**CSI 5V93-01 Spring 2015**

**Programming Homework (Part 2)**

Due: 3/19/15 (Thu) 11:59pm.

**Part 2: Mini tf-idf based focus web-crawler**

The goal of this part of the project is to develop a mini web-crawler for you to do a focus web crawl.

The idea is that you will have a test set of documents as basis. You first (using your program in part 1) calculate the tf-idf vector for each document. Then when you crawl the web, you generate the document vector for each web page downloaded, and compare it with the document set. If enough documents in the test set have cosine similarity above a certain threshold, you accept the document and store it. Otherwise, you ignore the document.

As mentioned in class, your web crawler should contains the following parts

* Scheduler: Maintain a list of URLs to be downloaded, and select the next one to download. If you implement it as a method, it should output the next URL to be downloaded
* Fetcher/Downloader: Receive the URL from the scheduler and actually fetch the page. If you implement it as a method, it should output the content downloaded (or the fact the page cannot be downloaded.
* Post-processing/Storage: Receive the content that is downloaded and decide whether to store the page, and also output the links’ URL to the scheduler.

Obviously, you need to maintain other data structures to be used.

The rest of the outline describe the various component and requirement of the system

***Restriction, Initialization and parameters***

Your crawler should crawl only web pages in the “.edu” domain. It should only save either HTML files or Text file. (Either recognized by the file extension, or by examining meta-tags of the download file).

In terms of politeness, your crawler should follow the robot exclusion protocol, and you should have a gap of at least 15 seconds between accesses web pages from the same domain. Also you should NOT open multiple connection to the same web site at the same time.

Your program should read in a file call “5V93crawler\_parameters.txt” that contains the parameters for the crawler. Each line is a parameter:

* The first line contains a number that denotes the maximum number of pages to be stored. If the number is <= 0, the program exits immediately with an error message
* The second line contains 2 terms: the first is the name of the directory where the downloaded file is to be stored. The second contains instruction whether to erase existing results and start from stretch. If the second word is “O”, then you should remove everything from that directory and start over, otherwise, you should just append to the data.
* The third line is the number of maximum processes/threads to use for fetching web pages. If the number is <= 0, the program exits immediately with an error message.
* The fourth to sixth lines contains 3 URLs which are to be used as the seed URLs.
* The seventh line contains two numbers. The first number denote the cosine similarity threshold, and the second number denotes the portion (between 0-1) of the document that have cosine similarity with the threshold in order for the crawler to accept the document and save it (and follow its links). Any numbers that are out of range will cause the program to exit immediately with an error message
* The eighth line contains two numbers. The first number denote the cosine similarity threshold, and the second number denotes the portion (between 0-1) of the document that have cosine similarity with the threshold in order for the crawler to follow its links (but NOT save the document). Any numbers that are out of range will cause the program to exit immediately with an error message.

You should also use your “5V93query.txt” file from part 1 to load information about your test files and other related info.

***Scheduler***

You should maintain a list of web pages to be downloaded, and your scheduler should pick an URL to be downloaded next. You have freedom to decide which download should occur next, bear in mind the restriction, and also to maintain efficiency.

***Fetcher***

You should allow yourself to have multiple processers/threads. Each thread is tasked to download a single web page.

In order to make life easier, you are encouraged to use the text based web browser lynx to download individual pages, extract links and convert HTML to text. The lynx program can be downloaded from the following web site: http://lynx.isc.org/

To download a web page, you should use the following command:

*lynx -dump –nolist <URL> > <output file>*

It will download the page, convert it to text, and redirect the result to an output file. On the other hand, if you only need the links, you should run the following:

*lynx -dump –listonly <URL> > <output file>*

You can call lynx from your program using the following ways:

* C/C++: using the system() system call
* Java: using the Runtime class
* Perl: using the system() functions or backtits (`)
* Python: using call()

You are welcomed to use other means to download individual web pages

***Post-Processers***

Your post processor should process the text and link information based on the criteria specified to determine whether to store the web page and/or follow the links. If you decide to store the web page, you need to assign a unique filename to that page and store it in the directory specified. You also need to remember the URL corresponding to the page (even after the program has finished).

**What to hand in**

You will need to upload your final program (zip it into a zip file if there is more than one file) to canvas by the deadline date.

You are allowed to form group of 2s for this part of the project. Notice that the next part will likely be individual again, and the two of you will share the programs you developed in this step for the next.