Restructuring robots.txt for better Information Retrieval

Bhavin M. Jasani
Department of computer Science,
Saurashtra University, Rajkot, Gujarat (India).

C. K. Kumbharana
Department of computer Science,
Saurashtra University, Rajkot, Gujarat (India).

ABSTRACT
Now a days the users of the WWW are not only the human. There are other users or visitors like web crawlers and robots which are generated by the search engines or information retrievers. The direct visitors of your website are very less than those who reach to your website by using search engines or through other links. To collect information from your website search engines use crawlers or robots to access your website. There must be an access mechanism or protocol for such robots which restrict them to access unwanted content of the website.robots.txt is a partial mechanism for such facilities but not fully functional. This paper gives an enhancements to fully make use of the functionality of robots.txt file.

Keywords
Crawling agents, robots, spammer, harvesters, User Agent, <META> tag, Directive Overriding, Web Crawling, Web Tree, Web Spamming, Crawling, Querying.

1. INTRODUCTION
Crawling agents called robots or crawlers are used by search engines for retrieval of Information from websites. The website has to access its resources to serve the requests coming from various web crawlers. It takes more time and resources of the web servers. There must be some mechanism to manage the crawler activities because the intention and purpose of retrieving content of all crawlers may not be identical. To overcome this problem the robot exclusion protocol(REP)[1] is a solution but this is not fully utilized by site administrators. REP provides set of access rules to be followed by web crawlers. A place in the website where these rules can be specified is a robots.txt file, Which contains allow or disallow rules to define access of content for specific web crawlers.

By using this technique specified in this paper, Website administrators can specify the set of rules to indicate the visiting robots that which parts of the site should be avoided and which part should be scanned at the time of website visit. It is purely depends up to the visiting robot to consult the instructions written into the robots.txt file and act accordingly. Because some bad robots like spammer or email harvesters may not act according to the robot exclusion protocol or instructions written in the robots.txt file. There is no control of site administrators over the scanning of robots. To overcome this problem some techniques are defined in this paper.

2. STRUCTURE & ACCESS METHOD OF ROBOTS.TXT
This section covers the encoding of the instructions and access method to retrieve these instructions by robot or crawler. Any visiting robot or crawler must first read these instructions before visiting any other URLs of that website. And decide if the other URLs of that website are accessible or not. Such instructions must be accessible through HTTP as text/plain format from specific path websiteurl/robots.txt as other pages are accessible from the internet.

Some examples of URLs for sites and URLs for corresponding "/robots.txt" files:

If the domain name of a website is http://www.example.com/ than the robots.txt file must be at root of the site i.e. *http://www.example.com/robots.txt* another domain name like http://www.myweb.com:8001/, than the robots.txt file must be available at *http://www.myweb.com:8001/robots.txt*

At the time of accessing robots.txt file if the server response returns success then robot or crawler must read the content or instructions from the robots.txt file and act accordingly if applicable to that visiting robot or crawler. If server returns page not found (404) error then visiting robot can assume that no instructions are available and hence the site is not restricted by robots.txt, robot will scan all available pages of that website.

3. FORMAT OF ROBOTS.TXT FILE
The instructions are encoded in the plain text format formally records are separated by new lines and each record is in the form of <field>:<value>.

The instructions start with the list of user agents to specify which instructions are for which robot with the value of "Allow" and "Disallow" fields. As an example:

User-agent:crawlername

User-agent:robotname

Allow:/public/page.html

*Allow:**

Disallow:/public/page_1.html

Where *User-agent* is the name of the crawler or robot to identify themselves at the time of website visit. Here *crawlername* and *robotname* are the name of the visiting robots. The *Allow* directive indicates the page /public/page.html is accessible for both user agents, while the *Disallow* directive indicates the page /public/page_1.html is not permissible to read for both user agents. The value of "*" defines the remaining all pages of the site are accessible for both user agents.

