World Journal of Engineering Research and Technology
WJERT
www.wjert.org
SJIF Impact Factor: 4.326

INTELLIGENT CROWLING TECHNIQUES FOR LOCATING THE SITES

Dr. Sulochana Sonkamble[1] and Dr. Balwant Sonkamble*[2]

[1]JSPM Narhe Technical Campus, Pune, Savitribai Phule Pune University, India.
[2]Pune Institute of Computer Technology, Pune, Savitribai Phule Pune University, India.

Article Received on 08/05/2017 Article Revised on 29/05/2017 Article Accepted on 18/06/2017

***Corresponding Author**
Dr. Balwant Sonkamble
JSPM Narhe Technical Campus, Pune, Savitribai Phule Pune University, India.

ABSTRACT

There are various techniques for crawling the web sites for searching the desired information. The paper titled "Intelligent Crawling Techniques For Locating the Sites" is focusing on the intelligent technique for searching the desired information which is available on the internet. The variety of information in the form of web pages on the internet is growing tremendously day to day. In this case searching relevant information on the Internet is a crucial and difficult task. This information is hidden behind query forms that interface to unexplored databases containing high quality structured data. Traditional search engines cannot access and index this hidden part of the information. A new term 'Web Retrieving' arises to find this hidden information and become the challenge to the researcher. We have presented a two-stage framework, namely SmartCrawler, for effectively harvesting deep web interfaces. The first stage is called as Site Locating, and a second stage is called as In-Site Exploring. At the first stage of Site locating, the center pages are searched with the help of search engines which in turn avoid visiting a large number of pages. To achieve more precise results for a focused crawl, SmartCrawler ranks websites to prioritize highly relevant ones for a given topic. In the second stage, adaptive link-ranking achieves fast in-site searching by exploring most relevant links. The in-site exploring stage uses adaptive link-ranking to search within a site; and we design a link tree for eliminating bias toward certain directories of a website for wider coverage of web directories.

INDEX TERMS: SmartCrawler, Site Locating, In-Site Exploring, Intelligent Crawling,

Link-Ranking.

1. INTRODUCTION

Web crawlers are used for a variety of purposes. A web crawler also known as a spider is a system for the bulk downloading of web pages. Most prominently, they are one of the main components of web search engines. A systems that assemble a corpus of web pages, index them, and allow users to issue queries against the index and find the desired information web pages that match the queries.

The function of crawler is to crawl around the ground. In web crawling, the crawler crawls around the internet, gathers the web-pages, categorizes them and searches the desired information on the WWW. The crawler contains of three parts: first is the crawler, second is the page indexing, third one is the information processing. The crawler works like a spider. The spider visits the pages, fetches the information and then follows the links in other pages within a site. The spider returns the crawled site over regular interval of time. The links are forwarded to the index for indexing the crawled pages. The information found in the first stage will be given to the third stage, the index through the links. The index is used as a catalog of information. The index is like a database, containing every copy of web-page which is found by crawler. If a web-page changes then the copy is updated with new information in the database. Third part is software.[12] This is a program that sort out millions of web pages recorded in the index. The searched records are compared and leveled them in order to get most relevant information. The figure 1 describes the different parts of crawler.

A large amount of data on the WWW remains inaccessible to crawlers of Web search engines because it can only be exposed on demand as users fill out and submit forms. The Hidden web refers to the collection of Web data which can be accessed by the crawler through an interaction with the Web-based search form instead of traversing hyperlinks.

Web crawling has many inherent challenges with respect to scale, content selection tradeoffs, social obligations, and adversaries, etc.

