

Assignment 1

CS834 Introduction to Information Retrieval

Fall 2017

Polykarpos Thomadakis

September 20, 2017

Question 1.1

Think up and write down a small number of queries for a web search engine. Make sure that the queries vary in length (i.e., they are not all one word). Try to specify exactly what information you are looking for in some of the queries. Run these queries on two commercial web search engines and compare the top 10 results for each query by doing relevance judgments. Write a report that answers at least the following questions: What is the precision of the results? What is the overlap between the results for the two search engines? Is one search engine clearly better than the other? If so, by how much? How do short queries perform compared to long queries?

Answer

Bing and DuckDuckGo were the two search engines that I compared for the purposes of this assignment. I searched for something that I expected to be a bit hard for them to return the best results without enough words. My search was for examples of Message Passing Interface (MPI) code in C language.

The precision is defined as the number of relevant results recalled by the total number of results recalled :

$$Precision = \frac{Relevant\ results}{Total\ number\ of\ results}$$

The overlap is defined as the number of results that appear in both search engines for the same query:

$$Overlap = \frac{Results \in \mathcal{D} \cap \mathcal{B}}{Number\ of\ results\ per\ search\ engine}$$

where \mathcal{D}, \mathcal{B} are the sets of results for DuckDuckGo and Bing respectively.

Note for DuckDuckGo: I did not include the result that is an advertisement in my evaluations, since advertisements are usually biased to come up even when they might not be that relevant.

I started with a one word query, simply MPI:

The acronym MPI is used for many different things, so none of the engines did that well with the one-word query. Bing did not return any relevant result at all, as opposed to DuckDuckGo which returned 3.

As for the overlap, it was zero for the relevant results, however 3 out of the non relevant results are common. I think it is important to note that and also the fact that the top result for both engines was the same.

The results can be seen in figure 1 and the precision in each case is:

$$Precision_{DuckDuckGo} = \frac{3}{10} = 0.3$$

$$Precision_{Bing} = \frac{0}{10} = 0$$

$$Overlap_{Relevant} = \frac{0}{10} = 0 \quad Overlap_{total} = \frac{3}{10} = 0.3$$

The next query was with 2 words, MPI C:

This time the engines did a much better job. Seven of the results were relevant for DuckDuckGo with the top 4 being relevant and two of the three irrelevants at the last two positions. Bing came short with 6 relevant results, however it made a big difference compared to the previous attempt which had no relevant results. The third, the seventh, the ninth and the tenth results were irrelevant. Six of the relevant results appear in both search engines and nine of the total results are common. The results are presented in figure 2.

$$Precision_{DuckDuckGo} = \frac{7}{10} = 0.7$$

$$Precision_{Bing} = \frac{6}{10} = 0.6$$

$$Overlap_{Relevant} = \frac{6}{10} = 0.6 \quad Overlap_{total} = \frac{9}{10} = 0.9$$

For the last attempt I used 3 words, MPI C examples:

In this case both engines did their best, returning only relevant results. Seven of the results are common for this case with the top two results being in the same ranking order. Another interesting finding is that the total overlap of the two engines has decreased even though their precision has increased.

The results are presented in figure 3.

$$Precision_{DuckDuckGo} = \frac{10}{10} = 1.0$$

$$Precision_{Bing} = \frac{10}{10} = 1.0$$

$$Overlap_{Relevant} = \frac{7}{10} = 0.7 \quad Overlap_{total} = \frac{7}{10} = 0.7$$

The overall results can be seen in table 1. It can be easily noted that the DuckDuckGo search engine outperforms Bing in all cases. In addition to that, we can see that increasing the length of the query results in a better precision.

	$Prec_{DuckDuckGo}$	$Prec_{Bing}$	Relevant Overlap	Total Overlap
MPI	0.3	0	0	0.3
MPI C	0.7	0.6	0.6	0.9
MPI C examples	1.0	1.0	0.7	0.7

Table 1: Overall results

b mpi

Web Images Videos Maps News My saves

Also try: [mpi number](#) · [mpi test](#) · [mpi medical term](#)

26,100,000 RESULTS Any time ▾

Meeting Professionals International

<https://www.mpiweb.org> ▾
Who is MPI? Live events connect people, inspire ideas and propel society forward. Meeting Professionals International (MPI) is the largest meeting and event industry ...

Membership

Do you have a question for MPI members services? Whatever your need, get ...

Events

Join MPI Foundation at Rendezvous at IncentiveWorks, the official opening night ...

Industry Careers

Be the first to know when a new job is posted and receive free, personalized career ...

