

Assignment 2

CS834, Information Retrieval, Fall 2017
Old Dominion University, Computer Science Dept

Hussam Hallak

CS Master's Student
Prof: Dr. Nelson

Question 1:

Exercise 4.3:

Try to estimate the number of web pages indexed by two different search engines using the technique described in this chapter. Compare the size estimates from a range of queries and discuss the consistency (or lack of it) of these estimates.

Answer:

I have chosen Google and Bing search engines. The technique described in this chapter uses the formula:

$$N = \frac{f_a \times f_b}{f_{ab}}$$

Where:

a and b are the independent terms

N is the estimated total number of indexed pages

f_a is the number of pages containing the term a

f_b is the number of pages containing the term b

f_{ab} is the number of pages containing both a and b

The two terms in the query need to be as independent as possible, so I have chosen the following queries:

1. Diesel Memory
2. Hairy Phone
3. Computer Manifold
4. Electronic Lumber

1. Diesel Memory: I ran the first query “Diesel Memory” and found the following:

A. Google:

Google found about 207,000,000 results for the term “Diesel”.

Figure 1: Query: Diesel, Search Engine: Google

The screenshot shows a Google search results page for the query "Diesel". The search bar at the top contains the word "Diesel". Below the search bar, the Google logo is visible, along with navigation links for "All", "Maps", "News", "Images", "Shopping", "More", "Settings", and "Tools". The search results are displayed below the navigation links, showing "About 207,000,000 results (0.99 seconds)".

The first search result is "diesel.com - Official Diesel® Site - New Arrivals", which includes a link to "shop.diesel.com/". Below this, there are four sub-results: "Diesel™ Men's Denim", "Diesel™ Women's Denim", "Diesel SKB Sneakers", and "Diesel Official JoggJeans".

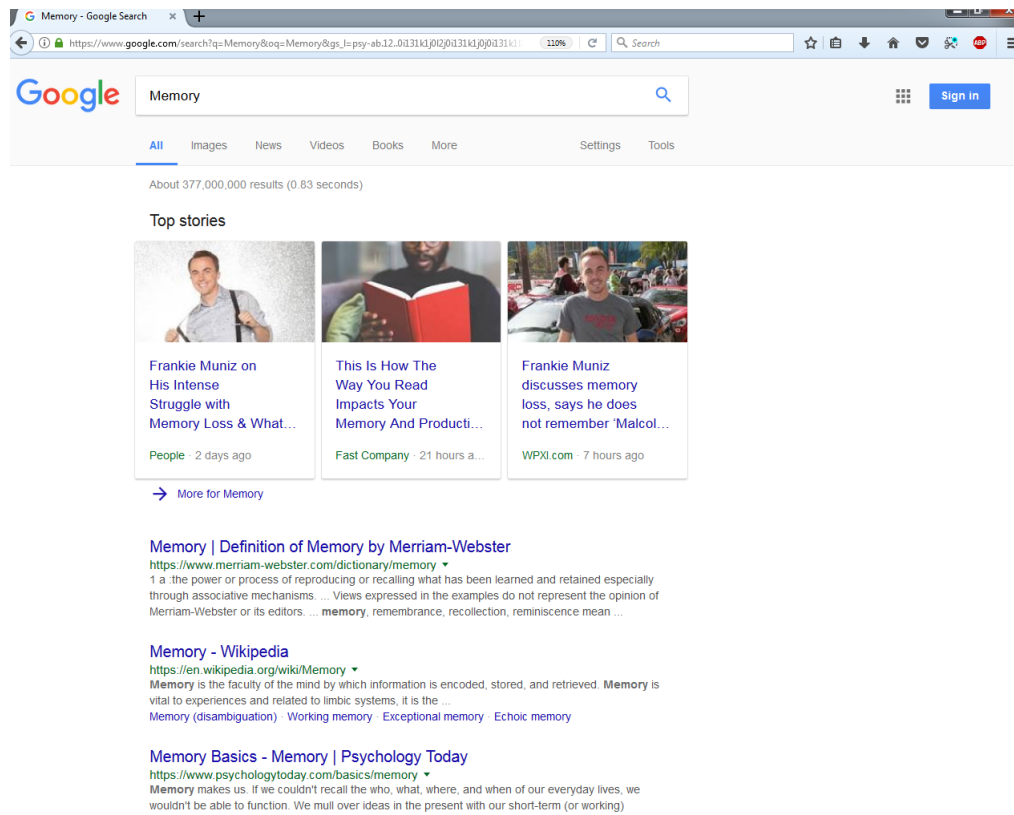
Below these sub-results, there is a section titled "Diesel Online Store: jeans, clothing, shoes, bags and watches" with a link to "shop.diesel.com/". This section includes a description of the "DIESEL BLACK GOLD, FALL WINTER 17 COLLECTION" and a link to "Men's · New Arrivals · Women's · New Denim".

