**WEB CRAWLER**

A Report for the Evaluation 1 of Project 2

**Student Name- Deep Anand**

**Shubham Kumar Singh**

Admission No.: 15SCSE101021

15SCSE101020

Under the Supervision of

**Yadvendra Pratap Singh**



**School of Computing Science and Engineering**

**Greater Noida, Uttar Pradesh**

**January 2019**

**Contents-**

1. Acknowledgement

2. Abstract

   3. Introduction

       (i)   Overall description

       (ii)  Purpose

       (iii) Motivations and Scope

   4. Literature Survey

   5. Proposed model

       (i)   System architecture

   6. Implementation

   7. References

**1. ACKNOWLEDGEMENT**

It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to our highly respected and esteemed guide Mr. Yadvendra Pratap Singh

School of Computer Science & Engineering Galgotias University, for his valuable guidance, encouragement and help for completing this work. His useful suggestions for this whole work and co-operative behaviour are sincerely acknowledged. We would like to express our sincere thank to, Dean, School of computing Science & Engineering, Galgotias University, for giving us this opportunity to undertake this project. We also wish to express our gratitude to Dr. Sanjeev Pippal (Coordinator, School of Computing Science and Engineering) for his kind hearted support.

We also wish to express our indebtedness to our parents as well as our family member whose blessings and support always helped us to face the challenges.

At the end we would like to express our sincere thanks to all our friends and others who helped us directly or indirectly during this project work.

Date:25/01/2019

Deep Anand

1513101182

Shubham Kumar Singh

1513101594

**Abstract-**

The World Wide Web is an interlinked collection of billions of documents formatted using HTML. Ironically the very size of this collection has become an obstacle for information retrieval. The user has to shift through scores of pages to come upon the information he/she desires. Web crawlers are the heart of search engines. Web crawlers continuously keep on crawling the web and find any new web pages that have been added to the web, pages that have been removed from the web. Due to growing and dynamic nature of the web; it has become a challenge to traverse all URLs in the web documents and to handle these URLs. A focused crawler is an agent that targets a particular topic and visits and gathers only relevant web pages. In this dissertation I had worked on design and working of web crawler that can be used for copyright infringement. We will take one seed URL as input and search with a keyword, the searching result is based on keyword and it will fetch the web pages where it will find that keyword. This focused based crawler approach retrieve documents that contain particular keyword from the user's query; we are implementing this using breadth-first search. Now, when we retrieved the web pages we will apply pattern recognition over text. We will give one file as input and apply the pattern recognition algorithms. Here, pattern symbolizes text only and check how much text is available on the web page. The algorithms that I had used for pattern search are Knutt-Morri-Pratt, Boyer-Moore, finite automata algorithm.

**Introduction**

**Overall Description-**

A **Web crawler**, sometimes called a **spider** or **spiderbot** and often shortened to **crawler**, is an [Internet bot](https://en.wikipedia.org/wiki/Internet_bot" \o "Internet bot) that systematically browses the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web" \o "World Wide Web), typically for the purpose of [Web indexing](https://en.wikipedia.org/wiki/Web_indexing" \o "Web indexing) .

[Web search engines](https://en.wikipedia.org/wiki/Web_search_engine" \o "Web search engine) and some other sites use Web crawling or spidering software to update their [web content](https://en.wikipedia.org/wiki/Web_content" \o "Web content) or indices of others sites' web content. Web crawlers copy pages for processing by a search engine which [indexes](https://en.wikipedia.org/wiki/Index_(search_engine)" \o "Index (search engine)) the downloaded pages so users can search more efficiently.

Crawlers consume resources on visited systems and often visit sites without approval. Issues of schedule, load, and "politeness" come into play when large collections of pages are accessed. Mechanisms exist for public sites not wishing to be crawled to make this known to the crawling agent. For example, including a [robots.txt](https://en.wikipedia.org/wiki/Robots.txt" \o "Robots.txt) file can request [bots](https://en.wikipedia.org/wiki/Software_agent" \o "Software agent) to index only parts of a [website](https://en.wikipedia.org/wiki/Website" \o "Website), or nothing at all.

