ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

Constructing and Maintaining Large Web Repositories through Continuous Web Crawling

J.Tamilselvan[#1], Dr. A.Senthirajan[*2]
[1]*Department of Computer Science*
[2]*Director, Computer Centre*
[1]tamilselvan@ymail.com
[2]agni_senthil@yahoo.com
[1]*K.S.Rangasamy College of Arts and Science (Autonomous)*
Tiruchengode, Tamilnadu, India
[2]*Alagappa University, Karaikudi, Tamilnadu, India*

***Abstract*— This paper describes a scalable, extensible web crawler with continuous crawling to build and maintain the corpora. While crawling web sites, a crawler has to decide an optimal order in which to crawl and re-crawl web pages. Web crawlers are used for a variety of purposes. Most prominently, they are one of the main components of web search engines, systems that assemble a corpus of web pages, index them, and allow users to issue queries against the index and find the web pages and relevant content that match the queries.**

***Keywords*— crawler, corpora, web, frontier, seed, URL.**

I. INTRODUCTION

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. In addition, the crawler should avoid putting too much pressure on the visited Web sites and the crawler's local network, because they are intrinsically shared resources.

A Web crawler is a program that downloads Web pages, commonly for a Web search engine or a Web cache. Roughly, a crawler starts off with an initial URL S_1. It first places S_1 in a queue, where all URLs to be retrieved are kept and prioritized. Spidering a website, link by link, will work for most of the websites. However, it can be a kind of tedious to examine each different kind of page to figure out the link structure. But when we do a little survey and experimentation, we may find a pattern in the site's URL that we use to save ourselves a considerable amount of time.

The most obvious examples are sites that paginate their information or with numbered URL parameters. The Judgments Information system consists of the Judgments of the Supreme Court of India and several High Courts has 30,000+ datasets (Judgments). **http://judis.nic.in/supremecourt/imgst.aspx?filename=1** Gets the page of the judgment S_1 **http://judis.nic.in/supremecourt/imgst.aspx?filename=2** Gets the judgment of S_2 and subsequently when the filename (URL parameter value) is incremented with the continuous values for filename it produces the S_3, S_4, S_5 and so on.

A. *Crawler*

The basic operation of any hypertext crawler is as follows. The crawler begins with one or more URLs that constitute a seed set. It picks a URL from this seed set, and then fetches the web page at that URL. The fetched page is then parsed, to extract both the text and the links from the page. The extracted text is fed to a text indexer. The extracted links (URLs) are then added to a URL frontier, which at all times consists of URLs whose corresponding pages have yet to be fetched by the crawler. Initially, the URL frontier contains the seed set; as pages are fetched, the corresponding URLs are deleted from the URL frontier. The entire process may be viewed as traversing the web graph. In continuous crawling, the URL of a fetched page is added back to the frontier for fetching again in the future if needed. This is a simple traversal of the web graph which is complicated by the many demands on a practical web crawling system, the crawler has to be distributed, scalable, efficient, polite, robust and extensible while fetching pages of high quality.

B. *Features of Crawler*

Distributed: The crawler should have the ability to execute in a distributed fashion across multiple machines.
Scalable: The crawler architecture should permit scaling up the crawl rate by adding extra machines and bandwidth.
Performance and efficiency: The crawl system should make efficient use of various system resources including processor, storage and network band-width.
Quality: Given that a significant fraction of all web pages are of poor utility for serving user query needs, the crawler should be biased towards fetching "useful" pages first.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

Freshness: In many applications, the crawler should operate in continuous mode: it should obtain fresh copies of previously fetched pages. A search engine crawler, for instance, can thus ensure that the search engine's index contains a fairly current representation of each indexed web page. For such continuous crawling, a crawler should be able to crawl a page with a frequency that approximates the rate of change of that page.
Extensible: Crawlers should be designed to be extensible in many ways to cope with new data formats, new fetch protocols, and so on. This demands that the crawler architecture be modular.

C. Crawler Architecture

The simple scheme given for crawling demands several modules that fit together as shown in Figure 1 and in Figure 2 the flow of continuous crawler is given.

1. The URL frontier, containing URL to be fetched in the current crawl (for continuous crawling, a URL have been fetched previously).
2. A DNS resolution module that determines the web server from which to fetch the page specified by a URL.
3. A fetch module that uses the http protocol to retrieve the web page at a URL.
4. A parsing module that extracts the text and stores it as corpus.
5. A duplicate elimination module that determines whether an extracted link is already in the URL frontier or has recently been fetched.