The name of the robot can found in the HTTP request made by the visiting robot for example the "search data" robot sends HTTP request as :

GET / HTTP/1.0
User-agent: search_data/0.1 Robot libwww-perl/5.04 might scan the "/robots.txt" file for records with:
User-agent: search_data

The comparison of user_agent value is case insensitive. If there is not match found for the visiting robot then the access of website is unlimited for that robot. Matching process compares each octet found in the URL with the records in robots.txt file. Possible matches are given in the bellow table as robots.txt file record value and URL path.

Record Value	URL Path	Matches
/tmp	/tmp	**Yes**
/tmp	/tmp.html	**Yes**
/tmp	/tmp/a.html	**Yes**
/tmp/	/tmp	**No**
/tmp/	/tmp/	**Yes**
/tmp/	/tmp/a.html	**Yes**
/a%3cd.html	/a%3cd.html	**Yes**
/a%3Cd.html	/a%3cd.html	**Yes**
/a%3cd.html	/a%3Cd.html	**Yes**
/a%3Cd.html	/a%3CD.html	**Yes**
/a%2fb.html	/a%2fb.html	**Yes**
/a%2fb.html	/a/b.html	**No**
/a/b.html	/a%2fb.html	**No**
/a/b.html	/a/b.html	**Yes**
/%7ebmj/index.html	/~bmj/index.html	**Yes**
/~bmj/index.html	/%7Ebmj/index.html	**Yes**

3.1 Working Example of robots.txt file

This section contains an example of how a /robots.txt is used. A fictional site may have the following URLs:
http://www.bmj.org/
http://www.bmj.org/index.html
http://www.bmj.org/robots.txt
http://www.bmj.org/server.html
http://www.bmj.org/services/fast.html
http://www.bmj.org/services/slow.html
http://www.bmj.org/orgo.gif
http://www.bmj.org/org/about.html
http://www.bmj.org/org/plans.html
http://www.bmj.org/%7Ejim/jim.html
http://www.bmj.org/%7Emak/mak.html

The following instructions are written into /robots.txt of the site http://www.bmj.org for three user agents “Robot 1/1.0”, “Crawler 1/0.2” and “Info Spiders/3.0”.

 # /robots.txt for http://www.bmj.org/
 # comments to webmaster@bmj.org
User-agent: robot1
Disallow: /

User-agent: crawler1
User-agent: infospider
Disallow:

*User-agent: **
Disallow: /org/plans.html
Allow: /org/
Allow: /serv
Allow: /~mak
Disallow: /

The following matrix shows which robots are allowed to access which URLs:

URLs	**Robot1**	**Crawler1**	**other& infospider**
http://www.bmj.org/	No	Yes	No
/index.html	No	Yes	No
/robots.txt	Yes	Yes	Yes
/server.html	No	Yes	Yes
/services/fast.html	No	Yes	Yes
/services/slow.html	No	Yes	Yes
/orgo.gif	No	Yes	No
/org/about.html	No	Yes	Yes
/org/plans.html	No	Yes	No
/%7Ejim/jim.html	No	Yes	No
/%7Emak/mak.html	No	Yes	Yes

3.2 Robots <META> Tags

The page requests coming from other websites may not read the instructions of robots.txt file. The visiting robot may read that page even this page is restricted in the robots.txt file. Apart from robots.txt file, by using html <meta> tags we can instruct the visiting robots to index the content of the HTML page or to follow the links found in the page as following:

<html>
<head>
<title>...</title>
<META NAME="ROBOTS" CONTENT="NOINDEX, NOFOLLOW">
</head>

the NOFOLLOW directive only applies to links on this page. It's entirely likely that a robot might find the same links on some other page without a NOFOLLOW (perhaps on some other site), and so still arrives at your undesired page. The "NAME" attribute must be "ROBOTS".

