DESIGN OF A STRUCTURE DRIVEN COOPERATIVE MIGRATING CRAWLER FOR RETRIEVING QUALITY DATA

THESIS

Submitted in fulfillment of the requirement of the degree of

DOCTOR OF PHILOSOPHY

to

YMCA UNIVERSITY OF SCIENCE & TECHNOLOGY

by

DEEPIKA

Registration No: YMCAUST/Ph18/2010

Under the Supervision of

Dr. ASHUTOSH DIXIT

ASSOCIATE PROFESSOR

Department of Computer Engineering

Faculty of Engineering & Technology

YMCA University of Science &Technology

Sector-6, Mathura Road, Faridabad, Haryana, INDIA

JUNE, 2017

CANDIDATE'S DECLARATION

I hereby declare that this thesis entitled "**DESIGN OF A STRUCTURE DRIVEN COOPERATIVE MIGRATING CRAWLER FOR RETRIEVING QUALITY DATA"** by **DEEPIKA,** being submitted in fulfillment of the requirements for the Degree of Doctor of Philosophy in Department of Computer Engineering under Faculty of Engineering and Technology of YMCA University of Science & Technology, Faridabad, during the academic year 2016-2017, is a bona fide record of my original work carried out under the guidance and supervision of **Dr. ASHUTOSH DIXIT, ASSOCIATE PROFESSOR, DEPARTMENT OF COMPUTER ENGINEERING, YMCA UNIVERSITY OF SCIENCE & TECHNOLOGY, FARIDABAD** and has not been presented elsewhere.

I further declare that the thesis does not contain any part of any work which has been submitted for the award of any degree either in this university or in any other university.

(DEEPIKA)

Registration No: YMCAUST/Ph18/2010

CERTIFICATE

This is to certify that this thesis entitled **"DESIGN OF A STRUCTURE DRIVEN COOPERATIVE MIGRATING CRAWLER FOR RETRIEVING QUALITY DATA"** by **DEEPIKA** submitted in fulfillment of the requirements for the award of Degree of Doctor of Philosophy in Department of Computer Engineering, under Faculty of Engineering and Technology of YMCA University of Science and Technology, Faridabad, during the academic year 2016-17, is a bona fide record of work carried out under my guidance and supervision.

I further declare that to the best of my knowledge, the thesis does not contain any part of any work which has been submitted for the award of any degree either in this university or in any other university.

Dr. ASHUTOSH DIXIT
Associate Professor
Department of Computer Engineering
Faculty of Engineering and Technology
YMCA University of Science & Technology, Faridabad

Dated:

ACKNOWLEDGEMENT

I express my gratitude to almighty God for giving me strength and courage to complete this thesis.

I would like to express my sincere and deep gratitude to my Supervisor Dr. Ashutosh Dixit, Associate Professor, Department of Computer Engineering, YMCA University of Science & Technology, Faridabad for giving me the opportunity to work in this area. It would never be possible for me to take this thesis to this level without his innovative ideas, invaluable guidance, continuous support and encouragement. His knowledge of different perspectives of research provided me with the opportunity to broaden my knowledge and to make significant progress.

I gratefully acknowledge my university colleagues for their encouragement, support and invaluable suggestions in completing this research. I am also thankful to my students who helped me directly and indirectly in completing my research work.

I would like to express my heartfelt thanks to my husband Rajiv Sindwani for providing me constant support, encouragement and unconditional love. I also express my sincere gratitude to my parents, sister, brother, in-laws and other relatives for being a constant source of motivation. My special thanks to my lovely daughter Enaya for understanding me and giving me time for doing my research work. I would also like to express my thanks to my friend Sandhya for being a constant source of motivation.

(Deepika)

ABSTRACT

The large usage of World Wide Web has created challenges for the technology which is responsible for maintaining it. Search Engines are the means of providing relevant and useful information from the Web. They are also responsible for providing quality and accurate information to its end users. Web Crawlers are the main component of the Search Engines. They play a fundamental role in providing useful information from the vast Web. Search Engines provide a convenient user interface through which user can submit their query whereas web crawler traverses/crawls the Web and downloads the relevant data to maintain its repository/database. The freshness of the repository/database of any Search Engine is highly dependent on working of a Crawler.

A traditional crawler has only one crawling instance to crawl the huge World Wide Web. As the nature of web is dynamic, so the traditional crawler fails to keep the pace with such dynamic and rapid growth of web. There is need to increase the crawling instances so that they can cover the web as maximum as possible. The existing crawlers have multiple crawling instances but these instances still have some limitation. Moreover, the load distribution is also not uniform among these instances.

In this thesis, a design of a structure driven cooperative migrating crawler for retrieving quality data is being proposed that has capability of creating its instances called migrants as per the need. With this dynamic creation feature, it tries to cover the web as maximum as possible. And with the help of sitemaps crawler crawls all the links available on any particular site.

In order to maintain up-to-date database, crawler revisit a particular site. To preserve network bandwidth, it is necessary to prevent the crawler to download the same data on revisit. The crawler should download only the modified data. When there are migrants executing for crawling then there may be chances of redundancy in database. There should be cooperation between migrants so that they crawl the web in uniform manner and do not crawl the duplicate content. A duplicate removal module is proposed to eliminate redundancy from database. A novel mechanism is developed that checks the duplicate content before storing the webpage in database. It also checks duplicate URLs to prevent the redundancy in crawling.

Moreover with multiple migrants there is need of URL scheduling policy to schedule the URL to appropriate migrants so that load is uniformly distributed and network resources are effectively utilized. This is achieved by using Analytic Hierarchy Process (AHP) which is multi variant decision making technique.

To get relevant results in response to users' query, users' interest should also be considered. Here, the users' interest is observed by analyzing their browsing behaviour on web pages. These browsing behaviours can be mouse click, print, save as, scroll, key up, key down, bookmark etc. A data mining technique based Apriori algorithm is applied to these behaviours to get the frequent patterns so that the users' interest may be identified.

Another improvement in the process of crawling is introduced in the form of structure driven crawling, along with content matching. It is suggested that the structure of a document should also be taken into consideration while crawling the web. It is found that the structure driven crawling is highly efficient in crawling the related webpages.

A comparison is made between proposed migrating crawler and conventional general migrating crawler. With the help of implementation results, it has been observed that proposed crawler works well in all areas like web coverage, load distribution, elimination of redundant crawling and getting webpage of users' interest in response to a query.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

WWW:	World Wide Web
HTTP:	Hypertext Transfer Protocol
URL:	Uniform Resource Locator
BFS:	Breadth First Search
DFS:	Depth First Search
HITS:	Hyperlink Induced Topic Search
PR:	Page Rank
LVS:	Label Value Set
XML:	Mark-up Language
D:	Domain
DNS:	Domain Name Resolver
URI:	Uniform Resource Identifier
TEC:	Total External Count
UEC:	Unique External count
TIC:	Total Internal Count
UIC:	Unique Internal Count
ACC:	Accuracy
TAC-L:	Total Access Count-Learning
SAC-L:	Split Access Count-Learning
FTP:	File Transfer Protocol
IP:	Internet Protocol
DUST:	Different URLs Same Text
SIG:	Signature
DOM:	Document Object Model
HTML:	Hyper Text Mark-up Language
XHTML:	Extensible Hyper Text Mark-up Language
DTD:	Document Type Definition
AHP:	Analytical Hierarchy Process
CPU:	Central Processing Unit
N/W:	Network
CR:	Consistency Ratio
CI :	Computational Index

RI:	Random Index
ChangeFreq:	Change Frequency
LOC:	Location
LDB:	Local Database
DOC:	Document
OC:	Occurrence Count
SHA-1:	Secure Hash Algorithm-1
FP-TREE:	Frequent Pattern Tree
SC:	Support Count
GPS:	Global Positioning System
IDF:	Inverse Document Frequency

CHAPTER I

INTRODUCTION

1.1 WORLD WIDE WEB (WWW)

The large usage of World Wide Web [1, 2] has created challenge for the tools which are responsible for maintaining it. Search Engines [3] are the means of providing relevant and useful information from the Web. They are highly responsible for providing quality and accurate information to its end users. Web Crawlers are the main component of the Search Engines. They play a fundamental role in providing useful information from the vast Web. Search Engines provide a convenient user interface through which user can submit their query. It is the responsibility and function of Web Crawler to traverse the Web and download the most relevant data for the user. So, the functioning of Web Crawler is significant in the architecture of search engine. A general architecture of the search engine is given in figure 1.1.

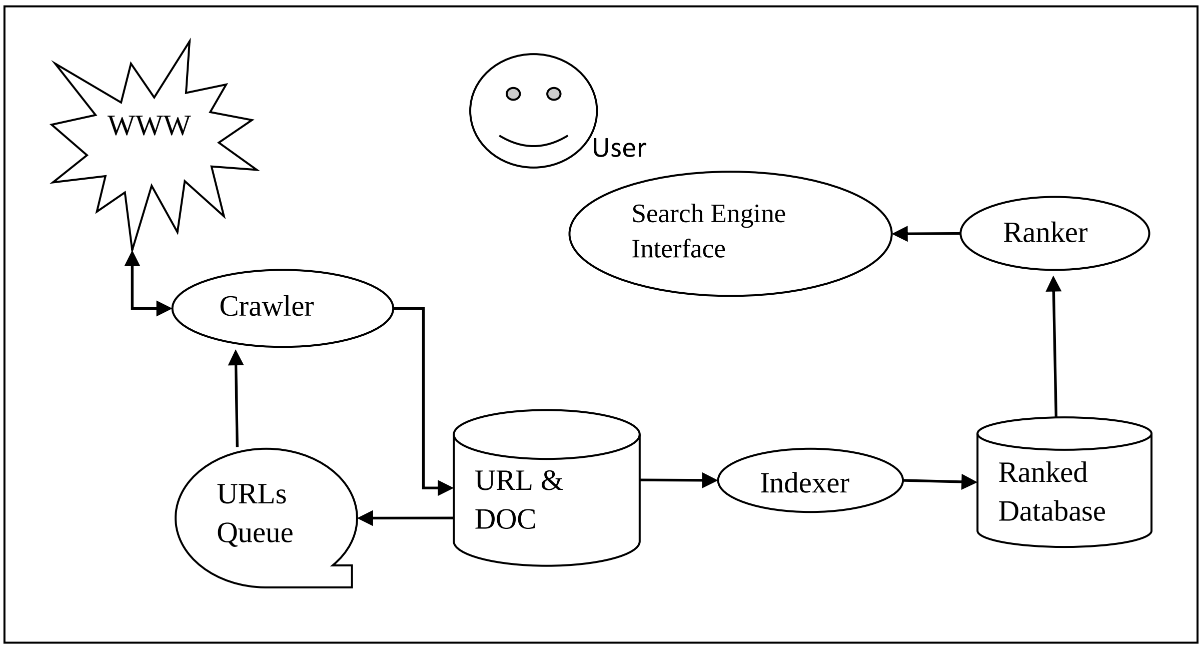


Figure 1.1: General Architecture of a Search Engine

A Web Search Engine consists of following main components:

- Crawler Module: It is a component which traverses the Web, collects the webpages and categorizes them.

- DOC & URL Repository: The downloaded web pages are temporarily stored in a local storage of search engine called DOC & URL repository. The new pages remain in the repository until they are sent to the indexer module for indexing.
- Indexer Module: The indexer module takes each new uncompressed page from the DOC & URL repository extracting suitable descriptors, creates a compressed description of the page and stores them in the database.
- Ranked Database: It contains web pages in compressed form. After compressing and indexing, indexer module stores the web pages in the database.
- Ranker Module: The ranker module takes the set of relevant pages from database and ranks them according to some criterion such as popularity score, content score etc. and provides them to the search engine interface.
- Search Engine Interface: The user supplies the query on search engine interface and gets the results for the same on this in the form of a list of relevant document links.

For downloading the web pages, the web search engines rely on *Crawlers.* A crawler is a program that downloads and stores web pages. A general architecture of crawler is shown in figure 1.2.

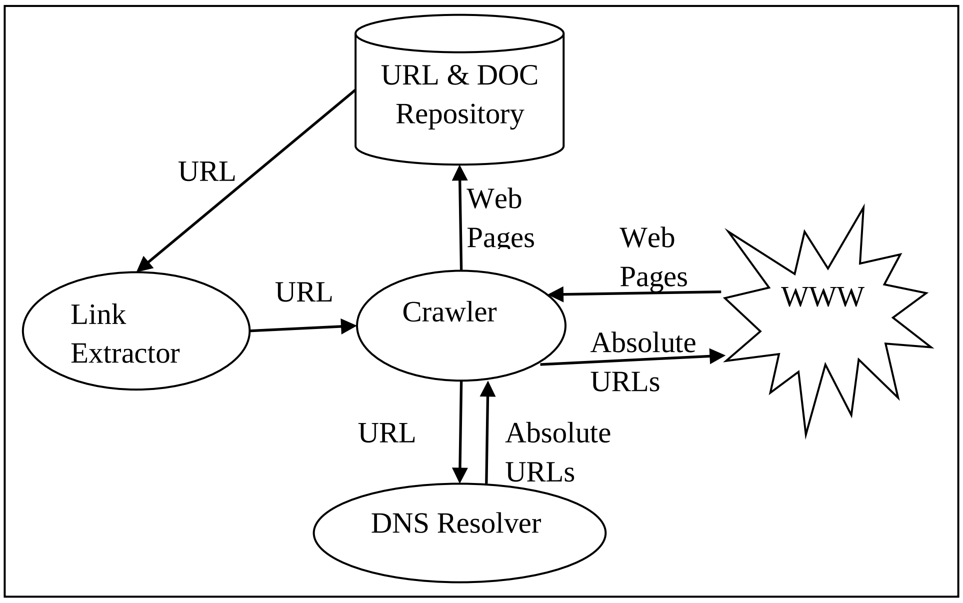


Figure 1.2: General Architecture of a Crawler

Generally, a crawler starts off by picking from an initial set of URLs queue, where all URLs to be retrieved are kept. From this queue, the crawler gets an URL, downloads the page and stores in the URL & DOC repository. Link extractor extracts URLs from the downloaded pages and adds them to the URL queue. This process is repeated until the URL queue exhausts or crawler decides to stop. There are many design issues related to crawler like efficient crawling, better network utilization, better cooperation between migrants, maximum coverage of the web etc.

Based on their working the crawler may be categorized in to the following types:-

- Parallel Web Crawler: Parallel Web Crawler [4] runs multiple processes in parallel to retrieve the whole or significant portion of the Web so that download rate is maximized.
- Focused Crawler: A focused crawler [25] may be described as a crawler, which returns relevant web pages related to a specific topic while traversing the web.
- Incremental Crawler: An incremental crawler [5] is one, which updates the selective set of downloaded pages instead of restarting the crawl from scratch each time.
- Migrating Crawler: A migrating crawler [6] distributes the downloading task among the downloading instances. The major benefits of migrating crawlers are scalability, reliability, and better network utilization.
- Hidden Web Crawler: A crawler which has the capability of extracting information from the hidden web is called as hidden web crawler [7]. Hidden Web consists of web pages that are created dynamically by filling the search query forms. As search forms are the entry-points into the hidden Web, so Hidden Web Crawler is designed to automatically process, analyze, and submit forms, using an internal model of forms and form submissions.

Since the work presented in this thesis is based on Migrating Crawler, a discussion on Migrating Crawler is given in next section.

1.2 MIGRATING CRAWLER

A migrating crawler [6] distributes the downloading task among the downloading instances. The major benefits of migrating crawlers are scalability, reliability and better network utilization. Unlike the centralized architecture of traditional search engines, migrating crawler distributes its crawling instances at geographically far away locations for crawling.

Migratability [8] can be defined in the context of Web crawling as the ability of a crawler to migrate to the data source (e.g., a Web server) before the actual crawling process starts on that Web server. Thus, migrating crawlers are able to move to the resource, which needs to be accessed in order to take advantage of local data access. The migrating crawler can perform a complete local crawling (either through HTTP [11], the file system, RPC or Aglets). The architecture of migrating crawler is shown in figure 1.3.

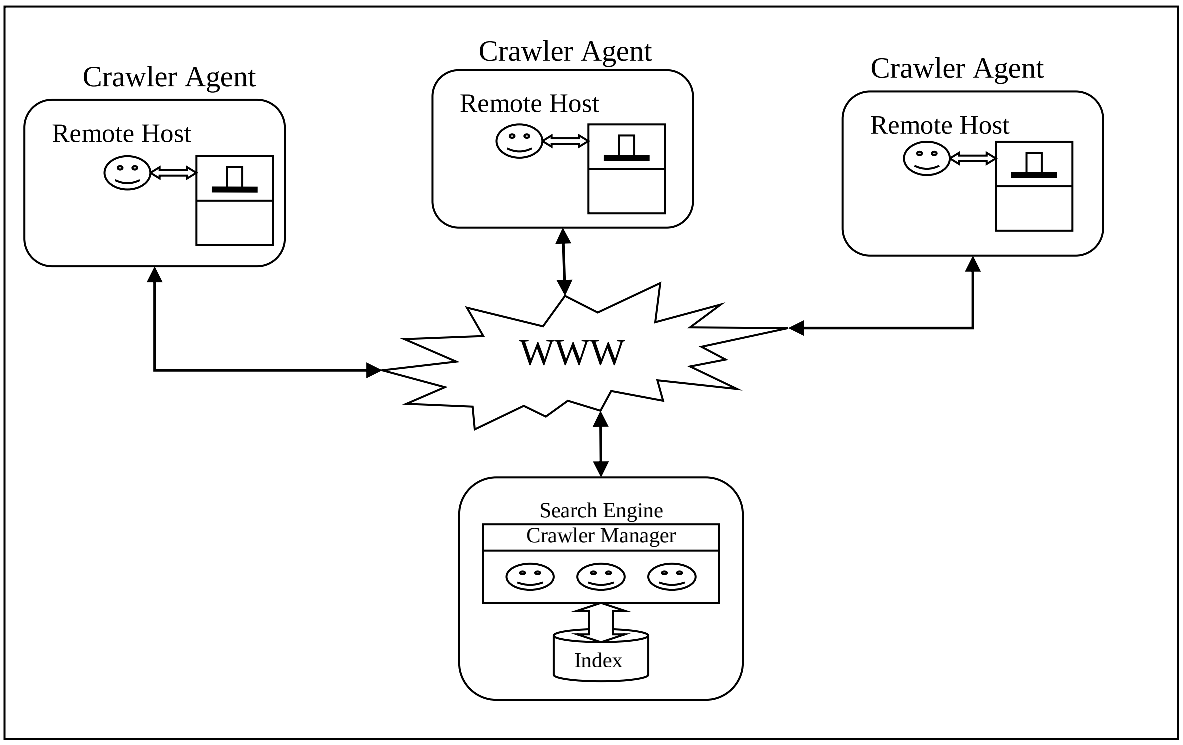


Figure 1.3: Architecture of Migrating Crawler

The migrating crawler creates its agents, called migrants [6, 121] and with the help of these migrants the crawling process takes place. The algorithm of Migrating Crawler is given in figure 1.4.

```
Migrating_ Crawler ( )
Steps 1: Begin
        2: Migrate to web server;
         3: Put server URL in url_list;
         4: For all URL belongs to url_list
         5: do begin
         6: Load page;                /* local data access */
         7: Store page in page_list
         8: if relevant; /* page analysis */
                     8.1 Extract page links;      /* recursive crawling */
        9: For all links belongs to page
        10: do begin
        11: If link is local then
                     11.1 Add link to url_list;
                     11.2 Else
                     11.3 Add link to external_url_list;
                     11.4 End
         12: End
         13: End
```

Figure 1.4: Algorithm: Migrating Crawler

These migrants help in achieving benefits like scalability, maximum web coverage, better network resource utilization etc.

1.3 MOTIVATION

The web is growing at a very fast rate and moreover, the existing pages are changing rapidly. There are several issues which need to be considered for an efficient web crawler design. Some major issues are discussed below:-

- **How to cover maximum web?** The size of the web is too large and it is difficult to cover the entire web. The crawler should be capable of covering the maximum web.
- **How to migrate the migrants in a proper way?** The migrating crawler migrates its migrants to many machines. The downloading process run in parallel and it may be the case that some machines are overloaded while some are sitting idle. If URLs are properly scheduled, then such type of situation can be avoided.
- **How to maintain databases uniqueness and freshness?** The crawler downloads the web pages to get stored in the database. This process is done by

multiple migrants. So, there is a possibility that the same URL may be crawled by multiple migrants. There should be some mechanism to eliminate such type of redundancy.

- **How to get more relevant results?** When a user fires a query, results on the basis of content matching is displayed. There is no involvement of user feedback as well as of the structure of web page while getting relevant pages. The browsing behaviour of user highly effects the relevancy of pages shown to the user. So, there should be some mechanism to incorporate users' browsing behaviour while ranking of pages. Along with content matching, the structure of web page should also be considered.

1.4 RESEARCH OBJECTIVES OF THE THESIS

When a user fires a query on search engine interface, the results should be relevant, unique and of user interest. The research objectives of the proposed work are as follows:

- **Structure-Driven Crawling: -** By knowing the structure of a required page beforehand, desired page(s) can be searched more efficiently. So, instead of fetching all pages related to search topic, modified crawler will fetch preferably those pages which have a similar structure to that of sample pages in terms of relevancy. While many works in the literature have addressed the issue of content-driven Web crawling, the use of the structure of the pages as a criterion to guide the traversal of crawlers has been almost neglected and will be one of focus area in this research.

  ***Proposal**: - In this work, in order to get more relevant results, along with content matching, the structure of web page is considered. A structure of web page can be taken out in the form of DOM tree. The structure is extracted from a downloaded web page and matched with supplied sample structure of a web page before storing them to the URL & DOC repository. The pages whose structure matches get the preference in while storage.*

- **URL scheduling: -** The size of the web is too large. To crawl this big size web URL plays an important role. The list of URLs selected for crawling highly affects the coverage of web. If this selection is appropriate then the coverage of web can be increased. Similarly, selection of a migrant can also be very beneficial in load distribution.

Proposal*: - In the proposed architecture of migrating crawler, a module called as URL ordering and URL scheduling is incorporated, which is responsible for ordering the URLs while crawling and then scheduling the URLs to appropriate migrants by using AHP technique [124]. Having a better URL Scheduling policy, a web crawler is able to download the useful information efficiently in minimum time. It also tries to cover the web as maximum as possible.*

- **Volatile information and change frequency of web pages**

  Web pages are changing at different frequencies [122, 127]. Due to resource constraints, search engines usually have difficulties keeping the local database completely synchronized with the Web. So with limited system resources, Web Crawler must be capable of knowing these frequencies so that it can re-visit accordingly and collect fresh information as far as possible.

  Proposal*: - In order to maintain the database up-to-date, instead of revisiting all web pages, again and again, the proposed crawler is revisiting only those web pages that have been undergone updation. This has been done with the help of Sitemap information.*

- **Relevancy**

  Whenever user supplies a query, search engine provides a list of document links relevant to his/her query. While providing these results, the user has no involvement. It has been observed that for getting relevant results users' interest should be considered. So, there should be some mechanism that incorporates users' interest while showing results in response to his/her query. Users' interest can be calculated in terms of their browsing behaviour on webpages.