Below the online store section, there is a "Top stories" section with three articles: "Petrol, Diesel Prices Fall Sharply In Gujarat, Maharashtra. Check T...", "Cut in VAT on petrol, diesel: Fuel cheaper in Maharashtra, poll-bou...", and "Oxford to ban petrol and diesel cars in 2020 in bid to be car-free by 2030".

On the right side of the search results, there is a "Knowledge Panel" for "Diesel". It includes the Diesel logo, a description of Diesel S.p.A. as an Italian retail clothing company, and various details such as "Customer service: 1 (877) 361-6150", "Headquarters: Breganze, Italy", "Founded: 1978, Molvena, Italy", "Industry: Fashion", "Parent organization: OTB Group", and "Founders: Renzo Rosso, Adriano Goldschmied". It also features social media profiles for Facebook, Instagram, and YouTube, and a section titled "People also search for" with links to Armani, Dolce & Gabbana, and Gucci.

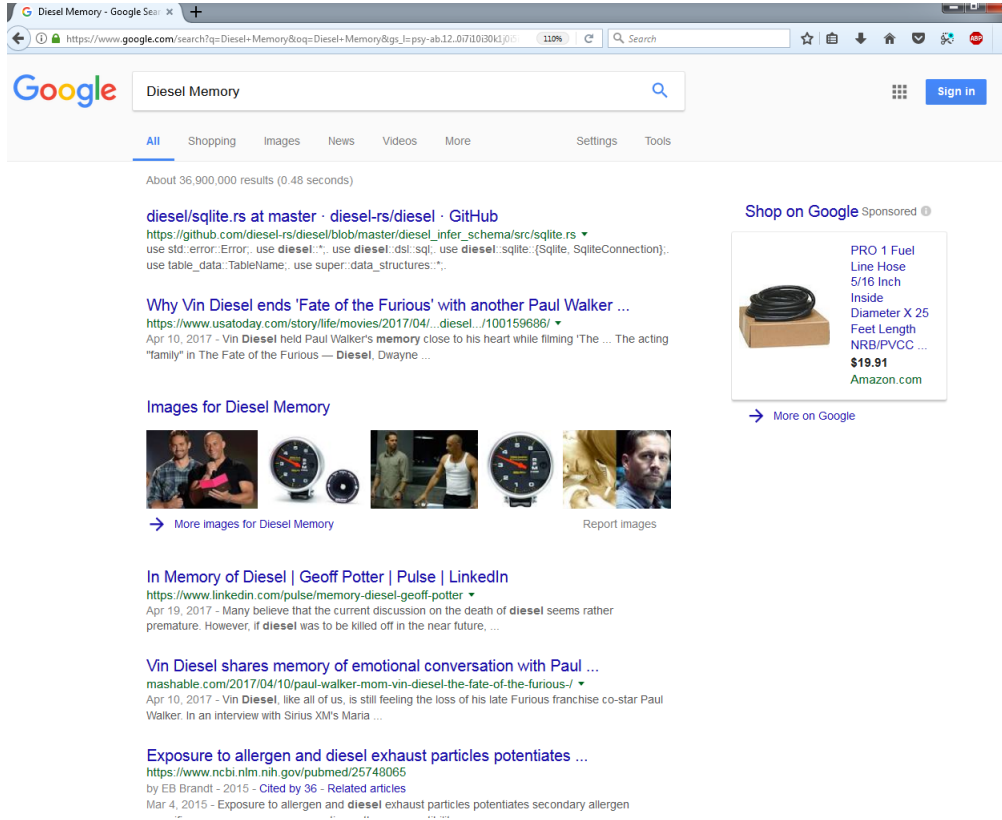
Google found about 377,000,000 results for the term “Memory”.