The number of Internet pages is extremely large; even the largest crawlers fall short of making a complete index. For this reason, search engines struggled to give relevant search results in the early years of the World Wide Web, before 2000. Today, relevant results are given almost instantly.

A Web crawler starts with a list of  URLs to visit, called the *seeds*. As the crawler visits these URLs, it identifies all the Hyperlinks in the page and adds them to the list of URLs to visit, called the crawl frontier. URLs from the frontier are recursively visited according to a set of policies. If the crawler is performing archiving of websites it copies and saves the information as it goes. The archives are usually stored in such a way they can be viewed, read and navigated as they were on the live web, but are preserved as ‘snapshots'.

The archive is known as the *repository* and is designed to store and manage the collection of web pages. The repository only stores HTML pages and these pages are stored as distinct files. A repository is similar to any other system that stores data, like a modern day database. The only difference is that a repository does not need all the functionality offered by a database system. The repository stores the most recent version of the web page retrieved by the crawler.

The large volume implies the crawler can only download a limited number of the Web pages within a given time, so it needs to prioritize its downloads. The high rate of change can imply the pages might have already been updated or even deleted.

**Purpose-**

A **web crawler** (also known as a web spider or webrobot) is a program or automated script which browses the World Wide web in a methodical, automated manner. This process is called web or spidering. Many legitimate sites, in particular search engines, use spidering as a means of providing up-to-date data.

**Motivation and Scope**

Motivations for Crawling There are several important motivations for crawling. The main three motivations are:

• Content indexing for search engines. Every search engine requires a web crawler to fetch the data from the web.

• Automated testing and model checking of the web application

• Identify the content of websites

• Determine how websites link to each other

• User friendly, efficient, fast, well-structured search results

• More accurate results.

• Prioritize highly relevant.

• Ranks sites URLs to prioritize potential deep sites.

• It provides personalize search to get result effectively.

• Automated security testing and vulnerability assessment. Many web applications use sensitive data and provide critical services. To address the security concerns for web applications, many commercial and open-source automated web application security scanners have been developed. These tools aim at detecting possible issues, such as security vulnerabilities and usability issues, in an automated and efficient manner. They require a web crawler to discover the states of the application scanned.

• Keen Crawler experiences an assortment of website pages amid a creeping procedure and the way to proficiently slithering and wide scope is positioning diverse destinations and organizing joins inside a webpage.

• The substantial volume of web assets and the dynamic idea of profound web, accomplishing wide scope and high productivity is a testing issue.

**Literature Survey**

**Crawling Traditional Web Applications-**

Web crawlers were written as early as 1993. This year gave birth to four web crawlers: World Wide Web Wanderer, Jump Station, World Wide Web Worm , and RBSE spider. These four spiders mainly collected information and statistic about the web using a set of seed URLs. Early web crawlers iteratively downloaded URLs and updated their repository of URLs through the downloaded web pages. The next year, 1994, two new web crawlers appeared: WebCrawler and MOMspider. In addition to collecting stats and data about the state of the web, these two web crawlers introduced concepts of politeness and black-liststo traditional web crawlers. WebCrawler is considered to be the first parallel web crawler by downloading 15 links simultaneously. From World Wide Web Worm to WebCrawler, the number of indexed pages increased from 110,000 to 2 million. Shortly after, in the coming years a few commercial web crawlers became available.

**Crawling Deep Web As server -**

side programming and scripting languages, such as PHP and ASP, got momentum, more and more databases became accessible online through interacting with a web application. The applications often delegated creation and generation of contents to the executable files using Common Gateway Interface (CGI). In this model, programmers often hosted their data on databases and used HTML forms to query them. Thus a web crawler can not access all of the contents of a web application merely by following hyperlinks and downloading their corresponding web page. These contents are hidden from the web crawler point of view and thus are referred to as deep web . In 1998, Lawrence and Giles estimated that 80 percent of web contents were hidden in 1998. Later in 2000, BrightPlanet suggested that the deep web contents is 500 times larger than what surfaces through following hyperlinks (referred to as shallow web) . The size of the deep web is rapidly growing as more companies are moving their data to databases and set up interfaces for the users to access them.

**Site Locating-**

• In this module, starts with a seed set of sites in a site database.