  Proposal*: A module called as User Behaviour Analyzer is proposed that is responsible to capture the users' browsing behaviour on webpages like print, copy, save as, click, hyperlink etc. Based on these browsing behaviours, users' interest is calculated and used in assigning rank to such pages.*

- **Redundant Crawling**

  Distributed crawlers may crawl the same region of web and as a result search engine shows multiple entries [9, 10]. There are many URLs whose syntax is different but they point to same page. This not only gives redundant results but also waste network resources. So, there should be a well defined cooperation between the crawling instances so that crawlers crawl large percentage of web and give more data without redundancy.

Proposal*: - To prevent redundant crawling, a Standard Normalization process has been applied to URLs. After applying this process syntactically similar URLs are detected and eliminated, thereby reducing the redundant crawling.*

- **Unique Database**

  In migrating crawler, migrants are responsible for crawling the web. It may be the case that migrants download the same set of webpages. So, there should be some mechanism that stores only unique webpages to the database.

  Proposal*: - A Duplicate eliminator is proposed here that detects duplicate web pages and stores only unique web pages and URLs in the repository. This not only prevents network resource misutilization but also provide unique results to the user*

- **Coverage and Scalability**

  There is always a need to develop some mechanism that will cover the web as maximum as possible and scale with the growth of the size of the web.

  Proposal*: -The proposed architecture of migrating crawler is capable of scale with the growth of the size of the web, by creating the migrants dynamically. It also covers the maximum web.*

1.5 ORGANIZATION OF THESIS

The following is an outline of the contents of this thesis:

- Chapter I explores some elementary aspects of Search Engine and crawlers. The challenges involved and proposed solutions are discussed in this chapter.
- Chapter II: A literature survey of selected publications related to different crawlers is given in this chapter along with the general architecture of a search engine, types of crawlers, a detailed description of migrating crawlers and some duplicate elimination policies. The identified problems are listed at the end of this chapter.

- Chapter III: In this chapter, the architecture of a structure driven cooperative migrating crawler for retrieving quality data along with their functional modules are discussed.
- Chapter IV: In this chapter, the URL Scheduling module is discussed in detailed. It schedules the migrant for crawling the URL using Analytical Hierarchy Process (AHP) technique. The architecture comprising of its functional modules along with their algorithms has been discussed in detail. The snapshots of implementation are given along with the result analysis.
- Chapter V: In this chapter, duplicate eliminator module is discussed in detailed. This module is based on hashing. It uses the concept of Cache for storage purposes. The detail of each module involved and associated are discussed. At the end of chapter snapshots of implementation and the result analysis is given.
- Chapter VI: In this chapter, a User Behaviour Analyzer is discussed. It is used for ranking based on the browsing behaviour of user using Apriori algorithm. The detailed discussion along with algorithms used is given. The snapshots of implementation are given along with the result analysis.
- Chapter VII: In this chapter, a structure driven crawling is discussed. It is based on documents matching technique using structure of web document. The DOM tree is used for the structure of web page. The detailed discussion along with the flowcharts and algorithms used are discussed here. The snapshots of implementation are given along with the result analysis.
- Chapter VIII concludes our contributions and provides guidelines for future work in this area.
- The bibliography includes references to publications in this area.

CHAPTER II

LITERATURE REVIEW

2.1 INTRODUCTION

World Wide Web (WWW) [12, 13, 14] is the largest collection of hyperlinked hypertext documents. The size of Web is growing and changing at a very rapid pace. Hypertext Transfer Protocol (HTTP) is used for traversing these hypertext links [11, 92]. A web browser provides a medium to access WWW. In 1993, first web browser Mosaic was released. But, the widespread use of web began after releasing of Microsoft's Internet Explorer in 1995. After that many more web browsers came into existence like Firefox, Mozilla, Google Chrome, Opera and Safari. The information which is available on the Web is actually gathered by the Search Engines and presented by these web browsers.

2.2 SEARCH ENGINES

Search Engine is an information tool which provides information regarding users' query [106, 109, 110]. It is also is responsible for providing an interface to User for searching information on World Wide Web. It has various components which traverse the web and maintain the databases [131]. In general, there are three types of Search Engines [16, 17, 18].

(i) Crawler-Based Search Engines

These types of search engines are automated by the crawler [129]. A crawler is a computer program that works automatically and without user intervention and [downloads the documents from the web and stores them to be used by other applications. Such type of search engines traverses large area of web gathered the information and stored in their repository. For example Google, Bing, AltaVista, AllTheWeb etc.

(ii) Human Powered Directory based Search Engines

These types of search engines involved human interventions for maintaining their database. Unlike crawler based search engine, the rank and position of the webpage in repository is not rely on keywords but relies on the authors' choice. The author decides whether the contents are relevant or not. Some of these directories may not contain complete information. These are purely powered by human. For example Yahoo, Open Directory and Look Smart.

(iii) Meta Search Engines

These types of search engines use other search engines database while showing results in response to users' query [128]. These search engines do not traverses the web on their own. According to them, the size of web is too large, so it is difficult to index all data in their database. They rely on other search engines databases. For example Dogpile, Metacrawler and Mamma.

Since the crawler-based search engine is used in the present work, therefore a detail discussion on it is presented here. The basic architecture of a search engine [15] consists of following components in two layers namely User layer and Crawler layer and it is shown in figure 2.1.

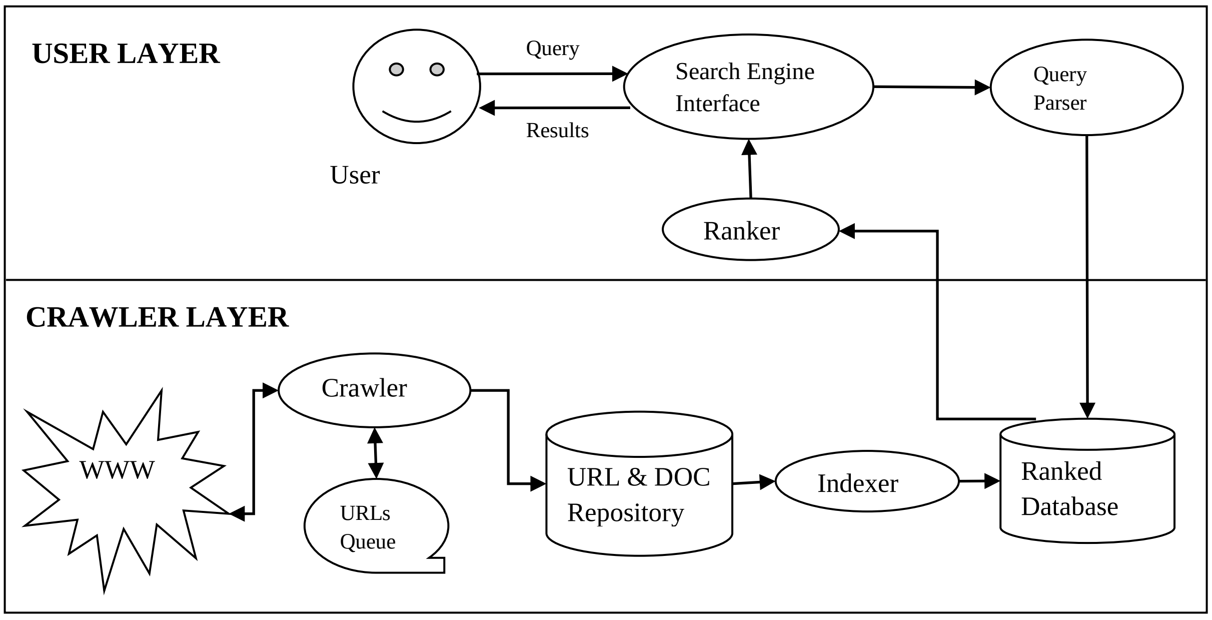


Figure 2.1: Layered Architecture of Search Engine

A Web Search Engine consists of following main components:

- **Crawler:** It is a component which traverses the Web, collect them and categorize them. There are many design issues are related with this module [20]. The detailed discussion about the crawler is given in subsequent sections.
- **URL Queue:** It consists of a list of URLs. It contains seed URLs also for initiating the crawling process.
- **URL & DOC Repository:** this repository is used for storing webpages along with their extracted links temporarily. The webpages are used by indexer for further processing.
- **Indexer:** The indexer takes each new uncompressed page from the page repository extracting suitable descriptors, creating a compressed description of the page. Indexing can be done using various techniques such as full text indexing, keyword indexing, human indexing, inverted index [116] etc.
- **Ranked Database:** After indexing web pages get stored in the ranked database. When User supplies the query, it gets the web pages from this database.
- **Ranker:** The ranking module takes the set of relevant pages from database and ranks them according to some criterion such as popularity score, content score etc. There are many ranking algorithms like Page Rank, HITS, Link based and much more[21][22][23].
- **Query Parser:** It receives the search requests from user and relies on indexer and repository in getting results.
- **Search Engine Interface:** It provides an interface to the User for searching information on World Wide Web. User submits a query here and gets the list of documents and corresponding URLs relevant to its query. For example, Google Chrome, Internet Explorer, Firefox etc.

2.3 WEB CRAWLER

It is the main module of the search engine. It is responsible for downloading the web pages from World Wide Web. It maintains the search engine repository. It traverses the World Wide Web in some manner like breadth first, depth first etc. and downloads the documents. The richness and freshness of a database of any search engine is highly depends on working of the crawler. The basic architecture of a crawler is shown in figure 2.2.

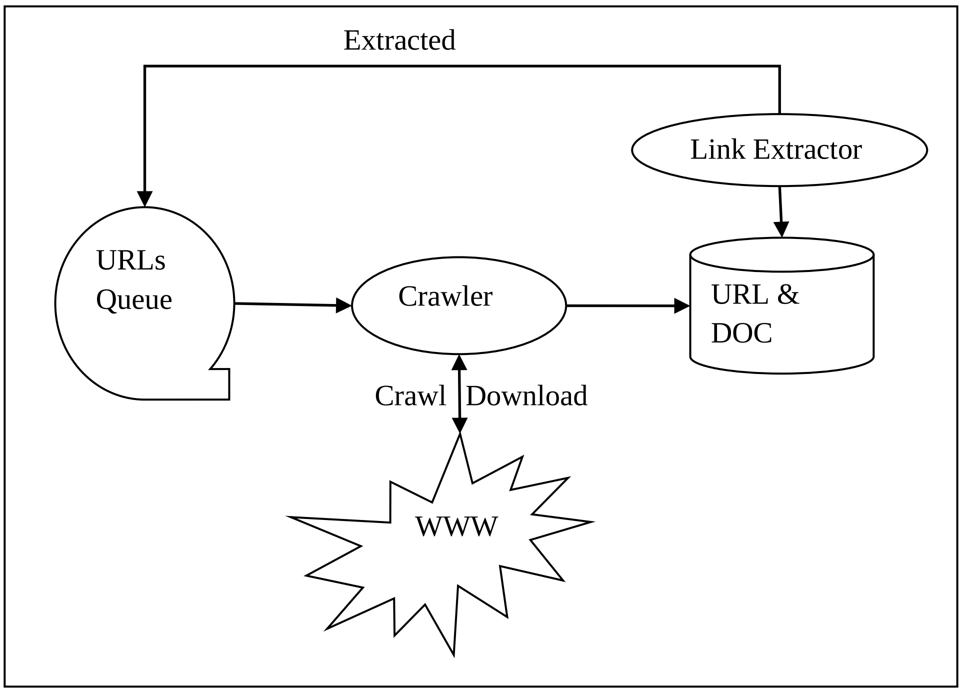


Figure 2.2: Architecture of Basic Crawler

The basic crawler consists of following components:-

i. URL Queue: It consists of list of URLs for crawling.
ii. Crawler: It is responsible for traversing, gathering and storing the webpages.
iii. URL & DOC Repository: It is a temporary storage of downloaded webpages used by crawler.
iv. Link Extractor: It is responsible for extracting embedded links from the downloaded webpages stored in repository and added them to URL Queue

The crawler has many types of traversing algorithm. Based on their working the crawler may be categorized in many types. Brief discussions of some are given in next sections.

2.3.1 Web Crawling Strategies

There are many ways in which crawler can traverse the web, gather the information and stored them in databases [130]. Some basic traversing strategies are given below:-

- Depth First Search (DFS)
- Breadth First Search (BFS)
- Page Rank

- HITS
- Fish- Search
- Shark Search

Breadth First Search

In this method, crawler starts at the root node and follows all the neighbour nodes which are at the same level. After crawling all nodes at first level, it crawls the next level nodes and the process continues [19].

Depth First Search

The web crawler starts from the root node and follows the depth up to the last child. If there is more than one child, then left most side child crawls first. After reaching at the end, it backtracked to the next unvisited node and continues the process till reaches at last child. It ensures that every path should be visited once.

Page Rank

In this approach, page rank of each link is calculated and then which has highest is choose for crawling [3]. The page rank is calculated as the sum of page ranks of all incoming links divided by the number of outgoing links as shown in equation 2.1.

$$PR(A) = (1-d) + d(PR(P1)/C(P1)+\ldots + PR(Pn)/C(Pn)) \qquad \textbf{(2.1)}$$

Where, PR (A) is the page rank of page A

PR (P1) is the page rank of a page P1 pointing to A

C (P1): number of out links from page P1

d: damping factor lies between 0 & 1

It is given by Brin & Page in 1998 and this Page Rank method is used by Google [3].

HITS algorithm

It is a link analysis algorithm. It uses hubs and authorities values to rate the page. A page which has many external links is considered as good hub page [22]. Similarly, a page which has many internal links is considered as good authority page [23].

Fish Search Algorithm

It works on the principle that relevant pages often have relevant neighbours [20]. Here, relevance is related to query supplied by the user. It treats the Internet as a directed graph where nodes are considered as a web page and edges as hyperlinks. It maintains a list of URLs which are to be searched. It scored web pages as 1 for relevant and 0 as non-relevant. It will traverse the graph in the direction where the relevant web pages are found.

Shark Search

It is an improvement over fish-search algorithm. The fish-search algorithm has some limitations and shark search removes these [21]. First of all, it changes binary values of relevancy as 0 & 1 to fuzzy score. This will give more priority to grandchildren of a relevant node than to the grandchildren of an irrelevant node. It also considers Meta information and Anchor text in calculating the relevancy score.

2.3.2 Web Crawler Types

Based on their working, the discussion of various types of crawlers are given below

a) Parallel Crawler

As the size of web increases, it gets more difficult to crawl the entire web. In Parallel crawler [24, 120], multiple instances of crawler are doing crawling process to cover the web as maximum as possible. It not only increases web coverage but also speed up the download rate.

It is clear from the figure 2.3 given below that now crawling is not depend only on single crawler but on multiple crawlers that are running in parallel.

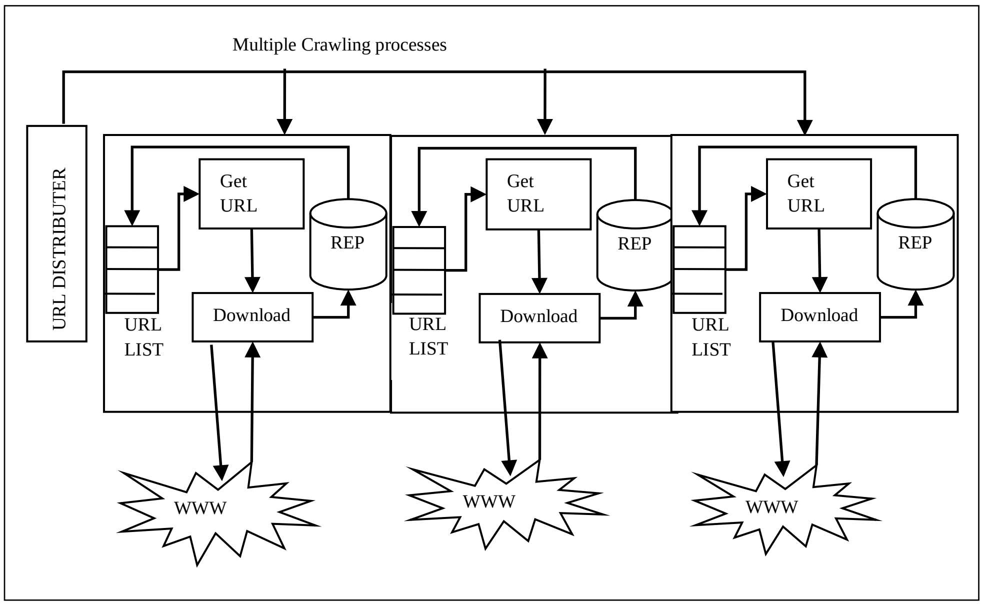


Figure 2.3: Flow of Multiple Crawling processes

The figure above shows how multiple crawling processes works in order to achieve the desired results in efficient and manageable manner. The crawling instances first gets the URL from URL list, traverses the web, downloads the webpages, stored in repository and from repository embedded links are extracted and again supplied to the crawling instances.

b) Focussed Crawler

The focussed crawler crawls only pre-defined topics related web documents [25, 82]. The goal is to select only those links that are of topic related other links are avoided. The focussed crawler crawls links that are given priority on the basis on given topic whereas other crawlers follow every link on a page in breadth first manner. For selecting the most promising links on related page, it uses an additional classifier. The seed URLs provided are of relevant to pre-defined topic. The relevancies of links that are embedded in web pages are calculated both based on content of the source page and links structure of the web. After finding relevancy, the crawling crawls the calculated relevant links and process go on. The architecture of focused crawler is given in figure 2.4.

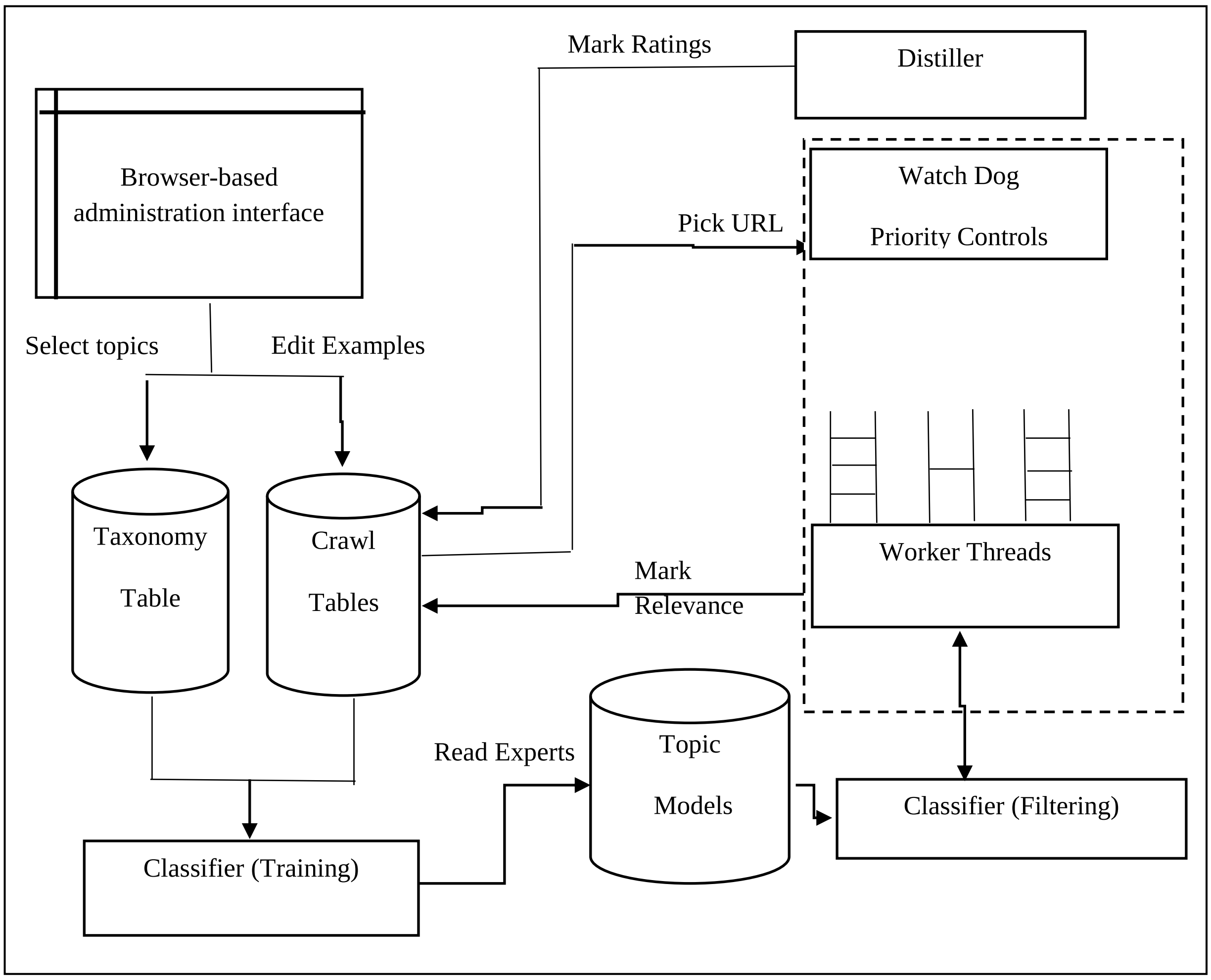


Figure 2.4: Architecture of Focused Crawler

The focused crawler [2] has three main components: a classifier which makes relevance judgments on pages crawled to decide on link expansion, a distiller which determines a measure of centrality of crawled pages to determine visit priorities and a crawler with dynamically reconfigurable priority controls which is governed by the classifier and distiller. The three classifier are used namely page classifier, form classifier and the link classifier. The page classifier is used to find whether the page belongs to the topic taxonomy or not. The link classifier identifies whether the links points to page that contain searchable forms or not. The form classifier is used to filter out useless forms. After classifying searchable forms, form classifier put them in database if they are not present there. The topic taxonomy is maintained with the help of users' feedback by asking interesting pages as they browse. The basic idea is to categorize crawled pages to be placed in this taxonomy. By searching to the limiting area, focussed crawler avoids crawling through unproductive paths.

c) Incremental Crawler

The main purpose of Incremental crawler is to crawls only those pages that have changed unlike other crawler that periodically checks the old pages and replaced them accordingly [104, 119, 120]. During this process, the crawler has two goals:

1. *Keep the local database updated*

   To keep the database updated, crawler should revisit the web pages in timely manner. The revisit frequency [117] for every page is estimated and then revisits them accordingly.

2. *Improve the local database quality*

   The crawler enhances the quality of database by fetching more important pages and replacing the less important one.

An incremental crawler [5] is one, which brings up to date a current set of downloaded pages instead of revisiting them from beginning every time. The revisiting policy has been defined and on that revisiting frequency web pages are re-crawled. The architecture of Incremental Crawler is given in figure 2.5.

