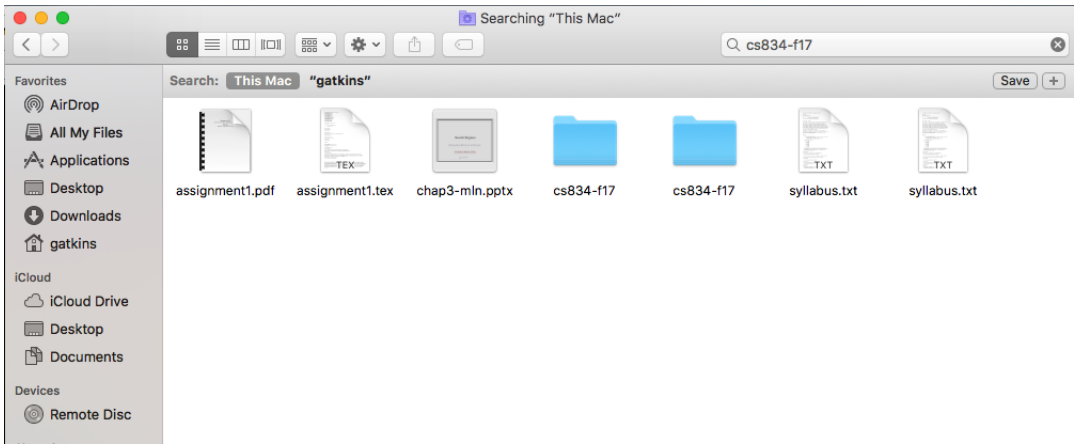

Figure 4: Enterprise Search using my own Desktop

```
Grants-MBP:~ gatkins$ grep -R "cs834-f17" ~/cs*
/Users/gatkins/cs734/cs834-f17/.git/config:     url = https://github.com/grantat/cs834-f17.git
/Users/gatkins/cs734/cs834-f17/.git/FETCH_HEAD:4bb91cc1dff40983b3d7d03f7dc9a4ab3c57c081
/Users/gatkins/cs734/cs834-f17/.git/logs/HEAD:0000000000000000000000000000000000000000 0033e52e
//github.com/grantat/cs834-f17.git
/Users/gatkins/cs734/cs834-f17/.git/logs/refs/heads/master:0000000000000000000000000000000000000000
m https://github.com/grantat/cs834-f17.git
/Users/gatkins/cs734/cs834-f17/.git/logs/refs/remotes/origin/HEAD:0000000000000000000000000000000000000000
one: from https://github.com/grantat/cs834-f17.git
/Users/gatkins/cs734/cs834-f17/index.md:[Syllabus](https://raw.githubusercontent.com/phonedude/
/Users/gatkins/cs734/cs834-f17/index.md:* [https://groups.google.com/group/cs834-f17](https://g
/Users/gatkins/cs734/cs834-f17/index.md:* Week 1 - August 31 - [Administrivia](https://raw.gith
/Users/gatkins/cs734/cs834-f17/index.md:), [LaTeX](https://www.overleaf.com/latex/learn/free-or
ercontent.com/phonedude/cs834-f17/master/slides/chap1.pptx), [2](https://raw.githubusercontent.
/Users/gatkins/cs734/cs834-f17/index.md:* Assignment 1: [due 2017-09-21](https://github.com/pho
/Users/gatkins/cs734/cs834-f17/README.md:[Schedule and Assignments](https://github.com/phonedu
/Users/gatkins/cs734/cs834-f17/syllabus.txt:   https://phonedude.github.io/cs834-f17/
/Users/gatkins/cs734/cs834-f17/syllabus.txt:   http://groups.google.com/group/cs834-f17/
/Users/gatkins/cs834-f17/.git/config:   url = https://github.com/grantat/cs834-f17.git
/Users/gatkins/cs834-f17/.git/logs/HEAD:0000000000000000000000000000000000000000 4bb91cc1dff40
.com/grantat/cs834-f17.git
/Users/gatkins/cs834-f17/.git/logs/refs/heads/master:0000000000000000000000000000000000000000 4
//github.com/grantat/cs834-f17.git
/Users/gatkins/cs834-f17/.git/logs/refs/remotes/origin/HEAD:0000000000000000000000000000000000000000
m https://github.com/grantat/cs834-f17.git
/Users/gatkins/cs834-f17/assignments/A1/docs/assignment1.tex:To answer this question I decided
/Users/gatkins/cs834-f17/assignments/A1/docs/assignment1.tex:Google offers the option to site s
 figure \ref{} \cite{googlesite}.
/Users/gatkins/cs834-f17/assignments/A1/docs/assignment1.tex:Atkins, Grant. ``CS734 Assignment
ee/master/assignments/A1}.
/Users/gatkins/cs834-f17/README.md:[Schedule and Assignments](https://github.com/phonedude/cs83
/Users/gatkins/cs834-f17/syllabus.txt:   https://phonedude.github.io/cs834-f17/
/Users/gatkins/cs834-f17/syllabus.txt:   http://groups.google.com/group/cs834-f17/
```

