Assignment 1

Fall 2016
CS834 Introduction to Information Retrieval
Dr. Michael Nelson

Mathew Chaney

September 18, 2016

Contents

Listings

List of Tables

1 Question 1.1

1.1 Question

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 an- swers 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?

1.2 Resources

The search engines Google [1] and DuckDuckGo [2] were used to obtain the results.

The following search queries were issued to each:

1. professional skateboarder
2. skateboarder
3. skateboarder from korea
4. skateboarder from south korea
5. korean skateboarder thrasher
6. skateboarder song
7. daewon song

The expected relevant pages will contain information regarding the professional skateboarder named Daewon Song.

2 Question 3.7

2.1 Question

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.

2.2 Resources

With the Python programming language [3], the script **sitemapgen.py**, in Listing 3, was created to perform the necessary tasks to complete Question 3.7. The output matches the format found at Sitemaps.org [4].

The script uses the last modified time of each file to estimate the change frequency and also escapes special characters to ensure URLs are valid. The DOM is built programatically while the script traverses the file system and when this process is complete the document is printed to standard out.

Running the script on the test directory matching the structure in Listing 1 and the output is shown in Listing 2.

```
[mchaney@mchaney-l code]$ tree smgen
smgen
+---sitemapgen.py
+---testdir
    +---dir1
    |   +---testfile
    +---dir2
        +---dir3
        |   +---testfile
        +---testfile

4 directories, 4 files
```

Listing 1: test directory structure

```
[mchaney@mchaney-l smgen]$ python sitemapgen.py -p .
<?xml version="1.0" ?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
    <url>
        <loc>http://www.example.com/testdir/dir1/testfile</loc>
        <lastmod>2016-09-18</lastmod>
        <changefreq>daily</changefreq>
        <priority>0.5</priority>
    </url>
    <url>
        <loc>http://www.example.com/testdir/dir2/testfile</loc>
        <lastmod>2016-09-18</lastmod>
        <changefreq>daily</changefreq>
        <priority>0.5</priority>
    </url>
    <url>
        <loc>http://www.example.com/testdir/dir2/dir3/testfile</loc>
        <lastmod>2016-09-18</lastmod>
        <changefreq>daily</changefreq>
        <priority>0.5</priority>
    </url>
    <url>
        <loc>http://www.example.com/sitemapgen.py</loc>
        <lastmod>2016-09-19</lastmod>
        <changefreq>always</changefreq>
        <priority>0.5</priority>
    </url>
</urlset>
```

Listing 2: site map generator output

3 Appendix A

```python
import sys
import os
import argparse
import datetime
import time
import urllib

from os.path import getmtime, isdir, isfile

import xml.dom.minidom as md


def append(doc, urlset, loc, lastmod, changefreq, priority='0.5'):
    url = doc.createElement('url')
    urlset.appendChild(url)

    loc_element = doc.createElement('loc')
    loc_element.appendChild(doc.createTextNode(loc))
    url.appendChild(loc_element)

    lastmod_element = doc.createElement('lastmod')
    lastmod_element.appendChild(doc.createTextNode(lastmod))
    url.appendChild(lastmod_element)

    changefreq_element = doc.createElement('changefreq')
    changefreq_element.appendChild(doc.createTextNode(changefreq))
    url.appendChild(changefreq_element)

    priority_element = doc.createElement('priority')
    priority_element.appendChild(doc.createTextNode(priority))
    url.appendChild(priority_element)


YEAR = 3.154e+7
MONTH = 2.592e+6
WEEK = 604800.0
DAY = 86400
HOUR = 3600
MINUTE = 60

def estimate_changefreq(posixtime):
    timenow = time.time()
    delta = timenow - posixtime
    if delta > YEAR:
        return 'never'
    elif delta > MONTH:
        return 'yearly'
    elif delta > WEEK:
        return 'monthly'
    elif delta > DAY:
        return 'weekly'
    elif delta > HOUR:
        return 'daily'
    elif delta > MINUTE:
        return 'hourly'
    else:
        return 'always'


def convertdate(posixtime):
    return datetime.datetime.utcfromtimestamp(posixtime).strftime('%Y-%m-%d')


def delve(root, folder, doc, urlset):
    items = os.listdir(root + folder)
    for item in items:
        filepath = root + folder + item
        if isfile(filepath):
            loc = args.host + urllib.quote(folder + item)
            lastmodsecs = getmtime(filepath)
            lastmod = convertdate(lastmodsecs)
            changefreq = estimate_changefreq(lastmodsecs)
            append(doc, urlset, loc, lastmod, changefreq)
        elif isdir(filepath):
            delve(root, folder + item + os.sep, doc, urlset)


def test(doc, urlset):
    append(doc, urlset, 'www.google.com', '2016-09-17', 'always', '0.8')
    append(doc, urlset, 'www.duckduckgo.com', '2016-09-17', 'daily')
    print doc.toprettyxml()


if __name__ == '__main__':
    # parse arguments
    parser = argparse.ArgumentParser('site map generator')
    parser.add_argument(
        '-test',
        '-t',
        action='store_true')
    parser.add_argument(
        '--path',
        '-p',
        default='.')
    parser.add_argument(
        '--host',
        default='http://www.example.com/')
    args = parser.parse_args()

    # create a document
    doc = md.Document()
    urlset = doc.createElement('urlset')
    urlset.setAttribute('xmlns', 'http://www.sitemaps.org/schemas/sitemap/0.9')
    doc.appendChild(urlset)

    # if desired, perform simple test and return
    if args.test:
        test(doc, urlset)
        sys.exit(0)

    # parse path from args
    path = args.path
    if path[len(path)-1] != os.sep:
        path = path + os.sep

    # iterate over all items in doc
    delve(path, '', doc, urlset)

    print doc.toprettyxml()
```

Listing 3: sitemapgen.py

4 References

[1] Google. Available at: http://www.google.com. Accessed: 2016/09/17.

[2] DuckDuckGo. Available at: http://www.duckduckgo.com. Accessed: 2016/09/17.

[3] The Python Programming Language. Available at: https://www.python.org/. Accessed: 2016/09/17.

[4] Sitemaps XML format. Available at: http://www.sitemaps.org/protocol.html. Accessed: 2016/09/17.