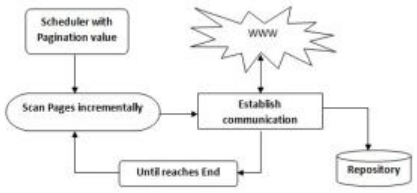

Figure 1: Components of Web Crawler

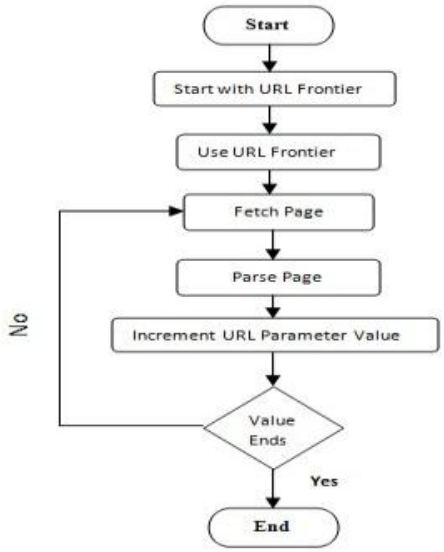

Figure 2: Flow of Continuous Crawler

D. Features Affecting Performance of Web Crawler

A critical look at the available literature [1] [2] [3] indicates the following issues that need to be addressed:
Issue 1: Overlapping of web documents: Overlap problem occurs when multiple crawlers running in parallel download the same web document multiple times.
Issue 2: Quality of downloaded web documents: The quality of downloaded documents can be ensured only when web pages of high relevance are downloaded by the crawlers.
Issue 3: Network bandwidth/traffic problem: In order to maintain the quality, the crawling process is carried out using either of the following approaches:

- Crawlers can be generously allowed to communicate among themselves or
- They cannot be allowed to communicate among themselves at all.

Both approaches put extra burden on network traffic.
Issue 4: Change of web documents: Changing and adding of web documents is a continuous process. This change must be reflected at the repository failing which a user may have to access an obsolete web document.

II. RELATED WORK

Web crawlers also known as robots, spiders, worms, walkers, and wanderers are almost as old as the web itself. Lots of previous work has focused on the crawling ordering strategy so far [5][6].. The first crawler, Matthew Gray's

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

Wanderer, was written in 1993, roughly coinciding with the first release of NCSA Mosaic [4]. Several papers about web crawling were presented at the first two World Wide Web conferences [7,8,9]. However, at the time, the web was two to three orders of degree smaller than it is today, so those systems did not address the scaling problems inherent in a crawl of today's web to improve the performance, but still these algorithms are computationally expensive.

As an alternative here a new URL numbering algorithm is proposed. Major advantage is that it will be relatively inexpensive. Website can process their contents efficiently.

III. PROPOSED ALGORITHM

To achieve better results for above mentioned factors of web crawler a URL numbering algorithm is proposed. In this algorithm a new site rank is calculated which covers all three types of web mining technique i.e. web content mining, web usage mining and web structure mining.

As a result of using all three web mining technique covering all issues it is believed to achieve an efficient site rank and corpora build algorithm.

Algorithm steps are as follows:-

1 Input a URL.
2 Extract whole site.
3 Remove the stop word and suffix.
4 Calculate tern weight using TF-IDF.
5 Now calculate content similarity.

A. *Algorithm Explanation*

A web crawler's working start with a seed URL. Every URL is associated with a web page or site. Then content of page are downloaded. We know that all content are not important. To weight the page in accordance to importance its stop word and suffix are removed. By this content to be used for ranking and querying it become less in size and more relevant.

1) *Stoplisting and Stemming*

When parsing a web page to extract content information or in order to score new URLs suggested by the page, it is often helpful to remove commonly used words or *stopwords*. This process of removing stopwords from text is called *stoplisting*. In addition to stoplisting, one may also stem the words found in the page. The *stemming* process normalizes words by conflating a number of morphologically similar words to a single root form or stem.

2) *TF - IDF (Term Frequency – Inverse Document Frequency)*

In information retrieval, the term frequency – inverse document frequency also called tf-idf, is a well known method to evaluate how important is a word in a document (page). tf-idf are also a way to convert the textual representation of information into a Vector Space Model (VSM).

The first step in modeling the document into a vector space is to create a dictionary of terms present in documents. To do that, all terms from the document are selected and converted it to a dimension in the vector space.

Term weight $w_i = tf_i * \log (D/df_i)$

Where

tfi = term frequency (term counts) or number of times a term *i* occurs in a document.

dfi = document frequency or number of documents containing term *i*.

D = number of documents in a database.

Weights are represented as the normalized product of Logarithmic Term Frequency and Inverse Document Frequency (L.T.F.-I.D.F.). The tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document. It is a way to score the importance of words (or "terms") in a document based on how frequently they appear across multiple documents.

Finding Word Associativity: Word associativity are measured by the probability of simultaneous occurrences of words present in the corpora.

$$P(w_1,w_2)=1/W \sum_{i=1}^{T} P(w_1|z_i)*(z_i|w_2))$$

Given a particular measure of page importance we can summarize the performance of the crawler with metrics that are analogous to the information retrieval (IR) measures of *precision* and *recall*. Many authors provide precision-like measures that are easier to compute in order to evaluate the crawlers.