Join MPI

As a Planner or Supplier, MPI's tiered membership approach gives you the ability ...

Chapters & Community

FIND A CHAPTER. MPI chapters allow our members to form powerful social and ...

Career Center

Meeting Professionals International offers the top jobs available in Event Planning. ...

[See results only from mpiweb.org](#)

MPI Fastech | The source for Industrial and marine ...
www.mpifastech.com/history.htm ▾
MPI Fastech is a wholesale distributor for industrial products. We are located in Virginia Beach, VA

MPI Virginia - Home Page
mpivirginia.org/index.php ▾
MPI Virginia. The Virginia Chapter of Meeting Professionals International was chartered in 1986 and became MPI's 42nd Chapter. The chapter consists of over 180 ...

Mpi Cafe in Norfolk, VA - Find.HamptonRoads.com
find.hamptonroads.com/mpi-cafe-norfolk-va.html ▾
Mpi Cafe at 5957 E Virginia Beach Blvd, Norfolk, VA 23502

MPI Virginia - Newsletter
mpivirginia.org/news.php?id=50 ▾
Hilton Norfolk at the Main - Norfolk, VA. MPI Virginia Charlottesville Forum June 11, 2017 Save the Date!
MPI World Education Congress - WEC 2017 June 19 - 22, 2017


Mpi Cafe - Norfolk, VA
loco.com ▾ Restaurants ▾ Caribbean
Caribbean, Restaurants - 5957 East Virginia Beach Blvd., Norfolk, VA 23502 · (757) 466-0220
Mpi Cafe, Caribbean business in Norfolk. See up-to-date pricelists and view recent announcements for this location.

MPI Virginia Chapter - Home | Facebook
<https://www.it-to-ro.fbjs.facebook.com/MPIVirginiaChapter> ▾
MPI Virginia Chapter, Richmond, VA. 360 likes. MPI Virginia normally meets the first Thursday and offers great opportunities for education and networking...

PDF MPI and GBTA Foundation Recognize CMM Recipients from ...
www.mpiweb.org/docs/.../cmm-norfolk-2016-class-press-release-final.pdf
MPI and GBTA Foundation Recognize CMM Recipients from Norfolk 2016 Class, Announce Destinations for 2017 Programs DALLAS, TX and ALEXANDRIA, VA - January 12, 2017 ...

Mpi in Downtown Norfolk Norfolk, Virginia with Reviews ...
<https://www.yellowpages.com/downtown-norfolk-norfolk-va/mpi> ▾
Find 2 listings related to Mpi in Downtown Norfolk on YP.com. See reviews, photos, directions, phone numbers and more for Mpi locations in Downtown Norfolk, Norfolk, VA.

MPI Training - Massachusetts Law Enforcement Police Training
municipalpoliceinstitute.org ▾
MPI Training offers Massachusetts Law Enforcement professionals with online In-Service E-Learning programs and our traditional classroom training seminars.

 mpi

Web Images Videos Meanings

All Regions ▾ Safe Search: Strict ▾ Any Time ▾

Cloverleaf for hospitals - Interoperability, data aggregation. ^{AD}
Interoperability, data aggregation. EMR, MPI, FHIR, HIE, CCD.
Infor.com/EHR_Cloverleaf | [Report Ad](#)

Meeting Professionals International
Who is MPI? Live events connect people, inspire ideas and propel society forward. Meeting Professionals International (MPI) is the largest meeting and event industry ...
<https://www.mpiweb.org>

MPI - Master Painter Institute
Home to MPI's online training program for professionals in the architectural paint industry.
mpi.net

MPI Training - Massachusetts Law Enforcement Police Training
MPI Training offers Massachusetts Law Enforcement professionals with online In-Service E-Learning programs and our traditional classroom training seminars.
municipalpoliceinstitute.org

mpiwi.com - MPI Property Management, LLC.
Milwaukee property management at its best. Let MPI Property Management, LLC's experienced Milwaukee property managers care for your rental home. If you are looking ...
mpiwi.com

MPI - What does MPI stand for? The Free Dictionary
Acronym Definition; MPI: Meeting Professionals International; MPI: Message Passing Interface; MPI: Migration Policy Institute (Washington, DC) MPI: Max-Planck ...
acronyms.thefreedictionary.com/MPI

Medical Professional Institute
Medical Assistant. Train for a career where your skills will always be in demand and you can have variety in your job every day! Click here to learn more about our 6 ...
mpl.edu

The Message Passing Interface (MPI) standard - mcs.anl.gov
Interface standard, tutorials, libraries, and links to other resources, as well as MPICH, an implementation of MPI.
mcs.anl.gov/research/projects/mpi/