Figure 5: Specified Enterprise Search using my own Desktop in a terminal

2

Question

1.4 List five web services or sites that you use that appear to use search, not including web search engines. Describe the role of search for that service. Also describe the search is based on a database or grep style of matching, or if the search is using some type of ranking.

Answer

Five websites with web services I use frequently that use some sort of search mechanism include: `quora.com`, `github.com`, `twitter.com`, `youtube.com`, and quite recently `docs.docker.com`.

Quora is a popular question answer website based on topics. This website include a mixed search type of database querying and grep style. As shown below in Figure 6, if I start to search for "webscience" it will start to autocomplete it showing topics where that entry contains grep but not an exact match of the term. It also uses databases for storing these topics and a full text search of a topic is allowed which return all entries of that topic.

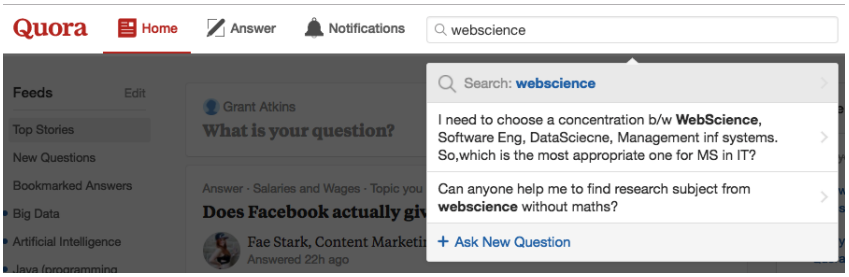

Figure 6: Quora search example

Github also includes a database and grep style of searching. When I search for "swift" on github it shows the following containing this term: repositories, code, commits, issues, wikis, and users as shown in Figure 7. This is obviously a grep for in-text occasions. However, the URI gives away a hint of database querying for a REST interface for example the following URI was sent for this term search: `https://github.com/search?utf8=%E2%9C%93&q=swift&type=`

Twitter is based on database searching. It uses services to take the input of a user then queries in their database. This request is then filtered down

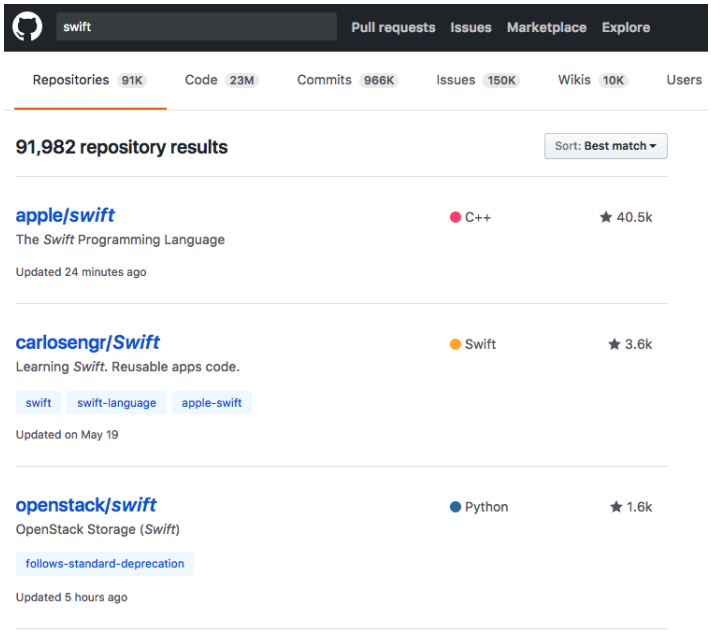

Figure 7: Github search example

to 10 items whether it be users, hashtags, or topics as shown in Figure 8. Each of these items is tagged which is noted in their database for tracking.

