INLS 490-154W: Information Retrieval Systems Design and Implementation. Fall 2009.

11. Web crawling

Chirag Shah*
School of Information & Library Science (SILS)
UNC Chapel Hill NC 27599
chirag@unc.edu

1 Introduction

This time we will look at collecting data off the web. A common way of doing so is by crawling the web, websites, or portions of them. We will focus our discussion on collecting data using 'wget', and then indexing and preparing it for searching. We will also introduce Heritrix crawler and look at a way to harvest data from YouTube.

2 wget

GNU 'wget'[1] is a utility that allows one to capture files off the web using HTTP, HTTPS, and FTP. This utility is usually available on UNIX platform and runs in non-interactive manner. This is very suitable for running background crawls and cron jobs.

To use 'wget', run it on the command line with a URL as an argument. For instance,

```
> wget http://www.cia.gov
```

'wget' provides a host of options, which can be seen by typing `wget --help`. For instance, `-r` allows one to recursively download from a site. The default depth for this recursion is 5, which means issuing `wget -r http://www.cia.gov` will download pages from up to five levels down from the CIA website. It is important to note that this crawl happens in breadth first manner, which means all the pages of a given level are collected before proceeding to the next level. If you want to change the default number of levels, use option `-l`. For example,

```
> wget -r -l 3 http://www.cia.gov
```

By default 'wget' crawls pages from the seed site only. However, often webpages point to other pages or sites outside of the current site. To get those pages too, use `-H` option, which

[1] http://www.gnu.org/software/wget/

tells 'wget' to span to any host. If you want to restrict these outside hosts to only certain hosts, you can specify them using `-D` option. For instance,
`wget -rH -D wikipedia.org http://answers.yahoo.com`
allows one to recursively crawl http://answers.yahoo.com, also following the links that point to host wikipedia.org.

3 Indexing and searching the crawled collection

Now that we have crawled a collection, it is time to index it and make it available for searching. Indri's 'buildindex' allows us to parse and index various file formats such as html, PDF, text, Word, and PowerPoint. For the CIA collection that we have downloaded, we will use html and PDF parsers. A sample parameter file for 'buildindex' is shown in Code 1.

Code 1: Parameter file for 'buildindex' for Indexing the crawled collection

```
<parameters>
<index>ciaIndex</index>
<indexType>indri</indexType>
<corpus>
    <path>./www.cia.gov/library/publications/the-world-factbook/</path>
    <class>html</class>
    <class>pdf</class>
</corpus>
</parameters>
```

Once the index it built, we need to get a search daemon running that can listen to search requests coming on a server. A parameter file for 'indrid' is given in Code 2.

Code 2: Parameter file for 'indrid'

```
<parameters>
    <index>ciaIndex/</index>
    <port>12345</port>
</parameters>
```

Finally, we are ready to start issuing queries. Indri's 'runquery' application lets us do this. Following is a parameter file for 'runquery'. The details of these parameters are discussed before.

Code 3: Parameter file for running a query on the crawled index

```
<parameters>
    <server>localhost:12345</server>
    <count>20</count>
    <trecFormat>false</trecFormat>
    <query>#combine(independence day)</query>
    <printSnippets>true</printSnippets>
</parameters>
```

4 Heritrix

Heritrix[2] is Internet Archive's web crawler. Unlike 'wget', Heritrix is designed to provide rich interaction and a sophisticated interface that allows one to create, configure, and manipulate crawl jobs. This crawler is implemented with a client-server architecture, where a server part runs in the background doing the crawls, and a client part connects to the server for creating and manipulating crawl jobs. Heritrix is based on Java and uses JRE to run the server component. The detailed explanation of Heritrix's configuration and functionality is beyond the scope of this document, but we will outline the steps to get Heritrix started below.

1. Download and unzip Heritrix.
2. Set `export HERITRIX_HOME` to the path where Heritrix is unzipped.
3. Set `export JAVA_HOME` to where Java is installed.
4. Set `export JMX_PORT` to the port number on which Heritrix server should listen to the client requests.
5. Go to 'bin' directory and start Heritrix with `heritrix --admin=login:password`.

Once the server component starts, you can access it by launching your web browser and entering `http://localhost:1234`, where `1234` is set with `JMX_PORT`. It is also possible to access the Heritrix server remotely, in which case the options for starting the server will be different.

On the web interface, you can login as an admin with the login name and password that you entered while starting the server. Once logged in, you can create/edit a profile, and start a job using that profile and seed(s). A running job with Heritrix looks something like what is shown in Figure 1.