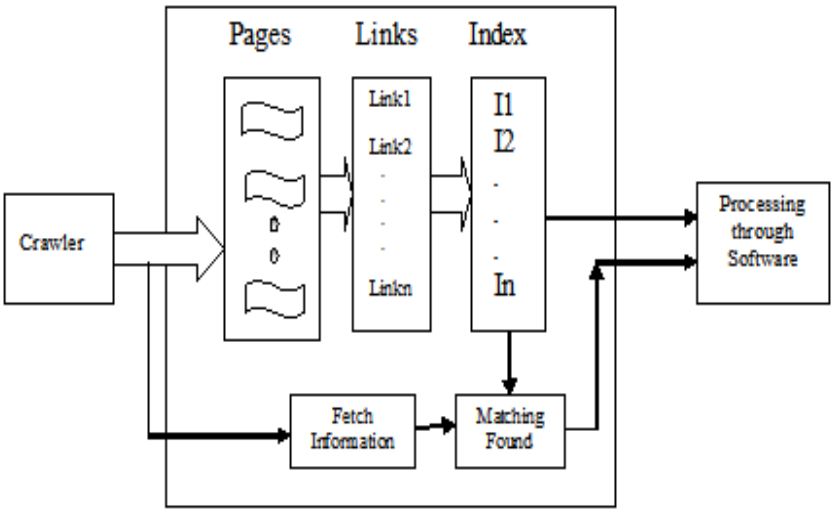

Figure 1: Parts of Crawler to fetch Information.

The web is very huge and continually growing information. It becomes the challenge to users to spider within it and search out the desired information. Crawlers that seek broad coverage and good freshness must achieve extremely high throughput, which poses many difficult engineering problems. Modern search engine companies employ thousands of computers and dozens of high-speed network links for this purpose. Even the highest-throughput crawlers do not purport to crawl the whole web, or keep up with all the changes.[13] Instead, crawling is performed selectively and in a carefully controlled order. The goals are to acquire high-value content quickly, ensure eventual coverage of all reasonable content, and bypass low-quality, irrelevant, redundant, and malicious content. The crawler must balance competing objectives such as coverage and freshness, while obeying constraints such as per-site rate limitations. A balance must also be struck between exploration of potentially useful content, and exploitation of content already known to be useful.

Crawlers should be "good citizens" of the web, i.e., not impose too much of a burden on the web sites they crawl. In fact, without the right safety mechanisms a high-throughput crawler can inadvertently carry out a denial-of-service attack. Some content providers seek to inject useless or misleading content into the corpus assembled by the crawler. Such behavior is often motivated by financial incentives and it should be avoided by commercial web sites.

2. LITERATURE SURVEY

The Web crawling is essential communicating media to all the users working with internet. Most of the business and other organizations prefer the internet for providing the services.

Therefore the web crawling became the most demanding area as everything is based on internet and websites. We have surveyed various research papers on the web crawling techniques.

In 2013 the A authors Yeye He, Dong Xin, Venkatesh Ganti, Sriram Rajaraman, and Nirav Shah explained in paper "Crawling Deep Web Entity Pages "that the deep-web crawl refers to the problem of surfacing rich information behind the web search interface of diverse sites across the Web.[2] It was estimated by various accounts that the deep-web has as much as an order of magnitude more content than that of the surface web. While crawling the deep-web can be immensely useful for a variety of tasks including web indexing and data integration, crawling the deep-web content is known to be hard. Authors focus on entity-oriented deep-web sites. These sites curate structured entities and expose them through search interfaces.

In paper "Toward Large Scale Integration: Building a Meta Queries over Databases on the Web" the authors Kevin Chen-Chuan Chang, Bin He, and Zhen Zhang Described Web has been rapidly "deepened" by myriad searchable databases online, where data are hidden behind query interfaces. Toward large scale integration over this "deep Web," authors have been building the Meta Queries system - for both exploring and integrating databases on the Web. To enable effective access to databases on the Web, since April 2002, authors have been building a "Meta Querying" system, the Meta Queries The goal is first, to make the deep Web systematically accessible, it will help users find online databases useful for their queries. Second, to make the deep Web uniformly usable, it will help users query online databases. Such deep-Web integration faces new challenges– for coping with the large scale: The deep Web is a large collection of query able. As the large scale mandates, first, such integration is dynamic: Since sources are proliferating and evolving on the Web, they cannot be statically configured for integration. Second, it is ad-hoc: Since queries are submitted by users for different needs, they will each interact with different sources. Toward the large-scale integration, the Meta Queries must achieve dual requirements one is Dynamic discovery and second is On-the-fly integration.

Dynamic discovery: As sources are changing, they must be dynamically discovered for integration– there are no pre-selected sources.

On-the-fly integration: As queries are ad-hoc, the MetaQuerier must mediate them on-the-fly for relevant sources, with no pre-configured specific per-source knowledge.