Figure 2: Query: Memory, Search Engine: Google



Google found about 36,900,000 results for the query “Diesel Memory”.

Figure 3: Query: Diesel Memory, Search Engine: Google



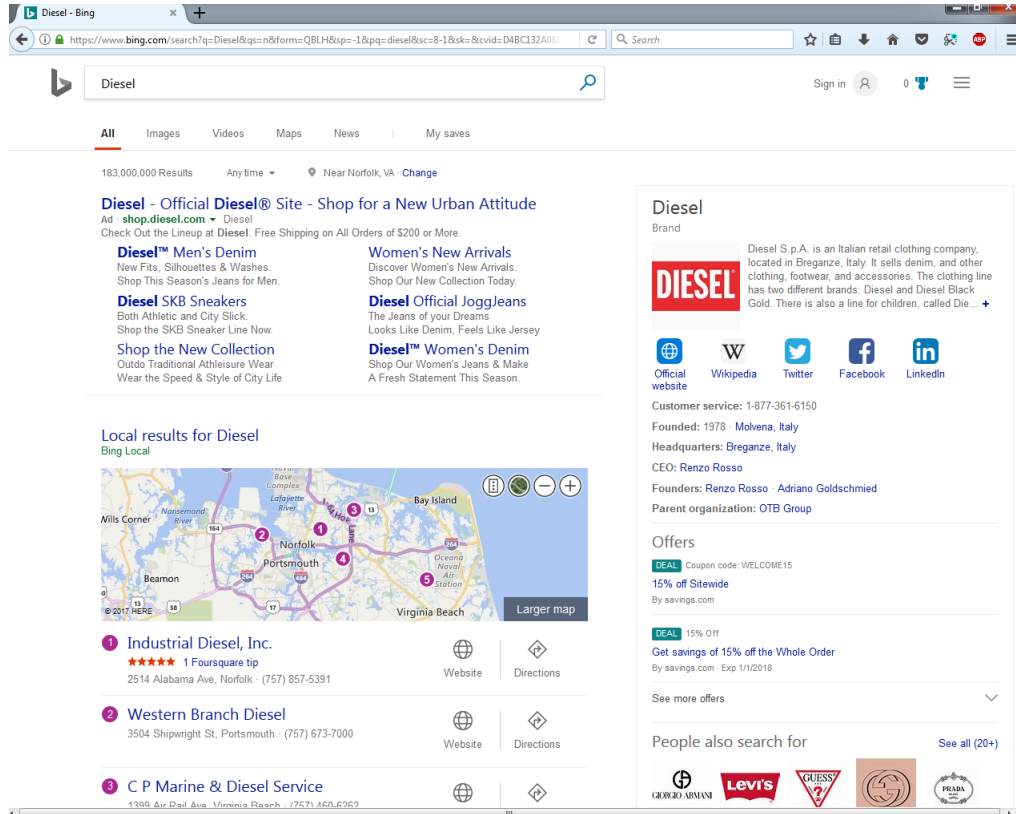
Plugging the results in the formula gives the following:

$$N_{google} = \frac{207,000,000 \times 377,000,000}{36,900,000} = 2114878048.780488 \approx 2114878049$$