Message Passing Interface - Wikipedia
Message Passing Interface (MPI) is a standardized and portable message-passing system designed by a group of researchers from academia and industry to function on a ...
https://en.wikipedia.org/wiki/Message_Passing_Interface

MPI Research - Preclinical and Early Clinical Contract Research
MPI Research strives to be the best CRO in preclinical and early clinical contract research.
mpiresearch.com

Open MPI: Open Source High Performance Computing
A High Performance Message Passing Library. The Open MPI Project is an open source Message Passing Interface implementation that is developed and maintained by a ...
<https://www.open-mpi.org>

Figure 1: Comparison of the results for query: MPI

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 whether the search is based on a database or grep style of matching, or if the search is using some type of ranking.

Answer

Here are five sites that I use that appear to use search:

1. **imdb.com**

The Internet Movie Database (abbreviated IMDb) is an online database of information related to films, television programs and video games, including cast, production crew, fictional characters, biographies, plot summaries, trivia and reviews. The user uses a search query to get information about movies, series or games that are relevant. The search looks to be done against a database.

2. **stackoverflow.com**

Stack Overflow is a website featuring questions and answers on a wide range of topics in computer programming. The website serves as a platform for users to ask and answer questions, and, through membership and active participation, to upvote or downvote questions and answers based on their content. This site seems to use a grep style search and also features ranking based on the upvotes of the relevant posts.

3. **play.google.com**

Google Play is a digital distribution service. It serves as the official app store for the Android operating system, allowing users to browse and download applications developed with the Android software development kit (SDK) and published through Google. Google Play also serves as a digital media store, offering music, magazines, books, movies, and television programs. Google Play seems to use a database search and using ranking based on the popularity of the results and their user provided reviews.

4. **craigslist.com**

Craigslist is a classified advertisements website with sections devoted to jobs, housing, personals, for sale, items wanted, services, community, gigs, resumes, and discussion forums. A user can choose a section and make a search for results relevant to it. The site seems to be using a database search style.

5. **github.com**

GitHub is a web-based Git or version control repository and Internet hosting service. It is mostly used for code. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project. Github seems to use a database style of search and offers optional ranking based on best match, highest rating, freshness, most forked.

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 that the files are accessible from a website at the URL `http://www.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

The program discovers all the files in the directory given from the user as a command line argument. Then it continues with the files and subdirectories of its subdirectories and goes on until all the files and directories under the given directory have been discovered recursively. To achieve this I use the `os.walk()` function from the `os` directory.

For each file found an xml url structure is generated where the location is registered and then the last modification date information is retrieved with the `os.stat()` function and is included in the url structure of the file.

Once all files in the tree under the origin directory have been discovered the procedure ends and the xml file is generated with the appropriate format and the structures generated are written on it.

The user can specify the path where the file will be created, the default location is the current directory. The source code used can be seen in listing 1 and the sitemap generated for the directory of the assignment can be found in listing 2.

```
1 import os
2 import time
3 import sys
4 import argparse
5
6 parser = argparse.ArgumentParser('smapgen')
7 parser.add_argument('path', help='Path to the directory that will be used to generate
8     sitemap')
9 parser.add_argument('--path-to-gen', '-p', help='Path where the generated sitemap.xml
10     will be put [default= Current directory]', default='./')
11 args = parser.parse_args();
12 p_to_gen = args.path_to_gen+'sitemap.xml';
13 root_dir = args.path
14
15 out=''
16 out+='xml version="1.0" encoding="UTF-8"?&gt;\n'
17 out+='<?urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"&gt;\n'
18 for root, dirs, files in os.walk(root_dir):
19     for f in files:
20         loc = root+'/' +f
21         url = root.replace(root_dir, "http://www.example.com")
22         out+= ' &lt;url&gt;\n'
23         out+= ' &lt;loc&gt;'+url+'/' +f+'&lt;/loc&gt;\n'
24         out+= ' &lt;lastmod&gt;'+time.strftime("%Y-%m-%d", time.localtime(os.path.getmtime(
25             loc)))+&lt;/lastmod&gt;\n'
26         out+= ' &lt;/url&gt;\n'
27 out+= '&lt;/urlset&gt;'
28
29 with open(p_to_gen, 'w') as f:
30     f.write(out)
31     f.close()</pre
```

Listing 1: Sitemap generator

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
3   <url>
4     <loc>http://www.example.com/A1.pdf</loc>
5     <lastmod>2017-09-03</lastmod>
6   </url>
7   <url>
8     <loc>http://www.example.com/compress.py</loc>
9     <lastmod>2017-09-03</lastmod>
10  </url>
11  <url>
12    <loc>http://www.example.com/sample-output</loc>
```