In paper "Searching for Hidden Web Databases" authors Luciano Barbosa, Juliana Freire focused on the recent studies estimate the hidden Web contains anywhere between 7,500 and 91,850 terabytes of information. As the volume of information in the hidden Web grows, there is increased interest in techniques and tools that allow users and applications to leverage this information.[6] In this paper, authors address a crucial problem that has been largely overlooked in the literature: how to efficiently locate the searchable forms that serve as the entry points for the hidden Web.[5] Having these entry points is a necessary condition to perform several of the hidden-Web data retrieval and integration tasks.

The searchable forms can be used as the starting point for deep crawls and for techniques that probe these databases to derive source descriptions; and in form matching they can serve as inputs to algorithms that find correspondences among attributes of different forms. Several factors contribute to making this problem particularly challenging. The Web is constantly changing – new sources are added, and old sources are removed and modified. A scalable solution, suitable for a large-scale integration or deep-crawling task, must automatically find the hidden-Web sources.

In addition, even for a well-defined domain it is hard to specify a schema that accurately describes the relevant forms. Authors propose a new crawling strategy that combines ideas from these two approaches. Authors use a page classifier to guide the crawler and focus the search on pages that belong to a specific topic. But in order to further focus the search, like in, our crawler learns to identify promising links, including links whose benefit may not be immediate, a link classifier selects links that are likely to reach pages that contain forms.

In "Focused crawling: a new approach to topic-specific web resource Discovery" paper the Authors Soumen Chakrabarti, Martin Van den Berg, and Byron Dom described the rapid growth of the World-Wide Web poses unprecedented scaling challenges for general-purpose crawlers and search engines. Authors describe a new hypertext resource discovery system called a Focused Crawler.[8] The goal of a focused crawler is to selectively seek out pages that are relevant to a pre-defined set of topics. The topics are specified not using keywords, but using exemplary documents. Rather than collecting and indexing all accessible Web documents to be able to answer all possible ad-hoc queries, a focused crawler analyzes its crawl boundary to find the links that are likely to be most relevant for the crawl, and avoids irrelevant regions of the Web. This leads to significant savings in hardware and network resources, and helps keep the crawl more up-to-date.

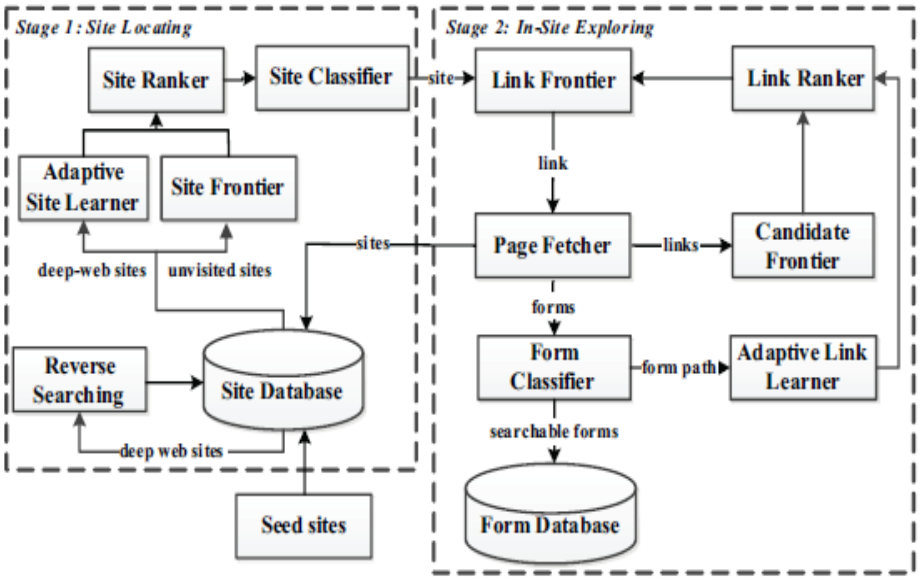

Figure 2: The two-stage architecture of *SmartCrawler.*

To achieve such goal-directed crawling, authors designed two hypertext mining programs that guide our crawler: a classifier that evaluates the relevance of a hypertext document with respect to the focus topics, and a distiller that identifies hypertext nodes that are great access points to many relevant pages within a few links. Authors report on extensive focused-crawling experiments using several topics at different levels of specificity. Focused crawling acquires relevant pages steadily while standard crawling quickly loses its way, even though they are started from the same root set. Focused crawling is robust against large perturbations in the starting set of URLs. It discovers largely overlapping sets of resources in spite of these perturbations. It is also capable of exploring out and discovering valuable resources that are dozens of links away from the start set, while carefully pruning the millions of pages that may lie within this same radius. Authors suggest that focused crawling is very effective for building high-quality collections of Web documents on specific topics, using modest desktop hardware.

