Web Stuggler : A New Tool for Mining Web Pages based on Page Traffic over Internet

M Sri Vidya [#1] , P. Srinivas [*2] , CH .Srinivas Reddy [*3]

[#1]M.Tech, Department of information Technology, Vignan's Institute of Information Technology, JNTU-KAKINADA, Andhra Pradesh, India
[#1]Srividya.mutha@gmail.com

[*2] Assistant Professor, Department of information Technology, Vignan's Institute of Information Technology, Visakhapatnam, Andhra Pradesh, India
[*2] Srinivasp3@gmail.com

[*3] Assistant Professor, Department of information Technology, Vignan's Institute of Information Technology, Visakhapatnam, Andhra Pradesh, India
[*3]Srinivasreddyviit@gmail.com

Abstract

A Web Stuggler is a program in internet, which automatically traverses the web by downloading documents and following links from page to page. They are mainly used by web search engines to gather data for indexing. Other possible applications include page validation, structural analysis and visualization; update notification, mirroring and personal web assistants/agents etc. Web Search/Web Crawlers is also known as spiders, robots, worms etc. A Search resides on a single machine. The Search simply sends HTTP requests for documents to other machines on the Internet, just as a web browser does when the user clicks on links. All the Search really does is to automate the process of following links. Web Searching speed is governed not only by the speed of one's own Internet connection, but also by the speed of the sites that is to be searched. Especially if one is a Searching site from multiple servers, the total Searching time can be significantly reduced, if many downloads are done in parallel. This work implements the "Breadth First Searching" algorithm, a refined version of one of the first dynamic Web search algorithm. By conducting several experiments on various types of websites comprising of educational websites like university, colleges, schools, public sites, booking sites, banking sites and so on. We finally got a conclusion that by using this proposed application or web crawler tool we can able to get the web site rank based on individual page traffic not based on overall page ranking. By this we clearly state that his proposed tool is mainly used by website administrators in order to reduce the workload that was been caused by participating users, by other users.

Keywords

Web Search,--Breadth First Search, Robots, Web Agents, Visualization, Page Validation

1. Introduction

The economic and cultural importance of the web has guaranteed considerable academic interest in it, not only for affiliated technologies, but also for its content. Research into web pages themselves has been **motivated** by attempts to provide improved information retrieval tools such as search engines, but also by the desire to know what is available, how it is structured and to determine its relationship with other meaningful human activities. The advanced facilities available in search engines such as AltaVista and Info seek, but their use has raised questions of reliability that have led to the creation of a specialist web spider/analyzer to produce the raw data by direct Searching and analysis of the sites concerned. Information scientists and others wishing to perform data mining on large numbers of web pages will require the services of a web Search or web-Search-based tool, either individually or collaboratively.

1.1 Fundamentals of a Web Search

Despite the numerous applications for Web Search, at the core they are all fundamentally the same. Following is the process by which Web Search work:

1. Download the Web page.

2. Parse through the downloaded page and retrieve all the links.

3. For each link retrieved, repeat the process.

Now let's look at each step of the process in more detail.

In the first step, a Web Search takes a URL and downloads the page from the Internet at the given URL. Oftentimes the downloaded page is saved to a file on disk or put in a database. Saving the page allows the Search or other software to go back later and manipulate the page, be it for indexing words (as in the case with a search engine) or for archiving the page for use by an automated archived.

In the second step, a Web Search parses through the downloaded page and retrieves the links to other pages. Each link in the page is defined with an HTML anchor tag similar to the one shown here:

```
<A
HREF="http://www.host.com/directory/file
.html">Link

</A>
```

After the Search has retrieved the links from the page, each link is added to a list of links to be searched.

The third step of Web Searching repeats the process. All Search work in a recursive or loop fashion, but there are two different ways to handle it. In our project the Links can be Searched using breadth-first manner.

2. Literature Survey

In this section we will describe the assumptions that are used in the proposed paper.

2.1 Main Motivation

The economic and cultural importance of the web has guaranteed considerable academic interest in it, not only for affiliated technologies, but also for its content. Research into web pages themselves has been motivated by attempts to provide improved information retrieval tools such as search engines, but also by the desire to know what is available, how it is structured and to determine its relationship with other meaningful human activities. The advanced facilities available in search engines such as AltaVista and Info seek, but their use has raised questions of reliability that have led to the creation of a specialist web spider/analyzer to produce the raw data by direct Searching and

analysis of the sites concerned. Information scientists and others wishing to perform data mining on large numbers of web pages will require the services of a web Search or web-Search-based tool, either individually or collaboratively.

The potential power of web mining is illustrated by one study that used a computationally expensive technique in order to extract patterns from the web and was powerful enough to find information in individual web pages that the authors would not have been aware of .A second illustration is given by the search engine Google, which uses mathematical calculations on a huge matrix in order to extract meaning from the link structure of the web. The development of an effective paradigm for a web-mining Search is, therefore, a task of some importance.

