Page Relevance based on Page Accessing Frequency

Nilufar Yasmin
Student, M.Tech
Manav Rachna College of Engineering, Faridabad, India

Komal Sachdeva
Assistant Professor
Manav Rachna College of Engineering, Faridabad, India

ABSTRACT
With the tremendous growth of the Internet, many web pages are available online. Search engines use web crawlers to collect these web pages from web for the purpose of storage and indexing. Most of the web pages are autonomous and are updated independently of the users that access the sources .As the web pages are updated autonomously, users are unaware of how often the sources change. An incremental crawler visits the web repeatedly after a specific interval for updating its collection. Users can be benefitted by knowing the page importance based upon the page accessing frequency. In this paper the page relevancy is finding out by a novel mechanism and a novel architecture is being proposed.

General Terms
Page Importance Based On Page Accessing Frequency**.**

Keywords
Search Engine, Web Crawler, Page access Frequency, Rank Updater.

1. INTRODUCTION
The World Wide Web [2, 3, 12] is a system that contains interlink hypertext documents, it uses web browser for accessing the hypertext documents from web servers. These hypertext documents can have text, images, also audio and video data. Hyperlinks allow the user to connect to the interconnected links in back and forth manner.

The web is a large repository of text documents, images, multimedia and vast amount of other information. Because the web contains very large number of web pages, search engine depends upon crawlers for gathering all the required web pages on the web.

A web crawler [13, 14] is a program that automatically retrieves and stores web pages and it creates the local collection of web pages. A crawler starts by getting an initial set of URLs, called as seed URLs. The crawler firstly retrieves the pages matched with the seed URL, then it extracts all the links that are present on the crawled web page and adds the new retrieved URLs to a queue for the further scan process.

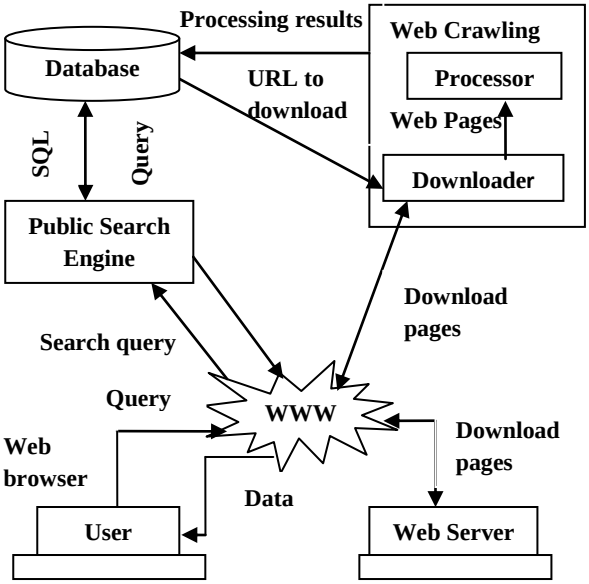

Fig 1: Architecture of search engine

In Fig 1, the search engine starts its searching by accepting the query from the user. The public search engine provides an interface to the user so that user submits the queries, and it contains the mechanism for serving these queries. This part is an important part for the search engines, because it is the only visible part to the end-users.

The database stores all the crawled web pages, crawled by the web crawler. The search engine sent queries to the database so that it answers to any user request. The database also feeds the downloader with the URLs to be downloaded .The processor processes the URLs taken from the downloader and updates the database with the fresh information (URLs).

In case of search engine, a web crawler is a component that downloads and stores the web pages for the search engine. A crawler starts by taking initial set of URLs into the queue, where all the retrieved URLs are kept and also prioritized. The crawler takes URL form this queue, downloads the web pages, extracts all the URLs present in the downloaded page, and place the new URLs in the queue. Crawler repeats this process until it crawls a desirable no of web pages. These collected web pages are later used by search engine or a Web cache.