```

14     <lastmod>2017-09-02</lastmod>
15 </url>
16 <url>
17     <loc>http://www.example.com/A1.synctex.gz</loc>
18     <lastmod>2017-09-03</lastmod>
19 </url>
20 <url>
21     <loc>http://www.example.com/crawl.py</loc>
22     <lastmod>2017-09-02</lastmod>
23 </url>
24 <url>
25     <loc>http://www.example.com/A1.log</loc>
26     <lastmod>2017-09-03</lastmod>
27 </url>
28 <url>
29     <loc>http://www.example.com/smapgen.py</loc>
30     <lastmod>2017-09-03</lastmod>
31 </url>
32 <url>
33     <loc>http://www.example.com/A1.tex</loc>
34     <lastmod>2017-09-03</lastmod>
35 </url>
36 <url>
37     <loc>http://www.example.com/decompress.py</loc>
38     <lastmod>2017-09-03</lastmod>
39 </url>
40 <url>
41     <loc>http://www.example.com/A1.aux</loc>
42     <lastmod>2017-09-03</lastmod>
43 </url>
44 <url>
45     <loc>http://www.example.com/sitemap.xml</loc>
46     <lastmod>2017-09-03</lastmod>
47 </url>
48 <url>
49     <loc>http://www.example.com/Wiki.html</loc>
50     <lastmod>2017-09-03</lastmod>
51 </url>
52 <url>
53     <loc>http://www.example.com/InformationRetrieval.pdf</loc>
54     <lastmod>2017-09-02</lastmod>
55 </url>
56 <url>
57     <loc>http://www.example.com/Wc.html</loc>
58     <lastmod>2017-09-03</lastmod>
59 </url>
60 <url>
61     <loc>http://www.example.com/NewFolder/test</loc>
62     <lastmod>2017-09-03</lastmod>
63 </url>
64 <url>
65     <loc>http://www.example.com/NewFolder/Another/anothertest</loc>
66     <lastmod>2017-09-03</lastmod>
67 </url>
68 </urlset>

```

Listing 2: Generated sitemap

Question 3.9

Write a simple single-threaded web crawler. Starting from a single input URL (perhaps a professors 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

The crawler starts with one page on its list, the one provided by the user or the default. When all links of the page have been found, the list is filled with the new pages to be crawled. The crawler continues crawling pages in this list, parsing their links.

Once all of the pages in the list have been parsed it will continue with their respective links and so on. In order to keep the running time reasonable the user can set the depth in which this procedure will continue crawling. The default depth is 2.

To avoid downloading big irrelevant files, the crawler will skip non-HTML files by checking the content-type in the header. Command line arguments can be used to set the path where the crawler will create a directory and store the downloaded files.

The source code for the crawler is presented in listing 3 and a sample of the output for depth 2 can be seen in listing 4.

```
1 from bs4 import BeautifulSoup
2 from urlparse import urljoin
3 import argparse
4 import requests
5 from time import sleep
6 import os
7
8 def crawl(depth, s_url):
9     i = 1
10    file_num = 0
11    s_url = s_url.strip()
12    print 'URL to start crawling:', s_url
13    print 'Maximum depth:', depth
14    urls = [s_url]
15    while i <= depth:
16        deeper_links = list()
17        for url in urls:
18            if url not in checked_urls:
19                try:
20                    # Request the link from the Internet and set a timeout to prevent
21                    # hanging
22                    request = requests.get(url, stream=True, timeout=1)
23                    # Will only parse html files, skips any other format
24                    if( 'text/html' not in request.headers['Content-Type'] ):
25                        print 'Skipping ', url
26                        continue
27                    print 'Crawling ', url,
28                except :
29                    continue
30                # Check if link is valid or not
31                if(request.status_code==200):
32                    checked_urls.add(url)
33                    print ' URL OK'
34                else:
35                    print ' URL NOT OK'
36                    continue
37                file_num +=1
38                with open(path+'/' +str(file_num)+'.html', 'w') as file :
39                    file.write(request.text.encode('utf-8'))
40                    file.close()
41                soup = BeautifulSoup(request.text, 'html.parser')
42                # Retrieve the links in the html file
43                links = soup.find_all('a')
44                for link in links:
```

