IOSR Journal of Engineering
e-ISSN: 2250-3021, **p**-ISSN: 2278-8719,
Vol. 2, Issue 12 (Dec. 2012), ||V4|| PP 57-62

Study of Webcrawler: Implementation of Efficient and Fast Crawler

Vineet Singh, Ayushi Srivastava, Suman Yadav
Department Of Information Technology,
Institute Of Technology And Management Gida Gorakhpur, Up , India

Abstract:- **A focused crawler is a web crawler that attempts to download only web pages that are relevant to a pre-defined topic or set of topics. Focused crawling also assumes that some labeled examples of relevant and not relevant pages are available. The topic can be represent by a set of keywords (we call them seed keywords) or example urls. The key for designing an efficient focus crawler is how to judge whether a web pages is relevant to the topic or not. It defines several relevance computation strategies and provides an empirical evaluation which has shown promising results. We developed a framework to fairly evaluate topical crawling algorithms under a number of performance metrics. Evolutionary crawlers achieve high efficiency and scalability by distributing the work across concurrent agents, resulting in the best performance/cost ratio.**

Keywords:- *Focus Crawler, Urls, Pbel, Web Spider, Seed, Frontier.*

I. INTRODUCTION

With the exponential growth of information on the World Wide Web, there is a great demand for developing efficient and effective methods to organize and retrieve the information available. A Web crawler is a computer program that browses the World Wide Web in a methodical, automated manner or in an orderly fashion. Because of limited computing resources and limited time, focused crawler has been developed. Focused crawler carefully decides which URLs to scan and in what order to pursue based on previously downloaded pages information. An effective crawler should have high precision (all the retrieved Web pages should belong to the topic) and recall (most of the relevant information available on the topic should be gathered). Moreover the crawler should be efficient; the relevant information should be collected in the least amount of time possible. Other terms for Web crawlers are ants, automatic indexers, bots, Web spiders, Web robots.

This process is called Web crawling or speeding. Many sites, in particular search engines, use speeding as a means of providing up-to-date data. Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages to provide fast searches. Crawlers can also be used for automating maintenance tasks on a Web site, such as checking links or validating HTML code. Also, crawlers can be used to gather specific types of information from Web pages, such as harvesting e-mail addresses (usually for sending spam). Focused crawling goes a step further than the classic approach. It is able to crawl particular topical (focused) portions of the World Wide Web quickly. WebCrawler is a metasearch engine that blends the top search results from Google, Yahoo!, Bing Search (formerly MSN Search and Live Search), Ask.com, About.com, MIVA, LookSmart and other popular search engines. WebCrawler also provides users the option to search for images, audio, video, news, yellow pages and white pages. WebCrawler is a registered trademark of InfoSpace.