Youtube is my go to for music and any other video content. Youtube uses a database for its search with an example query shown in Figure 9. When you track a request from youtube you'll see an outgoing request be sent to their services. When the request is done you'll see the videos starting with your query term but when you follow that link it is based on a hash which is stored somewhere in database.

The docker docs, `docs.docker.com` is something I've been using quite frequently lately and I've noticed that this search is strictly a grep search. Document websites for software are more often than not, built statically meaning they produce these files separately for each location on the website as shown in Figure 10. Also a sure tell for this is usually done by looking at network activity and for this website it showed no outgoing request on a search.

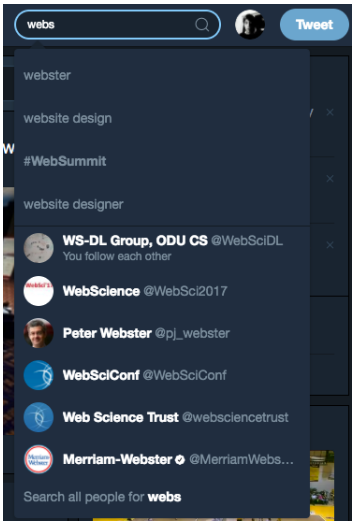

Figure 8: Twitter search example

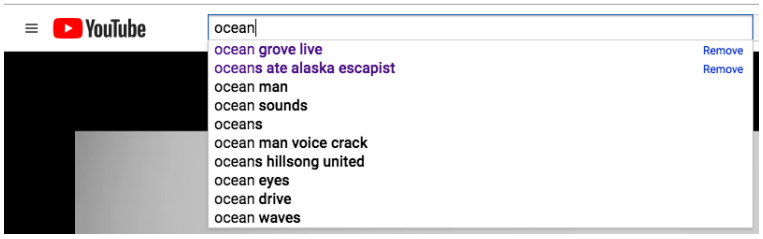

Figure 9: Youtube search example

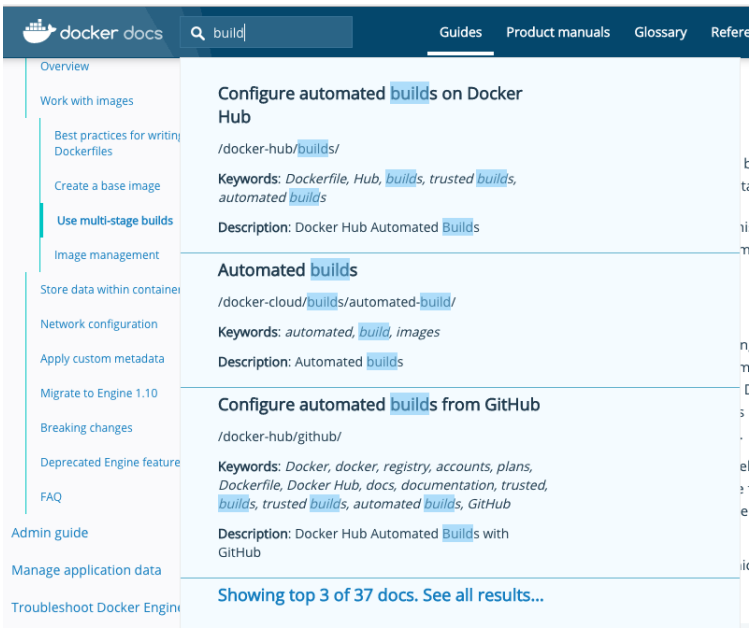

Figure 10: Docker docs search example

3

Question

3.7 Write a program that can create a valid sitemap based on the contents of a directory on your computer's hard disk. Assume the file are accessible from a website at the URL http://example.com. For instance, if there is a file in your directory called homework.pdf, this would be available at http://www.example.com/homework.pdf. Use the real modification date on the file as the last modified time in the sitemap, and to help estimate the change frequency.