```

45     new_link = link.get('href')
46     if new_link :
47         new_link = new_link.strip()
48         # Make link absolute if needed
49         new_link = urljoin(url,new_link)
50         # If link not crawled yet add to pending links
51         if new_link not in checked_urls:
52             deeper_links.append(new_link)
53         # Wait 5 seconds before downloading the next html file
54         sleep(5)
55     print len(deeper_links), 'links for next depth'
56     urls = deeper_links
57     print 'Finished depth ',i
58     i=i+1
59
60 # Set the command line arguments
61 parser = argparse.ArgumentParser('Crawler Parser')
62 parser.add_argument('--url', '-u', help='The url to start crawling from [default= http
63 ://www.cs.odu.edu/~mln/ ', default='http://www.cs.odu.edu/~mln/')
64 parser.add_argument('--depth', '-d', help='The crawl depth [default= 5]', default=5, type
65 =int)
66 parser.add_argument('--path', '-p', help='The path where the crawl directory will be
67 created to put the downloaded files [default= current directory]', default='.')
68 args = parser.parse_args();
69 depth = args.depth
70 s_url = args.url
71 path = os.path.abspath(args.path)+'/'
72 if not os.path.exists(path):
73     os.makedirs(path)
74 print 'Files will be downloaded at',path
75 # A set of all previously met links
76 checked_urls = set()
77 crawl(depth, s_url)
78 print 'Crawler finished, you can find the downloaded files at',path

```

Listing 3: Web crawler

```

1 Files will be downloaded at /home/poll/Desktop/crawl
2 URL to start crawling: http://www.cs.odu.edu/~mln/
3 Maximum depth: 2
4 Crawling http://www.cs.odu.edu/~mln/ URL OK
5 24 links for next depth
6 Finished depth 1
7 Crawling http://www.cs.odu.edu/ URL OK
8 Crawling http://www.odu.edu URL OK
9 Crawling http://www.cs.odu.edu/~mln/research/ URL OK
10 Crawling http://www.cs.odu.edu/~mln/pubs/ URL OK
11 Crawling http://www.cs.odu.edu/~mln/teaching/ URL OK
12 Crawling http://www.cs.odu.edu/~mln/service/ URL OK
13 Crawling http://www.cs.odu.edu/~mln/personal/ URL OK
14 Crawling http://www.larc.nasa.gov/ URL OK
15 Crawling http://sils.unc.edu/ URL OK
16 Crawling http://www.openarchives.org/pmh/ URL OK
17 Crawling http://www.openarchives.org/ore/ URL OK
18 Crawling http://www.mementoweb.org/guide/rfc/ID/ URL OK
19 Crawling http://www.openarchives.org/rs/toc URL OK
20 Crawling http://ntrs.nasa.gov/ URL OK
21 Crawling http://www.nsf.gov/awardsearch/showAward.do?AwardNumber=0643784 URL OK
22 Skipping http://www.cs.odu.edu/~mln/cv.pdf
23 Skipping http://www.cs.odu.edu/~mln/nsf-cv-2014.pdf
24 Crawling http://www.cs.odu.edu/~mln/lineage.html URL OK
25 Crawling http://www.cs.odu.edu/~mln/travel.html URL OK
26 Skipping http://www.cs.odu.edu/~mln/mln-ad.pdf
27 Crawling https://storify.com/michaelnelson/coverage-of-ws-dl-members-and-research
28 URL OK
29 Crawling http://ws-dl.blogspot.com/ URL OK

```

```

29 | Crawling http://twitter.com/phonedude_mln URL OK
    | Crawling https://twitter.com/phonedude_mln URL OK
31 | 2973 links for next depth
    | Finished depth 2
33 | Crawler finished, you can find the downloaded files at /home/poll/Desktop/crawl

```

Listing 4: Web crawler sample output

Question 3.12

Design a compression algorithm that compresses HTML tags. Your algorithm should detect tags in an HTML file and replace them with a code of your own design that is smaller than the tag itself. Write an encoder and decoder program.

Answer

To create the compression algorithm I introduced a 1-1 mapping between each html tag and my compressed tags. HTML tags with 3 or more letters are compressed to 2-3 at most. I start by loading the HTML file and finding all HTML tags in it using regular expressions to match them. Once a tag is matched it will be checked for transformation using the mapping introduced. For the decompression algorithm I follow the same procedure, but now I perform the mapping the reverse way i.e map the custom tags to the original ones and transform them when needed. The user must specify an input and output file name for both programs. Of course the output file should not exist, or it will be overwritten. The source code for both programs is found in listings 5 and 6 respectively. Sample input, output can be found in listings 7 and 8. Also the full files are available on my github repository but are not included here because of their size.

