Homework: Web Crawling

1. Objective

In this assignment, you will work with a simple web crawler to measure aspects of a crawl, study the characteristics of the crawl, download web pages from the crawl and gather webpage metadata, all from pre-selected news 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
Also see the document "Instructions for Installing Eclipse and Crawler4j" located on the Assignments web page for help.

3. Crawling

Your task is to configure and compile the crawler and then have it crawl a news website. In the interest of distributing the load evenly and not overloading the news servers, we have pre-assigned the news sites 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 **20,000** to ensure a reasonable execution time for this exercise. Also, maximum depth should be set to **16** to ensure that we limit the crawling.

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

USC ID ends with	News Sites to Crawl	Root URL
01~20	Wall_Street_Journal	https://www.wsj.com/
21~40	USA_Today	https://www.usatoday.com/
41~60	Boston_Globe	https://www.bostonglobe.com/
61~80	New_York_Daily_News	http://www.nydailynews.com/
81~00	Washington_Post	https://www.washingtonpost.com/

Limit your crawler so it only visits HTML, doc, pdf and different image format URLs and record the meta data for those file types

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_NewsSite.csv** (where the name "NewsSite" is replaced by the news website name in the table above that you are crawling). The number of rows should be no more than 20,000 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_NewsSite.csv**; clearly the number of rows will be less than the number of rows in fetch_NewsSite.csv
3. *all of the URLs (including repeats) 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 website (**OK**), or **b.** points outside of the website (**N_OK**). (A file points out of the website if its URL does not start with the initial domain name, e.g. when crawling USA Today news website all inside URLs must start with www.usatoday.com.) Name the file **urls_NewsSite.csv**. This file will be much larger than fetch_*.csv and visit_*.csv.

Note1: you should modify the crawler so it outputs the above data into three separate csv files; you may use them for processing later;
Note2: all uses of NewsSite above should be replaced by the name given in the table on page 1.

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 news 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 (including repeats) from all visited pages

[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 URLs extracted:
  The number of *unique* URLs encountered by the crawler
- # unique URLs within your news website:
  The number of *unique* URLs encountered that are associated with the news website, i.e. the URL begins with the given root URL of the news website, but the remainder of the URL is distinct
- # unique URLs outside the news website:
  The number of *unique* URLs encountered that were *not* from the news website.

- 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 (See Appendix A).
  - 1KB = 1024B; 1MB = 1024KB
- Content Type: a list of the different content-types encountered

These statistics should be collated and submitted as a plain text file whose name is CrawlReport_domain.txt, following the format given in Appendix A at the end of this document. Make sure you understand the crawler code and required output before you commence collating these statistics.

For efficient crawling it is a good idea to have multiple crawling threads. crawler4j supports multi-threading and our examples show setting the number of crawlers to seven (see the line in the code `int numberOfCrawlers = 7;`). However, if you do a naive implementation the threads will trample on each other when outputting to your statistics collection files. Therefore you need to be a bit smarter about how to collect the statistics, and crawler4j documentation has a good example of how to do this:

https://github.com/yasserg/crawler4j/tree/master/src/test/java/edu/uci/ics/crawler4j/examples/localdata

All the information that you are required to collect can be derived by processing the crawler output.

5. FAQ

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

A: These can be treated as different URLs.

Q: URL case sensitivity: are these the same, or different URLs?
`https://www.usatoday.com/foo` and `https://www.usatoday.com/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);
```

