Volume : 1 | Issue : 12 | September 2012 ISSN - 2249-555X

Research Paper Computer Science


# Web Crawler: An Intelligent Agent Through Intellect Webbot

*Shailesh A. Patel ** Dr. Jayesh M. Patel

*Asst. Prof., BCA Programme,A.M. Patel Institute of Computer Studies, Ganpat Vidyanagar

**Associate Prof., MCA Programme,A.M. Patel Institute of Computer Studies,Ganpat Vidyanagar

**ABSTRACT**

*To completely crawl the World Wide Web, web crawler takes more than a week period of time. This paper focuses on role of agents in providing intelligent crawling over the web. The role of building a proxy server at application level is clearly discussed. The web pages need to be cached for providing better response time. Within this time, there are changes occurred to various pages, so it cannot always be able to provide the updated content to the user. Intellect Webbot will reduce the latency taken for search results by enabling various agents, providing more updated links to the user, also the adeptness to view users' bookmarks anywhere through our system. Moreover, this system has distributed intelligent agents, which is used to index the web pages in the server with the updated information. The actual scenario is the user going to give the keyword in terms of query to this system.*

*The system contains several agents such as Link repository agent, Regional crawler agent, link maintenance agent, and bookmark agent. The results from this system are the list of URLs (Universal Resource Locator) along with description about that page. The link in the result page is called context link i.e., the text around a hyperlink within a web page. Forming the context link, based on the user given keyword and the related link that are available in the link repository, should be made and that would be included in the result page as a list of context links. Unlike other search engines, crawler provides context links to the user, according to the users' pursuit. This work is accomplished by storing the users' name along with their search history in the server.*

*The dynamic web cache management scheme is being tested across 30 nodes and its results are discussed. The proposed intelligent crawler is compared with LLIand dynamic web cache scheme and results are discussed. The results achieved from these experiments confirm the efficiency and adaptability of the proposed crawler*

## INTRODUCTION

The number of web pages available in the Internet is tremendously growing day to day and since this is the case, searching relevant information in the Internet is hard task. Usually, searching information in the World Wide Web can be done by searching the list of links crawled and sorted based on type of the web page or contents of the web page. Today, the main problems of search engines are the size and the rate of change in the web page daily. Web crawler is the process used by web search engines to index pages from the web. Web crawler takes approximately more than a week to crawl all the web pages in Internet once. The changes to the web pages are very frequently occurred today, so providing more relevant information for a specific request by a search engine is a challenging issue.

Intellect Webbot will overcome from above mentioned problems. Each agent will perform its own task in order to crawl the web information. The user needs to give their keyword for search, the crawler will search the links related with the help of link resolver agent to identify the Universal Resource Locators (URLs) available in the link repository. Intellect Webbot helps to crawl the web pages available in the Internet using distributed intelligent agents, which is used to index the web pages in the server with the updated information and produce the effective list of links as a result based on quality metrics such as speed, quality information, or related information.

A crawler is a program that retrieves and stores pages from the Web, commonly for a Web search engine. A crawler often has to download hundreds of millions of pages in a short period of time and has to constantly monitor and refresh the downloaded pages [3]. Context of a hyperlink or link context is defined as the terms that appear in the text around a hyperlink within a Web page.

## EXISTING METHODS

Latent Linkage Algorithm (LLI) can be used for efficiently retrieving web pages with lesser response time [17]. The system can be built an efficient hyper link - based Algorithm to find the relevant links for a given web page(URL). The algorithm is advantaged with linear Algebra Theories to reveal deeper relationship among the web page to identify relevant links more precisely and efficiently.

Whenever the client request the web page the request will be forwarded to the server. The server will forward it to the web server via ISP and DNS. The response from the web server will be given to the server, which in turn forwarded to the client and the data from the web server will be stored as cookie in the web browser for later use. An HTTP cookie (usually a simple cookie) is a packet of information sent by a web server to the web browser and then sent back by the browser each time it accesses that server. The existing server's uses data structures and BLOB data base management system.

## PROPOSED SYSTEM

In the proposed system we implement a proxy server, which makes use of randomized algorithm that combines the benefit of both RR scheme and utility function. This avoids the need for data structure. RR scheme is used to evict the documents randomly. The utility function assigns to each page a value based on -recentness of use -frequency of use -size of page -cost of fetching.