The basic algorithm for the Web Crawler is given below:

1. Read a URL from the set of seed URLs.

2. Find out the IP address for the host name.
3. Download the Robot.txt file that carries downloading permissions and also identifies the files to be excluded by the crawler.
4. Find out the protocol of underlying host like http, ftp, gopher etc.
5. Based on the protocol of the host, download the document.
6. Identify the document format like doc, html, or PDF etc.
7. Verify whether the document has previously been downloaded or not.
8. If the document is fresh one
   Then Read it and extracts the links.
9. Else
   Continue.
10. Convert the URL links into their absolute IP equivalents.
11. Add the URLs to queue of the set of seed URLs.

In this paper we discuss about estimating page importance based on page access frequency with the help of an incremental crawler. In section 2 we discuss about the approaches that are already in use in the incremental crawler for finding the relevant web pages. In section 3 we propose a novel mechanism and novel architecture for finding relevant web pages which is based on number of user accesses.

2. RELATED WORK

The incremental crawler updates the web pages of their local collection in a discriminately and incrementally manner. It does not periodically refresh the collection, but improves the "newness" of the local collection and fetches new pages in the local collection in more appropriate manner.

The architecture of Incremental Crawler is shown in Fig 2.

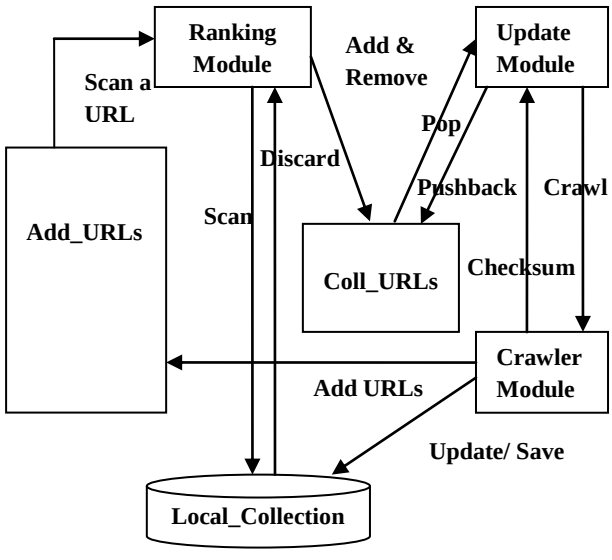

Fig 2: Architecture of Incremental Crawler

The All_URLs contains the set of all URLs that are accessed or to be accessed. The URLs in the Coll_URLs are chosen by the Ranking Module. The pages related to URLs in the Coll_URLs are downloaded by the Local_Collection. The Ranking Module continuously scans the All_URLs and the Local_collection and gives proper ranks to the web pages accordingly. When the Ranking Module finds that the page present in the Coll_URLs are not the updated pages, then the page in the Coll_URLs are replaced by the updated pages.
The Update Module takes the URLs from the Coll_URls, downloads the pages related to the URL, if the page changed, then it will be updated in the Local_collection. The Crawler Module has to be crawls the web page and it updates or saves the web page into the Local_collection according to the request of Update Module. It also extracts all the links (URLs) on the crawled page and adds those links in All_URLs.

While designing the incremental crawler [2, 6] two issues must be address:-

Maintain the local collection with fresh pages: Freshness of web pages in local collection is based on the strategy that used for it, that why the crawler should apply the best policies to maintain the local collection fresh.
In order to maintain the freshness of local collection, Revisit Frequency Calculator is use to finds the appropriate revisit frequency of the crawling so that crawler can update its local collection with fresh documents.

Improve quality by keeping relevant pages in the local collection: The web crawler should be improving the quality of the local collection by replacing less relevant pages with more relevant pages. This should be done so that the user can access more relevant pages while searching. It is essential because pages are continuously created and removed, and it may possible that some of the pages that were created may be more relevant than existing pages in the local collection. That's why the crawler needs to replace less relevant existing pages with more relevant new web pages.
Another reason is that, the relevancy of existing pages also changes over time. Thus, when some existing pages become less relevant than earlier ignored pages, then the web crawler should replace less relevant existing pages with earlier ignored new web pages.