B. Bing:

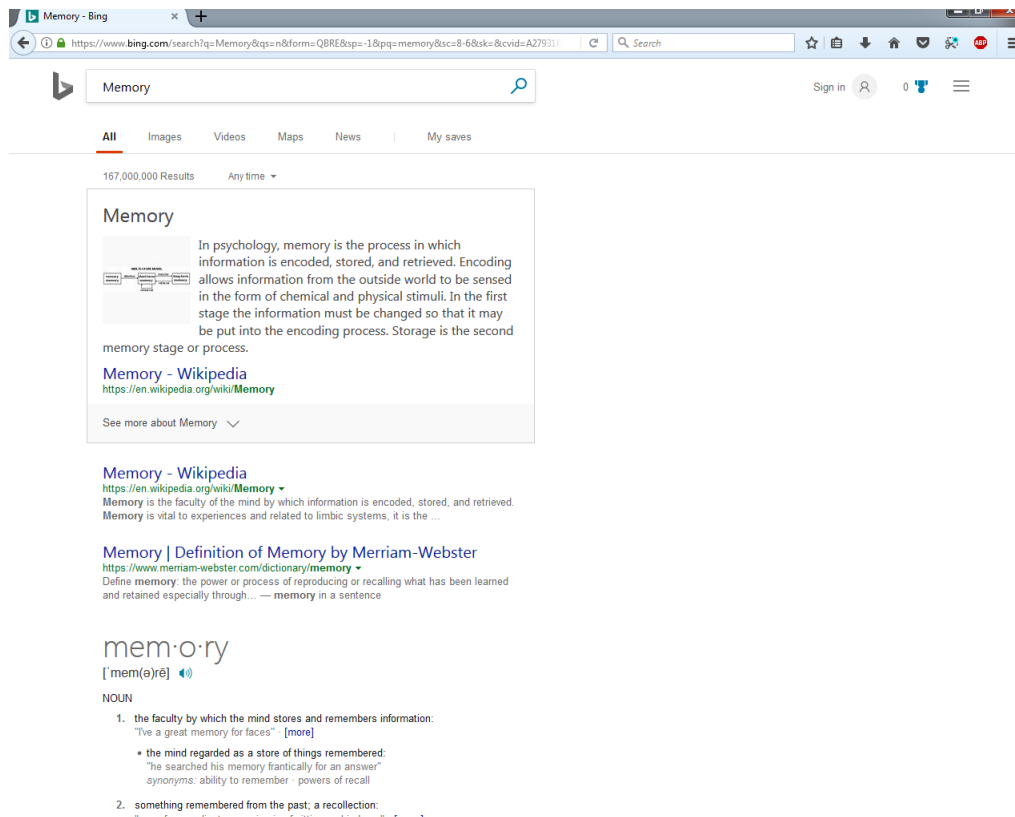
Bing found about 183,000,000 results for the term “Diesel”.

Figure 4: Query: Diesel, Search Engine: Bing



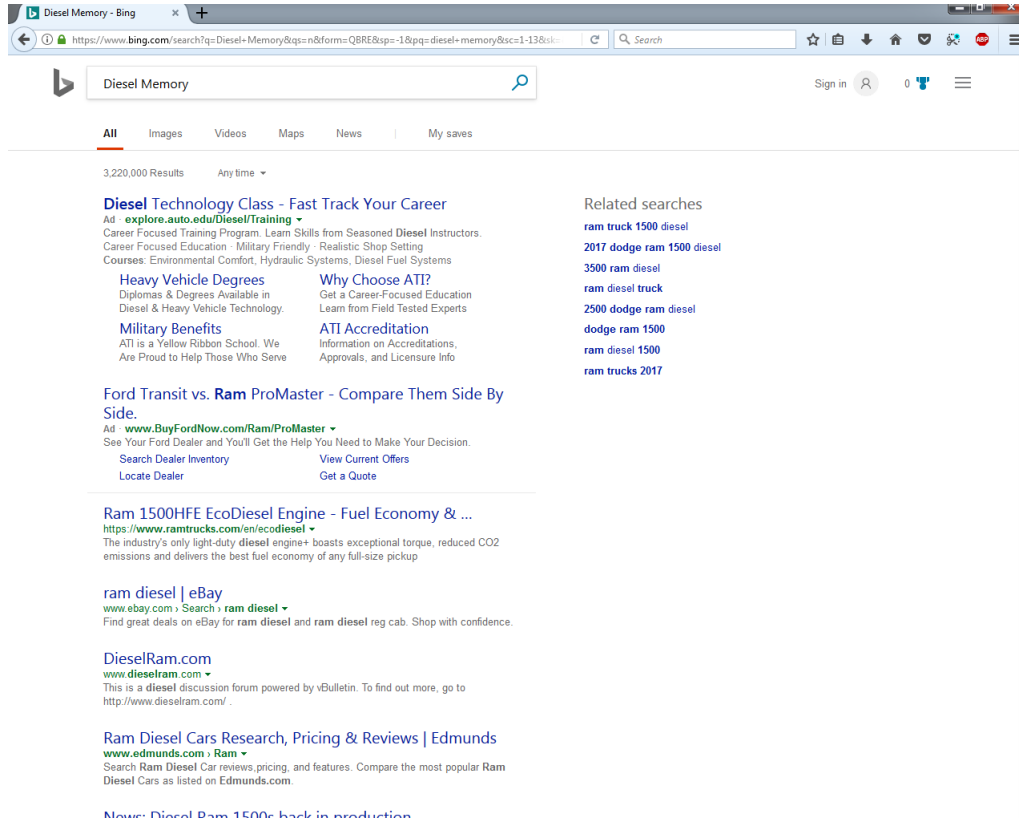
Bing found about 167,000,000 results for the term “Memory”.