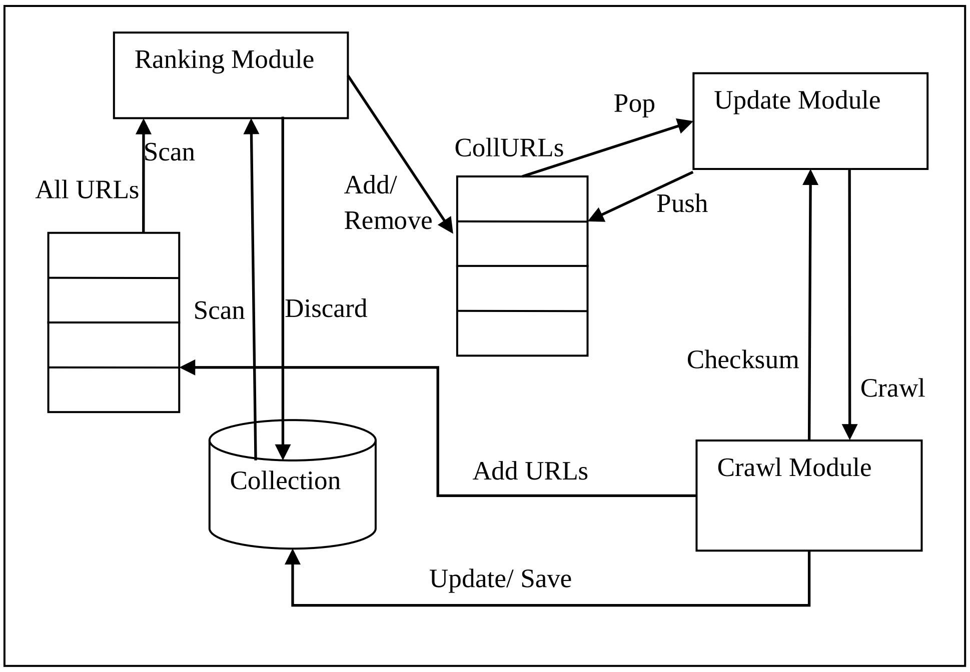


Figure 2.5: Architecture of Incremental Crawler

It consists of three modules:-

1. Ranking Module

   It is responsible of assigning importance to the URLs so that they will be selected first for crawling.

2. Update Module

   It is responsible for maintaining the freshness and quality of database. It takes the URLs from ranking module for maintaining quality and provided them to crawl module
3. Crawl Module

   It crawls the URLs provided by Update module.

Following lists are used in working of Incremental Crawler:-

1. AllUrls: - list of URLs crawled by crawler. It is used by ranking module to assign priority to them and placed in CollUrls list
2. CollUrls: - URLs are stored in priority order by ranking module and taken by update module to assign them revisit frequency. This list is then supplied to crawl module for crawling.

d) Distributed Crawler

In Distributed web crawler, a URL server scatters individual URLs to different crawlers, which download web pages in parallel [83]. The Crawler sends these downloaded pages to a central indexer for further processing. In this crawler, multiple crawling processes are running in parallel to increases the download speed. These multiple crawling instances may be geographically far away from each other.

A Distributed Crawler, IglooG [10] based on grid platform. Each crawler is arranging as grid service to improve the scalability of the system. The module Information services are responsible for distributing URLs in order to balance loads of the crawlers. Information services are structured as Peer-to-Peer overlay network.

The IglooG [10] is a distributed crawler and its architecture is shown in figure 2.6, where C denoted crawler.

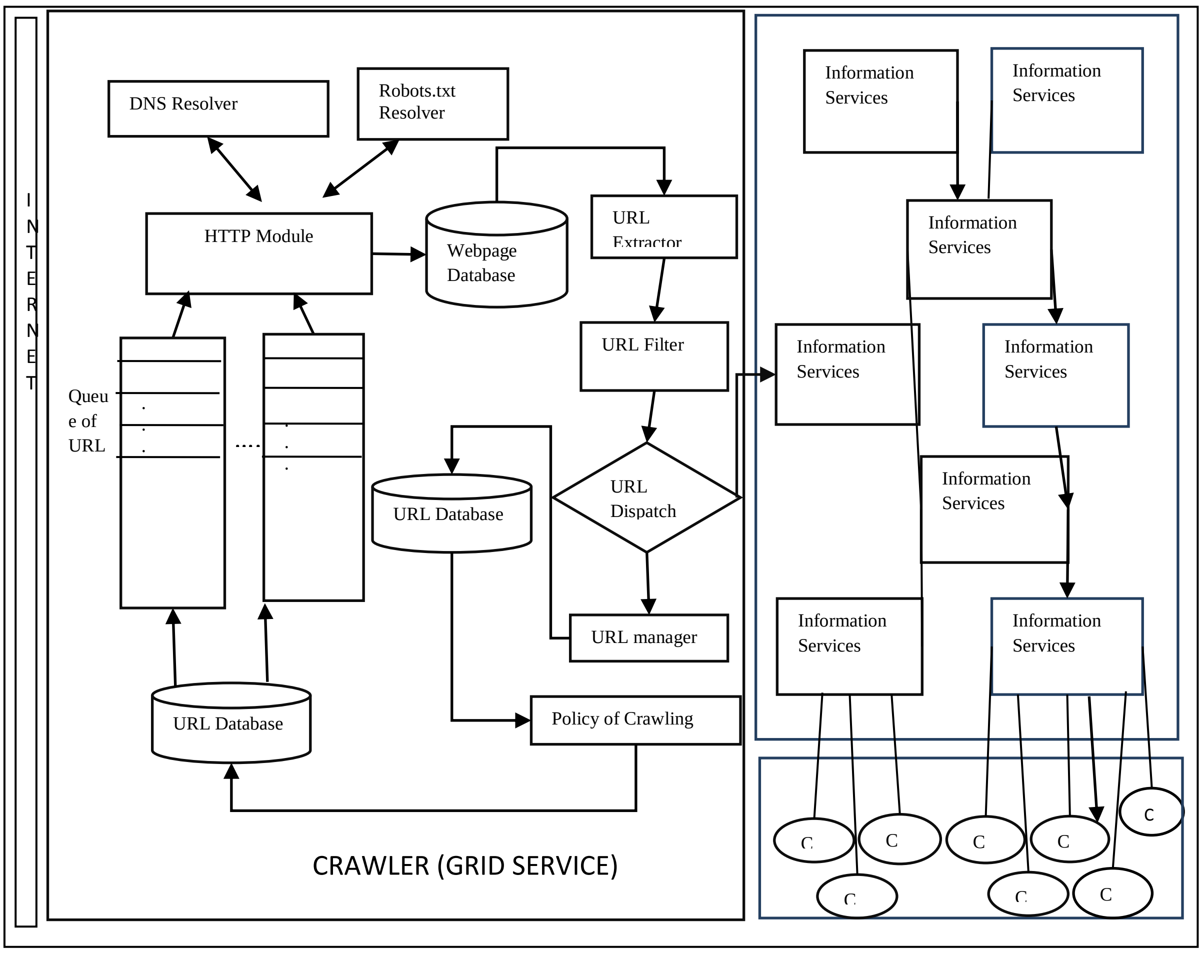


Figure 2.6: Architecture of Distributed Crawler

According to the ID of crawler and semantic vector of crawl page that is calculated by Latent Semantic Indexing, the crawler can decide whether pass on the URL to information service or hold itself. URL filter is used to filter out URLs that are according to the specified criteria. There are many ways of URL Filtering [88]

e) Hidden Web Crawler

A crawler which has the capability of extracting information from the hidden web is called as Hidden Web Crawler [84, 107, 113]. As search forms are the entry-points into the hidden Web, Hidden Web Exposer (HiWE) [7] is designed to automatically process, analyze, and submit forms, using an internal model of forms and form submissions. The architecture of Hidden Web is given in figure 2.7.

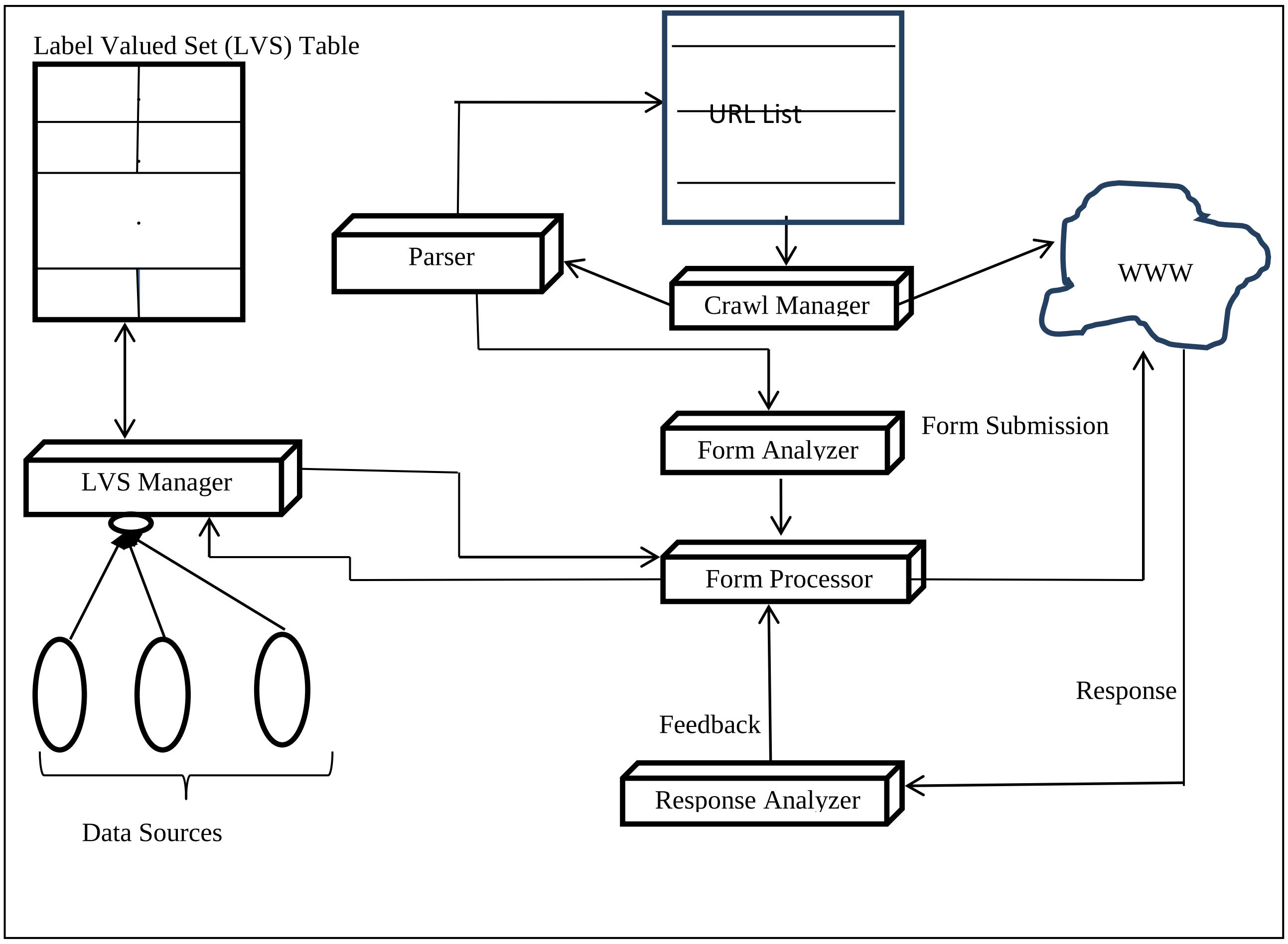


Figure 2.7: Architecture of Hidden Web Crawler

This model treats forms as a set of *(element, domain)* pairs. A form element can be any one of the standard input objects such as selection lists, text boxes or radio buttons. The architecture of HiWE has following steps:-

1. Analysis of Forms
2. Candidate Value Assignment: The value is assigned to form elements based on LVS (Label-Value Set) table and then weight is assigned by applying fuzzy aggregation function. These weights are then used as rank to these form elements.
3. Submission Form
4. Analysis of Response: The forms submitted are then analyzed for valid results.

The Hidden web performs inefficiently when the size of its crawl area increases. It is also unable to process the forms which are in different formats like .jpg, .pdf etc. Its efficiency depends on relevancy function and if this function computes wrong values, results will be wrong. So, its crawling highly depends on relevancy calculator and it should be accurate to get accurate results.

f) Migrating Crawler

In this work, the capabilities of migrating crawlers have been utilized to achieve scalability, reliability, maximum web coverage and better network utilization.

Odysseas et al [8] observed that the traditional centralized crawling model suffers from the following limitations:

- The task of processing the crawled data introduces a vast processing bottleneck at the search engine.
- The attempt to download thousands of documents per second creates a network and a DNS lookup bottleneck.
- Documents are usually downloaded by the crawlers in uncompressed form which increases the network bottleneck.

The authors also find that traditional centralized crawling cannot effectively catch up with the dynamic web. So, a novel architecture called UCYMicra [38] was developed. The UCYMicra Crawling System consists of three subsystems: The *Coordinator Subsystem*, The *Mobile Agents Subsystem* and a *Public Search Engine* as shown in Fig 2.8

.

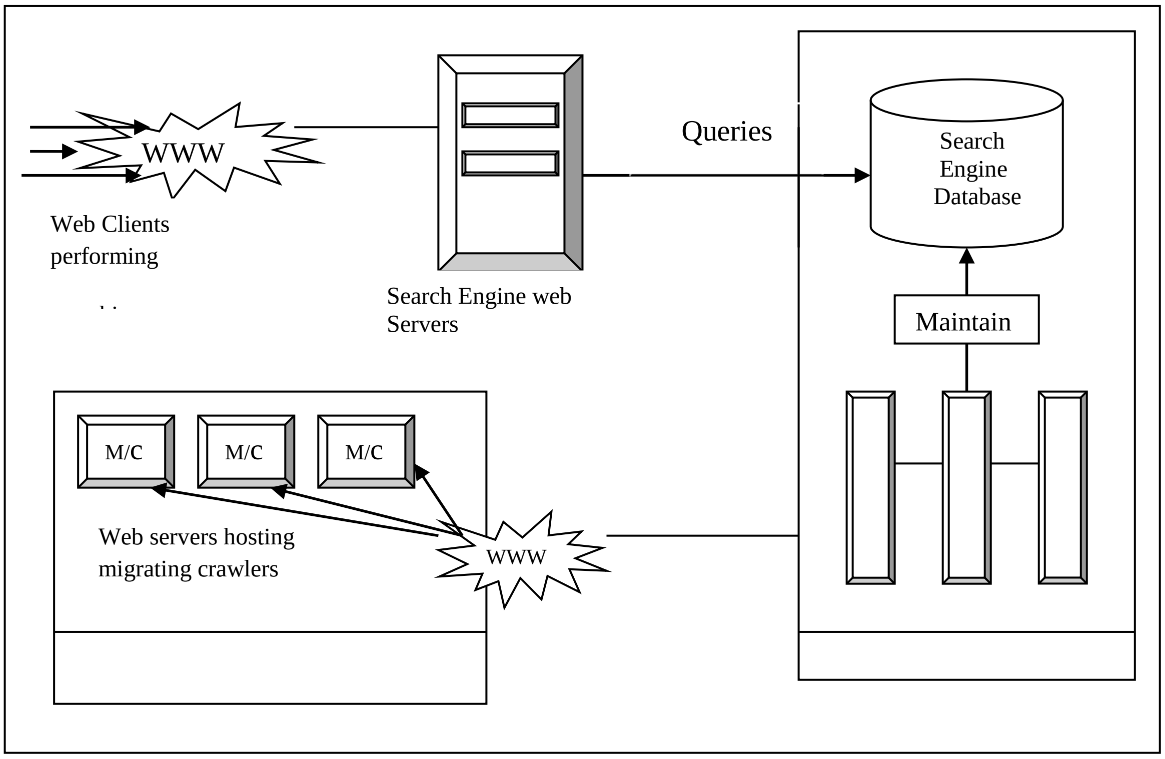


Figure 2.8: Architecture of Migrating Crawler

The coordinator subsystem resides at the search engine side and is responsible for maintaining the search database, providing online registration for new Web sites to participate in UCYMicra, and administering the *Mobile Agents Subsystem.* The Mobile Agent Subsystem is responsible for crawling the Web. It consists of two categories of mobile agents: The *Migrating Crawlers* and *the Data Carries*. The former are responsible for on-site crawling and monitoring of remote Web servers. Furthermore, they process the crawled pages, and send the results back to the coordinator subsystem for integration in the search engine's database. The latter are responsible for transferring the processed and compressed information from the Migrating Crawlers back to the Coordinator subsystem. The Public Search Engine is responsible for executing user queries on the database maintained by the Coordinator subsystem. Figure 2.9 shows UCYMicra at work.

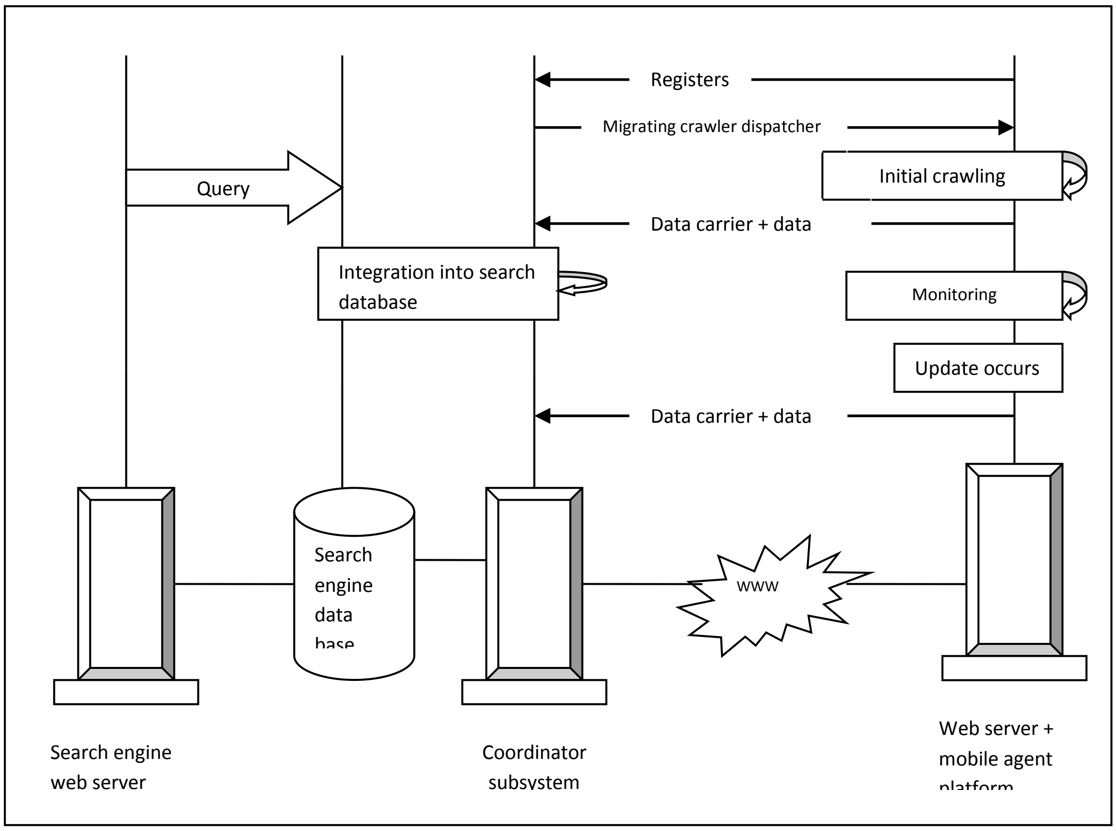


Figure 2.9: UCYMicra at Work

Powered by their inherent mobile capabilities, the Migrating Crawlers can perform the following tasks:

- Be **dispatched** to a newly registered web server that will participate in UCYMicra.
- **Crawling**: A Migrating Crawler can perform a complete local crawling (either through HTTP or the file system).
- **Processing**: Keywords are extracted from crawled documents, and are ranked based on their visual properties (font and color), position and occurrence frequency, in order to locally create a keyword index of the web server contents.
- **Data transmission**: The index is transmitted to the Coordinator subsystem by the Data Carriers. There it is integrated into the search database.
- **Monitoring**: The Migrating Crawler can detect changes on the Web server contents. Detected changes are instantly processed and transmitted to the Coordinator subsystem.
- **Real time upgrades**: New code for performing any of the above tasks can be easily deployed since UCYMicra's crawling architecture is based on Java.

Based on their working model, revisit, and many other characteristics, a brief comparison between the discussed types of crawler is given in table 2.1.

Table 2.1: Comparison of Different Crawlers

S. No	**Characteristics**	**Parallel**	**Focussed**	**Incremental**	**Distributed**	**Hidden Web**	**Migrating**
1.	Working Model	Crawling done by crawler instances in parallel	Crawling is done by crawler in a specific field	Crawling is done only on revisit	Crawling done in parallel but at far places	High quality Search forms	Crawling is done by multiple migrants
2.	Revisit Policy	Revisit on restart crawling	-	Revisit of high-rank pages first	Revisit on restart crawling	-	Revisit based on change frequency
3.	Crawling Strategy	Breadth First Search	Depth First Search	Breadth First Search	Breadth First Search	Depth First Search	Breadth First Search
4.	Speed	Fast	-	-	Fast	-	Fast
5.	Initial URL	Seed URL	Topic specific	Given by Priority Queue	Seed URL	Search Forms	Seed URL
6.	Scalable	Yes	No	No	Yes	No	Yes
7.	Web Crawler Example	PARCHAYD [64]	S.Chakrabarti [25]	Jungoo Cho[5]	IglooG [10]	HIWE [7]	Odysseas [8]

2.4 DESIGN ISSUES OF CRAWLER

When a user supplies a query on search engine interface, the results appeared should be of user interest, relevant and unique [108]. To provide relevant results, the repository and corresponding database should be fresh and rich. It is the responsibility of crawler to maintain repository rich and fresh. There are many design issues regarding designing of an efficient crawler. Some recent research work in the area of these design issues are described in following subsections [63, 85, 90].

2.4.1 Coverage

The coverage includes how much world wide web is captured and how much data is useful out of captured web. The information available on Web is increasing at a tremendous rate. To gather maximum information, crawler working should be optimized. The coverage can be defined in terms of the number of web page and links structure. The URL plays a vital role in expressing coverage metrics i.e. more number of URLs means more area covered by the crawler. There should be some efficient way for crawling the web in order to get maximum coverage.

Uri et al [26] introduce the concept of Sitemaps for crawling the web. Sitemap is an XML file that contains list of URLs along with some metadata [95, 96]. It includes change frequency, last modified and priority as Meta information. They classified URLs into following states:-

1. Seen: The URLs which are seen by crawler but not yet crawled.
2. Crawled: The URLs which are downloaded.
3. Unique: These are URLs which are left after eliminating duplicate ones.
4. Indexed: The URLs which are indexed by Indexer.
5. Results: The URLs which are shown to User in response to his query.
6. Clicked: The URLs clicked by User on the result page.

Then on the basis of these states, coverage metrics and their importance have been defined. To calculate coverage metrics, they define following relations for Domain D:-

$\text{Coverage}(D) = \text{Crawled}_{\text{sitemap}}(D) / \text{Crawled}(D)$

$\text{UniqueCoverage}(D) = |\text{Unique}_{\text{sitemap}}(D)| / |\text{Unique}(D)|$

$\text{IndexCoverage}(D) = |\text{Indexed}_{\text{sitemap}}(D)| / |\text{Indexed}(D)|$

$\text{PageRankCoverage}(D) = \sum \text{PageRank}_{\text{sitemap}}(D) / \sum \text{PageRank}(D)$

Where, $\text{Crawled}_{\text{sitemap}}(D)$ is the list of URLs crawled with the help of Sitemap

Crawled (D) is the list of URLs crawled without Sitemap.

With the help of above metrics, the difference in crawling with and without a sitemap was shown and it is found that Sitemap crawling provides better results.

