CS297 Project Report

Topic: Full Text Indexing for Heritrix

Prepared By: Darshan Karia (006960200)

1. Introduction

Heritrix is an open-source web-scale, archive-quality, extensible web crawler project [1]. Heritrix crawls the open web based on specified seed sites and stores the result of the crawl into archive files. The archive file follows the Internet Archive ARC File Format [7].

Once, we have crawled the web using Heritrix, NutchWAX (Nutch + Web Archive eXtensions) can be used to fetch data from web archives created by Heritrix[3] and create inverted index for all archives. If there exists a full-text index for the web, WERA can use that index to search and navigate through documents based on user query [WERA2006]. Here, NutchWAX provides the interface, which facilitates access module, in this case, WERA, to search and navigate through web archived documents [4]. My project is to provide Full-Text Indexing [5] to the web archive created by Heritrix. The basic concepts of indexing are explained in the book Information Retrieval [10]. It also explains phrase search, inverted index and various related techniques like using galloping search in the book. I can use all these things to develop my own indexing scheme for archived data of Heritrix.

2. Deliverables

2.1 Deliverable – 1: Experiment with Heritrix

Description:

To begin my project, I needed to understand working of Heritrix first as my project depends on archive files generated using Heritrix. Therefore, I studied basics about how Heritrix works from its manual. I obtained Heritrix project source code of Heritrix and built the project. I got Heritrix to run and did sample crawls using it.

Introduction to Heritrix:

Heritrix is an open-source web crawler project. It is designed to respect robots.txt and exclusion directives and META robots tags. That means, it will not crawl web sites, which specifies exclusion from being crawled for robots. Heritrix uses adaptive crawl method to avoid too much of traffic at web servers it is crawling.

Obtaining Source Code:

Download Source Code for Heritrix from sourceforge svn url using following command:

```
svn co https://archive-crawler.svn.sourceforge.net/svnroot/archive-crawler/release-branches/heritrix-1.14.4 heritrix-1.14.4
```

Building Heritrix:

Heritrix can be built from the source using Maven 2.x. Other versions may work, but they are not tested. The mentioned plug-ins are required to setup as described in the Developer's Manual of

Heritrix, point 2.2 [11]. To begin with, open command prompt and go to the heritrix-1.14.4 folder we have built: cd heritrix-1.14.4. Now, run maven command to start building from source code: maven dist. If any error occurs, that might be because of some missing package within maven. In that case, follow the onscreen instructions and download the required package for maven manually.

Running Heritrix:

If we are running Heritrix from the binaries, go to heritrix-1.14.4/bin directory and type following command to launch web-interface for heritrix: heritrix --admin=LOGIN:PASSWORD. Here, LOGIN is "admin" and PASSWORD is "letmein".

If we are running Heritrix from eclipse as a java project, you need to set up three parameters in run configuration before running. Set main class of the project to "org.archive.crawler.Heritrix". Set the program arguments to provide username and password "-a admin:letmein". Set the VM Arguments "-Dheritrix.development -Xmx512M".

After finishing the configuration as mentioned above, whether you are running from command prompt or from eclipse, you are ready to launch the Heritrix web based UI. To start, launch any Web Browser and go to http://127.0.0.1:8080 web address to access web based user interface for Heritrix [12].

Before we start crawling the web after our first run on the Heritrix, we need to configure a few default settings in the crawl profile. We need to provide value for http-headers like "user-agent" by replacing "PROJECT_URL_HERE" with valid project url and replace "from" with valid e-mail address of the person using the crawler. Also, add this new user-agent to "user-agents"

under “robots-honoring-policy” section of profile. At last, provide seed sites for the crawl under section “Seeds”.

2.2 Deliverable 2: Modify Source Code of the Heritrix

Description:

Modify source code of the Heritrix to store address of corresponding robots.txt for all files.

Understand and modify the Source Code:

Before I could modify the source code, I had to understand the source code and execution flow of Heritrix so that it is easier to find exact place for modification to make the desired changes in behavior of Heritrix.

To modify Heritrix source code and make it to store location of corresponding robots.txt for all crawled files, I added following code in write method of ARCWriterProcessor class located in org.archive.crawler.writer package. This code stores offsets of all the robots.txt files it has seen so far in a hash map with domain of robots.txt as the key and pass that along to arc file writer along with other meta-data:

```
String[] temp = curi.toString().split("/");
String tmp = "";
long robotPosition = 0;
flag = false;
for(int i = 0; i < temp.length - 1 && i < 3; i++)
{
        tmp += temp[i]+"/";
}
if(temp[temp.length-1].equalsIgnoreCase("robots.txt"))
{
        flag  = true;
        robotTable.put(tmp, new Long(position));
}
```

```
System.out.println("position: " + position);
if(!flag)
{
        for(String key: robotTable.keySet())
        {
                System.out.println("key: " + key + " vs " + tmp + "\n" + robotTable.get(key));
                if(tmp.equalsIgnoreCase(key))
                {
                        System.out.println("Robot: " + robotTable.get(key));
                        robotPosition = robotTable.get(key);
                }
        }
}
```

The given code basically keeps entry of offset of robots.txt file in a hashtable defined as below:

```
private Hashtable<String, Long> robotTable = new Hashtable<String, Long>();
```

Here, String key type is used to store domain name and Long value type is used for storing offset address for robots.txt associated with that domain.