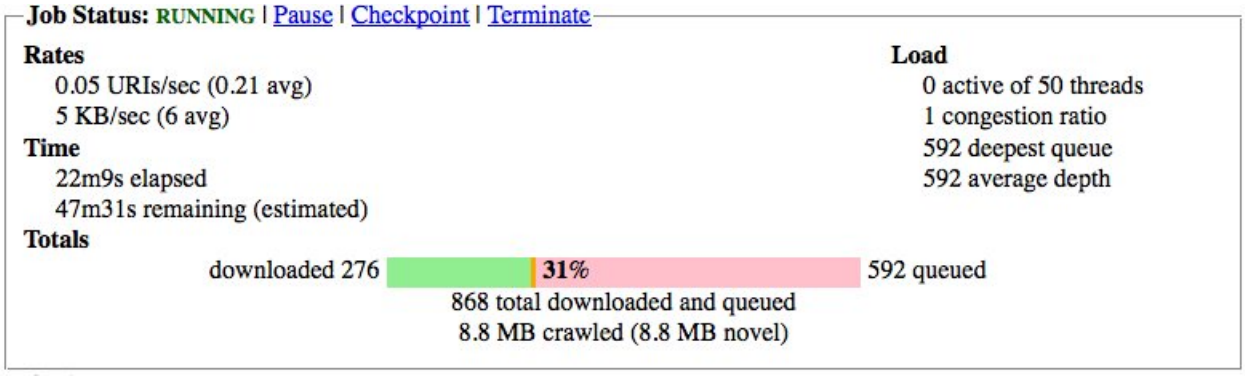

Figure 1: Running a job with Heritrix

Heritrix generates a very detailed report of the errors encountered during the crawl as well as many statistics. A snapshot of a crawl report is shown in Figure 2.

[2]http://crawler.archive.org/

Job name: CIA
Status: Finished - Ended by operator
Time: 3h49m29s216ms
Processed docs/sec: 0.13
Processed KB/sec: 5
Total data written: 71150660 (68 MB)

HTTP

Status code	Documents
HTTP-200-Success-OK	1833 (99.8%)
HTTP-404-ClientErr-Not Found	4 (0.2%)
Total:	1837

MIME type	Documents	Data
text/html	978 (53.2%)	60 MB
image/gif	761 (41.4%)	5.7 MB
image/jpeg	97 (5.3%)	2.0 MB
text/css	1 (0.1%)	2.2 KB
Total	1837	68 MB

Hosts	Documents	Data
www.cia.gov	1837 (100%)	68 MB
Total	1837	68 MB

TLD	Hosts	Documents	Data
gov	1 (100%)	1837 (100%)	68 MB
Total	1	1837	68 MB

Figure 2: Report of a crawl job with Heritrix

5 Harvesting YouTube

Several Web 2.0 services make it easier to crawl their websites for the data by providing API-based access. Here we will see how we can issue such requests to YouTube and collect data related to a video. For instance, sending `http://gdata.youtube.com/feeds/api/videos/H5h95s0OuEg` uses Google Data APIs and throws back a RSS response.

The RSS response will be in XML format, which we have seen how to parse. To aid in this parsing, you can use a supplied function called 'parseVideoEntry' in a PHP file. Include this file in the main file, which looks like the one in Code 4.

Code 4: Harvesting YouTube data

```
<?php
    require_once("parseRSS.php");

    $feedURL = "http://gdata.youtube.com/feeds/api/videos/H5h95s0OuEg";
    $entry = simplexml_load_file($feedURL);
    $video = parseVideoEntry($entry);

    $title = $video->title;
    $description = $video->description;
    $category = $video->category;
```

```
    $keywords = $video->keywords;
    $views = $video->viewCount;
    echo "$title\n$description\n";
    echo "$category\n$keywords\n$views\n";
?>
```

Running this script on the command line will issue the request to YouTube acquiring the information about a video (identified with a unique ID of 11 characters following '/videos/' in the URL). The response is parsed and certain entities such as title and description are extracted, and printed.

6 Summary

Given what we have seen so far, there are many ways in which we can combine different components and create a variety of applications. For instance, we can use 'wget' to crawl a blog site, use 'harvestlinks' application of Lemur to extract links from the collected pages, check if any of those URLs are pointing to a YouTube video, extract YouTube video ID if they do, and then collect various attributes of those videos. We can store those attributes in a database since that is structured information (fields are known) for easy access and presentation. We can index the blog site (unstructured textual information) and have it ready for searching in full text.

- Crawling refers to collecting web pages or sites for storage (e.g., site mirroring), archival (e.g., Internet Archive), and index (e.g., search engines).
- Search engines constantly crawl the web to get the latest snapshot of it.
- One needs to be careful about running crawl jobs since without proper control, it could overload the various resources, including the network.