Web Search Engine
CS172
Part A: Web Crawler for Edu Pages

Umesh Mogharíya, sid: 861083538 Kevin Ng, sid: 860929456

May 13, 2013

1 Overview of the System

The project constructs a Web Crawler which starts with a list of URLs to visit, called the seeds. As the crawler visits these URL, it extracts the hyperlinks and adds them to a Queue. It then downloads the html content based on certain rules.

1.1 Architechture

```
.CS172Crawler/
├── src/
│   ├── Main.java
│   ├── htmlprocess/
│   │   ├── HtmlProcess.java
│   │   └── SaveHTML.java
│   ├── jedis/
│   │   └── Jedis.java
│   ├── robot/
│   │   └── RobotExclusionUtil.java
│   └── Multiprocess/
│       └── WorkerThread.java/
├── lib/
│   ├── commons-pool-1.6.jar
│   ├── jedis-2.1.0.jar
│   └── jsoup-1-1.7.2.jar
├── saved/
│   └── *.zip
├── seed/
│   └── seed.edu
├── bin
├── apache-ant/
├── redis-2.6.13/
├── main.sh
├── InstallRedis.sh
└── build.xml
```

1.1.1 `Main.java`

It initiates the program by reading the `seed.edu` file and adds the links to a `Hash`. It manages the `WorkerThread` where the Crawling takes place.

1.1.2 `HtmlProcess.java`

The function here returns the CRC value of the webpages. This CRC value is determined using `java CRC2` for the `<body>` tag of the Web Page. This ensures higher duplicate content detection as 2 pages might have the same body but different `<head>`. And the ultimate goal is to crawl unique content.

1.1.3 `SaveHTML.java`

Here the Content of the Web Page is saved to the disk. All the files are Zipped in-memory and then transferred to the disk inorder to save space and time.

1.1.4 `Jedis.java`

`Redis Server - www.redis.io` is used to store all the URL frontier Data. It features are:

1. Key-Value in-memory storage.
2. Stays available even if Program crashes or restarts.
3. All functions like Hash and List are atomic and thread safe.
4. Several instances can be created for scaling the Crawler.
5. Being in-memory, it is very fast and efficient.

`Jedis.java` is a client library to connect to the Redis Server, and does the work of adding to Hash and Lists.

Each URL is added to a Global Hash and its `Hop List` like `hoplist1, hoplist2, etc` which keeps track of number of hops away from the seed.

While processing the `Hop List` URLs are popped out.

1.1.5 `RobotExclusionUtil.java`

Checks if the URL can be crawled or not. Source Code: TA.

1.1.6 `WorkerThread.java`

Here the crawler:

1. Crawls URL.
2. Finds URLs from the Web Page.
3. Checks if the URLs are unique or not.
4. Adds to Hash if-unique.
5. Adds to hop-list depending on hop number.
6. Calls Function to Compute CRC of the Web Page.
7. Adds the CRC to Global CRC Hash.
8. Checks if-unique CRC for the current document.
9. Calls function to save-to-disk if document is unique.
10. Returns the number of Pages saved.

1.1.7 `lib`

Stores the `jars` used in program.

1.1.8 `saved`

Stores the `saved pages [zip]` created by the program.

1.1.9 `seed.edu`

Stores the `seed URL` to be used in program.

1.1.10 `bin`

Stores the `class files` generated by the program.

1.1.11 `apache-ant`

To automate the build of the program.

1.1.12 `redis-2.6.13`

Redis Server Binaries which needs to be installed and started.

1.2 Crawler Strategy

The program obeys the robots.txt and parses the url. It halts based on the number of hops and pages downloaded provided by the user. It also handles errors related to `Time Out, Malformed URL, HTTP Errors, Incorrect Mime Types` and moves forward ignoring them. While collecting links apart from checking the normal `<a>`tags, the programs also looks for `<frameset>` tags and reconstructs URL from relative to absolute and adds it to the Hash. This is usefull for websites loading pages in frames rather than normal Page type display. The Crawler is multi-threaded which implies better efficiency. Using the Redis server to store the URLs ensures scalability if needed. Detailed description is presented in the section above.

1.2.1 Current Scope

The program is currently running in one `Amazon EC2 micro instance` crawling with seeds of all the possible 3 and 4 characters `.edu` domains. This list is generated by pinging all combinations of the 3 and 4 characters urls and adding them to list if the ping is a success.

1.2.2 Future Scope

Currently there is no measure for the Freshness of the webpage and the Relation between the pages is not stored to a `graph data structure`. Future works can include these as well as other modifications like making it runnable across many `EC2 instances`, etc.

1.3 Data Structures

Primary URL Data is stored in :

1. `URL Hash` : Stores all the Unique URLs with `fields like isCrawled, hop`.
2. `Hop List n` : Stores URLs which are popped out when crawled, where n is number of hops.
3. `CRC Hash` : Stores all the CRC of the Webpages. This is used to detect duplicates.

2 Limitations

1. Currently when using Multi-thread, a Thread sometimes skips/fails URLs which are in the list.
2. No explicit warning if the internet connection is broken.
3. Does not do Http POST.

3 Deployment Directions

3.1 Installing Redis Server

Run the shell file `InstallRedis.sh` This will install the Redis Server in your machine. You can safely delete the entire project folder to delete Redis.

3.2 Using Apache Ant and running Program

Run `main.sh`

3.3 Changing parameters

In `build.xml` change the arg line="2" and arg line="10" at the end of the file. Here arg 1 is Hops and arg 2 is pages. Note, none of them can be zero or negative. It needs to be a integer.

4 Screen Shots

Check Folder names ScreenShots.

5 Collaboration Details

1. Umesh :
   (a) Logic and Program Flow.
   (b) Datastructure Design.
   (c) Redis Implementation.
   (d) Multi- Threading.
   (e) Implementing Efficiency using CRC32 and Zipped save.
   (f) Report Writing.
   (g) Adding Frameset URL detection.
2. Kevin
   (a) Logic and Program Flow.
   (b) Datastructure Design.
   (c) jsoup implementation.
   (d) Cleaning URLs.
   (e) Detecting Errors.