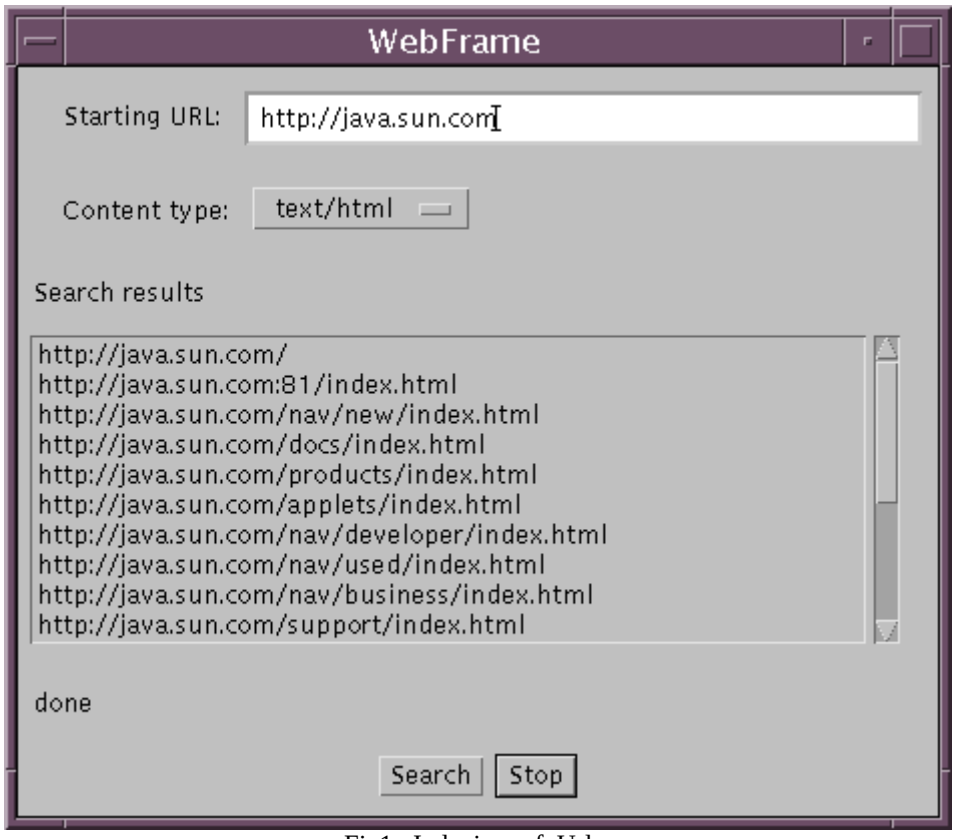

Fig1. Indexing of Urls.

II. HOW WEB CRAWLERS WORK

Web crawlers start by parsing a specified web page, noting any hypertext links on that page that point to other web pages. They then parse those pages for new links, and so on, recursively. Web-crawler software doesn't actually move around to different computers on the Internet, as viruses or intelligent agents do. A crawler resides on a single machine. The crawler simply sends HTTP requests for documents to other machines on the Internet, just as a web browser does when the user clicks on links. All the crawler really does is to automate the process of following links. The method used to exclude robots from a server is to create a file on the server which specifies an access policy for robots. This file must be accessible via HTTP on the local URL "/robots.txt"[5].Web crawlers are an essential component to search engines; running a web crawler is a challenging task. There are tricky performance and reliability issues and even more importantly, there are social issues. Crawling is the most fragile application since it involves interacting with hundreds of thousands of web servers and various name servers, which are all beyond the control of the system. Web crawling speed is governed not only by the speed of one's own Internet connection, but also by the speed of the sites that are to be crawled. Especially if one is a crawling site from multiple servers, the total crawling time can be significantly reduced, if many downloads are done in parallel.

Following Is the Process by Which Web Crawlers Work
1. Download the Web page.
2. Parse through the downloaded page and retrieve all the links.
3. For each link retrieved, repeat the process.

