Homework: Sample Web Crawl
Draft: February 8, 2016

---

1. Objective

In this assignment, you will work with a simple web crawler to measure and study the characteristics of selected USC school websites.

2. Preliminaries

To begin we will make use of an existing open source Java web crawler called crawler4j. This crawler is built upon the open source crawler4j library which is located on github. For complete details on downloading and compiling see
https://github.com/yasserg/crawler4j

3. Crawling

Your task is to configure and compile the crawler and then have it crawl one of the USC schools' websites. In the interest of distributing the load evenly and not overloading the school servers, we have pre-assigned the school to be crawled according to your USC ID number, given in the table below.

The maximum pages to fetch can be set in crawler4j and it should be set to 5000 to ensure a reasonable execution time for this exercise. Also, maximum depth should be set to 5 to ensure that we limit the crawling.

You should crawl only the school website assigned to you, and your crawler should be configured so that it does not visit pages outside of the given school website!

USC ID ends with	School to crawl	Root URL
01~14	Architecture	http://arch.usc.edu/
15~26	Cinematic Arts	http://cinema.usc.edu/
27~38	Dornsife (College)	http://dornsife.usc.edu/
39~49	Gould (Law)	http://gould.usc.edu/
50~60	Keck (Medicine)	http://keck.usc.edu/
61~71	Marshall (Business)	http://www.marshall.usc.edu/
72 ~ 78	Viterbi (Engineering)	http://viterbi.usc.edu/
79~88	Price (Public Policy)	http://priceschool.usc.edu/
89~00	Social Work	http://sowkweb.usc.edu/

Limit your crawler so it only downloads HTML, doc and pdf files. These files should be retained for use in exercises 3 and 4.

4. Collecting Statistics

Your first task is to enhance the crawler so it collects information about:

1. *the URLs it attempts to fetch*, a two column spreadsheet, column 1 containing the URL and column 2 containing the HTTP status code received; name the file **fetch.csv.** The number of rows should be close to 5,000 as that is our pre-set limit.
2. *the files it successfully downloads*, a four column spreadsheet, column 1 containing the URLs successfully downloaded, column 2 containing the size of the downloaded file, column 3 containing the # of outlinks found, and column 4 containing the resulting content-type; name the file **visit.csv;** clearly the number of rows will be less than the number of rows in fetch.csv
3. *all of the URLs that were discovered* and processed in some way; a two column spreadsheet where column 1 contains the encountered URL and column two an indicator of whether the URL a. resides in the school (**OK**), b. resides within USC, but outside of the school (USC), and c. points outside of USC (outUSC). Name the file **urls.csv**. This file will be much larger than 5,000 rows as it will have numerous repeated URLs

Note1: you should modify the crawler so it outputs the above data into three separate csv files; we may use them for processing later;
Note2: you should save all of the downloaded web pages, etc. for processing in the next exercise.

Based on the information recorded by the crawler in its output files, you are to collate the following statistics for a crawl of your designated school website:

- Fetch statistics:
  - # fetches attempted:
    The total number of URLs that the crawler attempted to fetch. This is usually equal to the `MAXPAGES` setting if the crawler reached that limit; less if the website is smaller than that.
  - # fetches succeeded:
    The number of URLs that were successfully downloaded in their entirety, i.e. returning a HTTP status code of 2XX.
  - # fetches failed or aborted:
    The number of fetches that failed for whatever reason, including, but not limited to: HTTP redirections (3XX), client errors (4XX), server errors (5XX) and other network-related errors.[1]

- Outgoing URLs: statistics about URLs extracted from visited HTML pages
  - Total URLs extracted:
    The grand total number of URLs extracted from all visited pages
  - # unique URLs extracted:
    The number of *unique* URLs encountered by the crawler
  - # unique URLs within School:
    The number of *unique* URLs encountered that are associated with the school website, i.e. the URL begins with the given root URL of the school.

[1] Based purely on the success/failure of the fetching process. Do not include errors caused by difficulty in parsing content *after* it has already been successfully downloaded.

- # unique USC URLs outside School:
  The number of *unique* usc.edu URLs encountered that were *not* from the school website.
- # unique URLs outside USC:
  The number of all other *unique* URLs encountered

- Status codes: number of times various HTTP status codes were encountered during crawling, including (but not limited to): 200, 301, 401, 402, 404, etc.
- File sizes: statistics about file sizes of visited URLs – the number of files in each size range.
  - 1KB = 1024B; 1MB = 1024KB

These statistics should be collated and submitted as a plain text file whose name is CrawlReport.txt, following the format given in Appendix A at the end of this document.

Computing Page Rank

For Assignment #3 you will need to compute the Page Rank of each page you download in this assignment. To do that you will need to keep a record of all URLs contained in a given page. Therefore you should also generate a csv file that includes every successfully downloaded html file, in column 1, and the outgoing URLs that were contained in the page, in subsequent columns 2, 3, 4, etc. Name the file **pagerankdata.csv.** You need not submit this file with your assignment #2, but you will need it when you get to assignment #3.

Make sure you understand the crawler code and outputs before you commence collating these statistics. All the information that you are required to collect can be derived by processing the crawler output.

5. FAQ

Q: What is the exact definition of a URL being within a specific school domain, such as `viterbi.usc.edu` ?