Valid values for the "CONTENT" attribute are: "INDEX", "NOINDEX", "FOLLOW", "NOFOLLOW". Multiple comma-separated values are allowed, but obviously only some combinations make sense. If there is no robots <META> tag, the default is "INDEX,FOLLOW", so there's no need to spell that out. That leaves:

<META NAME="ROBOTS" CONTENT="NOINDEX, FOLLOW">
<META NAME="ROBOTS" CONTENT="INDEX, NOFOLLOW">
<META NAME="ROBOTS" CONTENT="NOINDEX, NOFOLLOW">

NOINDEX, FOLLOW : Means do not Index the page, but follow the links found in the page to further scanning.
INDEX, NOFOLLOW : Means Index this page content for search, but do not follow the links found in the page for further scanning.
NOINDEX, NOFOLLOW : Means do not index this page for searching and also don’t follow the links found in this page for further scanning.

4 ENHANCING ROBOTS.TXT

As we have discussed in above sections that It is solely up to the visiting robot to consult this information and act accordingly. Because some bad robots like spammer or email harvesters may not act according to the robot exclusion protocol or instructions written in the robots.txt file.

As per the above study of robots.txt and Robot Exclusion Protocol there is no mechanism of how to manage the scanning flow of the robots and how to restrict them to visit unpublished or unwanted pages in the website, whether the intension of robot for scanning the website is good or bad.

Here, the researcher has suggested some improvements for robots.txt file as a protocol to follow each robot access in the website. These improvements are as under.

Up to now we have seen some basic keywords like user agent, allow, disallow, etc in the robots.txt files. But these information are only limited for controlling access of web pages into the website. But this information does not give any confirmation that whether the visiting robot has act according to this information or not.

The robots.txt standard is a very useful tool for both webmasters and the people who run web crawlers. This standard could be even more useful with several improvements.

As we know that, what is the main aim of any robots or crawler? The answer is to collect meaning full information from World Wide Web. Generally robots or web crawlers are run by search engines to collect information from WWW for indexing as per their structure and algorithms. Here the main aim of researcher is to improve the web crawling mechanism for effectiveness of search engine.

So far as the search engine mechanism and web crawling algorithm is concern the robots.txt file should also contain the information about the website, that what is the actual aim of the website, website content is for what purpose and category, the region of the website, language or target users of the website, description of the website, etc. So that a visiting robot can get basic information about website by reading only one file i.e. robots.txt. To collect such information if there is no such robots.txt file, the robot may need to scan more than five pages of the website. Let us discuss each topic in detail which is highlighted above.

4.1 Site Naming

Site naming creates many problems for web robots or web masters. A robots.txt file should contain the name of the site. Sites can be referred or referenced by many names. Even some large scale websites may have many physical IP addresses and virtual domain names. It becomes very crucial for robots or web master tools to determine which name to prefer for future use. This is useful when the site administrator has set the temporary domains to balance the server load. This naming guides the robot how to avoid that domain to reduce infinite looping and scanning. Site administrator may simply set the site name directive into the robots.txt file as under:

 # defines the site name for visiting robots.
Site-name: www.example.com | www.example1.com

Where "|" defines the alternatives as per BNF-like description, using the conventions of RFC 822[2].

4.2 Multiple Names and IP Addresses

Most of the websites are referred by multiple names and IP addresses to balance the server traffic as mentioned in above section. To avoid duplication crawlers generally convert these names to IP addresses. In practice when representing the search data it is required to use site names instead of IP addresses. It is bit confusing for web crawlers to determine which name should be used for a particular website. The robots.txt file should have an entry for the site name details.

Some heavy traffic websites use multiple servers to balance the load. Servers are added frequently and their IP addresses often change. Crawlers do not have any mechanism to understand and keep record of the frequently changing mapping of servers to their logical site names. These anomalies create undesired duplicate crawling and useless effort of the crawler and higher traffic at the websites. The robots.txt file is a perfect location to mention a list of IP addresses and their mapping to a logical site.

 # defines multiple site name and IP address mapping.
Site-temp-name: www.temp.abc.com=>127.0.0.1
Site-temp-name: www.temp.abc1.com=>192.168.0.1
Site-temp-name: www.temp.abc2.com=>192.168.1.123

4.3 Freshness of Web Pages and its Content

There are many ways to check freshness of the web page and web content. HTTP header provides a facility to check when the page or file was updated. This process is useful for saving labor work of robot to scan unchanged pages as the last scan was run before. It saves server time and bandwidth for scanning such pages.