Najork et al [27] approach suggests that breadth-first search is a good crawling strategy, as it tends to discover high-quality pages early on in the crawl. On early it means that as crawling increases progressively the quality of web pages deteriorates. It uses connectivity based metric, Page Rank given by Brin and Page to measure the quality of a page which is easy to measure as it is calculated on the basis of links pointed to any page. When more pages with high rank if pointing to a page then the Page rank of that page is also high. The document with high Page Rank should have less outgoing links. It uses the following relation to express Page Rank mathematically.

Let p1, p2 ... pk are the pages link to a page p. The Page Rank of page p is:

$R(p) = d/T + (1-d) \sum_{i=1}^{k} R(pi)/C(pi)$

Where, R(pi) is the Page Rank of pi

C(pi) is the number of out links of pi

T is the total number of pages

d is the parameter such that it values lies between 0.1 and 1.

With the help of this Page Rank values, there is increases in overall download rate and the burden on the server is reduced by downloading only important pages.

S S Vishwakarma et al [28] proposed a modified approach for crawling. It uses last visit time of crawler and applies the filter at the server side. This filter checks this last visit time and return list of those URLs only that are updated after crawler last visit. This is a query based approach as shown in the figure 2.10.

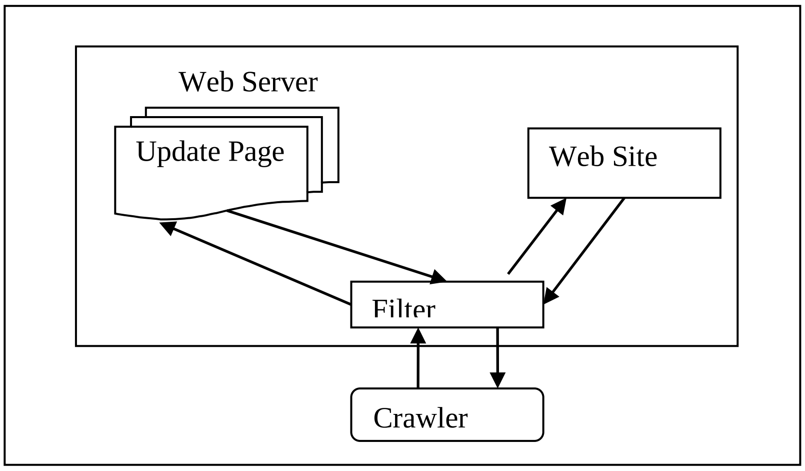


Figure 2.10: Query-Based Approach

HTTP uses it GET method to know the list of updated URLs. This approach reduces network traffic by avoiding crawling of non-modified web pages.

Brandman [29] introduces the concept of servers providing some meta information to the crawlers such as last modification and file size. With the help of last modified date, crawler crawls only modified pages. This saves a lot of network bandwidth and creates less traffic on the network. The proposed crawler works in the following manner:-

1. Start from the seed URL, it starts crawling. It follows some order for initial crawling of URL set.
2. Each web page has meta information associated with it. So, when web page downloaded along with its meta information also downloaded. For revisiting them, their meta information is examined and only updated web pages are downloaded.
3. After downloading L pages i.e. daily page limit, next L modified pages are crawled next day in some predefined order. Depending upon a number of modified pages, the crawler continues their downloading for that particular day. If a number of updated pages are less than L then crawler discontinue their downloading.

Damien Lefortier Yandex et al [30] worked on a different section of web pages called as ephemeral new pages. These are those pages on which users' interest grows within hours as they appear but remain only for few days. He found the sources of such pages and then re-visits them in order to get newly created such pages at a faster rate. The quality of sources estimated from user feedback. Along with content sources, the time of the page discovered is also contributed in determining the quality of ephemeral pages.

Tripathy A & Patra P.K [31] describe the design of a web crawler that uses PageRank algorithm proposed by Brin & Page for distributed searches and can be run on a network of workstations. The formula for evaluating Page Rank is given in equation 2.2.

$$PR(p) = (1-\gamma) + \gamma \sum_{\{d \in in(p)\}} \frac{PR(d)}{|out(d)|} \qquad \textbf{(2.2)}$$

Where p is the web page whose rank is to calculate,

d is another page

out (d) is the number of out links from d

γ is the constant damping factor.

On the basis of this Page rank, the importance of web page is calculated. There are many methods to calculate importance of any webpage [115]. The PageRank was the criteria to decide the quality of URL to be crawled. A distributed crawler is proposed in order to increase the crawling task. The proposed architecture is given below in figure 2.11.

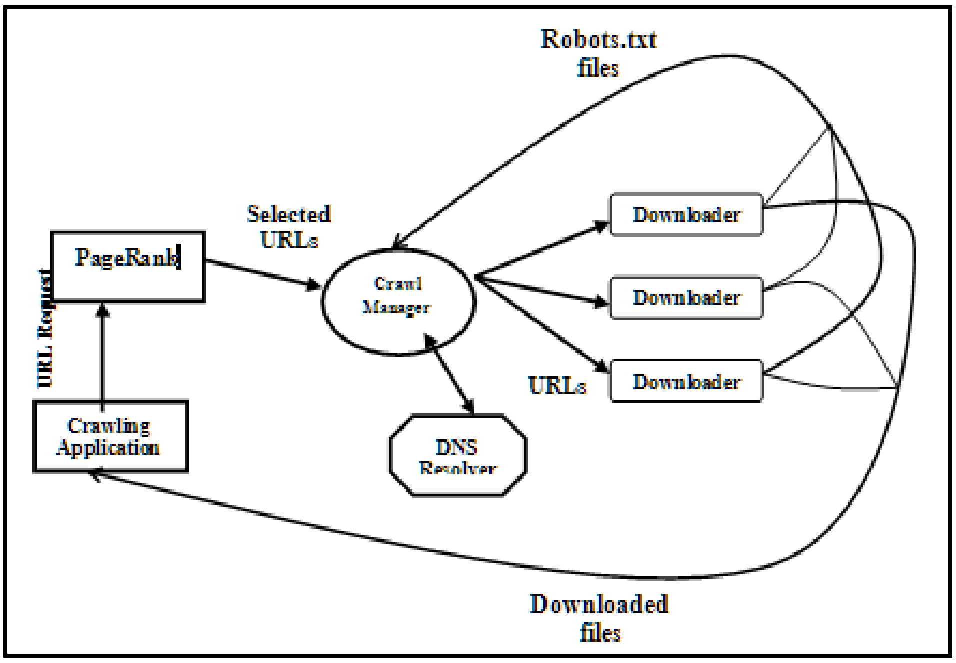


Figure 2.11: A Proposed Architecture of Distributed crawler

The crawl manager, downloader, DNS resolver and crawling application are distributed on different machines. To increase the system performance, these components can be replicated for increasing the scalability and reliability.

The summarized comparison of discussed work, with respect to the coverage design issue given in table 2.2.

Table 2.2: Coverage Solutions

Algorithm	**Uri et al**	**Najork et al**	**SS Vishwakarma et al**	**Brandman**	**Damien Lefortier Yandex et al**	**Tripathy A et al**

Technique Used	Sitemap	BFS	Query based approach	Meta information	User feedback	Page Rank algo
Domain	General	General	General	General	Ephemeral pages	General
Crawling Limit	No	No	No	Yes	Yes	No
Scalable	Yes	Yes	No	No	No	Yes
Efficiency	Freshness	High Quality	Revisiting	Freshness	Revisiting	Important pages

2.4.2 URL Scheduling

To increase the performance of crawler, the migrating crawler has been proposed. As discussed above, migrating crawler can create migrants for crawling the web on behalf of a migrating crawler [6]. When more than one crawling task is executed then there is the probability of increasing duplicate contents. The migrants may crawl the same URL and this not only wastes time but also the network resources. There is need of cooperation between the migrants so that multiple crawling of the same URL should be avoided. To achieve this goal, URLs should be scheduled properly so that every crawler has its own unique set of URLs set. Scheduling is the process of assigning a suitable URL to appropriate migrants in order to get unique URL download and proper utilization of network resources [80, 81, 111].

Some of the prevalent work in the related area is discussed below:-

Mohd Shoaib et al [32] proposed a web crawler which ordered the URLs on the basis of web pages content and structure similarity to the query. The occurrence of keywords in crawled pages corresponding to any given query is used for content similarity whereas neighbouring nodes linking is used for calculating structural similarity. The SimRank [69] algorithm also uses the same structural similarity calculation. The experiment was done on a set of similar websites and result was compared in terms of top URLs precision, crawling time, ordering time and similarity scores. The comparison showed that proposed crawler works well.

A.Guerriero et al [33] proposed a dynamic URL assignment method based on the clustering of URLs and then scheduling them. Clustering should be done in such a way that the same URL shouldn't be crawled by multiple crawlers. A distributed architecture of parallel web crawler was proposed. Its components cooperate in efficient manner in order to get desired results. The architecture is shown in figure 2.12:

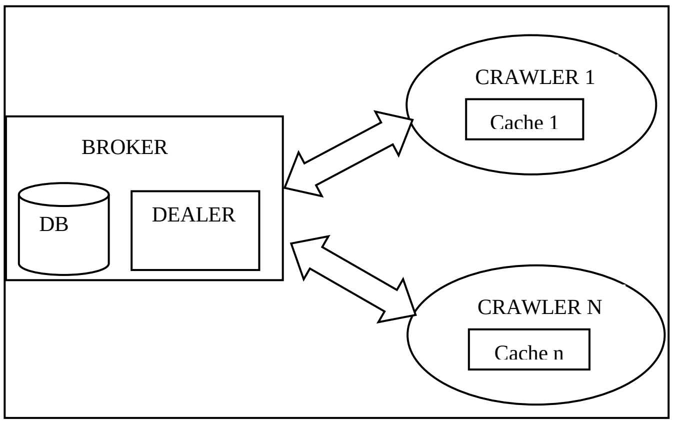


Figure 2.12: Dynamic Assignment of URL model

It consists of following components:-

1. Broker: - It is responsible for scheduling of URLs and also to create a communication link between crawler and database. It picks the URL from database and with the help of dealer scheduled it to crawlers.
2. Dealer: -It is responsible for crawler efficient working. It manages load optimization with the help of fuzzy clustering method.
3. Crawlers: -They are responsible for crawling the web and downloading the page in the repository.

The working of Dynamic URL Assignment is as follows in figure 2.13.

```
Step 1:  Get the URL from the database.
     2: Divide the URL structure into different components as mentioned by URI
        standards [7].
     3: Apply Hash function to each component and convert it into integer form.
     4: Represent integer URL components into 3D coordinates.
     5: Apply Fuzzy Clustering [8] to these URLs.
     6: Assign these clusters to different crawlers for crawling.
     7: Extracted links checks for duplicity with already stored in the database.
       7.1 If already found in database, then
           7.1.1 Discard
          Else
           7.1.2 Stored in database.
     8: Go to Step 1.
```

Figure 2.13: Algorithm: Working of URL Assignment

It relies heavily on the communication link, so link failure brings down the system completely. Load Balancing is also a major issue here.

Yuan Wan et al [34] designed and implemented a URL assignment method based on hashing. It works on parallel systems [4] where systems are physically independent but they are cooperating with each other through some mechanism. Each system downloads the web pages on their local machine and when internal links are extracted from these web pages, there is need of scheduling of these internal links. Either they are scheduled to be downloaded on the host machine or to some other machine. The crawlers are not communicated with each other. Host Machine is the central coordinator through which communication and scheduling take place between the crawlers. The architecture of the system is as shown in figure 2.14:

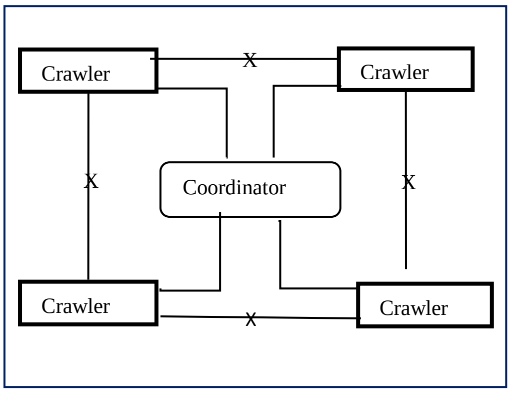


Figure 2.14: Model of Distributed Parallel Crawler

Coordinator assigns the URLs to different crawler based on its hostname. Hostname decides that whether the URL goes to other machine or downloads on its home machine. For this purpose, a hashing scheduling algorithm was designed. It will take URL as input and then hash function was applied to this URL. In hashing function, the host name was extracted from URL and converted into the integer format and then matched with an id of the crawler. It is given by following equation 2.3:

$$Key_i = (\sum_{i=1 \text{ to } l} \text{Transfer (Host (URL}_i)) \bmod n \qquad \textbf{(2.3)}$$

If a match takes place, then it downloaded on the same machine otherwise go to other. The coordinator has an id of all registered crawler and on the basis of this information, it schedules the URLs. It prevents duplicate URLs to download again. It partitioned the URLs list in such a way that ensures that no URL repeated at any machine. But it is not scalable i.e. if number of URLs increases then it is not sure that they performed in the same manner as it does now.

Dajie et al [35] introduces a scheduling algorithm for URLs which works on Round Robin Scheduling. It works on master-slave architecture. One node stores all information of other nodes and runs the scheduling algorithm. The architecture is given in figure 2.15.

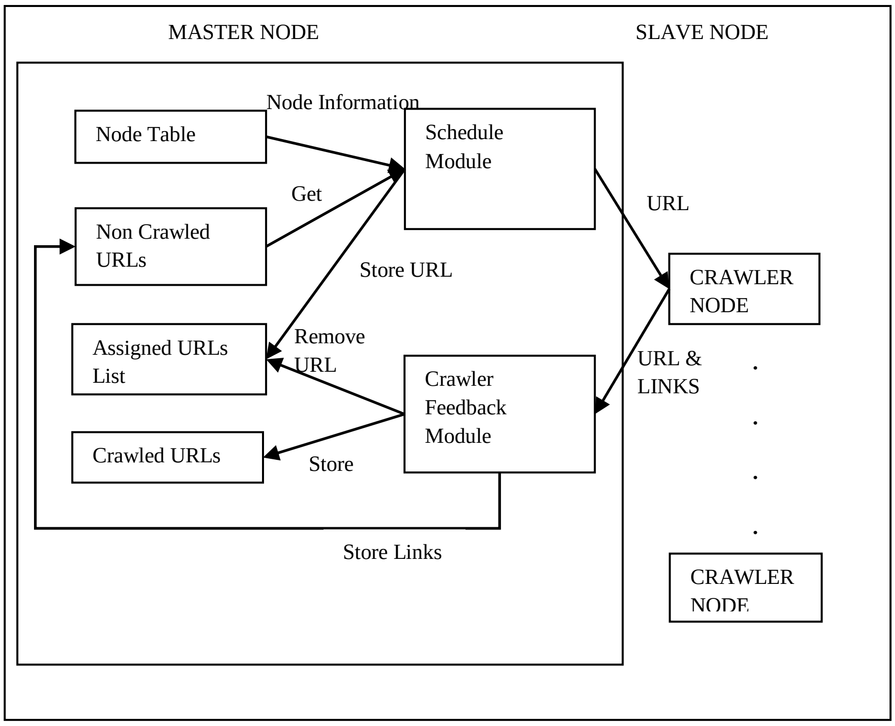


Figure 2.15: Master-Slave Architecture of URL Scheduling

For calculation of weight, time as an important factor was used. More time value means crawler has tasks that are yet not completed. It means it should not get more tasks.

Here, weight and time are the reciprocal of each other. So, weight is low for that node which has more time to finish its task.

In master-slave architecture, at the master node, various data structures are used to store the information of crawler node as well as the status of URLs i.e. whether they are scheduled to be crawled or completely crawled. The concept of Round Robin algorithm for assigning URLs to the crawler was used. A weight is assigned to each crawler node. This weight shows the status of that node. If weight is low then it means crawler is heavily loaded and vice versa. The weight is assigned with the help of the equation 2.4.

$$W = \frac{k}{\sum_{i=1}^{k} ti * (m+1)} \qquad \textbf{(2.4)}$$

Where, k=no. of tasks finished recently

t_i =finished time of i tasks

m=no. of tasks yet not finished

URLs get scheduled on the basis of this weight value of crawler node. A threshold value is taken to ensure that low weight crawler node will not leave unattended.

Chandramouli et al [36] works on the principle of the popularity of links. It calculates the popularity by mining the web logs available on the website. It counts the total access of particular web page and considered them as popularity. URL ordering was classified into two approaches. One is *non-learning* algorithms that use predetermined ordering function and other is *learning* algorithms that will order the URLs based on training set of URLs with quality information.

In the *non-learning* algorithm, high the access count the more important is the page. But, it is also suggested that it may be case that website owner itself access its website several times and this cause increase in access count and considered as an important page. To avoid such situation, four types of accesses to a website were defined. These are:

1. Total External Count (TEC): - access to URL on website from outside the local network
2. Unique External Count (UEC): - unique access from outside the local network
3. Total Internal Count (TIC): - access to URL on website from local network
4. Unique Internal Count (UIC): - unique access from local network

Thus, total access count is obtained by equation 2.5.

$$\boxed{Total= TEC+UEC+TIC+UIC \qquad \textbf{(2.5)}}$$

To predict the importance of every count value their accuracy is calculated and then assigns weights to each count value with the help of these accuracy values was assigned. Thus weighted score of each URL is calculated as per the equation 2.6.

$$\boxed{WeightScore=a*TECacc/Total+b*UECacc/Total+c*TICacc/Total+d*UICacc/Total \qquad \textbf{(2.6)}}$$

Where, TECacc =TEC accuracy algorithm

UECacc =UEC accuracy algorithm

TICacc = TIC accuracy algorithm

UICacc = UIC accuracy algorithm

a, b, c, d are raw external, unique external, internal and unique internal counts for the URL.

In *learning* algorithm, the best combination of above four count values was used. Two learning algorithms were implemented, Total Access Count Learning (TAC-L) and Split Access Count Learning (SAC-L). Both algorithms have training and testing phases. In these algorithms, access counts as input and supplied to any learning algorithm like decision tree or k-nearest neighbour and model is prepared. To measure the quality of a page, Page Rank algorithm was used. Higher the rank, more important is the page. The working of learning algorithm is shown in figure 2.16.

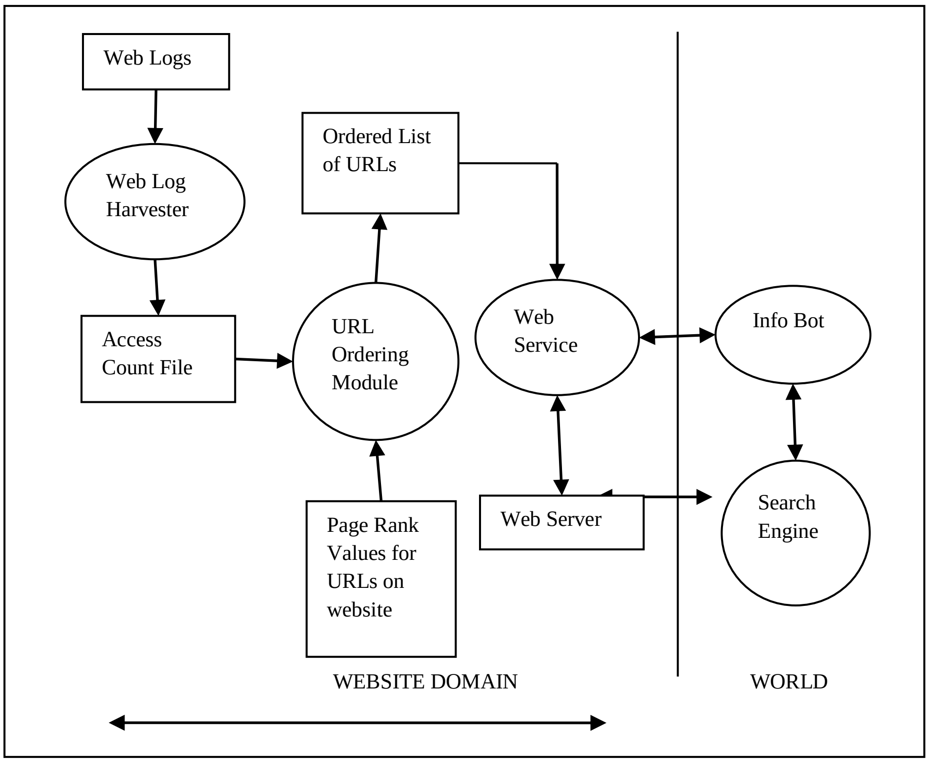


Figure 2.16: Popularity Based Architecture of Search System

Here, Web log server files are used to calculate the access counts of each URL and supplied as input to learning algorithm. With the help of Page Rank algorithm, URL can be ordered and provide to Info Bot for crawling.

The summarized results of all URL scheduling related research in shown in table 2.3.

Table 2.3: Summary of URL Scheduling Algorithms

Algorithm	**Task Scheduling**	**URL Hash**	**Popularity Based**	**Dynamic URL**
Technique Used	Round Robin	Hashing	Links Count	Fuzzy clustering
Descriptio n	Works on round robin policy i.e. every crawler gets its turn and weight assigned get surety of load balancing	In hashing method, URL's host part is find out and then convert the characters into its integer equivalent and matched with crawler's ID.	External and internal links counts are calculated.	Clustering is done to schedule URLs.

Algorithm	**Task Scheduling**	**URL Hash**	**Popularity Based**	**Dynamic URL**
Input	Time	URL	Link Count	URL
Resource Utilization	Efficient	Efficient	No consideration	Efficient
Duplicity Check	Yes	Yes	No	Yes
Recovery	Yes	Not Consider	Not applicable	No
Advantages	Simple & efficient, no starvation, load balancing	Fast, Load balancing	Simple and better than BFS, less burden on search engines	Efficient, Load balancing, remove duplicates, low cost
Disadvantages	Scalability, Single point of failure	Lack of scalability, less realistic, single point of failure	Small data set, not applicable to new pages	Dependency on communication link, single point of failure

2.4.3 Quality Of Database

The quality of Search Engine Database is achieved by having a unique collection of web pages. Uniqueness is obtained by eliminating duplicate web pages [75, 86]. Duplicity can be at URL level as well as at the level of downloaded web pages. A URL is composed of five components: the scheme, authority, path, query, and fragment components [37].

$$\underbrace{\text{http://}}_{\text{scheme}}\underbrace{\text{www.jabong.com}}_{\text{authority}}\underbrace{\text{/women/clothing/Biba/}}_{\text{path}}\underbrace{\text{?q=biba}}_{\text{query}}\underbrace{\text{\#pos=3}}_{\text{fragment}}$$

Scheme Component*:* The scheme is the first part of URLs. It basically tells about the protocol through which communication is takes place between sender & receiver i.e. web server & client and vice-versa. There are various protocols:

ftp	File Transfer protocol
http	Hypertext Transfer Protocol
gopher	The Gopher protocol
mailto	Electronic mail address
news	USENET news
nntp	USENET news using NNTP access
telnet	Reference to interactive sessions
wais	Wide Area Information Servers
file	Host-specific file names
prospero	Prospero Directory Service

Each protocol has different syntax for writing a URL and each has different purpose for writing. For example, FTP is used for files & directories, HTTP is used for internet resources, the news is used for news group etc.