Along with the changes I mentioned above, I made changes to some other classes of the source code for Heritrix to adapt to this new change in archive format, like ARCReader, ARCWriter, ARCWriterTest in org.archive.io.arc package, Warc2Arc class in org.archive.io.

2.3 Deliverable - 3: See capabilities of NutchWAX and WERA

Description: Tested workings of NutchWAX and WERA.

2.3.1 The Workings of NutchWAX: [8]

NutchWAX is a variant of Nutch to crawl through the arc files instead of the open web and generate index based on those arc files. The map reduce jobs of NutchWAX can be run using hadoop. NutchWAX requires certain requirements to fulfill to run. It requires Linux environment because all the scripts to execute various NutchWAX jobs are designed to run on Linux in form of shell scripts. It also requires JDK installed on the system it is running. It requires Tomcat 5.5.x

or higher to be able to run the search component of NutchWAX. The Search component is provided as a war file with NutchWAX. NutchWAX also requires Hadoop installed and configured to be able to run map reduced jobs for indexing.

NutchWAX contains both war file and jar file. The NutchWAX war file is used for searching, while the NutchWAX jar file is used for indexing and performs various map reduce tasks related to indexing. These tasks include importing ARC files, updating database of all URLs, inversion of the database of link information in readiness for indexing, actual indexing, deduplication and finally merging all indices into one final index.

After configuring required environment variables NUTCHWAX_HOME and HADOOP_HOME, following command can be used for importing ARC files:

sh $HADOOP_HOME/bin/hadoop jar nutch-1.0.jar org.archive.nutchwax.Importer arcs/inputs /arcs/outputs test

2.3.2 The Workings of WERA:

WERA is an open source project [2]. WERA provides the user with interfaces for searching, browsing and navigating the archived web pages [4].

WERA requires a search-engine with full-text indexing of archived documents and document retriever as interface between Access Module and the web archive. One of the search engines, which fulfill these requirements, is NutchWAX. The simplest configuration of WERA with NutchWAX can be as shown as in Figure 2.1. Firstly, user submits query to wera. Now wera constructs search request and sends it to NutchWAX to get XML formatted result set. Wera formats the result set to user-viewable content. Here, when user clicks on any result, WERA

retrieves the result page from the archive file using the archive retriever with the information present in the result set for that selected result page.

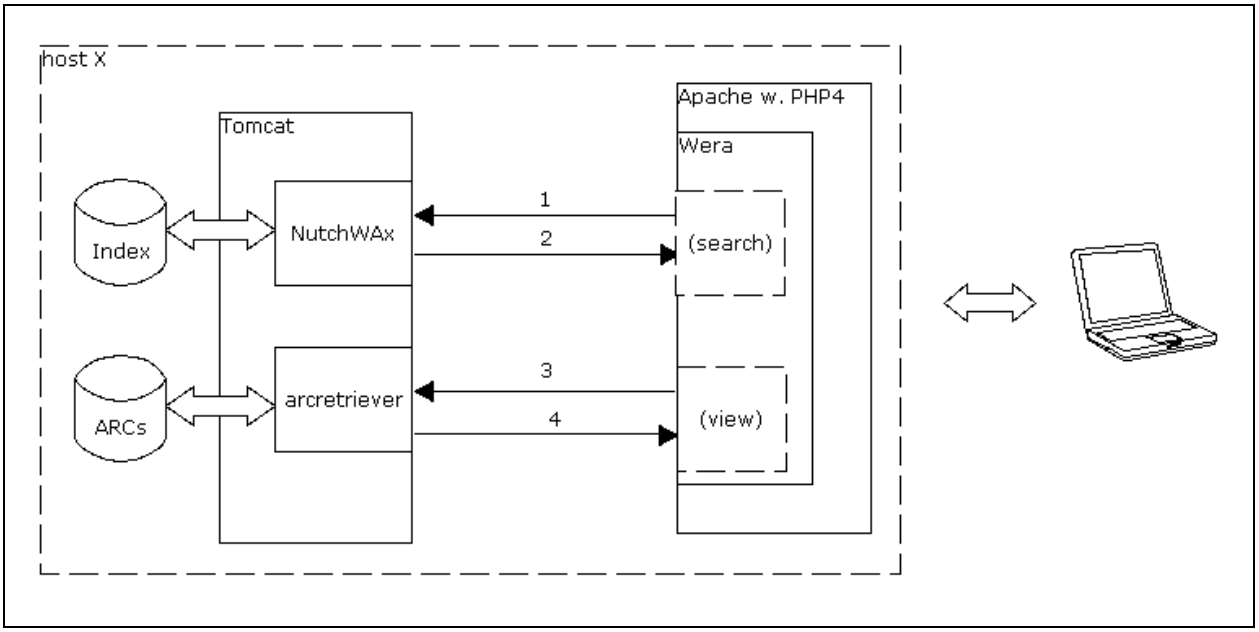

Figure 2.1: Configuration of WERA [9]

2.4 Deliverable - 4: Learn about Full-text indexing and inverted indexes

Description: Read chapter about inverted indexing from book titled "Information Retrieval" [10].