Answer

To answer this question a wrote a program in Python 3.0+ as shown in Listing 1. This program selects a directory from which to build a sitemap. The directory I decided to use for this question was actually this assignment's directory, **A1**, for this course's repository [2]. The XML sitemap genered is shown in Listing 2. It should be noted that you won't see a **.DS_Store** or some of the other LaTeX files located in the **/A1/docs/** folder in this repository due to them being ignored in my repository's **.gitignore** file, these files are in fact on my computer locally.

```
1  import os
2  from datetime import datetime
3  import xml.etree.ElementTree as ET
4
5
6  def indent(elem, level=0, more_sibs=False):
7      '''
8      Pretty print xml etree
9      '''
10     i = "\n"
11     if level:
12         i += (level - 1) * '  '
13     num_kids = len(elem)
14     if num_kids:
15         if not elem.text or not elem.text.strip():
16             elem.text = i + "  "
17             if level:
18                 elem.text += '  '
19         count = 0
20         for kid in elem:
21             indent(kid, level + 1, count < num_kids - 1)
22             count += 1
```

```
23          if not elem.tail or not elem.tail.strip():
24              elem.tail = i
25              if more_sibs:
26                  elem.tail += '  '
27      else:
28          if level and (not elem.tail or not elem.tail.strip()):
29              elem.tail = i
30              if more_sibs:
31                  elem.tail += '  '
32
33
34  def getLastModified(filePath):
35      '''
36      Get last modified datetime of a file
37      '''
38      stat = os.stat(filePath)
39      dtime = datetime.fromtimestamp(
40          stat.st_mtime).strftime("%Y-%m-%dT%H:%M:%SZ")
41      return dtime
42
43
44  def buildSitemap(outfile, baseURL, rootdir):
45      '''
46      build xml tree for directory based sitemap
47      '''
48      urlset = ET.Element(
49          'urlset', xmlns="http://www.sitemaps.org/schemas/sitemap
                /0.9")
50
51      with open(outfile, 'w'):
52          for subdir, dirs, files in os.walk(rootdir):
53              for f in files:
54                  filepath = subdir + os.sep + f
55                  lastmod = getLastModified(filepath)
56                  url = ET.Element('url')
57                  # remove './' from path as start of a string
58                  if filepath.startswith(rootdir + os.sep):
59                      filepath = filepath[len(rootdir + os.sep):]
60                  elif filepath.startswith(rootdir):
61                      filepath = filepath[len(rootdir):]
62
63                  filepath = filepath.replace(' ', '+')
64                  loc = ET.SubElement(url, 'loc')
65                  loc.text = baseURL + filepath
66                  lastmods = ET.SubElement(url, 'lastmod')
67                  lastmods.text = lastmod
68                  urlset.append(url)
69
70          indent(urlset)
```

```
71          tree = ET.ElementTree(urlset)
72          tree.write(outfile, encoding='utf-8', xml_declaration=
                True)
73
74
75  if __name__ == "__main__":
76      baseURL = "http://example.com/"
77
78      if baseURL.endswith('/') is False:
79          baseURL += '/'
80
81      outfile = 'data/sitemap.xml'
82      buildSitemap(outfile, baseURL, '../..')
```

Listing 1: Python script create a sitemap from a directory's contents

```
1  <?xml version='1.0' encoding='utf-8'?>
2  <urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
3    <url>
4      <loc>http://example.com/.DS_Store</loc>
5      <lastmod>2017-09-20T10:54:45Z</lastmod>
6    </url>
7    <url>
8      <loc>http://example.com/A1/a1.txt</loc>
9      <lastmod>2017-09-04T10:40:57Z</lastmod>
10   </url>
11   <url>
12     <loc>http://example.com/A1/README.md</loc>
13     <lastmod>2017-09-20T10:54:24Z</lastmod>
14   </url>
15   <url>
16     <loc>http://example.com/A1/Test+file.doc</loc>
17     <lastmod>2017-09-20T18:17:45Z</lastmod>
18   </url>
19   <url>
20     <loc>http://example.com/A1/docs/assignment1.aux</loc>
21     <lastmod>2017-09-21T11:27:55Z</lastmod>
22   </url>
23   <url>
24     <loc>http://example.com/A1/docs/assignment1.log</loc>
25     <lastmod>2017-09-21T11:27:55Z</lastmod>
26   </url>
27   <url>
28     <loc>http://example.com/A1/docs/assignment1.out</loc>
29     <lastmod>2017-09-21T11:27:55Z</lastmod>
30   </url>
31   <url>
32     <loc>http://example.com/A1/docs/assignment1.pdf</loc>
33     <lastmod>2017-09-21T11:27:55Z</lastmod>
```

```
34    </url>
35    <url>
36      <loc>http://example.com/A1/docs/assignment1.synctex.gz</loc>
37      <lastmod>2017-09-21T11:27:55Z</lastmod>
38    </url>
39    <url>
40      <loc>http://example.com/A1/docs/assignment1.tex</loc>
41      <lastmod>2017-09-21T11:28:07Z</lastmod>
42    </url>
43    <url>
44      <loc>http://example.com/A1/docs/NOTES.md</loc>
45      <lastmod>2017-09-13T11:37:42Z</lastmod>
46    </url>
47    <url>
48      <loc>http://example.com/A1/src/simpleCrawler.py</loc>
49      <lastmod>2017-09-20T08:40:09Z</lastmod>
50    </url>
51    <url>
52      <loc>http://example.com/A1/src/sitemap.py</loc>
53      <lastmod>2017-09-21T11:55:11Z</lastmod>
54    </url>
55    <url>
56      <loc>http://example.com/A1/src/data/sitemap.xml</loc>
57      <lastmod>2017-09-21T11:55:14Z</lastmod>
58    </url>
59  </urlset>
```