```

1 | import re
  | import argparse
3 |
  | def repl(matchobj):
5 |     for k,v in mapping.iteritems():
        if matchobj.group() == '<'+k :
7 |         return '<'+v
        elif matchobj.group() == '</'+k :
9 |         return '</'+v
    return matchobj.group()
11 |
13 | mapping = { '!DOCTYPE': '!DT', 'abbr': 'ab', 'acronym': 'ac', 'address': 'ad', 'applet': 'ap', '
    area': 'ar', 'article': 'art'
    , 'aside': 'as', 'audio': 'au', 'basefont': 'bf', 'base': 'ba', 'bdi': 'bi', 'bdo': 'bo', 'big': '
    bg', 'blockquote': 'bq', 'body': 'bd',
15 | 'button': 'bu', 'canvas': 'cv', 'caption': 'ca', 'center': 'ce', 'cite': 'ci', 'code': 'cd', '
    colgroup': 'cg', 'datalist': 'da', 'del': 'de',
    'details': 'da', 'dfn': 'df', 'dialog': 'dg', 'dir': 'dr', 'div': 'dv', 'embed': 'ed', 'fieldset'
    : 'fs', 'figure': 'fi', 'font': 'ft', 'footer': 'fr',
17 | 'form': 'fm', 'frame': 'fe', 'frameset': 'fa', 'header': 'he', 'head': 'hd', 'html': 'hl', '
    iframe': 'ir', 'img': 'im', 'input': 'ip', 'ins': 'in',
    'kbd': 'kb', 'keygen': 'kg', 'label': 'la', 'legend': 'le', 'link': 'ln', 'main': 'mi', 'map': 'mp
    ', 'mark': 'mk', 'menuitem': 'mn', 'menu': 'mn',
19 | 'meta': 'mt', 'meter': 'mr', 'nav': 'na', 'noframes': 'nf', 'nscript': 'ns', 'object': 'ob', '
    optgroup': 'og', 'option': 'op', 'output': 'ou',
    'param': 'pa', 'picture': 'pi', 'pre': 'pr', 'progress': 'pg', 'ruby': 'rb', 'samp': 'sa', '
    script': 'sc', 'section': 'se', 'select': 'sl',
21 | 'small': 'sm', 'source': 'so', 'span': 'sp', 'strike': 'st', 'strong': 'sn', 'style': 'sy', 'sub'
    : 'su', 'summary': 'sr', 'table': 'ta', 'tbody': 'tb',
    'textarea': 'ta', 'tfoot': 'tf', 'thead': 'te', 'time': 'ti', 'title': 'tl', 'track': 'tk', 'var'
    : 'vr', 'video': 'vd', 'wbr': 'wb'}

```

```

23 parser = argparse.ArgumentParser('compress')
25 parser.add_argument('path', help='Path to the file that will be compressed')
25 parser.add_argument('compressed', help='Path to the compressed file that will be
    generated')
27 args = parser.parse_args();
27 p_comp = args.compressed
29 p_path = args.path

31 f = open(p_path, 'r')
31 out=""
33
33 for line in f:
35     line = re.sub('</?[A-Za-z0-9]+', repl, line)
35     out+=line
37 f.close()
37 f = open(p_comp, 'w')
39 f.write(out)
39 f.close()

```

Listing 5: Compress source

```

import re
2 import argparse

4 def revrepl(matchobj):
    for k,v in mapping.iteritems():
        if matchobj.group() == '<'+v :
            return '<'+k
        elif matchobj.group() == '</'+v :
            return '</'+k
10    return matchobj.group()