Authority Component: The authority is the second part of URL. It has three subparts:

1. User information: it has username followed by @ and it is according to the scheme used. It is an optional part.
2. Hostname: it basically contains the location of a web server. The location of a web server can be a domain name or Internet Protocol (IP) address.
3. Port: it is a network port number for the server. Many protocols have default port number. The colon symbol (:) is prefixed before port number.

Path Component: It is the third part of URL. It based on authority or scheme. It contains data like file name, web page etc. It contains multiple paths which are separated by '/'.

Query Component: It is the fourth part of URL. It starts with the question mark symbol (?). It has information that is supplied to the web application and interpreted by the resource. The information it contains is in the form of parameter names and parameter values. They are separated by equals' symbol (=).

Fragment Component: It is sometimes an optional part. It contains information that indicating a particular part of a document. It starts with the hash symbol (#).

There are URLs which are different but point to the same page [89]. This type of problem is designated as DUST [39, 71] i.e. different URLs with similar text. The impact of DUST effects the whole working of Search Engines i.e. crawling, indexing, ranking etc. There are many reasons of DUST. Some are listed below:-

1. To Balance Load
2. To served as Backups
3. More user-friendly i.e. By creating shortcuts
4. To reduce network traffic

There can be many more reasons for duplicate URLs. Creating duplicate URLs may be benefits to Internet Users or webmasters but creating trouble to working of the search engine. By downloading the same page again waste network bandwidth then creating an index of these duplicate URLs waste time and effort and then at search engine interface when user see similar links again then he or she may get irritated.

Some of them which are different slightly in syntax can be identified by applying standard normalization steps. The elimination of duplicate URLs, as well as duplicate webpages, saves considerable network resources and this should be detected as early as possible because such duplicate identities point to other duplicate content. Some work for removing duplicate URLs done so far is discussed below. Thereafter various existing duplicate elimination policies for downloaded web documents are discussed.

Berner et al [37] introduces the concept of URL Normalization process. These steps are considered as standards in this field. These steps are broadly classified in three categories namely syntax based, scheme based and protocol based normalization process.

Syntax-Based Normalization

i. Case normalization – Change letters in the scheme & host component into the lower-case letters

ii. Percent-encoded normalization – decode unreserved character, such as %2D for hyphen and %5F for underscore.

iii. Path segment normalization – remove dot-segments from the path component, such as '.' and '... '.

Scheme-Based Normalization

i. Add trailing '/' after the authority component of URL.

ii. Remove default port number, such as 80 for http scheme.

iii. Prune the fragment of URL.

Protocol-Based Normalization

i. Remove last trailing slash if the results of accessing the resources are equivalent i.e. same protocol used.

The results are promising in the sense that non identical URLs were never converted into identical string and the URLs that are syntactically identical got identified.

Lee et al [38] gave extension to standard normalization process. They suggested converting the path part of URL into lowercase, removing the slash symbol at end of URL and also removing default pages. By adding these three steps to standard normalization process, more unique URLs were identified.

Agarwal et al [40] worked on crawl logs and then from these logs generate rules. These rules then utilized for finding duplicate web pages. They form clusters of similar pages from crawl logs. These clusters were then used to generate rules for detecting duplicate pages. These rules were then generalized and with the help of these rules the URL itself were able to detect identical pages. The proposals affect crawling, indexing in an effective way.

Farah et al [41] worked on document similarity along with the URL similarity. According to them, by taking the signature of whole body text is time-consuming. So, they considered to reduce the body text and then applied MD5 fingerprinting mechanism [68]. They suggested some ways to reduce the body text and by doing this they reduced the time wastage in the fingerprinting method. The step-wise working is shown in figure 2.17.

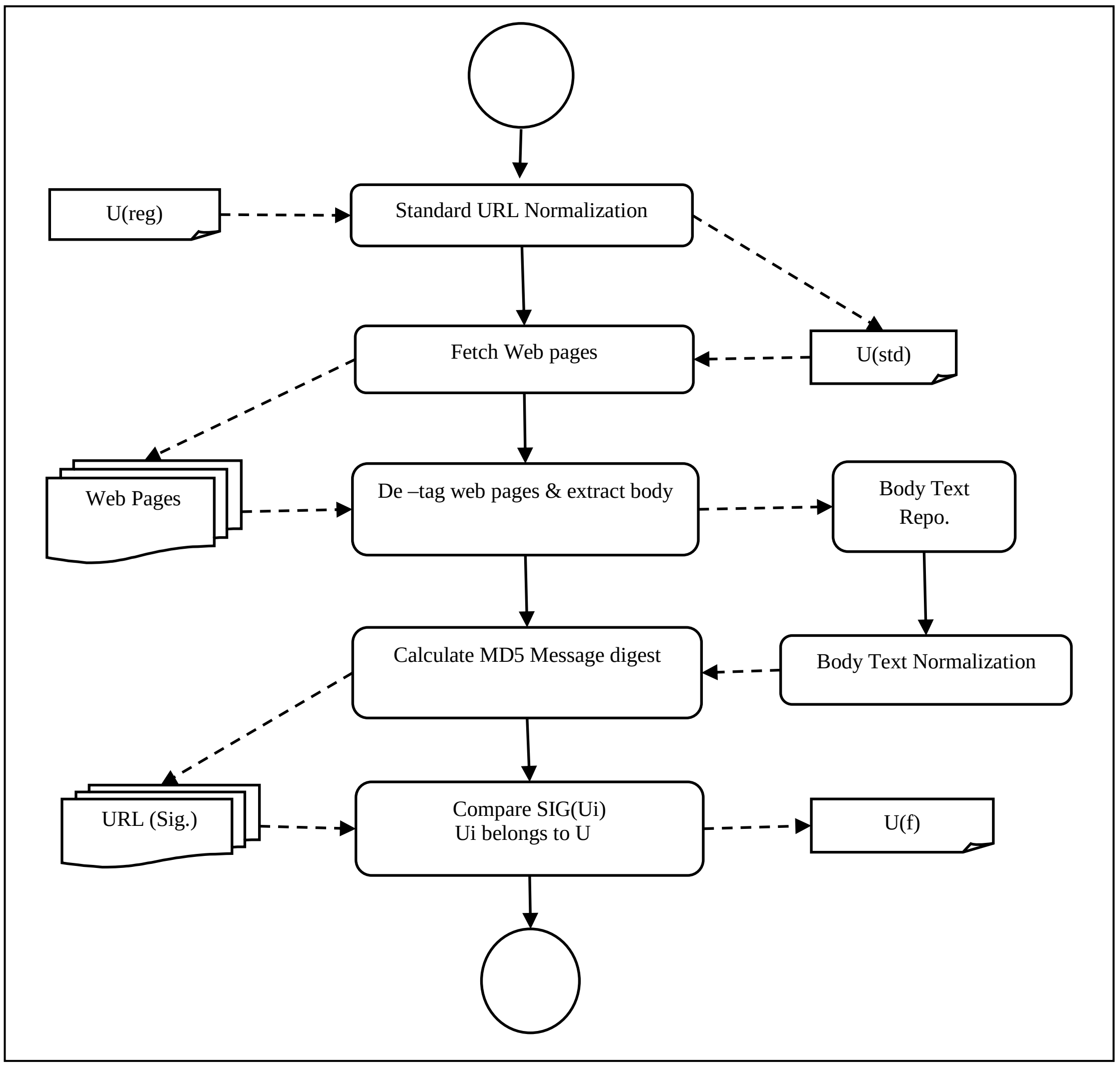

U(reg)
Standard URL Normalization
Fetch Web pages
U(std)
Web Pages
De –tag web pages & extract body
Body Text Repo.
Calculate MD5 Message digest
Body Text Normalization
URL (Sig.)
Compare SIG(Ui) Ui belongs to U
U(f)

Figure 2.17: Step-Wise Working of identifying syntactically Similar URLs

In this method, MD5 hashing algorithm was used to calculate message digest. But this hashing algorithm has limitation of creating same digest values of different webpages.

Luis et al [42] proposed an algorithm for complex hierarchal datasets like XML datasets [87]. A Bayesian network to check whether the two XML documents are duplicates or not was used. The two XML documents nodes are considered as duplicate only when their values are equal and their children nodes are duplicates too. It shows the probability of how many the documents are duplicates. The proposed technique not only works on content within the documents but also on how the information is structured.

Wei Li et al [43] works on both the syntax and semantic part of web documents to eliminate duplicate web documents. It considers keyword sequences for syntax i.e.

structure and semantics i.e. intension feature of web documents. The keywords sequences of different documents are compared to identify duplicate documents. If comparison found similar then documents are considered as same and one of them is discarded.

The summarized comparison on duplicate elimination policies both at URL as well as at Document level is given in table 2.4.

Table 2.4: Duplicate Elimination Policies

Algorithm	**Berner et al**	**Lee et al**	**Agarwal et al**	**Farah et al**	**Luis et al**	**Wei Li et al**
Technique	Normalization Steps	Extension to Normalization Steps	Generate rules based on crawl logs	Hashing	Bayesian Network	Keyword Sequences
Input	URLs	URLs	Crawl logs	HTML document	XML document	HTML document
Results	Unique URLs	Unique URLs	Unique webpages	Unique URLs and webpages	Unique webpages	Unique webpages

2.4.4 Lack of Relevancy

When a user supplies a query, the results shown by the search engine is based on user's query keyword matching. The documents shown by the search engine are based on content matching. But these results are not always relevant to user's query. To provide results more relevant, database should be richer. So, the crawling should also be done on the basis of structure of webpages. The structure of documents is in HTML form which is called as DOM tree i.e. Document object model [91]. Dom is HTML representation of a web document. The composition of HTML documents consists of all nodes whether it is an element, attribute, text etc. In DOM tree, the start node is document node and its branches are extended till all text nodes covered. The structure of a document is given in figure 2.18.

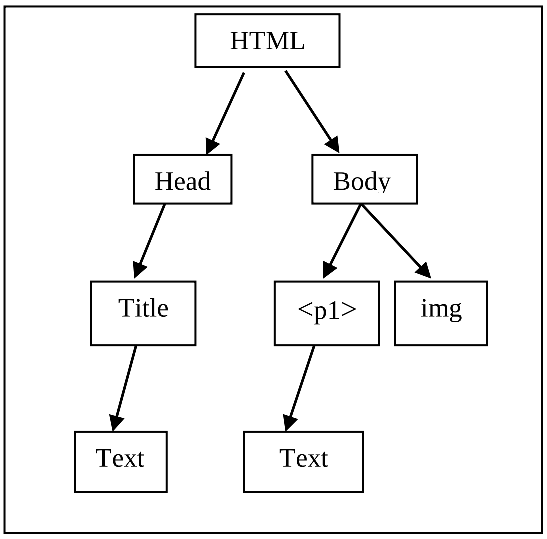


Figure 2.18: DOM Tree

With the help of this DOM tree, HTML structure of web documents is used to check similarity between them. If the structures matched then web documents are related to each other. There are various ways of matching structure of HTML documents. Some are discussed below:-

Ling Yu. et al [44] structure similarity was based on the idea that there exists a similar structure for pages belonging to a specific domain. He considered a web page in the form of a tree with specific tags as the nodes of a tree. Structure x Structure [0...1] returns the degree of similarity of a page structure. If the case is ideal, then if this function has the property that the value of x1 is greater than x2 then it can be concluded that the similarity of x1 to a page is greater than x2.

Vidal [45] proposed a method in which all the pages similar to a given page are known before hand. A tool has used that records all the target pages on a website and hence generate a navigation pattern to reach these targeted pages. The navigation pattern [79] consists of a sequence of patterns that a crawler follows to reach the targeted pages. This tool is then used to create a crawler based on these patterns which can be used to find out similarly structured pages even if similar pages are added later on.

Wang et al [46] considered web pages in the form of HTML structure. HTML pages are then represented in the form of XHTML pages. A document object model of web documents is then presented. For example, if A and B are used to represent two DOM trees corresponding to two HTML documents. Then similarity of A and B is given by equation 2.7.

$$\text{Similarity}(A, B) = \frac{\text{SimpleTreeMatching}(A, B)}{(\text{sizes}(A) + \text{sizes}(B))/2} \qquad \textbf{(2.7)}$$

Where, simple Tree Matching (A, B) represents the number of maximum matching nodes of tree A and tree B;

Sizes (X) represents the number of nodes on tree X.

When similarity (A, B) is closer to 1, tree A and tree B are very similar to each other, and the HTML documents they represent are also very similar.

For a given specific threshold θ ($0 \leq \theta \leq 1$) , if the Similarity(A,B) $\geq \theta$, then the two trees are considered to be matched successfully, and the Web data will be extracted correspondingly; otherwise, the two trees does not match.

Chunying Kang [47] also decomposed these web pages in DOM tree. Then these DOM trees are then traversed in breadth first manner. On the basis of traversal, layer by layer DOM node tree comparison takes place and then the sum of all floors of the changes are computed. Some threshold value is fixed on the basis of which it is decided that if their value is less than some threshold then pair of pages are structurally similar otherwise not.

Nierman et al [48] gives the idea to measure structural similarity between two XML documents. Tree edit distance based measures are used here. The algorithm developed by them is dynamically finds the distance between any pair of documents. A collection of documents are derived from multiple Document Type Descriptors (DTDs), pair-wise distances between documents in the collection are computed and cluster the documents using these distances. It is observed that the resulting clusters match the original DTDs and has better results than previously used similarity methods.

Bertino [49] proposed a matching algorithm to compute the structure similarity between an XML document and its Document Type Definition (DTD). The matching algorithm by comparing the document structure against the one the DTD requires is able to identify the commonalities and differences. Differences are due to the presence of extra elements and not the required elements. The evaluation gives the numerical rank of structure similarity.

The summarized results of all structure driven crawling methods in shown in table 2.5.

Table 2.5: Summary of Structure Driven Crawling methods

Algorithm	**Ling Yu et al**	**Vidal et al**	**Wang et al**	**Chunying Kang**	**Nierman et al**	**Bertino et al**
Input	DOM tree	HTML page	DOM Tree	DOM Tree	DTDs	DTDs
Technique Used	HTML Tag Matching	Navigation Pattern	Convert HTML to XHTML	BFS traversing	Edit distance	DTD matching
Domain	Specific	Input based	General	General	General	General
User Intervention	No	Yes	No	No	No	No

2.4.5 Ranking based on Users' browsing behaviour

In order to get relevant results, a search engine has to modify their page rank methods [112, 114]. It is suggested that before showing the results to the user, their interest should also be taken into consideration i.e. Users' browsing behaviour actions should be involved in showing results [100, 123]. There are many actions on a web page that indicate users' interest. Some of them are discussed below:-

Duration on Web Page

The user opens the web page and then after sometimes closes it. This time interval is called as duration on web page spend by the user. **Morita et al**. [50], **Konstan** [51], **Claypool** [52] and many others conclude that user spends much time on the page which is of his/her interest. According to them, the longer the user spends time on some page, the more interested he/she is.

But this is not always true. As **H. Weinreich et al**. [53] found that in nearly 50% cases user spend much time in deciding whether to move to the next page or not rather in reading the content. It may also possible that due to images on the web page which take time to show is the reason of long duration of user on that web page. Hence, it should not be the only indicator to show user interest on that web page.

Mouse Clicks

The user uses mouse click on the web page for many reasons like for to open hyperlink, to copy the content or may be as habitual behaviour. **Goeck's** browser [54] and **Letizia** [55] examined this indicator and include in the list of users' interest shown actions. The number of mouse clicks shows user interest on that web page. More is the number. More interested the user is.

Keys UP and DOWN and Scrollbar Clicks

The user uses keys UP and DOWN or Scrollbar clicks for scrolling the web page. If the number of keys uses is high that it means the user is interested in that web page. It may be the case that due to slow downloading of web page user uses these keys again and again. Due to high use of these keys may be longer length of the web page.

Print, Copy, Save as buttons

The user uses these buttons often when he is interested in that web page. When a user finds something of his interest on a web page, he may want to take print or may copy the content or may save it to his local drive for future use. So, these buttons are high indicators of users' interest on that web page.

Add to Favorites

This button is the clear indicator of Users' interest. He stores the particular web page under the category of Add to Favorites so that he can revisit it and saves his time. By pressing the Add to Favorites button, stores the URL of that web page for user prompt access.

Open URL

When any URL is opened by directly typing its address in address bar and then click on go indicates that this URL is of user interest.

In other research done by **Ying Xiaomin** [56] obtained the following conclusions:

- The user's browsing behaviour can be classified into three categories, namely physical behaviour (eye rotation, heart rate changes etc.), significant behaviour (save the page, print page and other acts) and indirect behaviours (browsing time, mouse keyboard operation etc.).
- Indirect behaviours are the main source to estimate user interest rate. Significant behaviours act in the event that the user is a high degree of interest in the corresponding page, but fewer significantly behaviours happen, a large number of pages have no corresponding significant acts, as a result, the significant behaviours only play a supporting role in the estimation of user interest.
- With the analysis of user's indirect behaviours, the smallest combinations of browsing behaviours draw as follows: save a page, print a page, store a page in the Bookmark, the number of times to visit the same page, dwell time on a page.

Many studies have found that the precision of user interests extracting without considering user behaviours is inaccurate. **Claypool** [52] proposed that user behaviours including residence time, visiting frequency, saving, editing could reveal user interests. **Weinreich** [53] presented that average reading speed plays a key role in determining the grade of user interests.

Goecks and Shavlik [54] proposed an approach for an intelligent web browser that is able to learn a user’s interest without the need for explicitly rating pages. They measured mouse movement and scrolling activity in addition to user browsing activity (e.g., navigation history). It shows the somewhat better results are obtained as compared to previous work.

Kun Xing et al [57] suggested that some actions like scrolling, mouse clicking should be included in the total browsing time. They worked on user browsing history and analyze documents which user has visited. On analysis, he concludes that browsing time and printing are important actions in showing users’ interest.

Zhao et al [58] proposed a mechanism for service providers to understand the users’ needs. After analyzing user browsing behaviour, this mechanism understands users’ needs and accordingly providing services to them. Each user has individual needs and required different services. With the help of this mechanism, the service provider provides different services to different users. Support Vector Machine is used here for analyzing users' browsing behaviour. This is supervised learning method in which browsing indicators are classified and helps in predicting their interest.

Yang et al [59] proposed personalized teaching software based on students interest. Students' interest was calculated by analyzing their browsing behaviour. By knowing their interest, they developed the software and cater the need of students in efficient way.
The all summarized browsing indicators in shown in table 2.6.

Table 2.6: Summarized Browsing Indicators

Browsing Indicators ↓	**Xiaomin, Claypool, Weinreich**	**Goecks & Shavlik**	**Kun Xing et al**	**Zhao et al**	**Yang et al**
Hyperlinks Clicked	No	Yes	No	No	Yes
Scrolling Activity	No	Yes	No	No	No
Mouse Activity	No	Yes	No	No	Yes
Keyboard Activity	No	No	No	No	No
Time on Page	Yes	Yes	Yes	No	Yes
Print	No	No	Yes	Yes	Yes
Prediction user's interest for the page	No	Yes	Yes	Yes	Yes
Explicit rating	No	No	No	No	No
Saving	No	Yes	Yes	Yes	Yes
Number of visit	No	No	No	Yes	Yes

2.5 PROBLEM IDENTIFIED IN EXISTING APPROACHES

A critical look at the available approaches indicates the following issues need to be deal with towards building an effective Migrating Crawler:

- In distributed architecture, multiple crawling instances download each and every page whether relevant or not which may leads to the unnecessary congestion on the network.

- In existing URL assignment strategies due to wrong selected threshold value, some of the crawler node may be left unattended and which may lead to uneven load distribution.
- The existing URL scheduling methods fails when the system grows i.e. methods are non scalable.
- To maintain quality of database, MD5 hashing is used to eliminate duplicate contents. It is observed that this method generates the collision i.e. same digest values are computed for different inputs.
- None of the structure based techniques consider structure matching at crawling level to maintain database richer with more relevant pages
- None of the users' browsing behaviour technique considers all the indicators to find the actual users' interest in existing researches. Also no ranking technique considers users' browsing behaviour while ranking the webpages before presenting them to the user.

A design of a structure driven cooperative migrating crawler for retrieving quality data, deals with all the identified shortcomings/problems found in existing work. The architecture of the proposed work with brief discussion on each component is given in next chapter.

CHAPTER III

DESIGN OF A STRUCTURE DRIVEN, COOPERATIVE MIGRATING CRAWLER FOR RETRIEVING QUALITY DATA

3.1 INTRODUCTION

The proposed Search Engine works in three layers namely remote web layer, crawler layer and user layer. Each layer is well interfaced with other layers to perform the search engine function in appropriate manner. The figure 3.1 shows the three layer architecture of the proposed work.