Listing 2: Sitemap created from my assignment 1 (A1) directory

4

Question

`Suppose that, in an effort to crawl web pages faster, you set up two crawling machines with different starting seed URIs. Is this an effective strategy for distributed crawling? Why or why not.`

Answer

This is not an effective strategy for distributed crawling. An important idea to remember is that the web is a graph structure where many links will eventually connect to one another. With this in mind, as these web crawlers start crawling through their assigned URIs and the depth in which the URIs in that page start to branch to other websites they will eventually overlap URIs which have either already been crawled or already in line to be crawled. Recently for our paper presentations, I had to present a paper on IRLbot, a single server crawling bot which had answers to this sort of issue [4].

With crawlers being separated by threads, IRLbot incorporated URI lookup and storage algorithms to track the URIs which I agree would be the best option. Whenever a URI is discovered it should be looked up inside the storage in which they track URIs and then cache the URI so the current crawlers can know even more quickly that they don't have to do a long search for this URI knowing its already been tracked. If the system recognizes it as a new URI it will be able to add it to an active queue system to be crawled on. Again, since there are multiple crawlers the other crawlers can check the cache, the current URI queue, or a long lookup from storage if necessary. Overall this system would prove faster than time lost performing requests to these servers and then checking if a URI can be stored.

5

Question

3.9 Write a simple single-threaded web crawler. Starting from a single input URL (perhaps a professor's web page), the crawler should download a page and then wait at least five seconds before downloading the next page. Your program should find other pages to crawl by parsing link tags found in previously crawled documents.

Answer

This question was relatively simple as we were posed a question very similar to this question previously in our CS532 class except it had the goal of extracting all pdf files from a webpage and not chasing further links. Therefore to answer this problem I used my code previously created from CS532 for assignment 1 but with some slight modifications as shown in Listing 3 [1]. For instance, I added: a time delay, a set to track the links already crawled, a queue system, and a hop counter for each website to track crawl level. In this problem I tested 1 and 2 hops on the website `http://www.cs.odu.edu/~gatkins/cs725/`. With a max of 1 hop I found that I had queue size of 142 unique URIs remaining to crawl as shown in Figure 11. With a max of 2 hops I found that I had queue size of 2626 unique URIs remaining to crawl as shown in Figure 12.

```
1  #!/usr/bin/env python3
2
3  import sys
4  import requests
5  from urllib.parse import urljoin, urlparse
6  from bs4 import BeautifulSoup
7  from collections import deque
8  import time
9
10
11 def crawl(uri, q, links_seen):
12     """
13     Take html string as parameter and parse through links ('a'
           elements).
14     """
15     print("Getting links from:", uri)
16     try:
17         useragent = ('Mozilla/5.0 (X11; Linux x86_64)
               AppleWebKit/537.36 '
```

```
18                          '(KHTML, like Gecko) Chrome/44.0.2403.157
                                Safari/537.36')
19
20          r = requests.get(uri, headers={'User-Agent': useragent},
                 verify=False)
21          content_type = r.headers.get('content-type').lower()
22
23          if 'text/html' in content_type and r.ok:
24              s = BeautifulSoup(r.text, 'html.parser')
25              all_a = s.find_all('a', href=True)
26
27              for link in map(lambda a: a['href'], all_a):
28                  if link not in links_seen:
29
30                      if isAbsolute(link) is False:
31                          fulllink = urljoin(r.url, link)
32                          if fulllink not in links_seen:
33                              q.append(fulllink)
34                              links_seen.add(fulllink)
35                      else:
36                          print("Link found:", link)
37                          q.append(link)
38                          links_seen.add(link)
39
40      except:
41          pass
42
43
44  def isAbsolute(url):
45      """
46      Taken from stackoverflow post
47      """
48      try:
49          return bool(urlparse(url).netloc)
50      except:
51          return False
52
53
54  if __name__ == '__main__':
55      try:
56          baseURL = sys.argv[1]
57      except:
58          print("Usage: python3 crawler.py {URL-R}")
59      q = deque()
60      q.append(baseURL)
61      hop_count = 1
62      links_seen = set()
63
64      uri = q.popleft()
```

```
65      links_seen.add(uri)
66
67      crawl(uri, q, links_seen)
68
69      qsize = len(q)
70
71      while hop_count > 0:
72          time.sleep(5)
73          uri = q.popleft()
74          qsize -= 1
75
76          crawl(uri, q, links_seen)
77
78          if qsize == 0:
79              qsize = len(q)
80              hop_count -= 1
81              print("Hop completed. With new links queue size is
                    :", qsize)
82
83      while True:
84          try:
85              uri = q.popleft()
86              print("Remaining URI", uri)
87          except:
88              break
```