The robots.txt file should have an entry of modification record for such frequently changing web pages or website areas by using MD5s or timestamp values. So that the robot can easily identify which area of the website is useful for scan and which area is unchanged since the last visit of that website.

Today the only way to tell what pages you want to update is to use the If-Modified-Since request-header field. This costs a connection per page. Having this information centralized in the robots.txt file would decrease server loads. This information could be presented at a directory or file level depending on the size of the site and the granularity of information the webmaster wants to present.

As the paths of the website inner pages or areas are defined in the robots.txt file we can put same changing timestamp[5] with the entry of the path in allow, disallow directives. Following example or directive can be used to apply such facility into the robots.txt file:

 # defines change timestamp of frequently changing pages.
Last-updated:/info/aboutus.html=>1431507724
Last-updated:/main/news.html=> 1431507798
Last-updated:/main/jobs.html=> 1431508278
Last-updated:/blog/recent.html=> 1431508758

4.4 Freshness of robots.txt file

The robots.txt file needs to include a time-to-live (TTL) value. This tells crawlers how often they should update the robots.txt information for that site. Some sites very rarely change their robots.txt files and do not want the extra traffic of having them frequently re-read by multiple crawlers. Even if the If-Modified-Since request-header field is used, a connection still

has to be created each time. On the other hand, some sites change their robots.txt files regularly and often. They are often hurt by extensive caching of robots.txt information by crawlers. Having an explicit TTL value would help crawlers satisfy each site's requirements.

4.5 Multi level robots.txt file

The scanning size of visiting robots may have scanning or reading limitations to read the robots.txt file at some extent. So it becomes danger to put large amount of text content in the robots.txt file because some robots leave the content at specific size or limit occurs. It is required to have more than one robots.txt file for the same website at different levels, so that the size of the file remain small and it is acceptable by almost all robots and webmasters.

For example in some website the content can be updated from many sources as there may be a more than one content updaters for a site. As an example a blogging site, social network site, university sites may have more than one content updaters.

In the university there may be multiple students updating their content on the website. the updating area of the site may be different for different student. And content access control may be changed and it is up to the students. if all students or all content updaters try to change the main global robots.txt file than it is not feasible in the practice and it is unreasonable to allow all of the content updaters to make changes in the robots.txt file.

The main robots.txt file should contain information or redirection facility to read further robots.txt file in the deep. The multilevel robots.txt files can be defined for various part of the site, may be this can be categorize by the access control or types of the website users.

4.6 Directive overriding for complex directory tree

The disallow statement of the current robots.txt standard could be made more powerful. For various reasons some sites cannot change their on-disk layout and may have very large directories. It is very cumbersome to exclude part of a large directory using the current disallow statement. A more powerful regular expression syntax or an 'allow' directive to override the disallow for specific files would be useful.

4.7 Site Description

the main task of any robot or crawler is to collect meaningful information from the world wide web to create some specific repositories. Repositories may be categorized by the content type, area of the content, language of the content, location of the website , origin of the website etc.
robots or crawlers generally read the content of the website and stores them to different categories for future use. The description of the website is also useful for SEO(search engine optimization) of the website.

It is recommended to have brief info of the website into the robots.txt file. Using this brief description of the website written in the robots.txt file robots or crawlers can identify that in which category the website will fall without reading the content of all pages of that website.

4.8 Host Details

Putting domain hosting details on robots.txt file may help the web crawler to collect meaning information like IP address of the website, domain provider, list of available name servers etc.

Using these information search engines can keep record of server location and list of available name servers for crawling purpose.

Following is the host details of http://www.w3c.org website as an example to make entry in the robots.txt file.

#Hosting Details.