A: Anything that starts with `http://viterbi.usc.edu`

Q: For the purposes of counting unique URLs, how to handle URLs that differ only in the query string? For example: `http://usc.edu/page?q=0` and `http://usc.edu/page?q=1`

A: These can be treated as different URLs.

Q: URL case sensitivity: are these the same, or different URLs?
`http://usc.edu/foo` and `http://usc.edu/FOO`

A: The path component of a URL is considered to be case-sensitive, so the crawler behavior is correct according to RFC3986. Therefore, these are different URLs.

The page served may be the same because:

- that particular web server implementation treats path as case-insensitive (some server implementations do this, especially windows-based implementations)
- the web server implementation treats path as case-sensitive, but aliasing or redirect is being used.

This is one of the reasons why deduplication is necessary in practice.

Q: Attempting to compile the crawler results in syntax errors.

A: Make sure that you have included crawler4j as well as all its dependencies.

Also check your Java version; the code includes more recent Java constructs such as the typed collection `List<String>` which requires at least Java 1.5.0.

Q: I get the following warnings when trying to run the crawler:

```
log4j:WARN No appenders could be found for logger
log4j:WARN Please initialize the log4j system properly.
```

A: You failed to include the `log4j.properties` file that comes with crawler4j.

Q: On Windows, I am encountering the error: `Exception_Access_Violation`

A: This is a Java issue. See: http://java.com/en/download/help/exception_access.xml

Q: I am encountering multiple instances of this info message:

```
INFO [Crawler 1] I/O exception (org.apache.http.NoHttpResponseException)
caught when processing request: The target server failed to respond
INFO [Crawler 1] Retrying request
```

A: If you're working off an unsteady wireless link, you may be battling network issues such as packet losses – try to use a better connection. If not, the web server may be struggling to keep up with the frequency of your requests.

As indicated by the info message, the crawler will retry the fetch, so a few isolated occurrences of this message are not an issue. However, if the problem repeats persistently, the situation is not likely to improve if you continue hammering the server at the same frequency. Try giving the server more room to breathe:

```
/*
 * Be polite: Make sure that we don't send more than
 * 1 request per second (1000 milliseconds between requests).
 */
config.setPolitenessDelay(2500);        // CHANGE THIS TO 2500 OR HIGHER.
```

Q: The crawler seems to choke on some of the downloaded files, for example:

```
java.lang.StringIndexOutOfBoundsException: String index out of range: -2
```

```
java.lang.NullPointerException: charsetName
```

A: Safely ignore those. We are using a fairly simple, rudimentary crawler and it is not necessarily robust enough to handle all the possible quirks of heavy-duty crawling and parsing. These

problems are few in number (compared to the entire crawl size), and for this exercise we're okay with it as long as it skips the few problem cases and keeps crawling everything else, and terminates properly – as opposed to exiting with fatal errors.

Q: While running the crawler, you may get the following error:
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.

A: It's described here - https://groups.google.com/forum/#!topic/crawler4j/urroQ5BRsWM. A simple fix is to add more JARs that can be found contained as a ,zip/.tar.gz in this link - http://logback.qos.ch/download.html. Adding all these external JARs to the project in the same way as the crawler-4j JAR will make the crawler display logs now.

6. Submission Instructions

- Save your statistics report as a plain text file and name it:

  `CrawlReport.txt`

- Also include the output files generated from your crawler run, using the names below:
  - `fetch.csv`
  - `visit.csv`
  - `urls.csv`
- Do **not** submit Java code or compiled programs; it is not required.
- Compress all of the above into a single zip archive and name it:

  `crawl.zip`

  Use only standard zip format. Do **NOT** use other formats such as zipx, rar, ace, etc. The zip file will contain:

  CrawlReport.txt, (the statistics file)
  fetch.csv
  visit.csv
  urls.csv

- To submit your file electronically to the csci572 account enter the following command from your UNIX prompt:

```
$ submit -user csci572 -tag hw2 crawl.zip
```

Appendix A

Use the following format to tabulate the statistics that you collated based on the crawler outputs.

Note: The status codes and content types shown are only a sample. The status codes and content types that you encounter may vary, and should all be listed and reflected in your report. Do **NOT** lump everything else that is not in this sample under an "Other" heading. You may, however, exclude status codes and types for which you have a count of zero.

Warning: The files you submit will be used for automated grading. Failure to follow the format strictly may result in inability to grade your submission. Use plain text files only (*.txt). Do **NOT** use rich text format (RTF).

`CrawlReport.txt`

```
Name: Tommy Trojan
USC ID: 1234567890
School crawled: Architecture

Fetch Statistics
================
# fetches attempted:
# fetches succeeded:
```

```
# fetches aborted:
# fetches failed:

Outgoing URLs:
==============
Total URLs extracted:
# unique URLs extracted:
# unique URLs within School:
# unique USC URLs outside School:
# unique URLs outside USC:

Status Codes:
=============
200 OK:
301 Moved Permanently:
401 Unauthorized:
403 Forbidden:
404 Not Found:

File Sizes:
===========
< 1KB:
1KB ~ <10KB:
10KB ~ <100KB:
100KB ~ <1MB:
>= 1MB:

Content Types:
==============
text/html:
image/gif:
image/jpeg:
image/png:
application/pdf:
```