Acquisition rate: In cases where we have Boolean relevance scores we could measure the explicit rate at which "good" pages are found. Therefore, if 50 relevant pages are found in the first 500 pages crawled, then we have an acquisition rate or *harvest rate* [1] of 10% at 500 pages.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

IV. EXPERIMENTS

A. *Data Collection*

In this section, experimental studies which will be carried on real data that will be acquired from internet by proposed crawler and it is the Data set which can be used for any kind of Information Retrieval. The proposed URL ordering crawler will be checked with the density of the word packed by the related documents built text corpora.

Web Crawler is implemented in Python on windows platform and experiments are done on Intel core2duo series CPU with 3GB RAM.

B. *Evaluation Method*

In order to measure the performance of the proposed ranking algorithm, it can be evaluated in two ways. First, top 100 URLs returned by the above mentioned algorithm will be used. A criterion is framed such as similarity to descriptions of relevant pages. This will be indication for site recommendation. Also, pages of spam sites should be identified. Minimum number of overlapping document, more relevant page, less traffic consume less bandwidth and most updated page storage are to be considered as far as this type of continuous ordering crawlers produce mere common contents as the crawler fetches from the same site of different contents.

The time taken by the crawler is completely based on the interval of visits to the same server; this interval is the most effective way of avoiding server overload. Commercial search engines like Google, Ask Jeeves, MSN and Yahoo! Search are able to use an extra "Crawl-delay" parameter in the robots.txt file to indicate the number of seconds to delay between requests.

V. RESULTS

As all the three web mining technique are employed in the above algorithm. Using website logs is inexpensive. Semantic relevance chooses more accurate probability. According to their relevance a weight factor is multiplied to obtain more accurate site score. Weight factor also plays an important role in obtaining more precious results. It is expected that it will give better result. It is able to fulfill those above mentioned issues. First is less overlapping, to be obtained as different content, page popularity and update frequency give precious score. Secondly, a good score will help in download a highly relevant page first, so better quality expected. Thirdly, when sites are carefully prioritized there are chances of less ambiguousness and frequent unnecessary traffic can be avoided. Finally, change frequency is also taken into consideration which helps to retrieve most updated page.

VI. CONCLUSION AND FUTURE WORK

In this paper we have proposed architecture for the web - crawling using continuous crawler techniques. This paper also shows the method to convert unstructured data into structured data. A new URL continuous ordering algorithm is proposed based on the content similarity, popularity information from web logs and site updating frequency. It is expected to perform well and better than traditional crawlers which are used to build web corpora. It also has a drawback that pages of different languages has not been accessed are dealt severely and also do not have good updating frequency. Focused crawling, proposed by Chakrabati([10]), is designed to narrow the acquisition to web segments that represent a specific topic. Only few approaches are known for language specific crawling. Our opinion is that focusing on a specific language and domain area are more specific when the content of the documents is taken into account.

REFERENCES

[1] Bhaskar Reddy, Kethi Reddy, "Improving efficiency of web crawler algorithm using parametric variations" Ph.d thesis submitted in June 2010 at Thapar University India.
[2] Shaojie Qiao, Tianni Li, Jiangtao Qiu, "SimRank: A Page Rank Approach based on Similarity Measure" 2010 IEEE.
[3] Hongzhi Guo, Qingcai Chen, Xiaolong Wang, Zhiyong Wang, Yonghui Wu, "STRank: A SiteRank Algorithm using Semantic Relevance and Time Frequency" Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics.
[4] Internet Growth and Statistics: Credits and Background. http://www.mit.edu/people/mkgray/net/background.html
[5] DilipKumar Sharma, A.K.Sharma "A Comparative Analysis of Web Page Ranking Algorithms" (IJCSE) International
Journal on Computer Science and Engineering Vol. 02, No. 08, 2010, 2670 - 2676
[6] Neelam Duhan, A. K. Sharma, Komal Kumar Bhatia, "Page Ranking Algorithms: A Survey", 2009 IEEE International Advance Computing Conference (IACC 2009).
[7] David Eichmann, "The RBSE Spider -- Balancing Effective Search Against Web Load", In Proceedings of the First International World Wide Web Conference, pages 113--120, 1994.
[8] Oliver A. McBryan, "GENVL and WWW: Tools for Taming the Web", In Proceedings of the First International World Wide Web Conference, pages 79--90, 1994.
[9] Brian Pinkerton, "Finding What People Want: Experiences with the WebCrawler", In Proceedings of the Second International World Wide Web Conference, 1994.
[10] S.Chakrabarti, M.VandenBerg, and B.Dom, "Focused crawling: a new approach to topic-specific Web resource discovery", Computer Networks (Amsterdam, Netherlands:1999), 31(11–16):1623–640,1999.