By using hyperlink graph of the web and algorithms like Page Rank algorithm, page importance is calculated. But now a day's people understand that the hyperlink graph is an incomplete and inaccurate way for finding page importance.

In approach, [1] it proposed a way to calculate page rank in incremental crawler .According to it, the hit counter stores the number of times the page is visited. And there must be some date on which that page is added on the web.

For estimating the page importance of that page it divides the hit counter by the total number of days for that page on the web. This will evaluate the access ratio of that page for the web.

Another approach, [3] it solves the problem of searching fresh information from the web in the incremental web search for estimating ranking of web pages which is changed. For solving this problem, it uses an integrated ranking framework by merging three metrics. The three metrics are Popularity Ranking, Content-based Ranking and Evolution Ranking which produce good Ranking for the changed web Pages.

3. PROPOSED WORK

There are three important characteristics of the web page that generate a scenario in which the web crawling is very difficult:

- Large Volume of web pages.
- Rate of change on web pages.
- Finding the relevant pages of the web.

A large volume of web pages implies that the web crawler can only download a fractional of the web pages and hence it is very essential that the crawler should be intelligently enough to prioritize download.

Whereas rate of change on web pages implies that web pages on the internet changes very frequently, as a result, by the time the crawler is downloading the last page from a site, the web page may change or a new page has been placed/ updated to the website.

Relevant pages of the web represents the 'value' of an individual page on the web, is a key factor for web search. Various search engines components such as, the crawler, indexer, and ranker are generally guided by this measure. Because the web is extremely large, and the web change dynamically, accurately calculating the importance of web pages becomes difficult, and it also creates a great challenge to web search engines.
Above difficulties can try to be solved with the proposed architecture.
The architecture is composed of the following Modules/Data Structure as shown in Fig 3.