• When the number of unvisited URLs in the database is less than a threshold during the crawling process, Smart Crawler performs reverse searching of known deep websites for center pages (highly ranked pages that have many links to other domains) and feeds these pages back to the site database.

• Site Frontier fetches homepage 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 (web sites containing one or more searchable forms) found.

• To achieve more accurate results for a focused crawl, Site Classier categorizes URLs into relevant or irrelevant for a given topic according to the homepage content.

**In-Site Exploring-**

• After the most relevant site is found in the site locating module, the in-site exploring module 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 Classier to and searchable forms.

• Additionally, the links in these pages are extracted into Candidate Frontier. To prioritize links in Candidate Frontier, Smart Crawler ranks them with Link Ranker.

PROPOSED MODEL

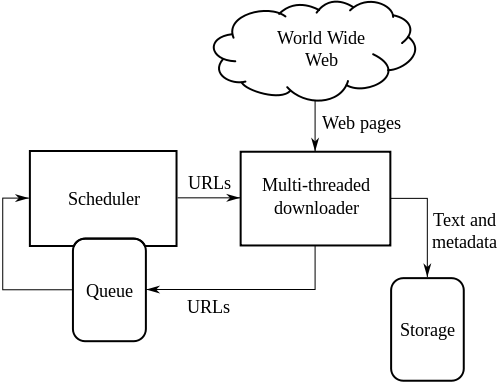
System Architecture-

Web search engine and some other sites use Web crawling or spidering software to update their web content or indices of others sites' web content. Web crawlers copy pages for processing by a search engine which indexes the downloaded pages so users can search more efficiently.

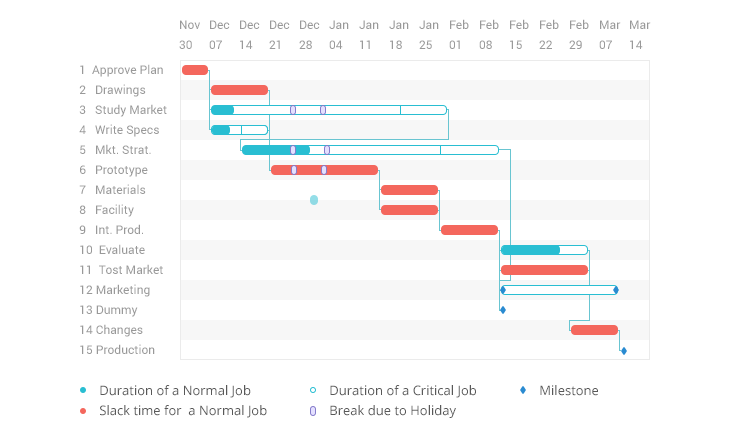
Crawlers consume resources on visited systems and often visit sites without approval. Issues of schedule, load, and "politeness" come into play when large collections of pages are accessed. Mechanisms exist for public sites not wishing to be crawled to make this known to the crawling agent. For example, including a robot.txt file can request bots to index only parts of a website, or nothing at all.

The number of Internet pages is extremely large; even the largest crawlers fall short of making a complete index. For this reason, search engines struggled to give relevant search results in the early years of the World Wide Web, before 2000. Today, relevant results are given almost instantly.

Crawlers can validate hyperlinks and HTML code. They can also be used for web scrapping (see also data-driven programming).



GANTT CHART



REFERNCES-

1. Mustafa Emmre Dincturk, Guy Vincent Jourdan, Gregor V Bochmann, and Iosif Viorel Onut. A model based approach for crawling rich internet applications.ACM Transactions on the web, (3): Article 19, 1–39, 2014.
2. Idc worldwide predictions 2014: Battles for dominance And survival on the 3rd platform. http://www.idc.com/.

1. Balakrishnan Raju, Kambhampati Subbarao, and Jha Manishkumar Assessing relevance and Trust of the Deep websources and results based on inter-source on Inter-source agreement. ACM Transactions on the Web.

1. Cheng Sheng, Nan Zhang, Yufei Tao, and Xin Jin Optimalalgorithms for crawling a hidden database in The web Proceedingsof the VLDB Endowment, 1112 –1123, 2012.