Hosting_provider: Massachusetts Institute of Technology
IP_Address: 128.30.52.45
Name_Server1:ns1.w3.org
Name_Server2:ns3.w3.org
Name_Server3:ns2.w3.org
Domain_ID:D1266030-LROR
Creation_Date:1994-07-06T04:00:00Z
Updated_Date:2015-02-16T16:48:01Z
Expiry_Date: 2022-07-05T04:00:00Z

4.9 Access log of each visiting robot/crawler

While visiting the website page, the robot or web crawler passes the header information to the web server as a request. Following is the example of http header sent by the HTTrack crawling application as a robot request :

GET /abc HTTP/1.1

Connection: keep-alive

Host: localhost

User-Agent: Mozilla/4.5 (compatible; HTTrack 3.0x; Windows 98)

Accept: text/html,image/png,image/jpeg,image/pjpeg,image/x-xbitmap,image/svg+xml,image/gif;q=0.9,/*;q=0.1*

*Accept-Language: en, **

Accept-Encoding: gzip, identity;q=0.9

Capturing User-Agent field from header information web servers can keep track of the visiting robots or web crawlers. This tracking process should also be performed from the website areas where any robot is not allowed to visit the pages, so that we can also identify that how many robots are entering in such restricted areas intentionally.

4.10 Authenticating Robots

As earlier mentioned in this section the it is up to the visiting robot/crawler to act according to the access rules written in robots.txt files. Some bad robots like email harvesters and spammers may not act according to robots.txt.

To make sure that robot/crawler is acting as per the instructions written in robots.txt file, web servers could provide an authentication for each robot/crawler as normal user's user name and password to access the website content. Web server or site administrator may also create a role for robots to access the website content.

For such authentication robots.txt file has such mechanism that each robot/crawler should be redirected to an authentication page when they enter to visit the website page or try to read the robots.txt file for further detail. After providing a username and password, robot/crawler can gain the access of scanning the web pages as per their roll and instructions written in the robots.txt file. As all web applications support sessions a site administrator can also track the activities of the robot/crawler during their logged in sessions.

4.11 Configuration Interface

Now a days the management of websites become crucial for site administrators. Maintaining an authenticity of the website is become a major problem for all online applications. We have discuss that the website should have multilevel robots.txt files for batter access control same way there may be more than one content updaters for the different areas of the website. To maintain these all one must have to make changes in robots.txt file every time and it becomes very difficult to track the record of each change made in robots.txt file.

Hence like the website admin panel, the robots.txt file also should be configurable through attractive user interface. The configuration settings can be stored in database and easily updatable using user interface by any person with little or no knowledge about robots.txt file and robot exclusion protocol.

4.12 Authenticating Robots

The web crawler or a robot will have its own independence to act as per the instructions written in the robots.txt file. We can list allow and disallow entries in the robots.txt file for a specific path or directories. But it becomes crucial to keep allow disallow entry of each page specifically when the website has more pages. Even it excide the size of the robots.txt file. Because some robots or web crawlers may read robots.txt file up to some defined size. So the size of robots.txt file should be as less as possible.

If you disallow the robot or crawler to enter into some restricted area of website by making entry in the robots.txt file, the bad intension robots even scan your page. Or sometimes the request of restricted page may comes from outside or crawling other website. At this time crawler may not read the target website robots.txt file and keep reading the restricted page. To overcome this problem we can use the meta tages for each page to inform the visiting robots that visit or not to visit this page.

But the problem remains still open that how to restrict all type of robots or crawler whether the intension of scanning robot is good or bad.

As we discussed earlier in the chapter section 3.8.9 that the web crawler or robot make a request at each page while crawling the website. By using header information web server can identify the name of requester i.e. the name of the robot or crawler in this case. At the top of the each page a site administrator can put a check mark whether the requesting robot is allowed to visit the current page or not. If the robots.txt file is configurable as discussed earlier, the checking entries of all robots from database is become easy irrespective of platform and programming language used to develop a website.

The following flowchart may help to understand the check mark process.