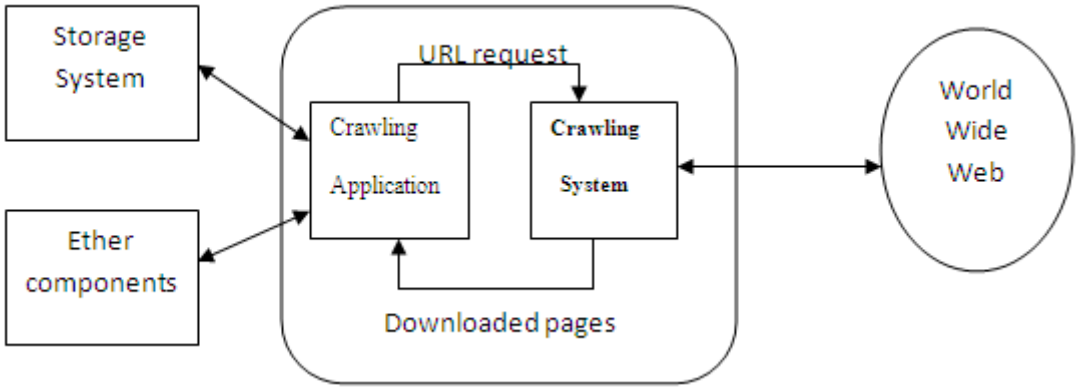

Fig 2. working of crowler

III. EXISTING ALGORITHMS AND TECHNIQUES

We Use the Following Algorithms and Techniques in Our Paper

A. PEBL

Web page classification is one of the essential techniques for Web mining because classifying Web pages of an interesting class is often the first step of mining the Web. However, constructing a classifier for an interesting class requires laborious preprocessing such as collecting positive and negative training examples. For instance, in order to construct a "homepage" classifier, one needs to collect a sample of homepages (positive examples) and a sample of nonhomepages (negative examples). In particular, collecting negative training examples requires arduous work and caution to avoid bias. This technique gives a framework, called Positive Example Based Learning (PEBL), for Web page classification which eliminates the need for manually collecting negative training examples in preprocessing. The PEBL framework applies an algorithm, called Mapping-Convergence (M-C), to achieve high classification accuracy (with positive and unlabeled data) as high as that of a traditional SVM (with positive and negative data). M-C runs in two stages: the mapping stage and convergence stage. In the mapping stage, the algorithm uses a weak classifier that draws an initial approximation of "strong" negative data. Based on the initial approximation, the convergence stage iteratively runs an internal classifier (e.g., SVM) which maximizes margins to progressively improve the approximation of negative data. Thus, the class boundary eventually converges to the true boundary of the positive class in the feature space. We present the M-C algorithm with supporting theoretical and experimental justifications. An experiments show that, given the same set of positive examples, the M-C algorithm outperforms one-class SVMs, and it is almost as accurate as the traditional SVMs[6].

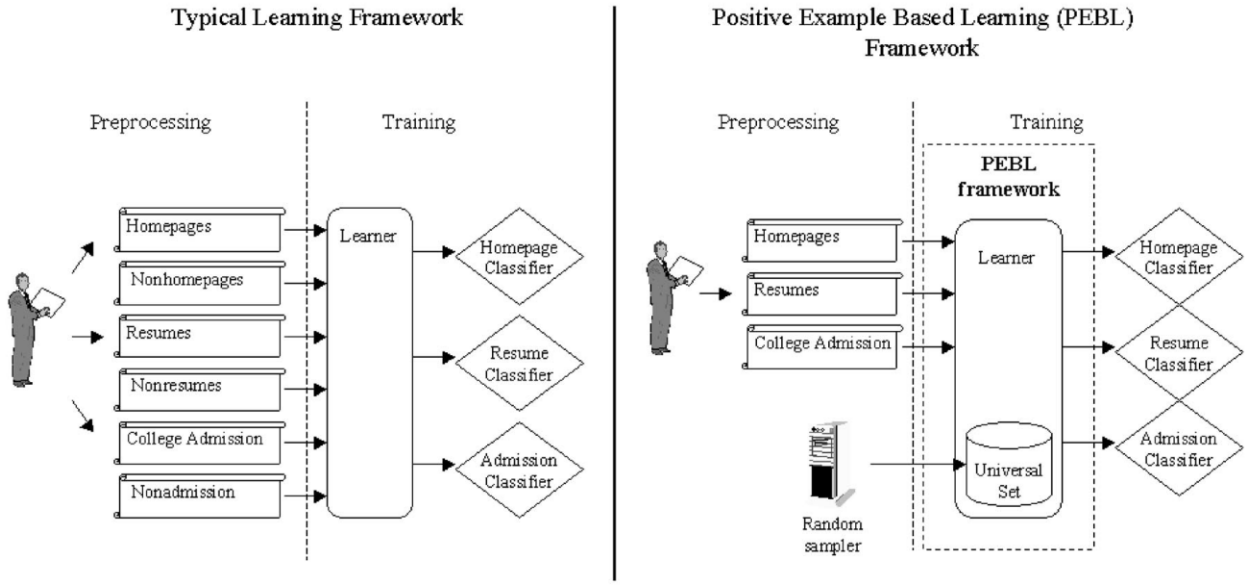

Fig 3. A typical l earning framework versus the Positive Example Based Learning (PEBL) framework. Once a sample of the universal set is collected in PEBL, the same sample is reused as unlabeled data for every class.

In Summary, The Contributions of Our Pebl Framework Are The Following

1. Preprocessing for classifier construction requires collecting only positive examples, which speeds up the entire process of constructing classifiers and also opens a way to support example-based query on the Internet. Fig. 1 shows the difference between a typical learning framework and the PEBL framework for Web page classification. Once a sample of the universal set is collected in PEBL, the sample is reused as unlabeled data for every class, therefore, users would not need to resample the universal set each time they construct a new classifier [3].
2. PEBL achieves accuracy as high as that of a typical framework without loss of efficiency in testing. PEBL runs the M-C algorithm in the training phase to construct an accurate SVM from positive and unlabeled data. Once the SVM is constructed, classification performance in the testing phase will be the same as that of a typical SVM in terms of both accuracy and efficiency [3].

The Algorithm Proceeds as Follows

1. Distinguish positive features based on their frequency in the positive data and the unlabelled data. Use these positive features to retrieve the strong negative points (containing none of the positive features).
2. Set P to the positive points and N to the strong negative points.
3. Learn an SVM based on the P and N as positive and negative points.
4. Classify the remaining points (not contained in N and P) using the current SVM. Points classified as negative are added to the set N.
5. Go to step 3, if points were added to N in the current iteration.