Examples include almost all online shopping sites, where each entity is typically a product that is associated with rich structured information like item name, brand name, price, and so forth. Additional examples of entity-oriented deep-web sites include movie sites, job listings, etc this is to contrast with traditional document-oriented deep web sites that mostly maintain unstructured text documents. Entity-oriented sites are very common and represent a significant portion of the deep-web sites.[3] The variety of tasks that entity oriented content enables makes the general problem of crawling entities an important problem.

The practical use of the system is to crawl product entities from a large number of online retailers for advertisement landing page purposes. While the exact use of such entities content in advertisement is beyond the scope of this paper, the system requirement is simple to state: Authors provided as input a list of retailers' websites, and the objective is to crawl high-quality product entity pages efficiently and effectively. There are two key properties that set our problem apart from traditional deep-web crawling literature. First, specifically focus on the entity-oriented model, because of our interest in product entities from online retailers, which are entity-oriented deep-web sites in most cases. While existing general crawling techniques are still applicable to some extent, the specific focus on entity-oriented sites brings unique opportunities. Second, a large number of entity sites like online retailers are provided as input to our system, from which entity pages are to be crawled.[7]

In this paper "Google's deep web crawl" the Authors Jayant Madhavan, David Ko, Lucja Kot, Vignesh Ganapathy, Alex Rasmussen, and Alon Halevy explained the Deep Web, i.e., content hidden behind HTML forms, has long been acknowledged as a significant gap in search engine coverage.[4] Since it represents a large portion of the structured data on the Web, accessing Deep-Web content has been a long-standing challenge for the database community. Authors describes a system for surfacing Deep-Web content, i.e., pre-computing submissions for each HTML form and adding the resulting HTML pages into a search engine index. The results of our surfacing have been incorporated into the Google search engine and today drive more than a thousand queries per second to Deep-Web content.

Surfacing the Deep Web poses several challenges. First, goal is to index the content behind many millions of HTML forms that span many languages and hundreds of domains. This necessitates an approach that is completely automatic, highly scalable, and very efficient. Second, a large number of forms have text inputs and require valid inputs values to be submitted. Authors present an algorithm for selecting input values for text search inputs that accept keywords and an algorithm for identifying inputs which accept only values of a specific type. Third, HTML forms often have more than one input and hence a naive strategy of enumerating the entire Cartesian product of all possible inputs can result in a very large number of URLs being generated. Authors present an algorithm that efficiently navigates the search space of possible input combinations to identify only those that generate URLs suitable for inclusion into our web search index. Authors present an extensive experimental evaluation validating the effectiveness of our algorithms.

```
// Sample program of Web Crawler
import java.io.*;
public class Crawler
{ String baseLink;
public static void main(String a[])
{ Crawler crawler=new Crawler( );
System.out.println("Enter Start String");
InputStreamReader converter = new InputStreamReader(System.in);
 BufferedReader in = new BufferedReader(converter);
try{ crawler.baseLink = in.readLine( ); }
catch(Exception ex){ }
System.out.println(crawler.baseLink);
crawler.crawl ( );}
public void crawl ( )
{ String arr[ ]=new String[1500];
String arr2[ ]=new String[1500];
try{  FileInputStream fstream = new FileInputStream("C:\\Web Crawler\\index.html");
DataInputStream in = new DataInputStream(fstream);
 BufferedReader br = new BufferedReader(new InputStreamReader(in));
String strLine;
 int i=0;
while ((strLine = br.readLine( )) != null)
 {  if(strLine.contains(baseLink))
{  arr[i]=strLine.substring (strLine.indexof (baseLink));
  int index=arr[i].indexof ("\"");
  if(index>0) {
  //System.out.println(index);
arr2[i]=arr[i].substring(0,index);
System.out.println(arr2[i]);  }
 //System.out.println (strLine.substring(strLine.indexof(baseLink)));
 i++;  }  }  in.close( );  }
catch (Exception e) {
System.err.println("Error: " + e.getMessage( )); }}}
```

Figure 3: Sample code for Crawl the pages.