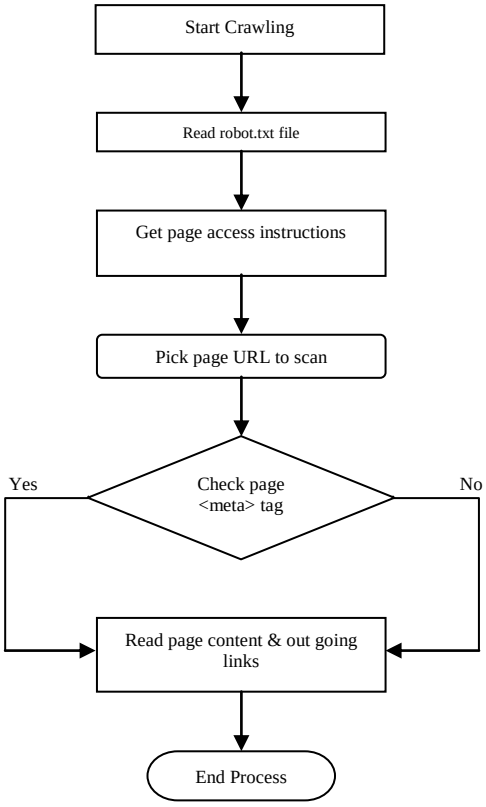

Fig.1 : [A process flow of crawler without page level check mark]

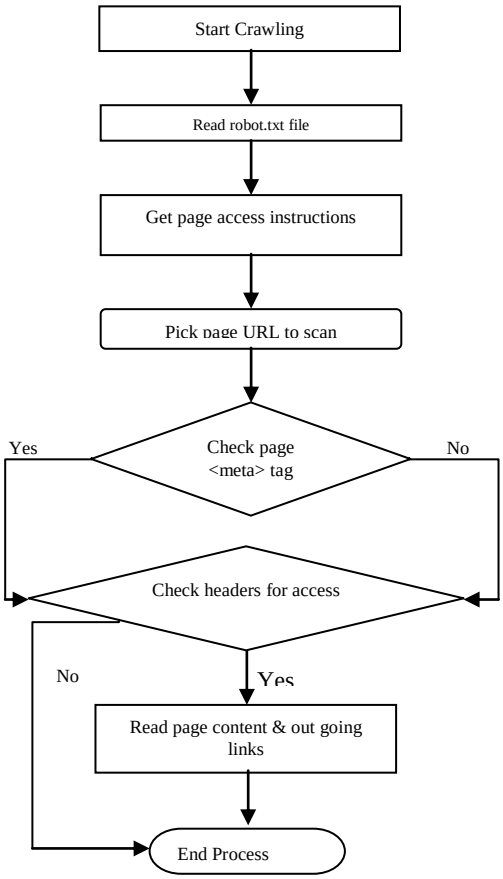

Fig.2 : [A process flow of crawler with page level check mark]

In figure 1 the web crawler or robot can ignore the page access rule and read the page. While in figure 2 the robot is identified by check mark to allow or disallow. If the robot is disallow to read the page, the robot won't find anything from the page. The pseudo code for check mark is as under:

```
Var allow=false;

Function read_Header(){

        Var user_agent="agent name";

        allow = Check_agent_entry(user_agent);

}

Function Check_agent_entry(user_agent){

        Var(array) list_of_robots=fetch from database;

        While(list_of_robots){

                If(robot==user_agent){

                        Return true;

                        Break;

                }

        }

        Return false;

}

If(allow){

// put html code of the page here.....

}else{

        Print("you are not allowed to scan or read this page.. bye bye.");

}
```

5. REFERENCES

[1] A Standard for Robot Exclusion: http://www.robotstxt.org/orig.html

[2] Standard for the Format of ARPA Internet Text Messages: https://www.ietf.org/rfc/rfc0822.txt

[3] The Web Robots Pages. http://www.robotstxt.org/

[4] W3C http://www.w3.org/

[5] Timestamp http://en.wikipedia.org/wiki/Timestamp

[6] Backus–Naur Form http://en.wikipedia.org/wiki/Backus%E2%80%93Naur_form