The second step is to build an intelligent webrobot using agents for different functionalities. We have implemented LLI algorithm and a model for dynamic web caching and compared the results with proposed intelligent webrobot.

**Intellect Webbot: Analysis & Design**

The architecture of web crawler should be broadly divided into two architectures, one is cyclic architecture as in Fig. 1 and another one is is detailed architecture as in Fig. 2. First step is the collecting pages from the World Wide Web, and then indexing all the pages. If any search keyword given, crawler will be able to give a list of relevant links by searching in this index. The following Fig. 1 clearly explicates the architecture.

Initially, crawler collects or indexes all the pages from World Wide Web, based on the downloaded information, crawler prepares database called link repository. Repository has the type of web page with corresponding link identification number. If a user request some information related to a specific topic crawler will search based topic given by the user.

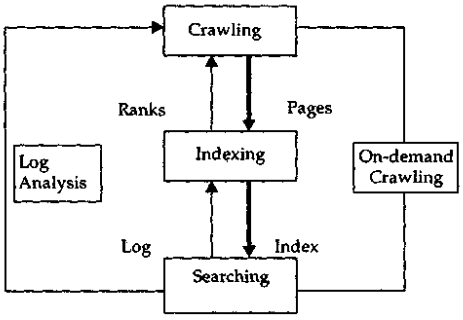



Fig1.

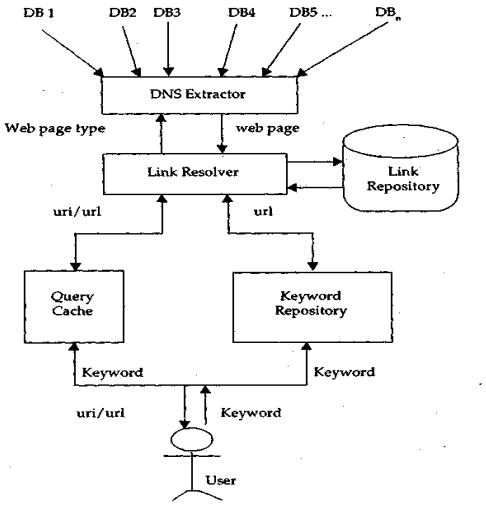
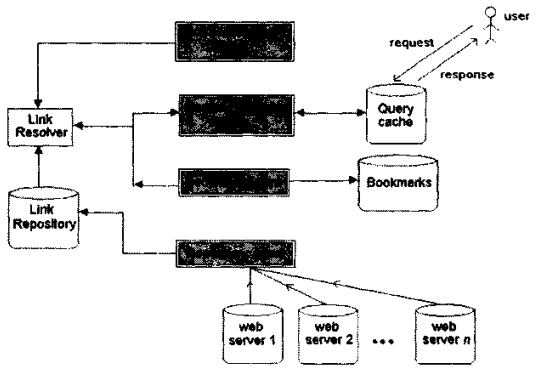



Fig2.

If result is not sufficient for the user, it will get results from link repository. Figure 2 clearly depicts the working of Intellect Webbot.

**Difficulties in Crawling**

There are two important characteristics of the Web that generate a scenario in which web crawling is very difficult: its large volume and its rate of change. The large volume implies that the crawler can only index a fraction of the Web pages within a given time, so it needs to prioritize work. The high rate of change implies that by the time the crawler is indexing the last pages from a site, it is very likely that new pages have been added to the site, or that pages have already been updated or even deleted. Figure 2 depicts the detailed architecture of the web crawler.




Based on the link id crawler will retrieve the full Name and description of the web page with the help of DNS Extractor.

**INTELLECT WEBBOT ARCHITECTURE**

The architecture contains four agents such as Link repository agent, Regional crawler agent, Link maintenance agent, and Bookmark agent. Figure 3 depicts the proposed architecture; double lined rectangle denotes the proposed agents in our architecture.

**Role of Agents**

The role of agents is to provide the faster indexing of web pages in link repository by implementing distributed web crawlers. By enabling agent control we store the user search history and bookmark results.