```
/*
 * READ ROBOTS.TXT of the website - Crawl-Delay: 10
 * Multiply that value by 1000 for millisecond value
 */
```

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: The problem is 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.

Q: What should we do with URL if it contains comma ?
A: Replace the comma with "-" or "_", so that it doesn't throw an error.

Q: What's the difference between aborted fetches and failed fetches?
A: failed: Can be due to HTTP errors and other network related errors
aborted: Client decided to stop the fetching. (ex: Taking too much time to fetch)

Q: For some reason my crawler attempts 19,999 fetches, even though max pages is set to 20,000, does this matter?
A: It can be possible because 20,000 is the limit that you will try to fetch (it may contain successful status code like 200 and other like 301). But the visit.csv will contain only the URL's for which you are able to successfully download the files.

Q: How to differentiate fetched pages and downloaded pages?
A: In this assignment we do not ask you to save any of the downloaded files to the disk. Visiting a page means crawler4j processing a page (it will parse the page and extract relevant information like outgoing URLs). That means all visited pages are downloaded.

You must make sure that your crawler crawls both http and https pages of the given domain

Q: For third CSV, should the discovered URLs include redirect URLs?
A: YES, if the redirect URL is the one that gets status code 300, then the url that redirects the URL to point to will be added to the scheduler of the crawler and waits to be visited.

Comment:
https://github.com/yasserg/crawler4j/blob/master/src/test/java/edu/uci/ics/crawler4j/examples/basic/BasicCrawler.java
has details on regular expressions that you need to take care.

Comment: Since many newspaper websites dump images and other types of media on CDN, your crawl may only encounter html files. That is fine.

Comment: File types css,js,json and others should **not** be visited.

Comment: Some sites may have less than the 20,000 pages, but as long as the formula matches. i,e
fetches attempted = # fetches succeeded + # fetches aborted + # fetches failed
your homework is ok. However, the variation should not be more than 10% away from the limit as it is an indication that something is wrong.

Comment: the homework description states that you only need to consider HTML, doc, pdf and different image format URLs . But you should also consider URL's with no extension as they may return a file of one of the above types.

Comment: The distinction between failed and aborted web pages.
failed: Can be due to content not found, HTTP errors or other network related errors

aborted: the client (the crawler) decided to stop the fetching. (ex: Taking too much time to fetch).

Q: In the visit_NewsSite.csv, do we also need to chop "charset=utf-8" from content-type? Or just chop "charset=urf-8" in the report?
A: You can chop Encoding part(charset=urf-8) in all places.

Q: REGARDING STATISTICS
A: #unique URLs extracted = #unique URLs within + #unique URLs outside
#total urls extracted is the sum of #outgoing links.
#total urls extracted is the sum of all values in column 3 of visit.csv

Q: How to handle pages with NO Extension
A: Use getContentType() and dont rely just on extension.
Note #1: Extracted urls do not have to be added to visit queue. Some of them which satisfy a requirement (e.g : content type, domain, not duplicate) will be added to visit queue. But others will be dumped by crawler.

However, as long as the grading guideline is satisfied, we will not deduct points.

Note#2: : 303 could be considered aborted. 404 could be considered failed.
To summarize: we consider a request to be aborted if the crawler decides to terminate that request. Client side timeout is an example. Requests can fail due to reasons like content not found, server errors, etc.

Note#3: 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

Q: Regarding the content type in visit_NewsSite.csv, should we display "text/html;charset=UTF-8" or chop out the encoding and write "text/html" in the excel sheet ?
A: ONLY TEXT/HTML, ignore rest.

Q: Should we limit the URLs that the crawler attempted to fetch within the news domain? e.g. if we encounter www.facebook.com, we should skip fetching by adding constraints in "shouldVisit()"? But do we need to include it in urls_NewsSite.csv?
A: Yes, you need to include every encountered url in urls_NewsSite.csv.

Q: All 3xx,4xx, 5xx should be considered as aborted?
A: YES

Q: ”cookie” domains are considered as original newsite domain ?
A: NO

6. Submission Instructions

- Save your statistics report as a plain text file and name it based on the domain names assigned below:

USC ID ends with	Site
01~20	CrawlReport_Wall_Street_Journal.txt
21~40	CrawlReport_USA_Today.txt
41~60	CrawlReport_Boston_Globe.txt
61~80	CrawlReport_New_York_Daily_News.txt
81~00	CrawlReport_Washington_Post.txt

- Also include the output files generated from your crawler run, using the extensions as shown above:
  - `fetch_NewsSite.csv`
  - `visit_NewsSite.csv`
- Do NOT include the output files
  - `urls_NewsSite.csv`

  where _NewSite should be replaced by the name from the table above.
- 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. For example the zip file might contain the following three files:

  1. CrawlReport_Wall_Street_Journal.txt, (the statistics file)
  2. fetch_Wall_Street_Journal.csv
  3. visit_Wall_Street_Journal.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.

`CrawlReport_NewsSite.txt`

```
Name: Tommy Trojan
USC ID: 1234567890
News site crawled: wsj.com

Fetch Statistics
================
# fetches attempted:
# fetches succeeded:
# fetches aborted:
# fetches failed:

Outgoing URLs:
==============
Total URLs extracted:
```

```
# unique URLs extracted:
# unique URLs within News Site:
# unique URLs outside News Site:

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:
```