12
mapping = {'!DOCTYPE': '!DOCTYPE', 'abbr': 'ab', 'acronym': 'ac', 'address': 'ad', 'applet': 'ap', '
    area': 'ar', 'article': 'art'
14 , 'aside': 'as', 'audio': 'au', 'basefont': 'bf', 'base': 'ba', 'bdi': 'bi', 'bdo': 'bo', 'big': '
    bg', 'blockquote': 'bq', 'body': 'bd',
    'button': 'bu', 'canvas': 'cv', 'caption': 'ca', 'center': 'ce', 'cite': 'ci', 'code': 'cd', '
    colgroup': 'cg', 'datalist': 'da', 'del': 'de',
16 'details': 'da', 'dfn': 'df', 'dialog': 'dg', 'dir': 'dr', 'div': 'dv', 'embed': 'ed', 'fieldset':
    'fs', 'figure': 'fi', 'font': 'ft', 'footer': 'fr',
    'form': 'fm', 'frame': 'fe', 'frameset': 'fa', 'header': 'he', 'head': 'hd', 'html': 'hl', '
    iframe': 'ir', 'img': 'im', 'input': 'ip', 'ins': 'in',
18 'kbd': 'kb', 'keygen': 'kg', 'label': 'la', 'legend': 'le', 'link': 'ln', 'main': 'mi', 'map': 'mp'
    , 'mark': 'mk', 'menuitem': 'mn', 'menu': 'mn',
    'meta': 'mt', 'meter': 'mr', 'nav': 'na', 'noframes': 'nf', 'nscript': 'ns', 'object': 'ob', '
    optgroup': 'og', 'option': 'op', 'output': 'ou',
20 'param': 'pa', 'picture': 'pi', 'pre': 'pr', 'progress': 'pg', 'ruby': 'rb', 'samp': 'sa', '
    script': 'sc', 'section': 'se', 'select': 'sl',
    'small': 'sm', 'source': 'so', 'span': 'sp', 'strike': 'st', 'strong': 'sn', 'style': 'sy', 'sub':
    'su', 'summary': 'sr', 'table': 'ta', 'tbody': 'tb',
22 'textarea': 'ta', 'tfoot': 'tf', 'thead': 'te', 'time': 'ti', 'title': 'tl', 'track': 'tk', 'var':
    'vr', 'video': 'vd', 'wbr': 'wb'}
nout=""

24
parser = argparse.ArgumentParser('decompress')
26 parser.add_argument('compressed', help='Path to the compressed file')
26 parser.add_argument('decompressed', help='Path to the decompressed file that will be
    generated')
28 args = parser.parse_args();
28 p_comp = args.compressed
30 p_decomp = args.decompressed
30 f = open(p_comp, 'r')
32 for line in f:
    line = re.sub('</?[A-Za-z0-9]+', revrepl, line)

```

```

34     nout+=line
36 f.close()
37 f = open(p_decomp, 'w')
38 f.write(nout)
39 f.close()

```

Listing 6: Decompress source

```

1  <!DOCTYPE html>
3  <html class="client-nojs" lang="en" dir="ltr">
4  <head>
5  <meta charset="UTF-8" />
6  <title>Wikipedia, the free encyclopedia</title>
7  <script>document.documentElement.className = document.documentElement.className.
      replace( /(^\s)client-nojs(\s|$)/, " $1client-js$2" );</script>
8  <link rel="stylesheet" href="/w/load.php?debug=false&lang=en&modules=ext.uls.
      interlanguage%7Cext.visualEditor.desktopArticleTarget.noscript%7Cext.
      wikimediaBadges%7Cmediawiki.legacy.commonPrint%2Cshared%7Cmediawiki.sectionAnchor
      %7Cmediawiki.skinning.interface%7Cskins.vector.styles&only=styles&skin=
      vector" />
9  <script async="" src="/w/load.php?debug=false&lang=en&modules=startup&
      only=scripts&skin=vector"></script>
10 <meta name="ResourceLoaderDynamicStyles" content="" />
11 <link rel="stylesheet" href="/w/load.php?debug=false&lang=en&modules=ext.
      gadget.charinsert-styles&only=styles&skin=vector" />
12 <link rel="stylesheet" href="/w/load.php?debug=false&lang=en&modules=site.
      styles&only=styles&skin=vector" />
13 <meta name="generator" content="MediaWiki 1.30.0-wmf.16" />
14 <meta name="referrer" content="origin-when-cross-origin" />
15 <meta property="og:image" content="https://upload.wikimedia.org/wikipedia/commons/
      thumb/9/94/Manon_Williams_Croeso_Cymru.jpg/1200px-Manon_Williams_Croeso_Cymru.jpg
      " />
16 <link rel="alternate" href="android-app://org.wikipedia/http/en.m.wikipedia.org/wiki/
      Main_Page" />
17 <link rel="alternate" type="application/atom+xml" title="Wikipedia picture of the day
      feed" href="/w/api.php?action=featuredfeed&feed=potd&feedformat=atom" />
18 <link rel="alternate" type="application/atom+xml" title="Wikipedia featured articles
      feed" href="/w/api.php?action=featuredfeed&feed=featured&feedformat=atom"
      />
19 <link rel="alternate" type="application/atom+xml" title="Wikipedia "On this day
      ..." feed" href="/w/api.php?action=featuredfeed&feed=onthisday&
      feedformat=atom" />
20 <link rel="apple-touch-icon" href="/static/apple-touch/wikipedia.png" />
21 <link rel="shortcut icon" href="/static/favicon/wikipedia.ico" />
22 <link rel="search" type="application/opensearchdescription+xml" href="/w/
      opensearch_desc.php" title="Wikipedia (en)" />
23 <link rel="EditURI" type="application/rsd+xml" href="//en.wikipedia.org/w/api.php?
      action=rsd" />
24 <link rel="license" href="//creativecommons.org/licenses/by-sa/3.0/" />
25 <link rel="canonical" href="https://en.wikipedia.org/wiki/Main_Page" />
26 <link rel="dns-prefetch" href="//login.wikimedia.org" />
27 <link rel="dns-prefetch" href="//meta.wikimedia.org" />
28 <!--[if lt IE 9]><script src="/resources/lib/html5shiv/html5shiv.min.js"></script><![
      endif]-->
29 </head>
30 <body class="mediawiki ltr sitedir-ltr mw-hide-empty-elt ns-0 ns-subject page-
      Main_Page rootpage-Main_Page skin-vector action-view"> <div id="mw-page-base"
      class="noprint"></div>

```

Listing 7: Sample of the original Wikipedia source file