Step 1 is called as the mapping stage while steps 2- 5 are known as the convergence phase. An important advantage of PEBL is that it can use the same random sample of web pages to learn a classifier for any topic of web pages. This helps in our scenario where different crawls for various topics can be initiated without the need to change the unlabeled data. The running time of PEBL is equivalent to the number of iterations needed to segregate the positive and negative points multiplied by the time take by one run of SVM. Once the SVM is constructed, classification performance is equivalent to that of a typical SVM in terms of both accuracy and efficiency. We won't necessarily use SVMs but would use this idea for a particular learning algorithm [6].

B. DEAN AND HENZINGER:

We plan to use techniques similar to the ones suggested in Dean and Henzinger for finding related pages to a set of web pages. They discuss two algorithms, the companion algorithm and the Cocitation Algorithm. Both find related pages on the web given a single URL. We are exploring on extensions to the algorithm where more than one URL can be accommodated.

This algorithm basically of two type

1. Companion Algorithm
2. Cocitation Algorithm

C. VIPS:

We plan to use a VIPS (Vision -based Page Segmentation) like algorithm to segment web pages to extract good keywords/phrases. This algorithm utilizes useful visual cues to better partition a page at the semantic level. It is based on the fact that semantically related content is usually grouped together and the entire page is divided into regions for different contents using explicit or implicit visual separators such as lines, blank areas, images, font sizes, colors, etc. The output of the algorithm is a content structure tree whose nodes depict semantically coherent content[6].

The Algorithm Spans Over Three Phases

1. Visual Block Extraction
2. Visual Separator Detection
3. Content Structure Construction

IV. ARCHITECTURE OF CRAWLER

The crawler maintains a list of unvisited URLs called the frontier. The list is initialized with seed URLs which may be pro- vided by a user or another program. Each crawling loop involves picking the next URL to crawl from the frontier, fetching the page corresponding to the URL through HTTP, parsing the retrieved page to extract the URLs and application specific information, and Nelly adding the unvisited URLs to the frontier. Before the URLs are added to the frontier they may be assigned a score that represents the estimated Benet of

visiting the page corresponding to the URL. The crawling process may be terminated when a certain number of pages have been crawled. If the crawler is ready to crawl another page and the frontier is empty, the situation signals a dead-end for the crawler. The crawler has no new page to fetch and hence it stops. Crawling can be viewed as a graph search problem. The Web is seen as a large graph with pages at its nodes and hyperlinks as its edges. A crawler starts at a few of the nodes (seeds) and then follows the edges to reach other nodes. The process of fetching a page and extracting the links within it is analogous to expanding a node in graph search. A topical crawler tries to follow edges that are expected to lead to portions of the graph that are relevant to a topic [8].