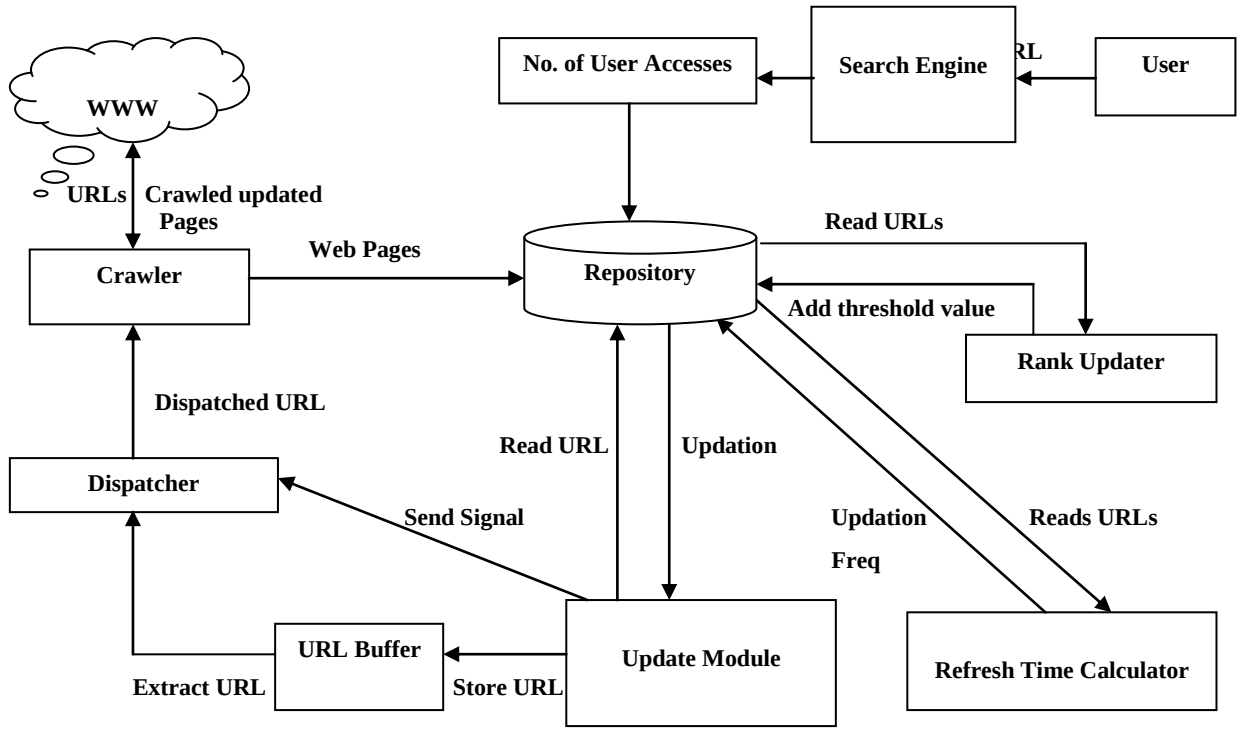

Fig 3: Architecture of proposed Incremental Crawler

2.1 Crawler Module

It crawl the web, download the web pages and stores the downloaded pages into the Page repository that maintains the documents crawled/updated by the "Crawler" along with their URLs.

Crawler ()
{
Read a URL from Repository
Download new page from the web and extracts new URLs found in the page
If URL exists in Repository
Then
If page changed, updates it in Repository
Else
Add URL, page to Repository
Add/replace new URLs to Repository
}

2.2 Update Module

Administrator is able to do the updation to the web pages with the help of Update Module. The Update Module read the URLs to check whether the web pages are changed or not, and if changed then it should be store the changed URLs to the URL Buffer for crawling the changed web pages. When this buffer will become full, the Update Module will send a signal called Fetch URL signal to Dispatcher. After getting the Fetch URL signal from Update Module, the Dispatcher will start fetching the URLs one by one and forward it to the crawler to download the web page again, so that the freshness of Page repository can maintain.

Update ()

```
{
 Wait (Repository empty)
 Fetch URLs and their link information from Repository
if URL is updated
Then add URL in URL Buffer
Else
Continue
If full (URL Buffer) then
Signal (fetch URL)
}
}
```

2.3 Dispatcher Module

It waits for the fetch URL signal from Update module and upon receiving this signal, it fetches an URL from the URL Buffer so that crawler can download the corresponding web pages. The algorithm for Dispatcher is as follows:

```
Dispatcher ()
{
 Wait (fetch URL)
 While (not (empty URL Buffer))
{
Fetch URL from Buffer;
Forward the URL to the crawler to download all the web pages related to that URL.
```

2.4 URL Buffer Module

It stores all the updated URLs which are need to be recrawled by the crawler. The URL Buffer is feeding by the Update Module.

2.5 Rank Updater Module

Rank Updater checks the repository after a particular interval of time. It calculates the difference between the no. of user accesses of before updation which is denoted by a1 and after updation, denoted by a2 of the URLs. And if the difference between a1 and a2 is equal to the constant value that is decided by the administrator then it will be add the threshold value to the no. of user accesses of after updation of those URLs. Then rank updater updates the relevancy of the web pages according to it. The algorithm for the Rank Updater is as follows.

```
Rank Updater ()
{
 While (URLs) // accept URLs from Repository
{
Find the no_of _accesses a1 // before updation
Find the no_of _accesses a2 // after updation
Diff = a1 - a2
If Diff > τ // τ equals to constant value
Then
 Add threshold value to the no_of_accesses of after updation of the URL.
```

2.6 Search Engine Module

It is used to the proposed architecture for calculating the no. of user accesses. The no_of_accesses is calculated as follows:

```
Init () /* initialize variables */
N = 0 /* total number of accesses */
After first search
N = N + 1
```

2.7 Repository:

The Repository stores all the web pages and related URLs that crawled by the crawler.

4. CONCLUSION & FUTURE WORK

The proposed architecture helps in maintaining the freshness of the repository and provides relevant web pages to the users. The calculation of the no. of user accesses helps in finding the importance of the web pages by efficiently managing the Rank Updater so that the relevant pages are provided to the users. Moreover, the architecture is suitable for the applications where relevant search have been required.