In this paper we studied the three stage web crowler and presented the paper for locating desired site. The queries are formulated to get the information on the web page on internet. Different stages of processing the query are divided and processed accordingly. In the first stage, intelligent web crawler performs site-based searching for centre pages with the help of search engines, in second stage achieves fast in-ste searching by excavating most relevant links with an adaptive link-ranking. In third stage, system will match the user keywords.

3. METHODOLOGY

The SmartCrawler is designed for discovering the knowledge efficiently and effectively in the deep web data sources. The SmartCrawler is designed with two stage architecture, site locating and in-site exploring, as shown in Figure 2. The first site locating stage finds the

most relevant site for a given topic, and then the second in-site exploring stage uncovers searchable forms from the site.

The site locating stage starts with a seed set of sites in a site database. Seeds sites are candidate sites given for SmartCrawler to start crawling, which begins by following URL from chosen seed sites to explore other pages and other domains. When the number of unvisited URLs in the database is less than a threshold during the crawling process, SmartCrawler performs ”reverse searching” of known deep web sites for center pages having high and feeds these pages back to the site database.

Site Frontier fetches home page URLs from the site database, which are ranked by Site Ranker to prioritize highly relevant sites. The Site Ranker is improved during crawling by an Adaptive Site Learner, which adaptively learns from features of deep-web sites. The deep-web site contains one or more searchable forms. To achieve more accurate results for a focused crawl, Site classifier categorizes URLs into relevant or irrelevant for a given topic according to the homepage content. After the most relevant site is found in the first stage, the second stage performs efficient in-site exploration for excavating searchable forms.

Links of a site are stored in Link Frontier and corresponding pages are fetched and embedded forms are classified by Form Classifier to find searchable forms. Additionally, the links in these pages are extracted into Candidate Frontier. To prioritize links in Candidate Frontier, SmartCrawler ranks them with Link Ranker. Both, the site locating stage and the in-site exploring stage are mutually intertwined. When the crawler discovers a new site, the site’s URL is inserted into the Site Database. The Link Ranker is adaptively improved by an Adaptive Link Learner, which learns from the URL path leading to relevant forms.

We have performed an extensive performance evaluation of Smart Crawler over real web data in the representative domains and compared with ACHE and site-based crawlers. As per our observation, our crawling framework is very effective, achieving substantially higher harvest rates than the state-of-the-art ACHE crawler. The theoretical study and existing results show the effectiveness of the reverse searching and adaptive learning.

4. DESIGN AND MODELLING

Smart Crawler is a focused crawler consisting of two stages: efficient site locating and balanced in-site exploring. Smart Crawler performs site-based locating by reversely searching

the known deep web sites for center pages, which can effectively find many data sources for sparse domains. By ranking collected sites and by focusing the crawling on a topic, Smart Crawler achieves more accurate results.[11] The in-site exploring stage uses adaptive link-ranking to search within a site.

We design a link tree for eliminating bias toward certain directories of a website for wider coverage of web directories. The model of the sample code for web crawling is shown in figure 3. The code tested and crawled the web pages. Our sample experimentation results are shown in figure 4.

The architecture model is proposed for two stage crawling the pages with higher rank. The representative set of domains show the effectiveness of the proposed two-stage crawler. This achieves higher harvest rates than other crawlers. In future work, we plan to combine pre-query and post-query approaches for classifying deep-web forms to further improve the accuracy of the form classifier. The requirements for external interface design are windows operating system, IDE Netbeans, Java and MySQL.

Figure 4: The output of the crawler is given below.

5. IMPLEMENTTION

When a search engine's web crawler visits a web page, it "reads" the visible text, the hyperlinks, and the content of the various tags used in the site, such as keyword rich meta tags. Using the information gathered from the crawler, a search engine will then determine what the site is about and index the information. The website is then included in the search engine's database and its page ranking process.

Web crawlers may operate one time only, say for a particular one-time project. If its purpose is for something long-term, as is the case with search engines, web crawlers may be programmed to comb through the Internet periodically to determine whether there has been any significant changes. If a site is experiencing heavy traffic or technical difficulties, the spider may be programmed to note that and revisit the site again, hopefully after the technical issues have subsided.

Crawlers can retrieve data much quicker and in greater depth than human searchers, so they can have a crippling impact on the performance of a site. Needless to say, if a single crawler is performing multiple requests per second and/or downloading large files, a server would have a hard time keeping up with requests from multiple crawlers.