**Link Repository Agent**

This agent runs on the server, which collects the list of keywords from WebPages stored in webserver and indexes it in the link repository. This will have the list of keywords and corresponding URL. It ignores the unwanted phrases and prepositions, etc. while accumulating the keywords

**Regional Crawler Agent**

The regional crawler agent is responsible for the grouping of the recently searched keywords and the corresponding URLs for a particular user. These details will be updated in the server whenever the user logs out of our system. Once the user enters a keyword and initiates search operation, the request is sent to the cache. The matching process will be done in cache (client) to list the recently fetched results. If not found, the request is sent to link, repository to list the results. After returning the results the user log is updated with recently searched information. The cache is maintained by using the LRU algorithm.

**Link Maintenance Agent**

The Link Maintenance Agent is responsible for the periodic updating process. It will periodically check for updates in the web server and sustains the update on links lost and links redirected. It will send the links present in the cache to the server and stores the updated content in the user log.

**Bookmark Agent**

The bookmark agent is responsible for maintaining the bookmarked URLs by the user. The bookmarked URLs are stored corresponding with the username in the server. Whenever the user gets the resultant links, user can bookmark the URL and the user can access the bookmarked URLs anywhere through our system.

**CRAWLING POLICIES**

**The behavior of a web crawler is the outcome of a combination of the following policies:**

- A selection policy that states which pages to download.
- A re-visit policy that states when to check for changes to the pages.
- A politeness policy that states how to avoid overloading websites.

• A parcillelization policy that states how to coordinate distributed web crawlers.

Selection Policy

As the size of the web is large, even very famous and large search engine covers the portion of the Internet. Not even one search engine searches more than 16% of web content. It is highly desirable that, from that crawled portion, finding relevant information. A good selection policy is required in order to do the best crawling task.

This requires a metric of importance for prioritizing Web pages. The importance of a page is a function of its intrinsic quality, its popularity in terms of links or visits, and even of its URL (the latter is the case of vertical search engines restricted to a single top-level domain, or search engines restricted to a fixed Website).

Najork and Wiener [12] performed an actual crawl on 328 million pages, using breadth-first ordering. They found that a breadth-first crawl captures pages with high Pagerank early in the crawl (but they did not compared this strategy against other strategies). The explanation given by the authors for this result is that "the most important pages have many links to them from numerous hosts, and those links will be found early, regardless of on which host or page the crawl originates".

Abiteboul et al. [16] designed a crawling strategy based on an algorithm called OPIC (On-line Page Importance Computation). In OPIC, each page is given an initial sum of "cash" which is distributed equally among the pages it points to. It is similar to a Pagerank computation, but it is faster and is only done in one step.

Boldi et al. [14] used simulation on subsets of the Web of 40 million from the .it domain and 100 million pages from the WebBase crawl, testing breadth-first against random ordering and an omniscient strategy. The winning strategy was breadth-first, although a random ordering also performed surprisingly well. One problem is that the WebBase crawl is biased to the crawler used to gather the data.

The importance of a page for a crawler can also be expressed as a function of the similarity of a page to a given query. This is called "focused crawling". The main problem in focused crawling is that in the context of a Web crawler, we would like to be able to predict the similarity of the text of a given page to the query before actually downloading the page. A possible predictor is the anchor text of links; this was the approach taken by Pinkerton in a crawler developed in the early days of the Web.

**Re-visit Policy**

The Web has a very dynamic nature, and crawling a fraction of the Web can take a long time, usually measured in weeks or months. By the time a Web crawler has finishedits crawl, many events could have happened. The events are characterized as creations, updates and deletions.

Creations: When a page is created, it will not be visible on the public Web space until it is linked, so we assume that at least one page update - adding a link to the new Web page - must occur for a Web page creation to be visible. A Web crawler starts with a set of starting URLs, usually a list of domain names, so registering a domain name can be seen as the act of creating a URL.

Updates: Page changes are difficult to characterize: an update can be either minor, or major. An update is minor if it is at the paragraph or sentence level, so the page is semantically almost the same and references to its content are still valid. On the contrary, in the case of a major update, all references to its content are not valid anymore.

