Web Search – An Application of Information Retrieval Theory

Term Project

Summer 2009

Overview Of This Phase Of The Project

You are to implement a crawler or a robot in this phase of the project to collect information from the web. The crawler starts with a given set of URLs. The number of URLs in this set can be one or more. It will retrieve the pages specified by this starting URL set. Parse the pages, extract some more URLs from these pages. Then visit the pages following these new URLs. This process will continue until either the time allowed has expired, or the number of retrieved pages has reached the limit, or there is no new page to visit.

The retrieved web pages are passed to the Indexer you built in the second phase of the project for processing. There an inverted index will be built for all the web pages retrieved, ready for an end user to search. The inverted indexing system was built as the third phase of the project.

General Algorithm

Traverse the web is very similar to traverse a general graph. Each web page can be considered as a node in a graph. Each hyper-link can be considered as a link in a graph. From this point of view, crawling web is not too much different from the graph traverse algorithm you learned in a typical data structure course. The following is a general web traversing algorithm that we discussed in lectures.

```
Initialize queue (Q) with initial set of known URLs
Until Q empty or page or time limit reached do  {
  Pop URL, L, from the front of the Q
  If L is not point to an HTML page (.gif, .jpeg, .ps, .pdf, .ppt, ...)
     continue loop
  If already visited L, continue loop
  If the page can't be downloaded (404 error, or robot exclusion)
     continue loop
  Download the page, P, for L
  Index P (or store it to a disk file)
  Parse P to obtain list of new URLs (this can also be done by Indexer)
  Append new URLs to the end of Q
}
```

Some Issues To Be Considered

Because of the vast size of the web, there are some technical and engineering issues we have to consider for a successful, less-intrusive crawler. We list here some of the issues to consider in crawling the web.

- Obey the robot protocol. See <http://www.robotstxt.org/robotstxt.html> for specific details. The basic idea when crawling the web is that first to check the server site (typically the root page) to see if the server administrator has put the `robots.txt` in place. If it is there, check the contents to see what are the directories that are excluded for visiting. We may ignore the individual page meta tags for now. Do not index and analyze the directories and pages that are excluded for visiting. If you don't follow the protocol, you may receive direct complaint from the server administrator. And your access to these web sites may be hindered. Pay attention to this issue.

- When visiting a web site you should report to the server the name of the user agent, the host where your program is running and a valid email address in case the site administrator wanted to contact you. This can be done when establishing the contact with the server, as shown in the following example in Java. Syntax in other languages vary, but the idea is the same.

```
// first get the information about local host
InetAddress inet = InetAddress.getLocalHost();

// now preparing the command to be sent to the server
String cmd = "GET " + path + " HTTP/1.0\n";
cmd += "Host: " + inet.getHostName() + "\n";
cmd += "User-Agent: " + "csci335x-course-project\n";
cmd += "From: " + "you@bucknell.edu\n\n";

// then send the command to the server
...
```

  The `InetAddress` class allows you to get various pieces of information about an Internet host. Here we retrieve the relevant Internet information of the local host. Then we send the name of the local host to the server as a part of the HTTP protocol. We also send the "User-Agent" and the email address of the person who is running the crawler to the server. Note that we now put two new-line characters at the end of the email address, instead of after the protocol `HTTP/1.0` as we did before, because all these pieces of information are part of the HTTP header.

  Check the example in http://localhost:9999//code/javaClient/webClient.java. If you use the `URLClient` as the base for a client, there is no easy to identify yourself. In this case, you may skip the identification part.

- Identifying yourself (the crawler) is a part of *good robot behavior*. When the site administrators see who is running the crawler and why, they are less likely to block the access and report it as intrusion. Other *good behaviors* include not to visit the same site with a lot of rapid requests. Rather wait a few minutes before next visit. In practice you may run a number of threads with each thread visit a site. Each one thread should wait a few minutes in between the visits to the same site. Because you have multiple threads running, the overall idle time for the crawler may not be high.

- If you decide to save the downloaded pages to disk files for further processing, make sure you use `synchronized` thread behavior to avoid inconsistence of the file status.

- Complete partial URLs. Many web pages contain partial URLs. That is, the URLs are relative to the current path. For example, you may encounter URLs within a page in the form of `../../home/page.html` or `mypage.html`. Your program needs to expand these URLs to a full URL so that the crawler can access them later. You may need to keep a current path and host for this purpose. For example, if the current host is `polaris.eg.bucknell.edu` and the current path is `work/example/` then the afore-mentioned two URLs will be expanded as `http://localhost:9999/home/page.html` and `http://localhost:9999/work/example/mypage.html`
- During the web crawling your program has to keep track of the pages that have been visited in order to avoid infinite loop. A `Hashtable` might be a reasonable choice for keeping the visited list. You can also devise your own data structure to maintain the list.
- Testing your crawler with some small Bucknell sites first. Do not crawl off-campus sites until you are pretty confident that your crawler is working properly.
- You may use the example `webClient.java` or `URLClient.java` as a starting point for a crawler. These two programs can be accessed from the course web site at <http://localhost:9999/code/javaClient/>

What to Hand In

Your team needs to hand in the following in the order given. Please staple all pages.

1. A team report for phase four of the project with a cover sheet. The report should include the following as a minimum.
   (a) A description of main contribution of each team member.
   (b) A summary of the working process, e.g. what the team started with, what the team has accomplished, any problems encountered, how the team solved them, any thoughts on the project, among others.
2. Source code for the programs (this phase only)
3. Snapshots of sample runs.
4. Email the instructor a copy of the complete source code (all phases up to now) in `zip` or `tar` format.