2.2 Mechanism

The proposed ranking model can be used as a web crawler, which is a program that automatically traverses the web by downloading documents and following links from page to page. They are mainly used by web search engines to gather data for indexing.

Other possible applications include:-

- Page validation
- Structural analysis and visualization
- Update notification
- Mirroring and personal web assistants/agents etc.

Web crawlers are also known as spiders, robots, worms etc.

"RANKING MODEL" which actually searches the data at the time when the URL is issued. While this "RANKING MODEL" does not scale up, it guarantees valid results for dynamic search. It is preferable to static search for discovering information in small and dynamic sub Webs. Finally, "*ranking adaptability"* measurement is proposed to quantitatively estimate if an existing ranking model can be adapted to a new domain with several experiments on various sites. By using our proposed ranking model using SVM, we get the following advantages:

- The proposed RA-SVM can better utilize by both the auxiliary models and target domain labeled queries to learn a more robust ranking model for the target domain data.
- Domain-specific features can steadily further boost the model adaptation, and RA- SVM-SR is comparatively more robust than RA-SVM- MR.
- Adaptability measurement is consistent to the utility of the auxiliary model, and it is deemed as an effective criterion for the auxiliary model selection

The emphasis in requirements analysis is on identifying *what* is needed from the system, not *how* the system will achieve its goals. This task is complicated by the fact that there are often at least two parties involved in software development-a client and a developer. The developer usually does not understand the client's problem domain and the client does not understand the issues involved in the system software systems developed by the developers. Hence causes a communication gap between them.

This communication gap is bridged during the analysis. This analysis phase ends with a document describing all the requirements called as SRS (Software Requirements Specification).

There are two major activities involved in this phase, Problem understanding and requirement specification. In problem analysis, the analyst has to understand the problem and its context. Such analysis typically requires through understanding of the existing system, parts which have to be automated. A clear understanding is needed of the important data entities in the system, major centers where action is taken, the purpose of the different actions that are performed and the inputs and outputs.

3. Proposed Algorithm and Methodologies

In this paper we are going to implement BFS (Breadth First Search) Algorithm for implementing the search traversal technique and finally the sorted URLS will be placed in the order of BFS Hierarchy where there will be no chance of repetance of visited URL to be replaced once again in between the searched URL's.By using this BFS Algorithm for identifying the crawled URL's, all the URL's will get distinct rank in the privilege of web crawling. If the same application is been used with DFS, we may get in correct URL's in the final output.

3.1 Breadth First Search (BFS) Algorithm

Breadth First Search, Algorithm is an uninformed search method that aims to expand and examine all nodes of a graph systematically in search of a solution. In other words, it exhaustively searches the entire graph without considering the goal until it finds it. It does not use a heuristic. The BFS algorithm is used in order to build a major search engine or a large repository such as the Internet Archive, high-performance Search start out at a small set of pages and then explore other pages by following links in a "breadth first-like" fashion. In reality, the web pages are often not traversed in a strict breadth-first fashion, but using a variety of policies, e.g., for pruning Search's inside a web site, or for Searching more important pages first.

The working principle of BFS Algorithm is it will crawl the root URL first and then from that root url it will search for the child url's that are linked directly with the root node and finally all the visited child url's will be entered into FIFO queue and finally once the url is visited it will be marked as visited and changes its status to the visited and the pointer will be keep on crawling new url's which was not yet repeated. During this crawling if already visited URL is again visited it will not consider that once again the same url into the list.

Steps for BFS Algorithm are:

1) Put the starting node (the root node) in the queue.
2) Pull a node from the beginning of the queue and examine it.
   a) If the searched element is found in this node, quit the search and return a result.
   b) Otherwise push all the (so-far-unexamined) successors of this node into the end of the queue, if there are any.
3) If the queue is empty, every node on the graph has been examined -- quit the search and return "not found".
4) Repeat from step 2.

3.2 Pseudo code for BFS Algorithm

The below pseudo code clearly represents the BFS algorithm procedure and its working principle.

Pseudo-Code for BFS Algorithm

```
function breadthFirstSearch (Start, Goal) {
    enqueue(Queue,Start)
    while notEmpty(Queue) {
      Node := dequeue(Queue)
      if Node = Goal {
         return Node  // the code below does not get executed
      }
      for each Child in Expand(Node) {
        if notVisited(Child) {
           setVisited(Child)
           enqueue(Queue, Child)
        } }    }
}
```


4. Implementation Modules based on Role

The following are the roles that are performed during the process of crawling the web pages based on individual page traffic. Here in this proposed tool we have two roles, one role is played by user and one role is played by Google server.