Deletions: A page is deleted if it is removed from the public Web, or if all the links to that page are removed. Note that even if all the links to a page are removed, the page is no longer visible in the Web site, but it will still be visible by the Web crawler. It is almost impossible to detect that a page has lost all its links, as the Web crawler can never tell if links to the target page are not present, or if they are only present in pages that have not been crawled.

Cost functions: From the search engine's point of view, there is a cost associated with not detecting an event, and thus having an outdated copy of a resource. The most used cost functions are freshness and age.

Freshness: This is a binary measure that indicates whether the local copy is accurate or not.

Age: This is a measure that indicates how outdated the local copy.

**IMPLEMENTATION**

Using Java and swing concept, the working model for our paper has been developed. In the working model, it is assumed there are two or more websevers present.

Link repository agent crawls each and every web page, upholds the keywords, and store it in link repository (a table is considered as link table).

Regional crawler agent groups the recently searched keywords and the corresponding URLs for a particular user (a table considered as userlog). The table is maintained by using the LRU algorithm.

Link maintenance agent does the periodic updating process. It will periodically check for updates in the Webserver and sustains the update on links lost and links redirected in link repository as well as user log.

Bookmark agent maintains the bookmarked URLs by the user. The bookmarked URLs are stored corresponding with the username in the server (a table considered asbookmark table). Whenever the user clicks the bookmark button, that particular URL is stored in bookmark table.

In order to make a search, the user needs to login to the server so as to maintain the user details and search log. Each and every query along with the keyword identified by an agent should be stored in query cache. This query cache will referred for each search made by that particular user in order to avoid the increased latency, provided if the same search keyword is given.

**RESULTS AND DISCUSSIONS**

The proposed system was tested with a test bed of 30 users trying to access similar links:

The system utilizes the bandwidth and directs request and retrieves the webpages using Proxy server software that is designed for use. Using the proposed intelligent crawler the response time for accessing keywords in WebPages is listed below:

| Key word | Server Response in Time (ms) | Cache response (size in bytes) | Web Site |
|---|---|---|---|
| Vista | 203 | 16 | www.microsoft.com |
| Operator in C | 172 | 16 | www.cs.umd.edu |
| Operator in C | 63 | 16 | www.cprogramming.com |

| | | | |
|---|---|---|---|
| Yahoo | 4.56 | 26 | www.yahoo.com |
| Gmail | 4.2 | 16 | www.google.com |
| Vlb | 2.4 | 27 | www.vlb.edu |
| Infoline | 2.1 | 100 | www.indianinfoline.com |
| Cricket | 3.5 | 120 | www.cricinfo.com |

The response time in comparison with LLI [17] and dynamic web cache management using randomized web cache management schemes [18] is shown below:

| Key word | Intelligent Crawler (ms) | Dynamic Web Cache Scheme | LLI (Latent Linkage Algorithm) |
|---|---|---|---|
| Vista | 203 | 205 | 210 |
| Operator in C | 172 | 190 | 180 |
| Operator in C | 63 | 70 | 70 |
| Yahoo | 4.56 | 5 | 5.3 |
| Gmail | 4.2 | 4.4 | 4.5 |
| Vlb | 2.4 | 3 | 3.2 |
| Infoline | 2.1 | 2 | 3 |

**The above result presented has driven for the following conclusions:**

1. The web page retrieval time is dependent on the response time of the servers being accessed like yahoo, google etc.
2. The web page content like text, image, video also affects the performance of the proposed model.
3. Since proxy server software does most of actions of managing request, security part is presented at the application level.
4. Dynamic web caching based on past eviction times is not always constant and is dependent on the network bandwidth.

**SUMMARY**

We have presented new agent based architecture for crawling the web and to provide efficient search results to the user which will work by satisfying the quality metrics such as quality or relevant information should be provided at acceptable speed. It will reduce the time for crawler to index web pages from the Internet in a distributed fashion. Also, refreshing to the content of crawler will be quickly done by having various agents, so that crawler will help to produce the more recent links to the user.

The above depicted graph (Fig. 4) clearly shows how the response from the cache as well as the server is differentiated. Moreover, the graph depicts the retrieval time from the cache and the server corresponding to the bandwidth used by the user.