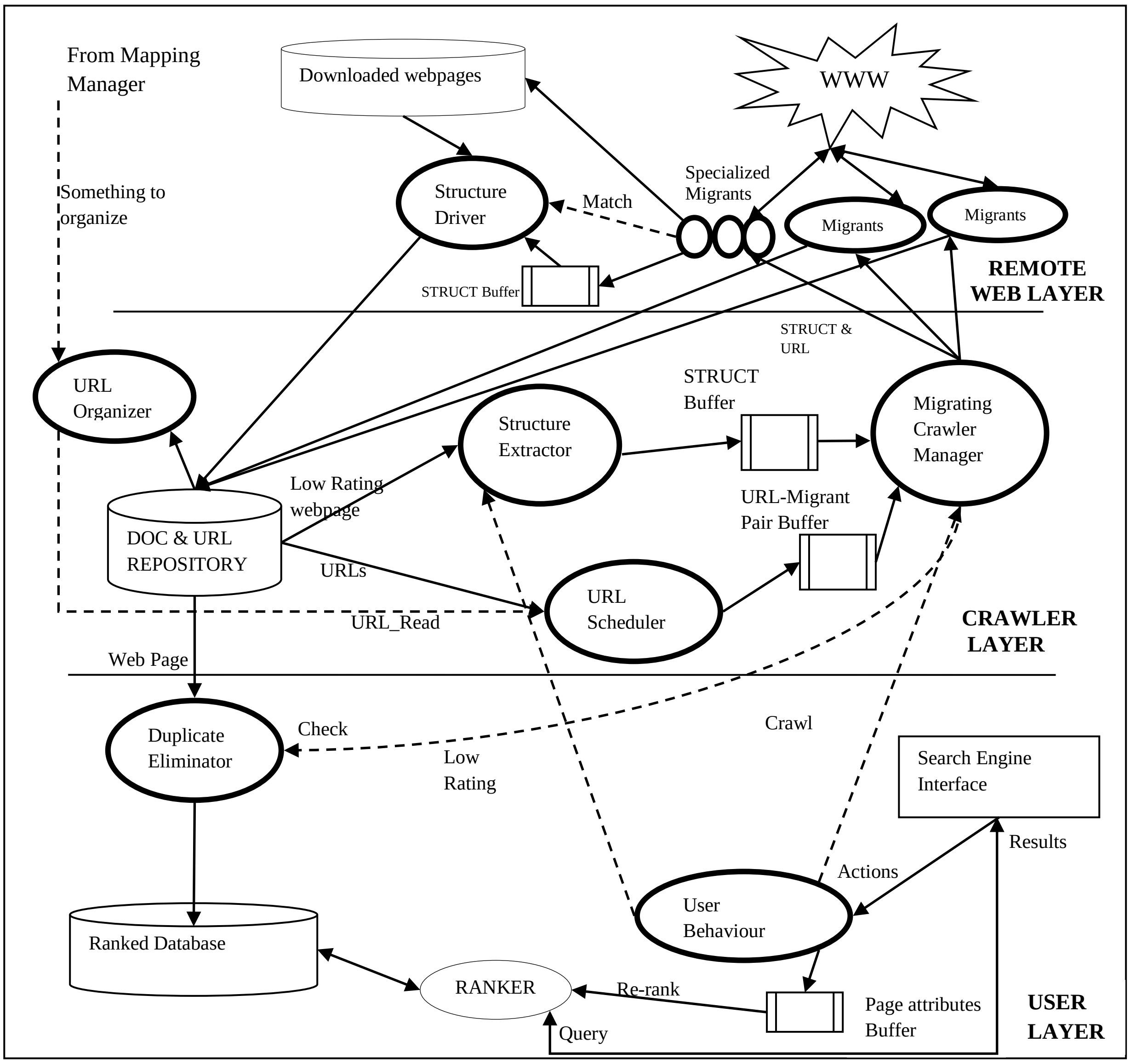


Figure 3.1: Three layer architecture of the proposed work

The working of all three layers is given below:-

a) Crawler Layer

At this layer, all the work is mainly done by crawler. This layer takes seed URLs as input. After getting URLs list, URL Organizer organizes them in some order to be distributed among the Migrants for crawling. URL Scheduler is responsible for picking up a migrant from the executing/available migrants which is then supplied a URL for downloading. Migrants crawl the web as depicted in Remote Web Layer. One more module is there which is responsible for extracting the structure of webpages called as Structure Extractor and supplied to Migrating Crawler Manager. Migrating crawler manager then creates more Migrants dynamically and supplies these structures along with URL for structure driven crawling of the web.

b) Remote Web Layer

At this layer, Migrants come into action. Two types of Migrants are specified at this layer. These are Migrants and Specialized Migrants. Migrants are the normal Migrant then gets the URLs from migrating crawler manager from crawler layer and crawls the web. Whereas Specialized Migrants are responsible for structure driven crawling. They are supplied with structures of webpages whose IDF is high. These structures are then stored in STRUCT Buffer which are then utilized by another module called as Structure Driver. The role of structure driver is to match the structure of those webpages that are crawl by specialized migrants and the selected webpages are then stored to DOC & URL Repository which is at Crawler Layer for further processing.

c) User Layer

This layer contains search engine interface where user supplies a query and gets the results. A ranked database is maintained by two modules namely duplicate eliminator and user behaviour analyzer. The role of duplicate eliminator is to remove redundancy from the downloaded webpages before storing them to ranked database. The User Behaviour Analyzer is responsible to capturing the users' interest and stores them as webpage attributes so that they may be used while ranking of the webpage.

3.2 DESIGN OF A STRUCTURE DRIVEN, COOPERATIVE MIGRATING CRAWLER FOR RETRIEVING QUALITY DATA

The detailed design of the Structure driven, cooperative, migrating crawler for retrieving quality data is shown in figure 3.2. It takes the input from URL queue, where pair of URL and its sitemap is stored.

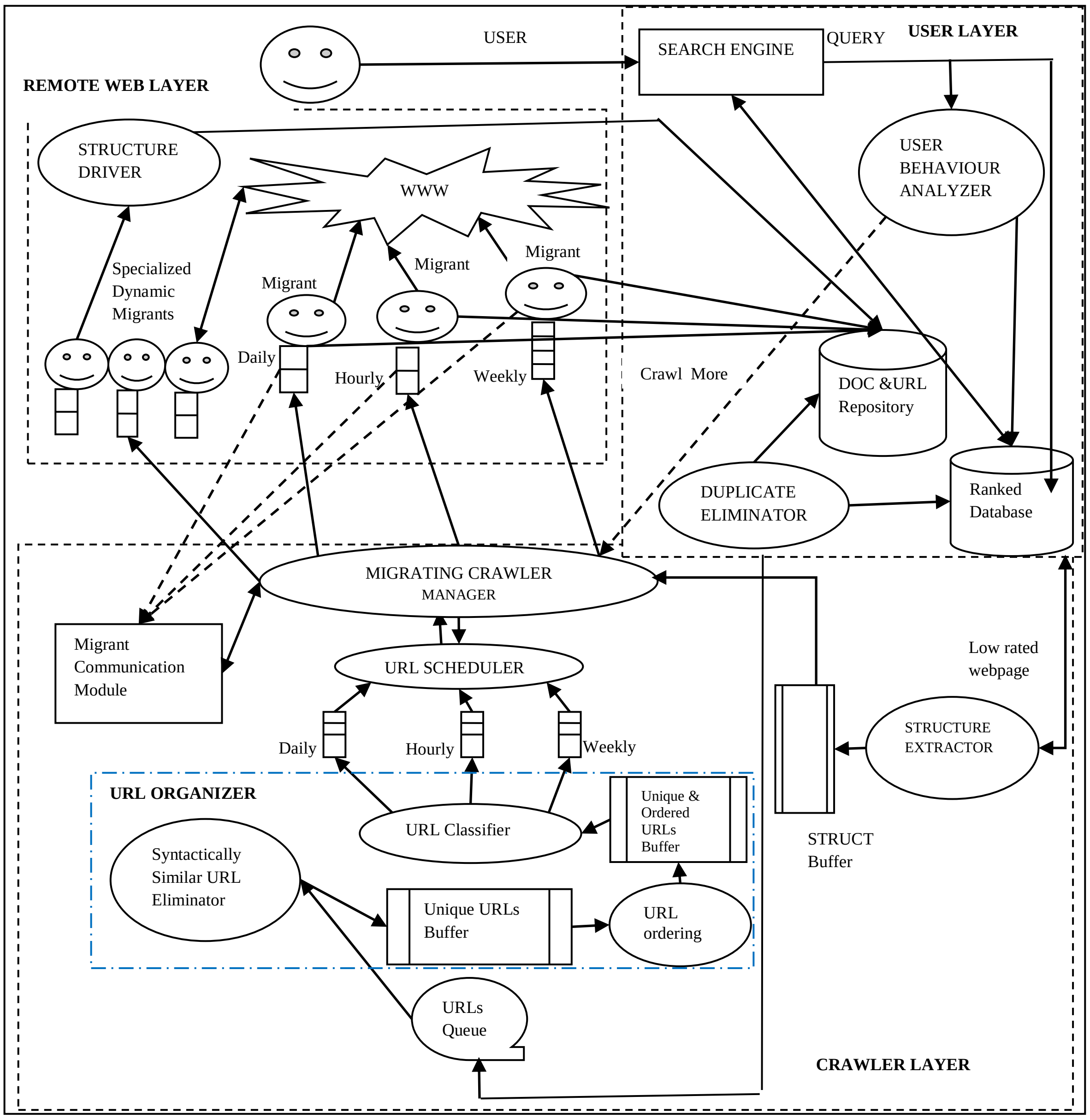


Figure 3.2: Architecture of Proposed Migrating Crawler

The system consists of the following major functional components:

i. URL Organizer

ii. URL Scheduler

iii. Migrating Crawler Manager

iv. Duplicate Eliminator
v. User Behaviour Analyzer
vi. Structure Extractor
vii. Structure Driver

A brief discussion on each of these functional components is given below:

3.2.1 URL Organizer

The role of URL organizer is to organize the URLs before scheduling them migrants. After getting signal (something to classify) from Mapping Manager, it takes URLs from DOC & URL Repository. It consists of following three functional components:-

(i) Syntactically Similar URL Eliminator

(ii) URL Ordering

(iii) URL Classifier

A brief discussion on each of these components is given below:

(i) Syntactically Similar URL Eliminator

It is responsible for eliminating URLs which has same syntax. This module gets the input from URL Queue and applies Standard Normalization process to those Urls for identifying the syntax similar URLs. There are six normalization steps that help in identifying the similar syntax URLs as discussed below:-

- Step 1: Change letters in the scheme component into the lower-case letters
- Step 2: Change letters in the host component into the lower-case letters
- Step 3: Eliminate the default port (i.e., “:80”)
- Step 4: Transform a null path string into the slash symbol
- Step 5: Decode unreserved characters
- Step 6: Eliminate the fragment component

With the help of these normalization steps, syntactically similar URLs are identified. By using string matching [105] the identified syntactically similar URLs are then eliminated.

After applying above steps on URLs list, the unique URLs list then stored in Unique URL buffer.

(ii) URL Ordering

The aim of this module is to order the URLs in efficient manner so that maximum coverage in less time can be achieved. It takes the input from Unique URL buffer, where along with URL its sitemap [61, 97] is also provided. A sitemap is an XML file in which a list of links available on that particular site is mentioned. Sitemap also contains the change frequency, last modification and priority of each link contained in that site. It is maintained by Website owner so that his site can be fully access by the crawler. With the help of this information, the URL which has maximum link is placed at the first position and so on. It may be the case that after getting hyperlinks of each webpage corresponding to an URL, some links appear in more than one webpage. So, URL Ordering module deletes one of the common links to prevent redundancy of same link. The ordered links are stored in unique and ordered URL list.

(iii) URL Classifier

The URL Classifier module takes the input from unique and ordered URL list and classifies them with the help of Sitemaps. The basis of classification is page change frequency (i.e. hourly, weekly, monthly etc) given in sitemap and makes their separate lists. This helps in assigning the URLs to different set of Migrants on the basis of their change frequency.

The algorithm for URL Organizer is given in figure 3.3.

```
URL_Organizer
Do Forever
Step 1: Wait (something_to_classify)
      2. Pick URLs from URL queue.
      3. Call Syntactically Similar URL Eliminator ()
         3.1 Check for duplicate URLs after undergoes normalization process.
      4. Call URL Ordering ()
         4.1 Calculate weight of each URL with the help of sitemap.
         4.2 Ordered the URLs with the highest weight first.
      5. Call URL Classifier ()
         5.1 Classify URLs on the basis of their change frequency i.e. hourly, daily,
         weekly etc.
      6. Pick the first URL and send it to URL Scheduling module.
      7. Signal (URLs_ ready).
```

Figure 3.3: Algorithm: URL Organizer

The detailed description of the URL Organizer module is given in subsequent chapters. The working of URL scheduler is given below.

3.2.2 URL Scheduler

The proposed URL Scheduler module gets various lists of URLs from URL classifier. URL Scheduler is responsible for picking up a migrant from the executing/available migrants which is then supplied a URL for downloading. The scheduling is done on basis of criterions of URLs as well as machine on which the migrants is presently executing. To get best alternative by using both criterions an Analytical Hierarchy Process (AHP) has been applied. It works on situation where multiple criterions are available for taking decision. In scheduling URLs, this AHP works well. The AHP methodology in brief is given below:-

a. First, design a structural modal with goal, criterions and alternatives. In this work, goal is to select a Migrant for URL, criterion are Migrants information like its load, URL capacity, N/W Bandwidth etc and alternatives are various available/executing Migrants.
b. Then, weights of different criterions are evaluated.
c. Next, a score is marked for each criteria of a migrant.
d. Then, a total weight is calculated for each given migrant.
e. Finally, Migrant with the maximum weight gets the next URL for downloading.

Following are the criterions which are used here:-

- Agent Load: the number of task executed at Migrant side is considered as its load.
- Agent Capacity: the size of hard disk available at Migrant side
- Network Latency: amount of time required to transfer a data from source to destination
- Network Bandwidth: Data rate supported by network connection
- Loading rate: ratio of the number of crawl tasks to memory capacity
- URL Capacity: it is measured as in terms of number of forward links
- CPU: it is expressed in terms of frequency
- URL Parent Rank: the rank of parent URL from which it is linked.

The relative importance of each criterion is calculated with the help of 9-point scale which is given by Saaty [60]. Similarly, the contribution of each criterion with available alternatives is calculated using same scale. In this case, Migrants are alternatives. So, contribution of every criterion for each Migrant is calculated. At end, total contribution in terms of weight is calculated and the Migrant with maximum weight has been assigned highest rank in terms of getting the next URL.

The algorithm for URL Scheduler is given in figure 3.4.

```
URL_Scheduler ()
Step 1: Do Forever
        2: wait (URL_Ready)
        3: Pick URLs from DOC & URL Buffer
        4: Get Migrants Information from Migrating Crawler
        5: Apply AHP technique on available Migrant data to schedule Migrant.
        6: Supply the URL to the migrant with the maximum weight.
        7: signal (Migrant_selected)
        8: End
```

Figure 3.4: Algorithm: URL Scheduler

After selecting migrant, URL scheduler send signal (Migrant_Selected) to the Migrating Crawler Manager and scheduled Migrant with maximum weight gets the URL for downloading. The detailed description with experiment analysis is given in subsequent chapters.

3.2.3 Migrating Crawler Manager

The Migrating Crawler Manager is one of the main modules of proposed architecture. It is the responsibility of migrating crawler manager to create multiple Migrants and with the help of these Migrants crawl the web as maximum as possible. It takes the Migrants information from Migrant Communication Module (i.e. load, URL capacity, N/W Bandwidth etc) and provided it to the URL scheduler so that the URL scheduler can schedule the appropriate Migrant based on available information. URL scheduler picks up an appropriate migrant on the basis of available information and provides it to the URL for downloading. It also creates the migrants depending upon the classification of URLs done by URL classifier like Migrants for daily change webpages, weekly change webpages and so on. For each list, it has a different set of migrants and accordingly schedules the URLs. After getting list of URLs, migrants crawls the web and send signal (done) to migrating crawler manager for further processing. The algorithm of Migrants is given in figure 3.5.

```
Migrant ()
Step 1: Pick the URLs assigned by the Migrating Crawler Manager and stores
        them in MainQ.
      2: While (remote MainQ ≠ empty)
        2.1: Download robot.txt.
        2.2: If unable to download
             2.2.1: Set IP part as blank.
             2.2.2: Store URL in local document & URL buffer.
             Else
             2.2.3: Read robot.txt.
             2.2.4: Download extracted links.
             2.2.5: Segregate the internal and external links.
             2.2.6: Add URL and internal links to LocalQ.
             2.2.7: Store the external links to DOC & URL Repository.
      3: While (remote LocalQ≠empty)
         3.1: Pick a URL from LocalQ.
         3.2: Download documents and store them in document and URL buffer
      4: Call Duplicate Eliminator ().
      5: Signal (check)
      6: Signal (done).
```

Figure 3.5: Algorithm: Migrant

The main functions of Migrating Crawler Manager are:-

1. Migrant Communication
2. Structure Driven Crawling

The detailed explanation of each of the function is discussed below.

- **Migrant Communication**

It is responsible for establishing and maintaining communication between the Migrants executing and Migrating Crawler Manager. The communication between migrants and Migrating Crawler Manager is implemented with the help of Aglets. Aglet is a java based mobile agent technology.

There are many communication models exist like home proxy, follower, email etc. In the present work, home proxy model is used for communication. The migrants get the absolute URL from Mapping Manager. Migrant carries itinerary along with it. Itinerary is a travel plan of Migrants. With the help of this itinerary aglets can roam the World Wide Web easily. The planning of itinerary can be done on various criterions like list of sites to be visited in some order, what are the actions done by Migrant, source & destination addresses, etc. In the present work, the itinerary has following attributes:-

(i) Sender & Receiver Address

(ii) URL to be fetch

(iii) Authenticity of Sender

After reaching at server side, Migrant's itinerary is first explored and then allowed to access the database. Migrants take the advantage of local downloading, filtering and duplicacy detection before transmitting the downloaded pages to the central machine. The proposed Migrating Crawler Manager has special property that it can create its Migrant dynamically also as per requirement. These dynamically created migrants download webpages on the basis of their structure. This structure driven crawling is explained in next section.

- **Structure Driven Crawling**

The Migrants are proposed to specialize in crawling the web on the basis of webpage structure instead of content matching. The structure of a webpage is taken in form of DOM tree. There is a module in proposed migrating crawler architecture called as User Behaviour Analyzer [62]. This module keep a track on users' activity on web page like saving the page, printing the page, time spend upon page, rating giving on page, closing of browser etc. On closing the browser or on retyping query, feedback is taken from user whether the information provided by search engine is sufficient? If he/she gives the answer as NO, this module sends signal to crawler manager to create dynamic specialized Migrants. These specialized Migrants crawls the web on the basis of structure of webpages.

A module at crawler layer called as structure extractor extracts the structure of web pages and sends it to the migrating crawler manager for supplying it to specialized Migrants. After receiving the structure from migrating crawler manager, these Migrants then crawl the web and stores only those web pages that have similar structure that of supplied web page structure. By doing this, the repository is getting richer with the more pages of users' interest.

The algorithm for Migrating Crawler Manager is given in figure 3.6.

```
Migrating_Crawler_Manager ()
Do Forever
Step 1: Get Migrants information from Migrant Communication Module and send it to
the URL scheduling module
        2: Wait (Migrant_selected)
        3: Get selected migrant from URL scheduling module
        4: Send selected migrant to the remote server for crawling
        5: Wait (crawl_more)
        6: Create specialized Dynamic Migrants
        7: Take structure from STRUCT Buffer.
        8: Send picked Structure to Migrants
        9: Migrants stored these structures to remote STRUCT Buffer
       10: Migrants crawls the web
       11: wait (done)
       12: Signal (check)
       13: End
```

Figure 3.6: Algorithm: Migrating Crawler Manager

3.2.4 Structure Extractor

The role of structure extractor is to extract the structure of webpage. The structure of web pages is in the form of DOM tree form. The DOM tree is extracted with the help of IE DOM Inspector. . After getting signal (Low_Rating) from User Behaviour Analyzer, it gets those webpages from the URL & DOC Buffer and extracted and stored it to the STRUCT buffer. The Migrating Crawler Manager can supply these structures to the migrants for the purpose of downloading the documents with similar structure. The algorithm for Structure Extractor is given in figure 3.7.

```
Structure_Extractor ()
Step 1: Do Forever
         2: wait (Low_Rating)
         3: Pick webpage from DOC & URL Buffer.
         4: Extract structure of picked webpage.
         5: Stored in STRUCT buffer.
         6: End
```

Figure 3.7: Algorithm: Structure Extractor

3.2.5 Structure Driver

The purpose of structure driver is to stores only those webpages which are similar in structure. This module is available at remote web layer. The structure driver performs two functions: structure extraction and structure matching. It extracts the structure of

those webpages that are downloaded by specialized migrants. These structures are then matched with structure stored in STRUCT buffer. The STRUCT buffer contains those webpages structures that are supplied by migrating crawler manager to migrants for downloading similar structure webpages. If matches, webpage gets stored in DOC & URL repository otherwise gets stored at Migrant side and can be used later on.
The algorithm for Structure Driver is shown in figure 3.8.

```
Structure_ Driver ()
Step 1: Do Forever
        2: Pick webpage from downloaded webpages
        3: Extract structure of downloaded webpage
        4: Match structure of downloaded webpage and stored in STRUCT Buffer
        5: Matched documents/ pages get stored in URL & DOC Buffer.
        6: End
```

Figure 3.8: Algorithm: Structure Driver

It will work only when it get signal form User Behaviour Analyzer. The detailed architecture and experiment analysis will explain in subsequent chapter.

3.2.6 Duplicate Eliminator

After the downloading by different Migrants, Migrating Crawler Manager sends signal (check) to the duplicate Eliminator. The Duplicate eliminator will check the webpage before storing them to ranked database. The webpages are first checked with stored webpages i.e. already crawled webpages and if they are found new then it will pass the webpages to link extractor. Otherwise, compare its last modification date and time because it may be the case that same webpage again crawled due to some changes in it. So, if it is updated only then it gets stored. Afterwards, it also checks duplicacy in extracted links. If extracted links matched with already visited stored links, then it gets discarded and if they are new only then they are added to URL Queue for crawling. Checking is done on the basis of hash values of web documents. With the help of hashing, the time of matching reduces sufficiently.

The algorithm for Duplicate Eliminator is given in figure 3.9.

```
Duplicate_Eliminator ()
Step 1: Do Forever
        2: Wait (check)
        3: Take webpage from DOC & URL Buffer
        4: generate its SHA-1 Hash
        5. Match with existing Digest values of each web page available in stored page
           Database.
        6. If (exists)
           6.1 Remove the web page
        7. Otherwise
           7.1 Store web page in ranked database
           7.2 Check duplicacy in extracted links
                 7.2.1 If (exists)
                       Check for modification
                 7.2.2 Otherwise
                       Added to URL Queue
          8. End
```

Figure 3.9: Algorithm: Duplicate Eliminator

For storing URLs cache and memory both are used. The advantage of cache is that it has faster access as compared to memory. The recently used URLs are stored in cache to prevent time delay in matching duplicate URLs. As the size of cache is small, so the limited data can only be stored on it. Many replacement algorithms are there such as LRU, FIFO etc for replacement.. Here, LRU is used i.e. the URL which is least recently used is removed. The detailed description and experiment analysis is discussed in subsequent chapters.

3.2.7 User Behaviour Analyzer

This module analyses the behaviour of user on webpages [62]. The behaviour are the actions that a user is performing on a particular webpage. It also maintains date of visit, URL and feedback information of each webpage related to each user. A browser is developed to store all these actions in a database. A data mining algorithm is applied on this stored information. Apriori data mining algorithm is used here. With the help of calculated values of support and confidence, the weight of webpage is calculated. In Apriori algorithm, Confidence value(C) can be calculated as per equation 3.1.

$$\textbf{Confidence value(C)} = \frac{\textbf{support\_count (most frequent action)}}{\textbf{support\_count (subset of most frequent action)}} \qquad \textbf{(3.1)}$$

Based on the confidence value of each subset the weight of a webpage can be calculated as per given equation 3.2.

$$\mathbf{P_{wt} = C_1 + C_2 + C_3 + \ldots\ldots\ldots\ldots + C_i = \sum C_i} \qquad \textbf{(3.2)}$$

where C_1, C_2, C_3…. are the confidence values of subsets of frequent occurring actions which satisfy minimum confidence threshold.

This weight is considered as rank of that particular webpage. The rank of webpage is highest whose page weight is highest. This module also stored the feedback of information provided by proposed browser. This information is stored with webpage who get highest rank at the time of feedback. Either on closing the browser or on retyping the query, a pop up window is opened and asked the User that is the information provided by the search engine is sufficient? If he says yes then fine otherwise it sends signal to crawler manager to do more crawling for similar information. On getting signal from user behaviour analyzer, crawler manager creates specialized Migrants to crawls the web.

The algorithm for User Behaviour Analyzer is given below in figure 3.10.

```
User Behaviour Analyzer ()
Do Forever
Step 1: Read visited pages.
     2: Analyze that page.
     3: Maintain log files of the browsing behaviours.
     4: If (satisfied==No)
        4.1: Signal (crawl_more)
        4.2: Signal (Low_Rating)
     5: Apply Apriori algorithm on log files.
     6: Calculate Page Weight on the basis of user behaviour
     7: Rank the webpage as per weight calculated.
     8: Store them in ranked database
     9: End
```

Figure 3.10: Algorithm: User Behaviour Analyzer

The detailed description of the each module of the proposed system is given in subsequent Chapters.