Figure 5: Query: Memory, Search Engine: Bing



Bing found about 3,220,000 results for the query “Diesel Memory”.

Figure 6: Query: Diesel Memory, Search Engine: Bing



Plugging the results in the formula gives the following:

$$N_{Bing} = \frac{183,000,000 \times 167,000,000}{3,220,000} = 9490993788.819876 \approx 9490993789$$

2. Hairy Phone: I ran the second query “Hairy Phone” and found the following:

A. Google:

Google found about 106,000,000 results for the term “Hairy”.

Google found about 1,410,000,000 results for the term “Phone”.

Google found about 8,780,000 results for the query “Hairy Phone”.

Plugging the results in the formula gives the following:

$$N_{google} = \frac{106,000,000 \times 1,410,000,000}{8,780,000} = 17022779043.28018 \approx 17022779043$$

B. Bing:

Bing found about 292,000,000 results for the term “Hairy”.

Bing found about 748,000,000 results for the term “Phone”.

Bing found about 51,500,000 results for the query “Hairy Phone”.

Plugging the results in the formula gives the following:

$$N_{Bing} = \frac{292,000,000 \times 748,000,000}{51,500,000} = 4241087378.640777 \approx 4241087379$$

3. Computer Manifold: I ran the third query “Computer Manifold” and found the following:

A. Google:

Google found about 861,000,000 results for the term “Computer”.

Google found about 21,500,000 results for the term “Manifold”.

Google found about 6,470,000 results for the query “Computer Manifold”.

Plugging the results in the formula gives the following:

$$N_{google} = \frac{861,000,000 \times 21,500,000}{6,470,000} = 2861128284.38949 \approx 2861128284$$

B. Bing:

Bing found about 448,000,000 results for the term “Computer”.

Bing found about 38,400,000 results for the term “Manifold”.

Bing found about 35,100,000 results for the query “Computer Manifold”.

Plugging the results in the formula gives the following:

$$N_{Bing} = \frac{448,000,000 \times 38,400,000}{35,100,000} = 490119658.1196581 \approx 490119658$$

4. Electronic Lumber: I ran the forth query “Electronic Lumber” and found the following:

A. Google:

Google found about 1,010,000,000 results for the term “Electronic”.

Google found about 43,300,000 results for the term “Lumber”.

Google found about 5,720,000 results for the query “Electronic Lumber”.

Plugging the results in the formula gives the following:

$$N_{google} = \frac{1,010,000,000 \times 43,300,000}{5,720,000} = 7645629370.629371 \approx 7645629371$$

B. Bing:

Bing found about 289,000,000 results for the term “Electronic”.

Bing found about 42,400,000 results for the term “Lumber”.

Bing found about 244,000,000 results for the query “Electronic Lumber”.

Plugging the results in the formula gives the following:

$$N_{Bing} = \frac{289,000,000 \times 42,400,000}{244,000,000} = 50219672.13114754 \approx 50219672$$

The estimated number of indexed pages in Google as I found are:

2114878049

17022779043

2861128284

7645629371

It is clear that the estimates varied markedly, by few billions, for different queries. I assume that one of the reasons behind that is the difference in “independence level” of the two terms among different queries.