Normally, crawler takes more than a week time to cover some part of the World Wide Web. By implementing Intellect Webbot model, crawler takes less than a week time since it is distributed and also it always give a most recent link as a result for the search.

**FUTURE WORK**

This paper describes a new architecture for agent based intelligent crawler. In order to clearly explain the architecture, working model for Intellect Webbot architecture has also been shown. Implementing any crawler in a real time is very tough task than the designing work. It requires getting various permissions from the owners of theweb servers administrators (not for all web servers). It needs huge storable node for keeping the details of the user and the URLs.

The future enhancement to this work will be developing and implementing Intellect Webbot by enabling agents such as link repository agent, regional crawler agent, link maintenance agent, bookmark agent in a distributed fashion. The model can be extending for incorporating attacks for large repository of web pages stored for a large organization.

**REFERENCES**

Alex Homer, Dave Sussman, Rob Howard, Brian Francis, Karli Watson and Richard Anderson, Professional ASP.NET1.1 from Wrox Publishing. | [2] A l e x L . G . Hayzelden and Rachel A. Bourne, Agent Technology for Communication Infrastructures, John Wiley & Sons, 1999. | [3]Birukov. A, Enrico Blanzieri and Paolo Giorgini, Implict: An Agent Based Recommendation System for Web Search. AAMAS'05. | [4]Carlos Castillo, Effective Web Crawling, Technical Publication. Department of Computer Science -University of Chile, November 2004. | [5]Gautam Pant and PadminiSrinivasan, Link Contexts in Classifier-Guided Topical Crawlers, /£££ Transaction on Knowledge and Data Engineering, Vol. 18 No. 1, pp. 81-92, January 2006. | [6]Junghoo Cho, Crawling the Web: Discovery and Maintenance of Large Scale Web Data, Technical Publication, November 2001. | [7] Junghoo Cho, Hector Garcya-Molina and Lawrence Page, Efficient Crawling Through URL Ordering, Proceedings of the Seventh Conference on World Wide Web, Brisbane, Australia, April 1998. | [8]Liren Chen and Katia Sycara, Web Mate: A Personal Agent for Browsing and Searching, Transaction on Internet Technology, pp. 1-16, September 1997. | [9] Mathias Bauer et al., Instructible Information Agents for Web Mining, /£££ Transaction on Data Mining, Vol. 10, No. 5, pp. 23-33, October 1999. | [10]Marc Najork and Janet L. Wiener, Breadth-first Crawling Yields High-quality Pages, Proceedings of the Tenth Conference on World Wide Web, Hong Kong, pp. 114-118, May 2001, Elsevier Science. | [11]Marina Buzzi. Cooperative Crawling, Proceedings of the First Latin American Web Congress (LA-WEB 2003). | [12]Paolo Boldi, Bruno Codenotti, Massimo Santini and SebastianoVigna, Ubi Crawler: A Scalable Fully Distributed Web Crawler, /£££ Transaction on Internet Technology, pp. 1-14. | [13]Paolo Boldi, et al., "Do Your Worst to Make the Best: Paradoxical Effects in Pagerank Incremental Computations", Vol. 3243, Springer Publ., Lecture Notes in Computer Science, pp. 168-180, October 2004. | [14]Ricardo Baeza-Yates, Carlos Castillo and Felipe Saint-Jean. Web Dynamics, Clip. Web Dynamics, Structure and Page Quality, pp. 93-109, Springer, 2004. | [15]Serge Abileboul, Mihai Freda and Gregory Cobena, Adaptive On-line Page Importance Computation. Proceedings of the Twelfth International Conference on World Wide Web, pp. 280-290, ACM Press, 2003. | [16]http://en.ivikipedia.org/ | [17]JingyuHou and Yanchun Zhang, Effectively Finding Relevant Web Pages from Linkage Information, /£££ Transactions on Knowledge and Data Engineering, Vol. 15, No. 4, July/August 2003. | [18] BalajiPrabhakaran, "Efficient Randomized Web-cache Replacement Schemes Using Samples From Past Eviction Time", IEEE/ACM Traits. Networking, Vol. 10, pp. 441-453, August 2002. |