CHAPTER IV

DESIGN OF A NOVEL URL SCHEDULING MECHANISM USING ANALYTIC HIERARCHY PROCESS (AHP)

4.1 INTRODUCTION

With the increase in the size of the web, it is necessary to ensure the richness and uniqueness of information available on it. The crawler is the main module responsible for gathering the information from the web. There are many design issues [63] while designing the crawler like uniqueness and richness of database, cooperation between migrants, fast and efficient crawling. There are many types of crawler exists such as Parallel Crawler, Migrating Crawler, Focussed Crawler, Hidden Crawler, Incremental Crawler etc. In this chapter, the capabilities of migrating crawler [6] are being utilized for designing an efficient crawling system. The proposed Crawler tries to achieve all the design issues needed for efficient crawling. In this work, the improvement in quality with minimum communication overhead of database is achieved by proper scheduling of Migrants.

4.2 UNIFORM RESOURCE LOCATOR (URL)

URL is uniform resource locator. It is used to address a resource. In World Wide Web, the resource is a web page and URL is the address of that web page. User searches by query and search engine in return show URLs of corresponding web pages from its repository. The Search Engines repository is maintained by a Crawler. Crawler gets the seed URL from URL Queue and starts the crawling process. It fetches a web page corresponding to that seed URL and extracts the internal URLs from that page and added to URLs Queue. Crawler picks the URLs from URL Queue and the process repeats until the Queue is empty. Now, as the size of the web increases tremendously, retrieving the information available on the net is becoming a very tedious task. The task becomes more difficult when crawler fetches the same page but with different URLs. This problem is designated as DUST [39, 74] i.e. different URLs with similar text.

DUST effect the whole working of Search Engines i.e. crawling, indexing, ranking etc. There are many reasons of DUST. Some are listed below:-

1. To Balance Load
2. To served as Backups
3. More user-friendly i.e. By creating shortcuts
4. To reduce network traffic

There can be many more reasons for duplicate URLs. Creating duplicate URLs may be benefits to Internet Users or webmasters but creating trouble to working of the search engine. By downloading the same page again will waste network bandwidth then creating an index of these duplicate URLs will waste time and effort. Moreover, at search engine interface when the user sees the similar page again then he or she may get irritated. In this work, duplicity at URL level is trying to eliminate.

The URL is composed of five components namely the scheme, authority, path, query and fragment components [37] as shown below:

$$\underbrace{\text{http:}}_{\text{scheme}}\underbrace{\text{//www.jabong.com}}_{\text{authority}}\underbrace{\text{/women/clothing/Biba/}}_{\text{path}}\underbrace{\text{?q=biba}}_{\text{query}}\underbrace{\text{\#pos=3}}_{\text{fragment}}$$

The brief description of each component of URL is given in Chapter II.

There are URLs which points to the same page. This problem is designated as DUST [39] i.e. different URLs with similar text. DUST effect the whole working of Search Engines i.e. crawling, indexing, ranking etc. In order to remove this duplicity at URLs level proper processing has been adopted. Before checking for duplicity in URLs, URL standard Normalization process [37] is applied to them. This process eliminates the syntactically similar URLs. There are three types of normalization process:

1. Case normalization

   Conversion of scheme component letters and hostname to lowercase is done in this type of normalization
2. Percent-encoding normalization

   All unreserved characters like ~, _ etc are decoded into %form.
3. Path segment normalization

Remove all ‘.’, ‘..’ from the path component of the URL.

Remove the fragment component from the URL i.e. after#.

Eliminate port number like 80.

Remove ‘/’ from the end and add ‘/’ at path location if it is null.

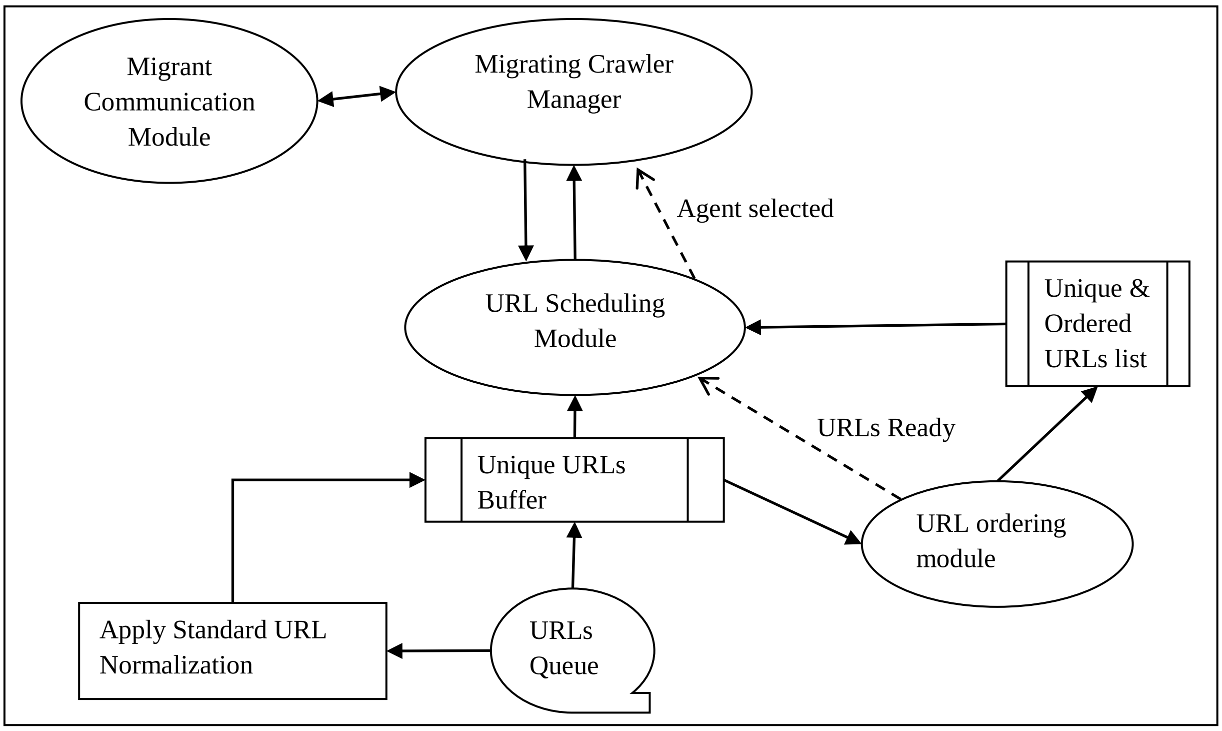
With the help of these normalization processes, syntactically similar URLs are identified. By using string matching the identified syntactically similar URLs are then eliminated.

Sitemap [61] is also used here to get the information of all links present in a web page. Basically, sitemap gives the number of links present in a web page. This helps in ordering the URLs before downloading them.

4.3 PROPOSED WORK

The architecture of proposed migrating crawler consists of following major modules and is shown below in figure 4.1.

1. URL Ordering Module
2. Migrant Communication Module
3. URL Scheduling Module
4. Migrating Crawler Manager



Migrant Communication Module
Migrating Crawler Manager
Agent selected
URL Scheduling Module
Unique & Ordered URLs list
URLs Ready
Unique URLs Buffer
URL ordering module
Apply Standard URL Normalization
URLs Queue

Figure 4.1: Architecture of Proposed Migrating Crawler with URL Scheduling

The detailed description of each module is given below:-

4.3.1 URL Ordering Module

It takes normalize URLs from URL Queue as input and removes duplicates if any. For ordering the URLs it takes the help of sitemap which contains the graph of internal links corresponding to each URL. The URL which has the highest number of links is considered for first downloading.
The number of internal links found in an URL with the help of sitemap is considering the weight of it. As discussed earlier, the multiple URLs may have the common links which may lead to duplicate downloads. So for removing these duplicate links DOM tree graph of each URL is used. And after every download, each URL will generate its new DOM tree graph with that commonly downloaded link may be deleted from it. Let’s take an example,

In the following example, 6 URLs from the URL Queue namely A, B, C, D, E, F have been taken and their internal links are obtained from Sitemap. The links which are common are marked black as shown in figure 4.2:

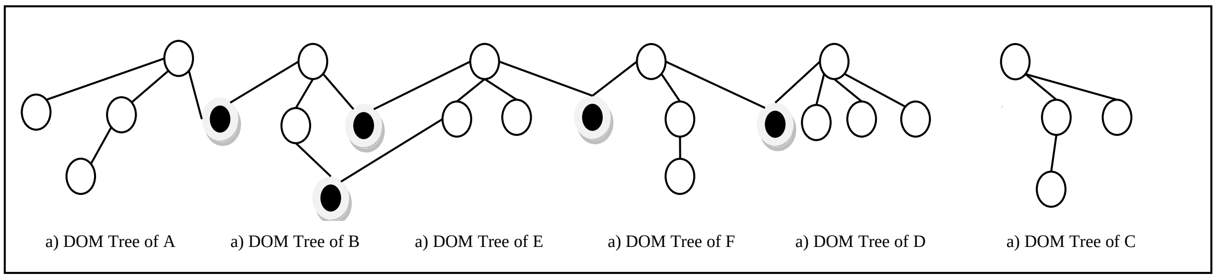


Figure 4.2: DOM Trees of URLs

Small case letters are used for representing the internal links as given below:-

For site A: a1, a2, a3, a4 are internal links

Likewise,

B: b1, b2, a1, b3.

C: c1, c2, c3.

D: d1, d2, d3, d4.

E: e1, e2, b3, e3, b1.

F: f1, f2, d1, e1.

With the help of DOM tree graph, internal links of each website are obtained. Now, URL with highest internal links is considered to be submitted to crawler manager and rest are ordered accordingly. The reason behind this consideration is more the number of internal links means maximum coverage of Web by crawling a maximum number of links.

So, the obtained order is E, A, B, D, F, C. After the selection, all other URLs are matched with this initially selected URL. The internal link which is common to both will be deleted from other URLs to avoid multiple crawling of the same URL and the process continues with other URLs. This can be done as follows:

$E \cap A = \emptyset$, $E \cap B = b1, b3$, $E \cap C = \emptyset$, $E \cap D = \emptyset$, $E \cap F = b1$.

By this comparison with E, b1 and b2 links may be removed because of duplicate matching. After crawling of E, the selection algorithm starts again with the following situation:

A=a1, a2, a3, a4.

B=b2, a1.

C=c1, c2, c3.

D=d1, c1, d2, d3.

F=f1, f2, f3, a1.

Likewise reapplying selection algorithm on above links, the next new order will be A, D, F, C and B. This process repeats until the URL Queue is empty. It may be observed that the order has changed after first selection because of common internal links available in other URLs. So, every time when URL is to be selected, the internal links of every URL are to be calculated every time before submitting to migrating crawler manager.
The algorithm for URL Ordering is given in figure 4.3.

```
URL Ordering Module ()
Do Forever
Step 1: Pick URLs from URL queue.
        2: Check for duplicate URLs after undergoes normalization process.
        3: Find out a number of links of each URL with the help of sitemap.
        4: Order the URLs with the highest number of internal links first.
        5: Pick the first URL and send it to URL Scheduling module.
        6: Signal (URLs_ ready).
        7: After ordering, all URLs matched with initially selected URL
            7.1: Internal common links get removed
        8: Again go to Step 4.
        9: End
```

Figure 4.3: Algorithm: URL Ordering

4.3.2 Migrant Communication Module

In migrating crawler, crawler manager create its Migrants and send them to WWW. These Migrants then behalf of migrating crawler crawls and send the downloaded pages to the Migrant machine. The communication between migrant and server is implemented with the help of Aglets. Aglet is a java based mobile agent technology. The communication architecture is given below in figure 4.4.

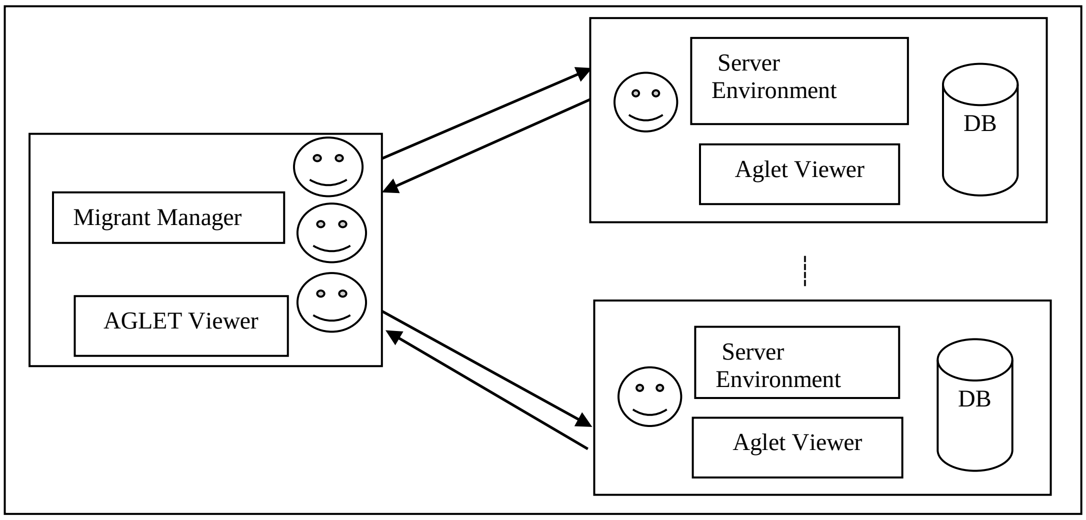


Figure 4.4: Migrant- Server Communication

When a migrant visits server then it is necessary that server machine also has aglet technology on its machine. Without aglet at both sides, communication can’t take place.

Aglet Model

It is an IBM's Agent Technology for programming migrant in Java. In this proposed work, home proxy model is used for communication. In this method, sender knows the receiver's name. It has lookup service through which it gets the receiver's address. The sender knows the URL as name of server and from DNS resolver gets the corresponding address.
The Aglet architecture has following key concepts:-

- **Aglet**: it is migrant in Java technology. It visits different machines and these machines have aglet technology in them.

- **Proxy**: It serves as aglet representative. It is responsible for protection of aglets from outside sources. It can also provide location transparency i.e. hide the location of aglet from other aglets.

- **Context**: it is aglet execution place. It provides aglet running environment where aglets can run conveniently and also protect from malicious aglets. The server's address and their name combined become the context name.

- **Identifier**: A unique number which is associated with each aglet and also it is permanent throughout the lifetime of an aglet.

During the lifetime of an aglet, it comes across the number of states or operations as shown in figure 4.5.

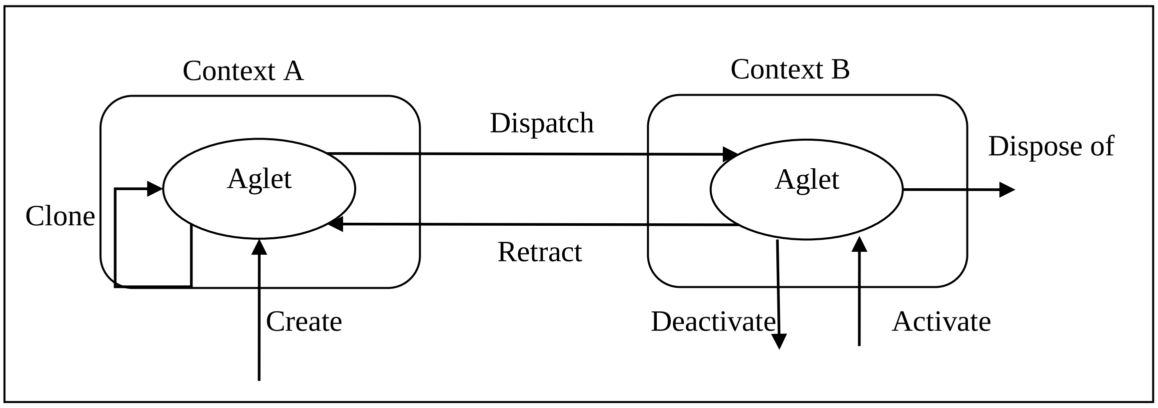


Figure 4.5: Migrant Life Cycle

The explanation of each state is given below:-

- **Creation**: it is created in context and an identifier is assigned to it during this phase.
- **Cloning**: it is the phase when aglet creates its identical copy called as clone. Its clone gets the different identifier.
- **Dispatching**: During this phase aglet changes its context from source to destination. After reaching in destination context, it is restarted its execution.
- **Retraction**: in this phase aglet will remove from the current context and inserted into requested context.
- **Activation and deactivation**: the ability of an aglet to halt its execution and store its state is called as deactivation. To restore its execution is called as activation of an aglet.
- **Disposal**: When aglet current execution will halt and remove it from its current context then an aglet is said to be in disposal state.

During the execution of migrants, they will undergo in any of the phases. The cloning ability of aglet helps in creating migrants dynamically. As per load increases or requirement arises, migrants will be available always. Similarly, after Migrants work finishes, it can dispose also. When any high priority migrant comes, current executed migrants can be halt and resume later.

Along with mobile code, aglet also carries itinerary along with it. Itinerary is a travel plan of migrants. With the help of this itinerary aglets can roam the World Wide Web easily. The planning of itinerary can be done on various criterions like list of sites to be visited in some order, what are the actions done by Migrant, source & destination addresses, etc. In this proposed work, the itinerary has following attributes:-

(i) Sender & Receiver Address

(ii) URL to be fetch

(iii) Authenticity of Sender

After reaching at server side, Migrant's itinerary is first explored and then allowed to access the database.

4.3.3 URL Scheduling Module

The URLs stored in Unique URL Buffer are required to assign to appropriate migrant. With the help of Migrant communication module, migrating crawler manager gets the information such as agent load, network latency, network load, URL capacity etc about every migrating crawler and URLs. On the basis of this information, the appropriate migrating crawler is selected. An URL scheduling module is designed in order to get an optimal migrant for each URL [93, 103].
The algorithm for URL scheduling is given in figure 4.6.

```
URL_Scheduling ()
Do Forever
Steps 1: Wait (URLs_ ready)
      2: Pick the URL from Unique URL buffer
      3: Ask Migrants information from Migrating Crawler manager.
      4: Find an optimal migrant for picked URL.
        4.1: Collect all the criterions for selecting Migrants
        4.2: Construct the structural model
        4.3: Compute comparison matrix of collected criterion
              4.3.1: Compare Pair-wise criteria using Saaty scale (1980).
              4.3.2: Construct Normalized pair-wise matrix and calculate $W_n$ as weight
              of matrix
        4.4: Calculate importance of each criterion with respect to available alternatives
        and also weight $W_i$ for each matrix for each criterion.
        4.5 Overall Score= $W_n x W_i$
      5: Assign the URL to the migrant whose Overall score is highest. (i.e. Rank 1)
     6: Repeat the steps 1-4 till the Unique URL buffer is empty.
     7: Signal (migrant_ selected)
     8: End
```

Figure 4.6: Algorithm: URL Scheduling Module

For finding the appropriate migrant for picked URL, the following are the criterion taken to design module and are discussed below:-

- Agent Load: the number of the task presently executing at agent side.
- Agent Capacity: the size of hard disk available at agent side
- Network Latency: amount of time required to transfer a data from source to destination

- Network Bandwidth: Data rate supported by network connection
- Loading rate: ratio of the number of crawl tasks to memory capacity
- URL Capacity: it is measured as in terms of number of forward links
- CPU: it is expressed in terms of frequency
- URL Parent Rank: the rank of parent URL from which it is linked.

To take a decision with these criterions, the decision-making method called as Analytic Hierarchy Process (AHP) is used. The AHP methodology, in brief, is given below:-

1. First, develop a structural model of the problem. It consists of three layers namely goal, criterion, and alternatives
2. Next, comparison matrix is prepared to find out how relative important of each criterion in achieving the goal.
3. Then, find out priority of each alternative in terms of their contribution to each criterion
4. Then, check the consistency of the given information on the relative importance of each criterion.
5. At the end, Rank all the alternatives and to find the best alternative based on the above steps.

Illustration

The Proposed decision-making procedure using AHP is given below:-

Step 1: Structural Model Design

In this step, the structural model is designed in which selecting a migrant as a goal at the first layer, characteristics of the migrant as criterions at middle layer and migrating crawlers as alternatives at the bottom layer as shown in figure 4.7.

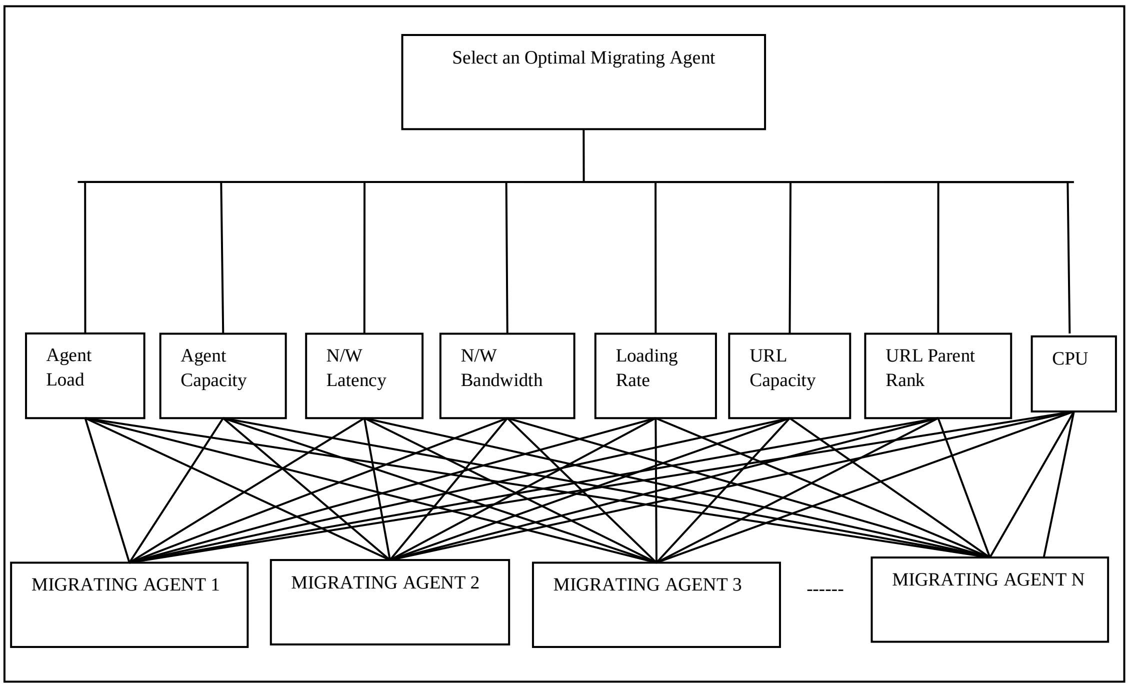


Figure 4.7: Structural Model

Step 2: Comparison Matrix

In this step, the relative importance of each characteristic of migrants is calculated. These characteristics can be obtained from migrating crawler manager as it has all the information about each migrant. A matrix is formed in which relative importance of criteria are calculated by making pair-wise comparisons in following two steps:-

Step 2.1 *Assigning the relative importance to criterion*

To develop the comparison matrix, the relative importance among criterion is required to be computed. It is difficult to measure the criteria of the migrant directly as each criteria has different measuring unit. To overcome this issue, nine-point given by Saaty [60] was used as shown in Table 4.1. A small workshop of experts in field of Search Engines was conducted to assign relative importance to criterion using the nine point scale. The Expert as per their opinion have assigned the weight to each criterion with respect to other criteria such as an expert finds the URL capacity is 8 times better than the Agent Load and most of other also agreed on same. This is shown in Table 4.2.