The estimated number of indexed pages in Bing as I found are:

9490993789

4241087379

490119658

50219672

The same inconsistency held for Bing. The estimates varied by few billions.

Question 2:

Exercise 4.6:

Process five Wikipedia documents using the Porter stemmer and the Krovetz stemmer. Compare the number of stems produced and find 10 examples of differences in the stemming that could have an impact on ranking.

Answer:

Question 3:

Exercise 4.8:

Find the 10 Wikipedia documents with the most inlinks. Show the collection of anchor text for those pages.

Answer:

Question 4:

Exercise 5.8:

Write a program that can build a simple inverted index of a set of text documents. Each inverted list will contain the file names of the documents that contain that word.

Suppose the file A contains the text the quick brown fox, and file B contains the slow blue fox. The output of your program would be:

```
% ./your-program A B
blue B
brown A
fox A B
quick A
slow B
the A B
```

Answer:

Question 5:

Exercise 5.10:

Suppose a company develops a new unambiguous lossless compression scheme for 2-bit numbers called SuperShrink. Its developers claim that it will reduce the size of any sequence of 2-bit numbers by at least 1 bit. Prove that the developers are lying. More specifically, prove that either:

SuperShrink never uses less space than an uncompressed encoding, or

There is an input to SuperShrink such that the compressed version is larger than the uncompressed input

You can assume that each 2-bit input number is encoded separately.

Answer:

We have 4 possible 2-bit numbers and 3 possible shorter bit sequences. Let's apply the pigeonhole principle, which states that if n items are put into m containers, with $n > m$, then at least one container must contain more than one item. This means that any mapping from 2-bit sequences to shorter sequences must have at least 2 sequences compressed to the same shorter sequence. Therefore, when trying to decompress this shorter sequence, mentioned above, it is impossible to know which of the original 2-bit sequences it was compressed from.

Listing 1: The content of crawler.py

```
import sys
import time
from bs4 import *
import urllib2
import re

def crawl(url):
    url = url.strip()
    page_file_name = str(hash(url))
    page_file_name = page_file_name + ".html"
    fh_page = open(page_file_name, "w")
    html_page = urllib2.urlopen(url)
    soup = BeautifulSoup(html_page, "html.parser")
    html_text = str(soup)
    fh_page.write(url + "\n")
    fh_page.write(page_file_name + "\n")
    fh_page.write(html_text)
    fh_page.close()
    children_links = set()
    for link in soup.findAll('a', attrs={'href': re.compile("^http://")}):
        children_links.add(link.get('href'))

    #convert relative links to absolute links
    for child in children_links:
        if isAbsolute(child):
            new_child = urllib2.urlparse.urljoin(url, child)
            children_links.remove(child)
            children_links.add(new_child)

    links_to_return = set()
    for child in children_links:
        try:
            r = urllib2.urlopen(child)
            http_message = r.info()
            full = http_message.type
            if full == "text/html" and r.getcode() == 200:
                links_to_return.add(child)
        except urllib2.HTTPError as e:
            print "There is an error crawling links in this page:"
            print "Error Code:"
            print e.code
```

```

    return links_to_return

def isAbsolute(url):
    try:
        return bool(urlparse(url).netloc)
    except:
        return False


if __name__ == "__main__":
    if len(sys.argv) != 3:
        print "Usage: python crawl.py <seed_url> <pages_count>"
        print "e.g: python crawl.py https://www.yahoo.com/ 20"
        exit()
    url = sys.argv[1]
    pages_count = int(sys.argv[2])
    print "Entered URL:"
    print url
    html_page = urllib2.urlopen(url)
    print "Final URL:"
    print html_page.geturl()
    print "*****"
    links_to_crawl = set()
    links_to_crawl.add(html_page.geturl())
    found_links = set()
    crawled_links = set()
    counter = 0
    while links_to_crawl and counter < pages_count:
        link = links_to_crawl.pop()
        print "crawling: " + link
        new_links = crawl(link)
        crawled_links.add(link)
        for new_link in new_links:
            if new_link not in crawled_links:
                links_to_crawl.add(new_link)
            if new_link not in found_links:
                print "New link found: " + new_link
                found_links.add(new_link)
        time.sleep(5)
        counter += 1

```

I ran the crawler on Dr. Nelson's web page and passed 10 "ten pages" as the number of pages I want to be downloaded.