4.1 Google API Role Flow Chart

In this Google API's Role, it shows that operations performed by the Google API after accepting the URL from the user. The default actions are specified by the Ranking Model Web Crawler which is clearly shown in figure 1.

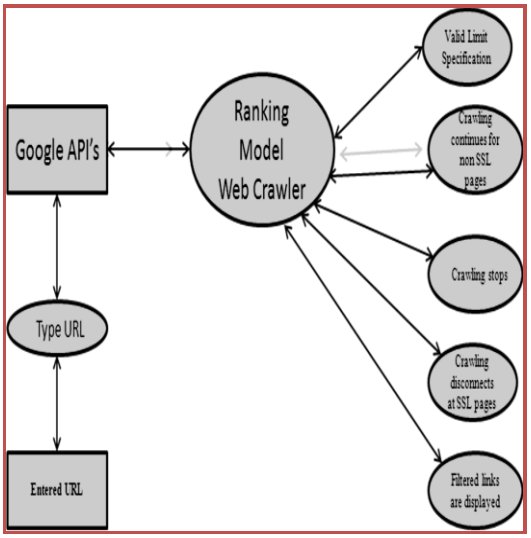


Figure 1. Role of Google API in our application

4.2 Role of User and his Flow of Events

In this User Role, it shows that the operations performed by the user after accepting the appropriate URL details from Google API's**.** The default actions which should be performed by the user is specified by the Ranking Model Web Crawler which is clearly shown in figure 2.

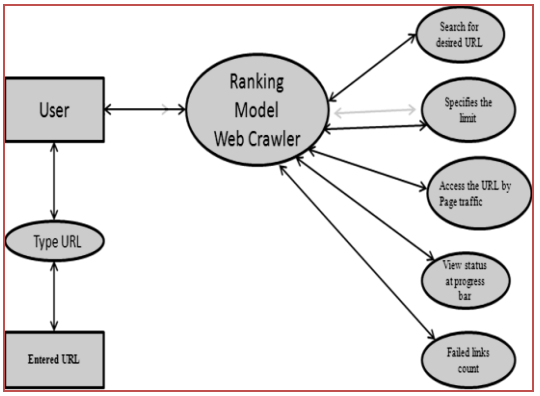


Figure 2. Role of User in our application

4.3 Main Source Code for Current Application

Here we will discuss the main source code for the proposed application implementation. For developing this application we use Java Swings as a front end interface and we are not using any back end data base for implementing this application. For implementing this application we need a google connection because all the searched urls traverse through internet and then come back to the user interface and displays as an output in the filtered order based on page traffic.

Main Logic Part for connecting URL to Internet

```
public class URLConnect extends Thread
 {
   String URLPass;
            StringBuffer str;

             URLConnect(String URLName)
             {
    try
                        {
          URLPass=URLName;
```



```
                                        URL url =
new URL(URLName);

        HttpURLConnection connection =
(HttpURLConnection) url.openConnection();
        connection.connect();

        if(connection.getResponseCode()!=200)
                                        {
          connection.disconnect();
                                        Furl++;
          UpdateStatus("Disconnected from
"+URLName);

UpdateResult("<br>"+URLName+": disconnected
<br>");


if(control>=0)
                                connOther();
                                        else
                                        {

        out();
          }
                                        }
                                        else
                                        {
           UpdateStatus("Connected to
"+URLName);

        BufferedReader in = new BufferedReader(
new
InputStreamReader(connection.getInputStream()));
                        String line;
                        str = new StringBuffer();

while((line=in.readLine())!=null)
                                {
                                 str.append(line);
                                }
                                in.close();
          start();
                                        }
                } // end of try

catch(MalformedURLException murle)
                {
      UpdateStatus("Type with a valid protocol!
ex: http://www.sun.com");

progress.setIndeterminate(true);
                         unlock();
                        }
                        catch(UnknownHostException
uhe)
                        {
      UpdateStatus("URL not found! Check
whether you are connected to internet / IP could not
be determined");

progress.setIndeterminate(false);
                        unlock();
                        }
                        catch(Exception e)
                        {
      UpdateStatus("Error : "+e);

                         if(control>=0)
                                connOther();
                                        else
                                        {

          out();
            }
                        } // end of catch
          } // end of URLConnect constructor
          public void run()
          {
    search(str,URLPass);
          }
 } // // end of URLConnect class
```

Logic Part for Searching the Web Pages based on Page Usage Traffic

```
public void search(StringBuffer str,String urlPass)
          {
    progress.setValue(Clist);

   try
                {

     UpdateStatus("\nCrawling . . . "+urlPass);

                if(URLlist>1)

                 UpdateResult("<br>Crawling .
. . "+urlPass);
```