```

2 <!DOCTYPE html>

```

```

<hl class="client-nojs" lang="en" dir="ltr">
4 <hd>
<mt charset="UTF-8"/>
6 <tl>Wikipedia, the free encyclopedia</tl>
<sc>document.documentElement.className = document.documentElement.className.replace(
/(^|\s)client-nojs(\s|$)/, "$1client-js$2");</sc>
8 <ln rel="stylesheet" href="/w/load.php?debug=false&lang=en&modules=ext.uls.
interlanguage%7Cext.visualEditor.desktopArticleTarget.noscript%7Cext.
wikimediaBadges%7Cmediawiki.legacy.commonPrint%2Cshared%7Cmediawiki.sectionAnchor
%7Cmediawiki.skinning.interface%7Cskins.vector.styles&only=styles&skin=
vector"/>
<sc async="" src="/w/load.php?debug=false&lang=en&modules=startup&only=
scripts&skin=vector"></sc>
10 <mt name="ResourceLoaderDynamicStyles" content="" />
<ln rel="stylesheet" href="/w/load.php?debug=false&lang=en&modules=ext.gadget
.charinsert-styles&only=styles&skin=vector"/>
12 <ln rel="stylesheet" href="/w/load.php?debug=false&lang=en&modules=site.
styles&only=styles&skin=vector"/>
<mt name="generator" content="MediaWiki 1.30.0-wmf.16"/>
14 <mt name="referrer" content="origin-when-cross-origin"/>
<mt property="og:image" content="https://upload.wikimedia.org/wikipedia/commons/thumb
/9/94/Manon-Williams-Croeso-Cymru.jpg/1200px-Manon-Williams-Croeso-Cymru.jpg"/>
16 <ln rel="alternate" href="android-app://org.wikipedia/http/en.m.wikipedia.org/wiki/
Main_Page"/>
<ln rel="alternate" type="application/atom+xml" title="Wikipedia picture of the day
feed" href="/w/api.php?action=featuredfeed&feed=potd&feedformat=atom"/>
18 <ln rel="alternate" type="application/atom+xml" title="Wikipedia featured articles
feed" href="/w/api.php?action=featuredfeed&feed=featured&feedformat=atom"
/>
<ln rel="alternate" type="application/atom+xml" title="Wikipedia &quot;On this day
...&quot; feed" href="/w/api.php?action=featuredfeed&feed=onthisday&
feedformat=atom"/>
20 <ln rel="apple-touch-icon" href="/static/apple-touch/wikipedia.png"/>
<ln rel="shortcut icon" href="/static/favicon/wikipedia.ico"/>
22 <ln rel="search" type="application/opensearchdescription+xml" href="/w/
opensearch_desc.php" title="Wikipedia (en)"/>
<ln rel="EditURI" type="application/rsd+xml" href="//en.wikipedia.org/w/api.php?
action=rsd"/>
24 <ln rel="license" href="//creativecommons.org/licenses/by-sa/3.0/" />
<ln rel="canonical" href="https://en.wikipedia.org/wiki/Main_Page"/>
26 <ln rel="dns-prefetch" href="//login.wikimedia.org"/>
<ln rel="dns-prefetch" href="//meta.wikimedia.org"/>
28 <!--[if lt IE 9]><sc src="/resources/lib/html5shiv/html5shiv.min.js"></sc><![endif]-->
</hd>
30 <bd class="mediawiki ltr sitedir-ltr mw-hide-empty-elt ns-0 ns-subject page-Main_Page
rootpage-Main_Page skin-vector action-view"> <dv id="mw-page-base" class="
noprint"></dv>

```

Listing 8: Sample of the compressed Wikipedia source file