Listing 3: Python script to for a single-threaded web crawler

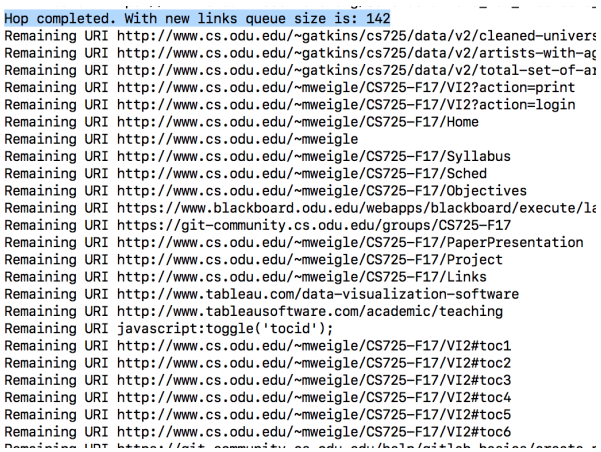

Figure 11: Terminal output of my crawler with 1 hop limit

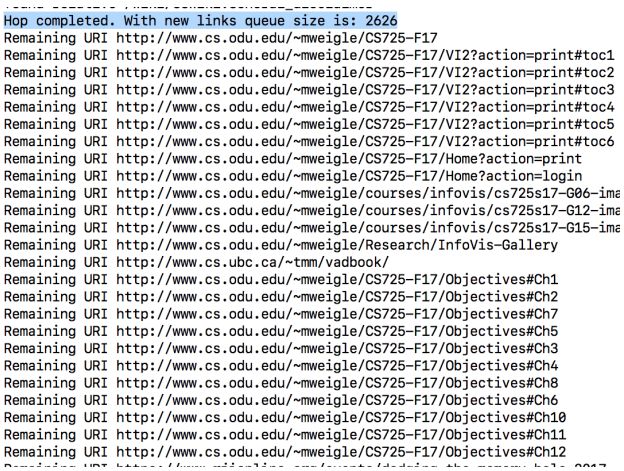

Figure 12: Terminal output of my crawler with 2 hops limit

References

[1] Atkins, Grant. "CS532 Assignment 1 Repository" Github. N.p., 23 March 2017. Web. 23 March 2017.https://github.com/grantat/cs532-s17/tree/master/assignments/A1/src.

[2] Atkins, Grant. "CS734 Assignment 1 Repository" Github. N.p., 21 September 2017. Web. 21 September 2017.https://github.com/grantat/cs834-f17/tree/master/assignments/A1.

[3] Boswell, Weny. "Advanced Google Search Shortcuts" Lifewire. N.p., 2 July 2017. Web. https://www.lifewire.com/advanced-google-search-3482174

[4] Hsin-Tsang Lee, Derek Leonard, Xiaoming Wang, and Dmitri Loguinov. 2009. "IRLbot: Scaling to 6 billion pages and beyond." ACM Trans. Web 3, 3, Article 8 (July 2009), 34 pages. http://dx.doi.org.proxy.lib.odu.edu/10.1145/1541822.1541823.