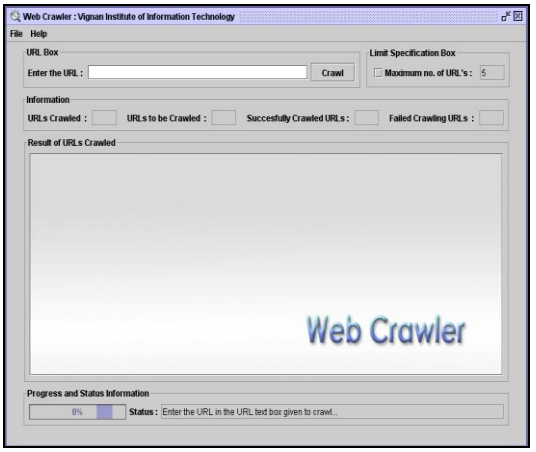


```
int j=0,sav=0,v=0,p=0,flag=1;

long i=0,len=0;

                String sov="";

            len = str.length();

// saves the length of source code

relative(str.toString());

                absolute(str.toString());

if(control>=0)

  connOther();

                else

out();

                }

                catch(Exception e)

                {          UpdateStatus("Error :
"+e);                 out();
                }          }
```

5. Experimental Results and Description

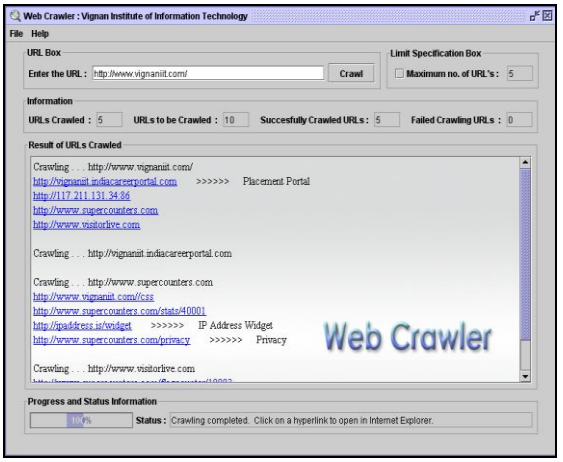
For developing this web page extractor tool based on page usage traffic, we use java technology with front end User interface designed with java Swings and back end is internet connection for crawling the URL's lively. The below screen is the main interface for executing this application which mainly contains a text area where we need to enter the url what we want to search .There was a button beside text box called as "Crawl", if you click on that button it will direct the requested url to the google server and then return back to the text area which is available in the results of url's crawled area.

Web Crawler Main Window



In the above window we will also find progress bar which is present below the window which indicates the progress of the application. The status bar what we have in the current application indicates the status of the web crawler. The next window clearly indicates the crawling of urls that was processed by the specified URL.

Main Window after Crawling the URL



The above window crawls the URLs of Vignan university website. The url's will be crawled based on page usage traffic.

6. Conclusion

We have described the architecture and implementation details of our Searching system. Searching forms the backbone of applications that facilitate Web information retrieval. The web Search is designed using Breadth-first Searching, which provides the highest page rankings. Itself capable of searching a large collection of web sites by using idle processing power and disk space. The testing of the system has shown that it cannot operate fully automatically for tasks that involve searching entire web sites without an effective heuristic for identifying duplicate pages.

7. Future Enhancement

As this application can only crawl URL's in BFS fashion and it is failed in crawling SSL protected pages. In future we may wish that these limitations can be resolved and we may also crawl the authenticated pages also from the web, so that there will be no limitation present in the web crawler over internet.

8. References

[1]. Herbert Schildt Java Complete Reference. Fifth Edition.

[2]. Core Java 2: Volume I & II - Fundamentals By Cay S. Horstmann, Gary Cornell

[3]. Java™ Tutorial, Third Edition: A Short Course on the Basics By Mary Campione, Kathy Walrath, Alison Huml

[4]. Data Mining Techniques By Arun K Pujari

[5]. Flippo Menczer, Gutam Pant, Padmini Srinivasan, Searching the Web.

[6]. Pankaj Jalote, An Integrate Approach to Software Engineering, 3rd Edition.

[7]. Searching the Web. Gautam Pant, Padmini Srinivasan and Filippo Menczer3

[8].S. Chakrabarti. Mining the Web. Morgan Kaufmann

[9].F. Menczer and R. K. Belew. Adaptive retrieval agents: Internalizing local context and scaling up to the Web. Machine Learning

[10]. J. Rennie and A. K. McCallum. Using reinforcement learning to spider the Web efficiently.Morgan Kaufmann, San Francisco, CA, 1999.

[11]. G. Salton and M.J. McGill. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.

[12]."Efficient Searching Through URL Ordering", Junghoo Cho, Hector Garcia-Molina, Lawrence Page. 7th International Web Conference (WWW 98)