As the future work, the limitation of the current work that the crawler crawl the web pages even after the administrator stops the crawling process is taken into account. Future work includes applying a technique to overcome this problem.

5. REFERENCES

[1] Sakshi Goel, Anjana, Akhil Kaushik, Kirtika Goel, "A Novel Approach for Page Rank in Incremental Crawler", IJCST Vol. 3, Issue 1, Jan. - March 2012.

[2] Niraj Singhal, Ashutosh Dixit, Dr. A. K. Sharma, "Design of a Priority Based Frequency Regulated Incremental Crawler", 2010 International Journal of Computer Applications (0975 – 8887) Volume 1 – No. 1.

[3] Arvind Kumar, Km. Pooja, "An effective method for ranking of changed web pages in incremental crawler", International Journal of Computer Applications (0975 – 8887) Volume 8– No.7, October 2010.

[4] Rosy Madaan, Ashutosh Dixit, A.K. Sharma, Komal Kumar Bhatia, "A Framework for Incremental Hidden Web Crawler", International Journal of Computer Science and Engineering (IJCSNE), Vol. 02, No. 03, 2010.

[5] Ravita Chahar, Komal Hooda, Annu Dhankhar, "Management Of Volatile Information In Incremental Web Crawler", IJCSI International Journal of Computer Science Issues, Vol. 4, No. 1, 2009(ISSN (Online): 1694-0784, ISSN (Print): 1694-0814).

[6] Ashutosh Dixit, Harish Kumar and A.K Sharma, "Self Adjusting Refresh Time Based Architecture For Incremental Web Crawler", International Journal of Computer Science and Network Security (IJCSNS), Vol 8, No12, Dec 2008.

[7] M.P.S.Bhatia, Divya Gupta, "Discussion on Web Crawlers of Search Engine". Proceedings of 2nd National Conference on Challenges & Opportunities in Information Technology (COIT-2008) RIMT-IET, Mandi Gobindgarh. March 29, 2008.

[8] Cho, J. and Roy, "Impact of search engines on page popularity". In Proc.13th International World Wide Web Conference, 2004.

[9] Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, S. Raghavan, "Searching the Web", ACM Transactions on Internet Technology, Vol. 1, Num. 1, August 2001, pp.2-43.

[10] Mark Najork, Allan Heydon, "High- Performance Web Crawling", September 2001.

[11] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Effective personalization based on association rule

discovery from web usage data. In Proceedings of the 3rd ACM Workhop on Web Information and Data Management, pages 9–15, November 2001. Atlanta, USA.

[12] Arvind Arasu, Junghoo Cho, Hector Garcia-Molina, Andreas Paepcke, and Sriram Raghavan, "Searching the Web", ACM Transactions on Internet Technology (TOIT), 1(1):2–43, August 2001.

[13] Junghoo Cho and Hector Garcia-Molina, "Estimating frequency of change", 2000, Submitted to VLDB 2000, Research track.

[14] Junghoo Cho and Hector Garcia-Molina. 2000a. "The evolution of the web and implications for an incremental crawler"., In Proceedings of the 26th International Conference on Very Large Databases.

[15] Brian E. Brewington and George Cybenko. "How dynamic is the web." In Proceedings of the Ninth International World-Wide Web Conference, Amsterdam, Netherlands, May 2000.

[16] Henzinger M. R. Link analysis in web information retrieval. IEEE Data Engineering Bulletin, 23(3):3-8, September 2000.

[17] Jenny Edwards, Kevin McCurley, John Tomlin, "An Adaptive Model for Optimizing Performance of an Incremental Web Crawler".

[18] Brin, Sergey and Page Lawrence, "The anatomy of a large-scale hypertextual Web search engine". Computer Networks and ISDN Systems, April 1998.

[19] Mike, Burner, "Crawling towards Eternity : Building an archive of the World Wide Web", Web Techniques Magazine, 2(5), May 1997.