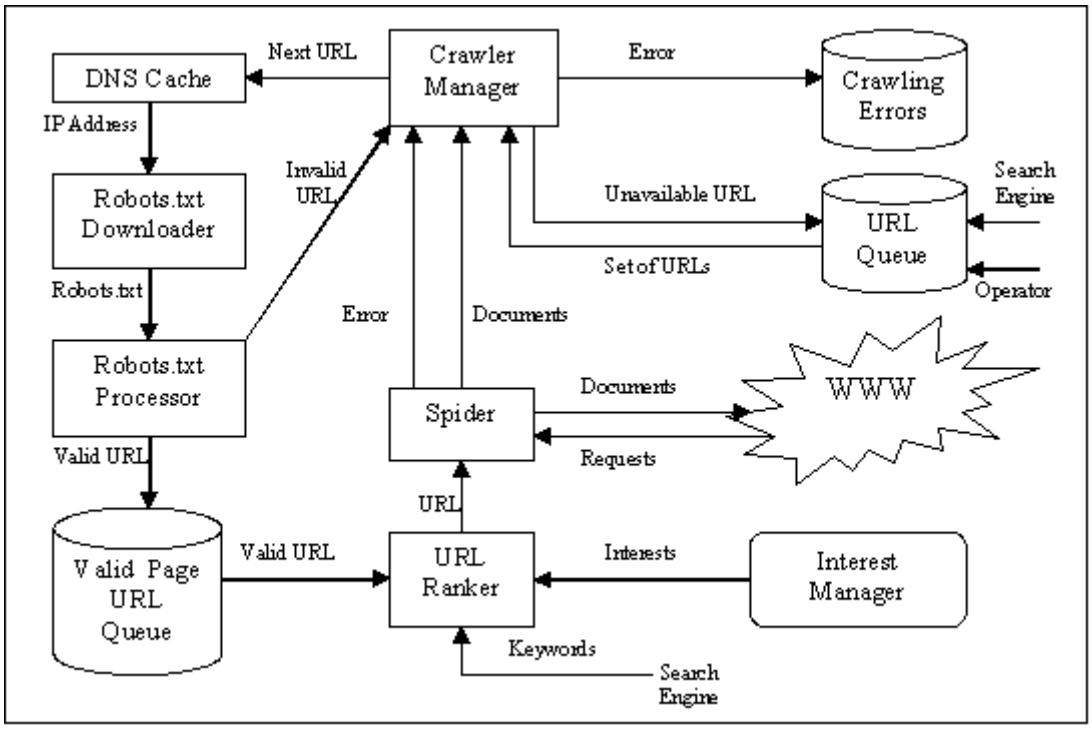

Fig 4.Architecture of Crawler

V. CRAWLING STRATEGIES

1). Breadth-First Crawling: Launched by following hypertext links leading to those pages directly connected with this initial set.
2). Repetitive Crawling: Once pages have been crawled, some systems require the process to be repeated periodically so that indexes are kept updated.
3). Targeted Crawling: Specialized search engines use crawling process heuristics in order to target a certain type of page.
4). Random Walks and Sampling: random walks on Web graphs via sampling is done to estimate the size of documents on line.
5). 5. Deep Web crawling: A lot of data accessible via the Web are currently contained in databases and may only be downloaded through the medium of appropriate requests or forms. The Deep Web is the name given to the Web containing this category of data.

VI. PREREQUISITES OF CRAWLING SYSTEM

1. Flexibility: System should be suitable for various scenarios.
2. High Performance (Scalability): System needs to be scalable with a minimum of one thousand pages/ second and extending up to millions of pages.
3. Fault Tolerance**:** process invalid HTML code, deal with unexpected Web server behavior, can Handle stopped processes or interruptions in network services**.**
4. Maintainability and Configurability: Appropriate interface is necessary for monitoring the crawling process including:
   - Download speed
   - Statistics on the pages Amounts of data stored

VII. CONCLUSION

In this paper we present a strategy to building an efficient, fast and reliable focus web crawler. The framework included the knowledge of architecture and its lexical presentment, relevance computation strategies, implementation algorithm, and an empirical evaluation of the overall framework. In this report, we want to discuss usage of vague techniques for enhancing the existing Focused Crawler for better resource discovery and user interface. We gave an overview of the existing Focused Crawler. A number of crawler strategies, policy have been discussed in this paper by which we can develop a fast crawler. The description of algorithms are also discussed here and the PBEL is the on of the useful technique which are described the in above section.

REFERENCES

1). S.Chakrabarti, M. van den Berg, B. Dom, "Focused crawling: a new approach to topic-specific Web resource discovery," in 8th International WWW Conference, May 1999.

2). .J. Cho, H. Garcia-Molina, and L. Page, "Efficient crawling through URL ordering," in *Proceedings of the Seventh World-Wide Web Conference*, 1998.

3). Krishna Bharat and Monika R.Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval, pages 104–111, Melbourne, AU, 1998.

4). S. Chakrabarti, K. Punera, and M. Subramanyam. Accelerated focused crawling through online relevance feedback. In WWW, Hawaii. ACM, May 2002.

5). Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific Web resource discovery. Computer Networks (Amsterdam, Netherlands: 1999), 31(11–16):1623–1640, 1999.

6). H. Yu, J. Han, and K.C.-C. Chang, "PEBL: Positive-Example Based Learning for Web Page Classification Using SVM," Proc. Eighth Int'l Conf. Knowledge Discovery and Data Mining (KDD '02), pp. 239-248, 2002

7). E.L. Allwein, R.E. Schapire, and Y. Singer, "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers," J. Machine Learning Research, vol. 1, pp. 113-141, 2000.

8). Aggarwal, F. Al-Garawi, and P. S. Yu. Intelligent crawling on the World Wide Web with arbitrary predicates. In *WWW10*, Hong Kong, May 2001.