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.
Note: You can use any IDE of your choice. But we have provided installation instructions for Eclipse IDE only

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	NewsSite Name	Root URL
01~20	NY Times	nytimes	https://www.nytimes.com
21~40	Wall Street Journal	wsj	https://www.wsj.com
41~60	Fox News	foxnews	https://www.foxnews.com
61~80	USA Today	usatoday	https://www.usatoday.com
81~00	Los Angeles Times	latimes	https://www.latimes.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 primary 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/HTTPS 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. Column names for this file can be URL and Status
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 (in Bytes, or you can choose your own preferred unit (bytes,kb,mb)), 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 host/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.
   For example for New York Times- the URL http://www.nytimes.com/section/sports and the URL http://nytimes.com/section/sports are both considered as residing in the same website whereas the following URL is ***not*** considered to be in the same website, http://store.nytimes.com/

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

Based on the information recorded by the crawler in the output files above, 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
  - # 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_NewsSite.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. ***You are required to use multiple threads in this exercise.*** 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. See both of the following links for details:

https://github.com/yasserg/crawler4j/blob/master/crawler4j-examples/crawler4j-examples-base/src/test/java/edu/uci/ics/crawler4j/examples/multiple/MultipleCrawlerController.java
and
https://github.com/yasserg/crawler4j/blob/master/crawler4j-examples/crawler4j-examples-base/src/test/java/edu/uci/ics/crawler4j/examples/localdata/LocalDataCollectorCrawler.java

[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.

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.nytimes.com/page?q=0` and `https://www.nytimes.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.nytimes.com/foo` and `https://www.nytimes.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. Download slf4j-simple-1.7.25.jar from https://repo1.maven.org/maven2/org/slf4j/slf4j-simple/1.7.25/ add this as an external JAR 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: Should the number of 200 codes in the fetch.csv file have to exactly match with the number of records in the visit.csv?

A: No, but it should be close, like within 2,000 of 20,000. If not then you may be filtering too much.

Q: "CrawlConfig cannot be resolved to a type" ?
A: import edu.uci.ics.crawler4j.crawler.CrawlConfig;

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)
You may sum up both the values and provide the combined result in the write up.

Q: For some reason my crawler attempts 19,999 fetches, even though max pages is set to 20,000, does this matter?
A: No, it doesn't matter. It can occur 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: How much time should it approximately take to crawl a website using *n* crawlers?
A: (i) Depends on your parameters set for the crawler
(ii) Depends on the politeness you set in the crawler program
Therefore, it can vary for everyone

Q: For the third CSV file, urls_NewSite.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.

Q: When the URL ends with "/", what needs to be done?
A: You should filter using content type. Please have a peek into Crawler 4j code located at https://github.com/yasserg/crawler4j/tree/master/crawler4j/src/main/java/edu/uci/ics/crawler4j/crawler
You will get a hint on how to know the content type of the page, even if the extension is not explicitly mentioned in the URL

Q: Eclipse keeps crashing after a few minutes of running my code. But when I reduce the no of pages to fetch, it works fine.
A: Increase heap size for eclipse using this.
https://wiki.eclipse.org/FAQ_How_do_I_increase_the_heap_size_available_to_Eclipse%3F

Q: What if a URL has an unknown extension?
A: Please check the content type of the page if it has an unknown extension

Q: Why do some links return True in shouldVisit() but cannot be visited by Visit()?
A: shouldVisit() function is used to calculate whether the page should be visited or not. It may or may not be a visitable page.

For example - If you are crawling the site http://viterbi.usc.edu/, the page http://viterbi.usc.edu/mySamplePage.html should be visited. but this page may return a 404 Not Found Error or it may be redirected to some other site like http://mysamplesite.com. In this case, shouldVisit() function would return true because the page should be visited but visit() will not be called because the page cannot be visited.
Comment:

https://github.com/yasserg/crawler4j/blob/13afe9460c1e72ffebe1b14c260ae0fd2eaa5f5e/crawler4j-examples/crawler4j-examples-base/src/test/java/edu/uci/ics/crawler4j/examples/multiple/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. E.g. you can add .json to your pattern filter. If the extension does not appear, use
`!page.getContentType().contains("application.json")`

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.

> **Scenario:**
> My visit.csv file has about 15 URLs lesser than the number of URLs with status code 200. It is fine if the difference is less than 10%.

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).
You may sum up both the values and provide the combined result in the write up.

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()` in visit() and don't rely just on extension. If the content type returned is not one of the required content types for the assignment, you should ignore it for any calculation of the statistics. This will probably result in more rows in visit.csv, but it's acceptable according to the grading guidelines.

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 the 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.

Note#4: Consider fetches failed and aborted as same similar to as mentioned in Note#3

Note#5: Hint on crawling pages other than html

Look for how to turn ON the Binary Content in Crawling in crawler4j. Make sure you are not just crawling the html parsed data and not the binary data which includes file types other than html.

Search on the internet on how to crawl binary data and I am sure you will get something on how to parse pages other than html types.

There will be pages other than html in almost every news site so please make sure you crawl them properly.

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: Are "cookie" domains considered as an original newsite domain ?
A: NO, they should not be included as part of the newsite you are crawling. For details see https://web.archive.org/web/20200418163316/https://www.mxsasha.eu/blog/2014/03/04/definitive-guide-to-cookie-domains/

Q. More about statistics
A: visit.csv will contain the urls which are succeeded i.e. 200 status code with known/ allowed content types. Fetch.csv will include all the urls which are been attempted to fetch i.e. with all the status codes.
fetch.csv entries will be = visit.csv entries (with 2xx status codes) + entries with status codes other than 2XX
visit.csv = entries with 2XX status codes.

(**Note**:-> fetch.csv should have urls from news site domain only)

Q. do we need to check content-type for all the extracted URLs, i.e. url.csv or just for visited URLs, e.g. those in visit.csv?
A. only those in visit_NewsSite.csv

6. Submission Instructions

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

USC ID ends with	Site
01~20	CrawlReport_nytimes.txt
21~40	CrawlReport_wsj.txt
41~60	CrawlReport_foxnews.txt
61~80	CrawlReport_usatoday.txt
81~00	CrawlReport_latimes.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_nytimes.txt, (the statistics file)
  2. fetch_nytimes.csv
  3. visit_nytimes.csv

- Please upload your homework to your Google Drive CSCI572 folder, in the subfolder named hw2

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. Also, note the use of multiple threads. ***You are required to use multiple threads in this exercise***.

`CrawlReport_NewsSite.txt`

```
Name: Tommy Trojan
USC ID: 1234567890
News site crawled: nytimes.com
Number of threads: 7

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

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:
```