This is a screen shot of running the crawler:

Figure 7: Running crawler.py



```
C:\Windows\system32\cmd.exe
C:\Users\Hussan-Den\Desktop\CS834\A1\Q5>python crawler.py http://www.cs.odu.edu/~mln/ 10
Entered URL:
http://www.cs.odu.edu/~mln/
Final URL:
http://www.cs.odu.edu/~mln/
*****
crawling: http://www.cs.odu.edu/~mln/
New link found: http://www.larc.nasa.gov/
New link found: http://www.cs.odu.edu/~mln/service/
New link found: http://www.cs.odu.edu/~mln/personal/
New link found: http://www.openarchives.org/ore/
New link found: http://www.cs.odu.edu/~mln/research/
New link found: http://www.cs.odu.edu/~mln/travel.html
New link found: http://ntrs.nasa.gov/
New link found: http://www.cs.odu.edu/
New link found: http://www.openarchives.org/rs/toc
New link found: http://sils.unc.edu/
New link found: http://www.mementoweb.org/guide/rfc/ID/
New link found: http://www.cs.odu.edu/~mln/teaching/
New link found: http://www.openarchives.org/pnh/
New link found: http://www.cs.odu.edu/~mln/pubs/
New link found: http://www.cs.odu.edu/~mln/lineage.html
New link found: http://www.cs.odu.edu/~mln/
New link found: http://www.odu.edu
New link found: http://www.nsf.gov/awardsearch/showAward.do?AwardNumber=0643784
crawling: http://www.nsf.gov/awardsearch/showAward.do?AwardNumber=0643784
New link found: http://www.nsf.gov/awards/managing/index.jsp?org=NSF
New link found: http://www.nsf.gov/awardsearch/index.jsp
New link found: http://www.nsf.gov/awards/managing/co-op_conditions.jsp?org=NSF
New link found: http://www.nsf.gov/publications/pub_summ.jsp?ods_key=gpm
New link found: http://www.nsf.gov/awards/presidential.jsp
New link found: http://www.nsf.gov/awards/managing/special_conditions.jsp?org=NSF
F
New link found: http://www.nsf.gov/awards/managing/fed_dem_part.jsp?org=NSF
New link found: http://www.nsf.gov/awards/about.jsp
New link found: http://dellweb.bfa.nsf.gov/
New link found: http://par.nsf.gov/
New link found: http://www.nsf.gov/bfa/dias/policy/
New link found: http://www.usa.gov/
New link found: http://www.nsf.gov/awards/managing/general_conditions.jsp?org=NSF
F
New link found: http://www.nsf.gov/div/index.jsp?div=IIS
crawling: http://www.nsf.gov/awards/managing/index.jsp?org=NSF
New link found: http://www.research.gov/
New link found: http://www.research.gov
crawling: http://www.larc.nasa.gov/
crawling: http://www.nsf.gov/awards/managing/fed_dem_part.jsp?org=NSF
crawling: http://ntrs.nasa.gov/
There is an error crawling links in this page:
Error Code:
404
There is an error crawling links in this page:
Error Code:
403
New link found: http://www.nasa.gov/
```

This is a screen shot of the 10 HTML files downloaded from crawled pages:

References

- [1] Stackoverflow. <https://stackoverflow.com/questions/tagged/python>.
- [2] Sergey Brin and Larry Page. Google search engine. <http://google.stanford.edu>.
- [3] http://www.dba-oracle.com/oracle_news/news_ebay_massive_oracle.htm
- [4] <https://www.wired.com/2012/01/amazon-dynamodb/>
- [5] <https://www.youtube.com/watch?v=a0OvgTfF8Pg>
- [6] <https://stackoverflow.com/questions/6526181/write-a-program-that-takes-text-as-input-and-produces-a-program-that-reproduces/6526266#6526266>
- [7] https://en.wikipedia.org/wiki/Pigeonhole_principle

Figure 8: Pages downloaded by crawler.py