As noted by Koster[9] the use of Web crawlers is useful for a number of tasks, but comes with a price for the general community. The costs of using Web crawlers include:

1. Network resources, as crawlers require considerable bandwidth and operate with a high degree of parallelism during a long period of time.
2. Server overload, especially if the frequency of accesses to a given server is too high.
3. Poorly-written crawlers, which can crash servers or routers, or which download pages they cannot handle.
4. Personal crawlers that, if deployed by too many users, can disrupt networks and Web servers.

Some crawlers intend to download as many resources as possible from a particular web site. So path-ascending crawler was introduced that would ascend to every path in each URL that it intends to crawl. One example of path finding data on is given as a seed URL[10] of http://llama.org/hamster/monkey/page.html, it will attempt to crawl /hamster/monkey/, /hamster/, and /Cothey found that a path-ascending crawler was very effective in finding isolated resources, or resources for which no inbound link would have been found in regular

crawling. The path-ascending crawlers are also known as Web harvesting software, because they're used to "harvest" or collect all the content , collection of photos in a gallery from a specific page.

6. CONCLUSION

In this paper, we propose the intelligence techniques for locating the sites. We presented a two-stage framework, namely SmartCrawler, for effectively harvesting deep web interfaces. The focus of stage one is locating the site, and stages two is exploring the site. In first stage , the center pages are searched which are desired. At the second stage most relevant links are collected. The design and model of the system is presented in the paper.

ACKNOWLEDGMENTS

The first author is professor in Computer Engineering Department at JSPM Narhe Technical Campus, Pune, SP Pune University, India. Her research areas are cloud computing and web mining, computer vision & soft computing. She has completed her PhD in Computer Science and Engineering in 2012, Master of Technology and Bachelor of Engineering in Computer Science and Engineering. The second author is professor in Computer Engineering Department at Pune Institute of Computer Technology - PICT, Pune, SP Pune University, India. His research areas are artificial intelligence, speech processing, soft computing and machine learning. He is guide for PhD students at University. He has completed his PhD in Computer Science and Engineering in 2012, Master of and Bachelor of Engineering in Computer Science and Engineering.

REFERENCES

1. Chen-Chua et al. “SmartCrawler: A Two-stage Crawler for Efficiently Harvesting Deep-Web Interfaces”, IEEE Transactions on Services Computing Volume: PP Year: 2015; 99: 1-14.
2. Yeye He, Dong Xin, Venkatesh Ganti, Sriram Rajaraman, and Nirav Shah “Crawling deep web entity pages” In Proceedings of the sixth ACM international conference on Web search and data mining, 355–364. ACM, 2013.
3. Roger E. Bohn and James E. Short, “How much information”, 2009 report on American consumers”, Technical report, University of California, San Diego, 2009.
4. Jayant Madhavan, David Ko, Łucja Kot, Vignesh Ganapathy, Alex Rasmussen, and Alon Halevy ” Google’s deep web craw” Proceedings of the VLDB Endowment PVLDB '08, August 23-28, 2008, Auckland, New Zealand , 2008 VLDB Endowment, ACM , 978-1-

60558-306-8/08/08, 2008; 1241–1252.

5. Kevin Chen-Chuan Chang, Bin He, and Zhen Zhang "Toward Large Scale Integration: Building a Meta Querier over Databases on the Web", CIDR, 2005; 44–55.
6. Luciano Barbosa and Juliana Freire, "Searching for hidden-web databases" Eighth International Workshop on the Web and Databases, Baltimore, Maryland, June 2005; 1-6, 16-17.
7. Peter Lyman and Hal R. Varian, "How much information" Technical report, UC Berkeley, 2003.
8. Soumen Chakrabarti, Martin Van den Berg, and Byron Dom "Focused crawling: a new approach to topic-specific web resource discovery", Computer Networks, 1999; 31(11): 1623–1640.
9. URL: https://www.myheritage.com/site-146419392/koster.
10. URL: http://www1.llama.org/?kw=llamas.
11. C. J. van Rijsbergen "Information Retrieval", Information Retrieval Group, University of Glasgow, UK.
12. Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.
13. Mini Singh Ahuja, Dr Jatinder Singh Bal Varnica, "Web Crawler: Extracting the Web Data", International Journal of Computer Trends and Technology (IJCTT), Jul 2014; 13(3): 132-137. ISSN: 2231-2803.