Basic Techniques: The chapter includes topics regarding phrase search, inverted index, functions related to that. Implementation using Binary tree search, sequential search and galloping search is explained in the chapter.

Abstract Structure of Inverted Index: If the collection is static and small enough, it can be maintained within memory with simple data structure like Hash Table, and posting list for each term in a simple array P_t[]. first and last methods of inverted index can be implemented in

constant time by returning 1st and last index content from collection. next and prev methods however needs be to implemented using binary search with time complexity $O(\log(l_t))$.

Implementation of next and prev methods of Inverted Index Using Binary Search:

Here, $P_t[]$ of length l_t contains all the posting of terms. This method works well for term search and phrase search where one of the term appears more frequently while other occurs rarely. E.g., "the" occurs more frequently in any page. It has to go through more and wasteful binary searches in case of terms of phrase appearing in similar frequencies. To overcome that, we can make changes in next and prev methods and make nextPhrase method to adapt this and perform better for phrases. This implementation has time complexity $O(n . L . \log(L))$ is appropriate when shortest posting list is considerably shorter than longest posting list in the collection ($l << L$).

Implementation of next and prev methods of Inverted Index Using Linear Search:

Here next method is implemented using linear search rather than binary search. It stores cached index offset c_t and uses that as starting point for next recursive call to "next" method if possible. This implementation has time complexity $O (n . L)$ is appropriate when all posting lists have approximately same length. ($l \approx L$).

Implementation of next and prev method of Inverted Index Using Galloping Search

The idea here is to use sequential search to form a small search region and then search based on binary search on that small region. Time complexity for galloping search is $O(l . \text{Log} (L / l))$. When $l << L$, this complexity is close to binary search performance and when $l \approx L$, it is close to sequential search. Therefore, here we achieved advantage of both binary search approach and sequential search approach.

3. Conclusion

I have studied working of Heritrix, procedure for crawling web using Heritrix, and storing them in archive file format. I have also studied how NutchWAX works and creates index for the documents along with usage of WERA to use the index and search through the archive files. Now, I need to develop my own indexing scheme and use that to create index for archive files recorded by Heritrix. I will use open source ARCReader of archive project to retrieve data from archive files and process them for indexing.

I need to test indexing on large set of data to test efficiency of my indexing scheme and algorithm in terms of processing speed and memory management. To get large set of data, I will run Heritrix on more seed sites and for longer duration.

By experimenting with Heritrix, I discovered how to crawl using Heritrix and get the archive files, which I can use for indexing during my project. I also studied and modified code of Heritrix to understand its workflow and to know about modules of Heritrix. Here, I learnt how archive files are getting created, which helps in understanding the archive file format more. By understanding archive file more, it is easier to retrieve data from that while indexing the archive files. I also studied NutchWAX to understand existing possible indexing solution for archive files. That can help me in designing my own solution for indexing archive files. I studied WERA to see how to react to user search queries, pass that request to index, and retrieve search result. This gave me better understanding of overall search process. This will also help in designing better index based on requirement.

4. References:

[1] Heritrix Web Crawler - http://crawler.archive.org/. 2010.

[2] WERA – Retrieved from Hetrix Web Site: http://archive-access.sourceforge.net/projects/wera/. 2007.

[3] NutchWAX – Retrieved from Nutch Web Site: http://archive-access.sourceforge.net/projects/nutch/. 2009.

[4] WERA Manual. Royal Library in Stockholm, Royal Library in Copenhagen, Helsinki University Library in Finland, National Library of Norway, National and University Library of Iceland. 2006.

[5] He, J., Yan, H., and Suel, T. 2009. Compact full-text indexing of versioned document collections. In Proceeding of the 18th ACM Conference on information and Knowledge Management (Hong Kong, China, November 02 - 06, 2009). CIKM '09. ACM, New York, NY, 415-424. DOI= http://doi.acm.org/10.1145/1645953.1646008

[7] ARC File Format – Retrieved from Heritrix Web Site: http://www.archive.org/web/researcher/ArcFileFormat.php

[8] NutchWAX Getting Started Guide and Overview - http://archive-access.sourceforge.net/projects/nutchwax/apidocs/overview-summary.html. 2007.

[9] What is WERA - http://archive-access.sourceforge.net/projects/wera/articles/what-is-wera.html 2005

[10] Büttcher, S., Clarke, C. L., & Cormack, G. V. (2010). *Information Retrieval: Implementing and Evaluating Search Engines.* MIT Press.

[11] Heritrix Developer Documentation - Retrieved from Heritrix Web Site: http://crawler.archive.org/articles/developer_manual/building.html

[12] Heritrix User Manual – Retrieved from Heritrix Web Site: http://crawler.archive.org/articles/user_manual/install.html