Table 4.1: Judgement Scale

Numerical Rating	Verbal Judgement
9	Extremely preferred
8	Very strongly to extremely preferred
7	Very strongly preferred
6	Strongly to very strongly preferred
5	Strongly preferred
4	Moderately to strongly preferred
3	Moderately preferred
2	Equally to moderately preferred
1	Equally preferred

Step 2.2 *Weight calculations of each Criteria*

Weight of each criterion is calculated by adding the values in each column, dividing each element by its column total and by taking the average of elements in each row. This calculated weight is considered as the relative importance of each criterion with respect to other. The highest value of weight is considered as most important criteria than others.

Example:-

In given table 2, criterion importance with each other is expressed such as URL capacity is 7 times more important than agent load, Agent load is 2 times more important than N/W bandwidth and so on. The weight of each criterion is calculated in following steps:-

(i) First, sum the values in each column

(ii) Then, divide each element by its column total.

(iii) Then, take the average of elements in each row.

For instance, the weight of the URL Parent Rank is calculated:-

(i) Total= 1/8+1/2+1+1/7+1/2+1/3+1/4=2.851

(ii) Divides each row element by its column total.

(iii) Take an average= (8/21.833+2/ 4.176+1/2.851+7/18+4/10.583+5/13.5+4/18)/7

This comes out to .373. Similarly, all weights are calculated as shown in table 4.2.

Table 4.2: Criterion Comparison

Attributes	**Agent Load**	**URL Capacity**	**URL Parent Rank**	**N/W Bandwidth**	**Loading Rate**	**Agent Capacity**	**N/W Latency**	**Weights**
Agent Load	1	1/7	1/8	2	¼	1	3	0.0735
URL Capacity	7	1	½	5	4	3	4	0.273
URL Parent Rank	8	2	1	7	4	5	4	0.373
N/W Bandwidth	½	1/5	1/7	1	1	1	1	0.055
Loading Rate	4	¼	½	1	1	2	3	0.116
Agent Capacity	1	1/3	1/3	1	½	1	2	0.061
N/W Latency	1/3	¼	¼	1	1/3	1/2	1	0.046
Total	21.833	4.176	2.851	18	10.583	13.5	18	

Step 3: Priority computation for each alternative

Next, priority is calculated of each alternative with respect to the every criterion. For this, alternatives are compared pair-wise and find out how well each alternative serves each criterion. The priority is calculated in terms of weight by comparing every migrant with respect to all criterions. Suppose there are N numbers of migrant alternatives, M are the number of criterions then M number of NxN matrices will be constructed. The matrix formed with the help of data of each migrant and URLs like its load, bandwidth, URL capacity etc. In given matrix, the number shows how much these criteria are important to each other like MC1 is 4 times more better criterion than MC2 in terms of agent load. The weight is calculated by the same step as done in step 2.

For example, for MC1, first, sum its column elements (1+1/4+1/5+1/6=1.62), and then divide the elements with the sum. Then, take an average of its row elements [(1/1.62+4/5.66+5/9.5+6/12)/4]. This comes out to be .597. Similarly, all weights are calculated with respect to every criterion as shown in table 4.3 to table 4.9.

Table 4.3: Migrants Comparison w.r.t Agent Load

Agent Load	**MC1**	**MC2**	**MC3**	**MC4**	**Weights**
MC1	1	4	5	6	0.597
MC2	1/4	1	3	3	0.222
MC3	1/5	1/3	1	2	0.108
MC4	1/6	1/3	½	1	0.073
Total	1.62	5.66	9.5	12	

Table 4.4: Migrants Comparison w.r.t URL Capacity

URL Capacity	**MC1**	**MC2**	**MC3**	**MC4**	**Weights**
MC1	1	1/6	3	½	.119
MC2	6	1	7	3	.580
MC3	1/3	1/7	1	1/7	.500
MC4	2	1/3	7	1	.250
Total	8.33	1.64	17	4.64	

Table 4.5: Migrants Comparison w.r.t URL parent Rank

URL parent Rank	**MC1**	**MC2**	**MC3**	**MC4**	**Weights**
MC1	1	1/3	1/7	½	.075
MC2	3	1	1/3	1	.191
MC3	7	3	1	4	.575
MC4	2	1	¼	1	.160
Total	13	5.33	1.72	6.5	

Table 4.6: Migrants Comparison w.r.t N/W Bandwidth

N/W Bandwidth	**MC1**	**MC2**	**MC3**	**MC4**	**Weights**
MC1	1	3	4	6	.547
MC2	1/3	1	2	3	.224
MC3	¼	½	1	4	.163
MC4	1/6	1/3	¼	1	.066
Total	1.75	4.83	7.25	14	

Table 4.7: Migrants Comparison w.r.t Loading Rate

Loading Rate	**MC1**	**MC2**	**MC3**	**MC4**	**Weights**
MC1	1	1/7	¼	1/3	.060
MC2	7	1	3	4	.551
MC3	4	1/3	1	3	.260
MC4	3	¼	1/3	1	.129
Total	15	1.72	4.58	8.33	

Table 4.8: Migrants Comparison w.r.t Agent Capacity

Agent Capacity	**MC1**	**MC2**	**MC3**	**MC4**	**Weights**
MC1	1	¼	¼	1	.094
MC2	4	1	1/4	3	.253
MC3	4	4	1	4	.555
MC4	1	1/3	¼	1	.099
Total	10	5.58	1.75	9	

Table 4.9: Migrants Comparison w.r.t N/W Latency

N/W Latency	**MC1**	**MC2**	**MC3**	**MC4**	**Weights**
MC1	1	2	3	1	.362
MC2	½	1	3	1	.255

N/W Latency	**MC1**	**MC2**	**MC3**	**MC4**	**Weights**
MC3	1/3	1/3	1	½	.111
MC4	1	1	2	1	.272
Total	2.83	4.33	9	3.5	

Step 4: Checking consistency using consistency ratio:

After calculating weights, consistency of the given information is checked corresponding to relative importance between criterion and also for available migrant alternatives using the values of Consistency Ratios. Consistency Ratio (CR) measures how consistent the judgments have been relative to large samples of purely random judgments. Consistency Ratio explained by Saaty [] as the ratio of consistency index (CI) and random consistency index (RI):-

CR= CI/RI

CI is consistency index and calculated by using formula given in equation 4.1.

$$\text{Consistency index } (\mathrm{CI}) = \frac{(\lambda\max - n)}{n-1} \qquad (4.1)$$

Where, λ max is summation of each weight and sum of columns of comparison matrix with respect to other criterion and n is number of criterion being compared (i.e. size of pair-wise comparison matrix).

As per Saaty [60], Random index (RI) is the consistency index of a randomly generated pair-wise comparison matrix. RI depends on the number of elements being compared (i.e. size of pair-wise comparison matrix) and takes on the following values as shown in table 4.10.

Table 4.10: Random Index

N	1	2	3	4	5	6	7	8	9	10
RI	0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49

If the value of consistency ratio is smaller or equal to 10%, the inconsistency is acceptable. For consistency ratio higher than 10% (i.e. 0.1), the judgment needs to be revised.

For instance, the CR of Table 4.3 is calculated by taking ratio of CI and RI. First, CI is calculated as per equation 4.2.

$$CI = \frac{(\lambda max - n)}{n - 1} \quad (4.2)$$

Where, λmax= (1.62*.597) + (5.667*.222) + (9.5*.108) + (12*.073) =4.12714 and

n=4

So, CI= 4.12714-4/3= .04238

And CR= .04238/0.90= .047

Similarly, CR is calculated for table 4.4 to table 4.9. The obtained values are given below:-

Table 4.4= .005

Table 4.5= .008

Table 4.6= .052

Table 4.7= .044

Table 4.8= .076

Table 4.9= .003

From the above-computed consistency values, it is observed that all are below 0.1 (Saaty, 1980). It means all the matrices are consistent and the information it contains is acceptable.

Step 5: Assigning Rank to various alternatives:

After checking the consistency of available information, the overall priority of each alternative i.e. migrant is calculated as shown in table 4.11. Then on the basis of these priorities rank will be calculated.

Table 4.11: Overall Priority Matrix

Criterions Weight	**0.073**	**0.273**	**0.373**	**0.055**	**0.116**	**0.061**	**0.046**	
Criterions / Alternatives	**Agent Load**	**URL Capacity**	**URL Parent Rank**	**N/W Bandwidth**	**Loading Rate**	**Agent Capacity**	**N/W Latency**	**Priority**
MC1	0.597	.119	.075	.547	.060	.094	.362	.164
MC2	0.222	.580	.191	.224	.551	.253	.255	0.350
MC3	0.108	.500	.575	.163	.260	.555	.111	0.315
MC4	0.073	.250	.160	.066	.129	.099	.272	0.171

Priority of MC1=

.073*.597+.273*.119+.373*.075+.055*.547+.116*.060+.061*.094+.046*.362=.164

Similarly, Priority for MC2, MC3 and MC4 are calculated as shown in table 4.11.

Table 4.12 represents ranking of priorities where highest priority has been assigned Rank 1 and the lowest one is assigned the last rank.

Table 4.12: Rank Matrix

Alternatives	**Priorities**	**Rank**
MC1	.164	4
MC2	0.350	1
MC3	0.315	2
MC4	0.171	3

This URL scheduling module waits on URL_ ready signal and then picks the URL to schedule it to appropriate migrant. After finding an optimal migrant for URL, it sends a signal to migrating crawler manager to inform that migrant is selected for crawling.

4.3.4 Migrating Crawler Manager

It is the main module responsible for downloading of URLs. It creates its multiple Migrants and distributed over the network. It waits for the signal called as migrant selected from the URL scheduling module. It assigns the URL to corresponding selected migrant. These Migrants work on behalf of migrating crawler and do the whole process of crawling. It also provides Migrants information to the URL scheduling module for scheduling the URL to the appropriate migrant.

4.4 EXPERIMENTAL EVALUATION OF THE PROPOSED SYSTEM

The performance of proposed migrating crawler was compared with a Conventional migrating crawler. In Conventional Crawling, load was unevenly distributed which takes more crawling time and misutilization of resources by leaving many machines unutilized while others are heavily loaded. Whereas proposed migrating crawler generates required number of migrants which crawl the web on behalf of a migrating crawler. With the help of URL scheduling module, load distribution was almost balanced, crawling time reduces and proper utilization of resource. The purpose of the migrant is to crawl the web as maximum as possible in efficient manner.

4.4.1 Procedure

The experiment conducted on 4 systems of different configurations in computer lab of YMCA University of Sciences and Technology. Migrating Crawler manager was running on one machine and rest of the machines are having instances of migrating crawler called as migrants. Migrating Crawler manager is responsible for distributing the crawling work on different machines and which is handled by migrants. The efficiency of proposed migrating crawler can be calculated in terms of number of URLs crawled in particular time. It also utilized network resources by selecting appropriate migrating crawler at an appropriate time. It also balances the load effectively by distributing load proportionally to the systems available.

The summary results obtained in table 4.13 support the results that proposed migrating crawler crawled more links in same time as compared to conventional one.

Table 4.13: Experiment Results

Parameters	**Proposed Migrating Crawler**	**Conventional Migrating Crawler**
Total Links Crawled	4169	3735
Total Links Saved	4035	3692
Time taken (in secs)	14400	14400
Duplicate URLs Identified	134	43

4.4.2 Load Distribution

The proposed migrating crawler performs the function of balancing loads across different migrants. In Conventional crawler, it has been observed that URLs on machines were not equally distributed and the variation across machines was huge. But the proposed URL scheduling module distributed the URLs almost equally on all the machines. The result comparison of load distribution using conventional and proposed migrating crawler is depicted in figure 4.8.

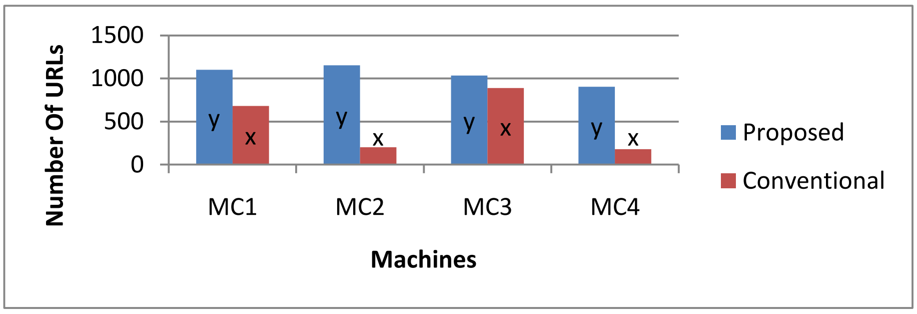


Figure 4.8: Load Distribution of Proposed Migrating Crawler

In above graph all x represents number of URLs crawled by conventional whereas all y represents number of URLs crawled by proposed one. It is observed from the above graph that variation in load distribution is almost uniform in case of proposed crawling method in comparison with conventional crawling method.

4.4.3 Efficiency

The efficiency can be calculated in terms of how many URLs were crawled in particular time. In given example, the proposed migrating crawler crawled more links as compared to Conventional migrating crawler in same time. The graph shows the efficiency of proposed as compared to Conventional one in figure 4.9

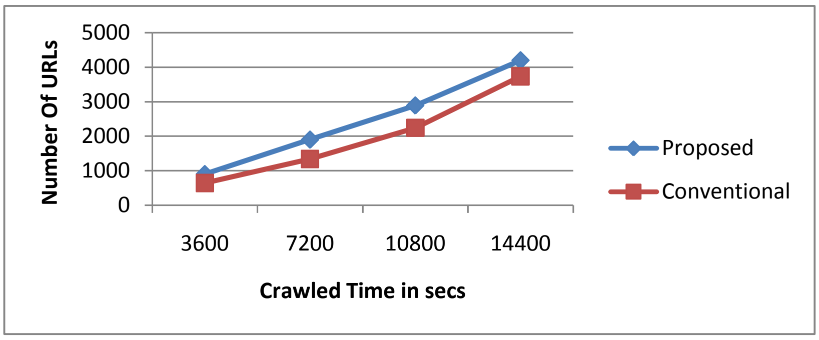


Figure 4.9: Efficiency of Proposed Migrating Crawler

From the above given figure 4.9, it may observed that proposed Migrating crawler crawls more links i.e.4169 than Conventional i.e. 1387 in same time i.e. 14400 secs.

4.4.4 Uniqueness

In this work, Standard Normalization techniques are applied on set of URLs to identified duplicate URLs. These steps works on syntax of URLs and after normalization URLs whose syntax are same are identified. This will make migrants to crawl unique set of Urls and thus preserves time and network resources.

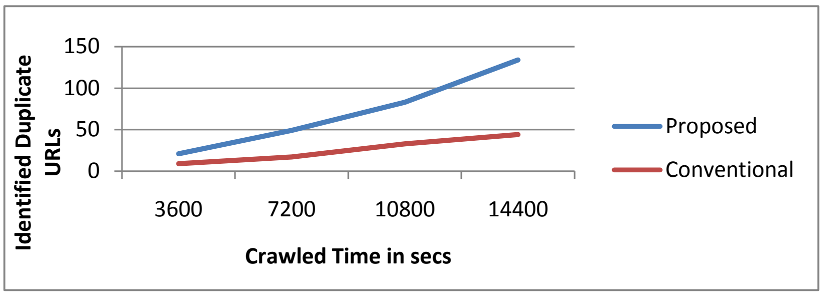


Figure 4.10: Uniqueness of Proposed Migrating Crawler

From the above figure 4.10, it is observed that proposed migrating crawler identified more duplicate URLs whereas Conventional crawler identified less in same time.

Different migrants before downloading the web pages passed them to Duplicate Eliminator Module. This module is responsible for preventing duplicate web pages to be stored in a database. This module checks the duplicacy both at web page and at URLs. The links extracted from the downloaded web page checked for duplicacy with already stored URL list before added to URL Queue for crawling. The detailed explanation of this module along with the architecture is explained in next chapter with implementation results.

CHAPTER V
DESIGN OF AN EFFICIENT MIGRATING CRAWLER BASED ON SITEMAPS

5.1 INTRODUCTION

Search Engines are the most common and widely used medium of finding information on the web. The user enters a keyword for searching the information and on the basis of that keyword search engine searches their databases and give the results related to users' query. These databases are created from a repository maintained by web crawlers. Web crawler crawls the web, downloads the documents and stored them in a search engines repository. They continuously crawl the web to get new and more relevant information. So, the web crawler is an important module of any search engines. There are many issues [63] related to design an efficient web crawler. For example, User interest [50, 51, 62] can also be considered while designing a crawler. In this chapter, more emphasis is on maximum coverage and freshness of database while keeping the network traffic low. As the size of the web grows exponentially, it is very difficult to crawl to the whole web and maintained the freshness of the search engines repository. Even with the presence of massive resources, the present crawlers are not able to do their task efficiently.

Sitemaps may be utilized to discover all the links present on a particular web page. It is an XML file that lists all the links of a web page and also the other information about that web page, e.g. when the page was last modified, how frequently the web page will change and how much the importance of any link in comparison to other links present on that web page. Sitemaps also help in extracting structure of a web page and then this extracted structure can be used for many purposes [61].

Although without sitemap crawler may discover most of the links but with a sitemap it will do the task more efficiently. Following are the reasons for this:

1. The Large size of Web Site: - it may be possible that due to the large size of the website, some links may be left can explore by a crawler while downloading the web pages.

2. New Web Site: - The web crawler always follows the same pattern of crawling and due to this it may miss the new entries.
3. Less External Links: - some websites have fewer links to other websites and crawler crawls to the web by following one page to another. So having less number of links causes crawler to rarely visit that particular site.

From above mentioned reasons, it is justified that with the help of sitemap crawler is work more efficiently while downloading the web pages.

Here an example of Gmail is taken where sitemap of Gmail technical support is designed with the help of the website www.web-site-map.com. A Sitemap protocol for Gmail technical support is given below:

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xsi:schemaLocation="http://www.sitemaps.org/schemas/sitemap/0.9
http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
<url>
        <loc>http://www.gmailtechnicalsupport.com/</loc>
        <changefreq>daily</changefreq>
        <priority>1.00</priority>
</url>
<url>
        <loc>http://www.gmailtechnicalsupport.com/services</loc>
        <changefreq>daily</changefreq>
        <priority>0.85</priority>
</url>
<url>
        <loc>http://www.gmailtechnicalsupport.com/privacy-policy</loc>
        <changefreq>daily</changefreq>
        <priority>0.85</priority>
</url>
<url>
        <loc>http://www.gmailtechnicalsupport.com/contacts</loc>
        <changefreq>daily</changefreq>
        <priority>0.85</priority>
</url>
<!-- Generated by www.web-site-map.com -->
</urlset>
```

The sitemap may consist of some essential fields and some optional fields as discussed below: -

Loc- it specifies the URL of particular page.

Lastmod: it is an optional field, which specifies when the page was last updated

Changefreq: it is an optional field, which specifies the frequency of web page changed like always, hourly, daily, weekly, monthly, never.

Priority: it is also an optional field, which specifies the priority in comparison with other URLs present in the page.

Other than above mentioned parameters various other parameters can be added to sitemap e.g. information about images, sitemap for mobile, etc. For large websites whose sitemaps are very large in size are difficult to download. So instead of downloading large size sitemap, SitemapIndex files are used. This allows big sitemaps breaking into smaller sitemaps and keeps their entries in SitemapIndex file. The size of these SitemapIndex files is small and they are easy to download and manage.

5.2 DESIGN OF AN EFFICIENT MIGRATING CRAWLER BASED ON SITEMAPS

With the help of sitemaps, crawler tries to crawl the web more efficiently so that updated information always gets stored in the database while keeping the network load low. The Sitemaps are used to provide checks at various levels before downloading the page. These checks are done by web crawlers. The fields which are used in this work are *changefrequency* and *lastmod*.

5.2.1 The Architecture

A migrating Crawler is a crawler which has its agents called as Migrants [6]. Crawler sends these Migrants to different servers and on the server side, Migrants do the processing e.g. downloading, compressing etc at server side only and return the results back to the crawler side.

The proposed architecture has following subsystems:

1. **Mapping Manager**

   It maps URLs into their IP addresses and also stores sitemap of the corresponding URL with the help of Sitemap Generator in resolving URL-IP-Sitemap Queue.

2. **URL Classifier**

   It will classify the URLs on the basis of their changed frequency like hourly, daily, weekly, monthly etc. And then inform the crawl manager about their classification.

3. **Migrating Crawl Manager**

   It will send the migrants to their assigned URL server as provided by URL Classifier. After calculating their revisit, documents are downloaded.

4. **Document & URL Buffer**

   This is a buffer for storing documents send by different Migrants from different servers. From this buffer, URLs are extracted and send back to Mapping Manager for further processing.

Here in proposed architecture, by using the sitemap Migrants will work in different ways and crawls the web efficiently. The workflow of proposed architecture is given in figure 5.1.

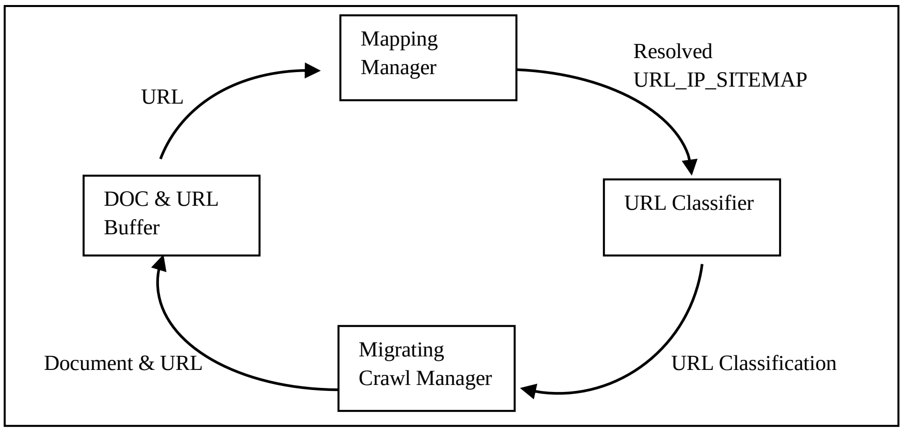


Figure 5.1: Workflow of proposed Architecture

The working of each involved module is described in the next section.

The working of Proposed Migrating Crawler is as follows: - URL_IP_sitemap values generated by Mapping Manager are used by URL Classifier for classifying URLs on the basis of their change frequency. It may change daily, hourly, weekly, monthly or never. Migrants are assigned to each such classified list and visit the URLs according to their change frequency. On reaching the websites, whether to download the pages or not will depend on whether it has updated or obsolete copy. If the Migrant is visiting the first time, then Migrant doesn't have any last crawled time of the web page and it simply downloads the page. But if the Migrant is revisiting it, then it will check the last crawled time of web page with the last mod values of the web page which it has got from the sitemap. If last crawled value is greater than last mod value, it will not download the page as Migrant already has its updated copy with it. But if the value of last crawled is lesser than last mod value, it shows there are some modifications have been done and the Migrant has an old copy with it. So it will download the page and replace its copy with the new updated one.

It consists of following functional components and detail of each component is given in next section:-

1. Mapping Manager
2. URL Classifier
3. Migrating Crawler Manager
4. Migrants
5. Duplicate Eliminator

The general architecture of the proposed work is as follows in figure 5.2.

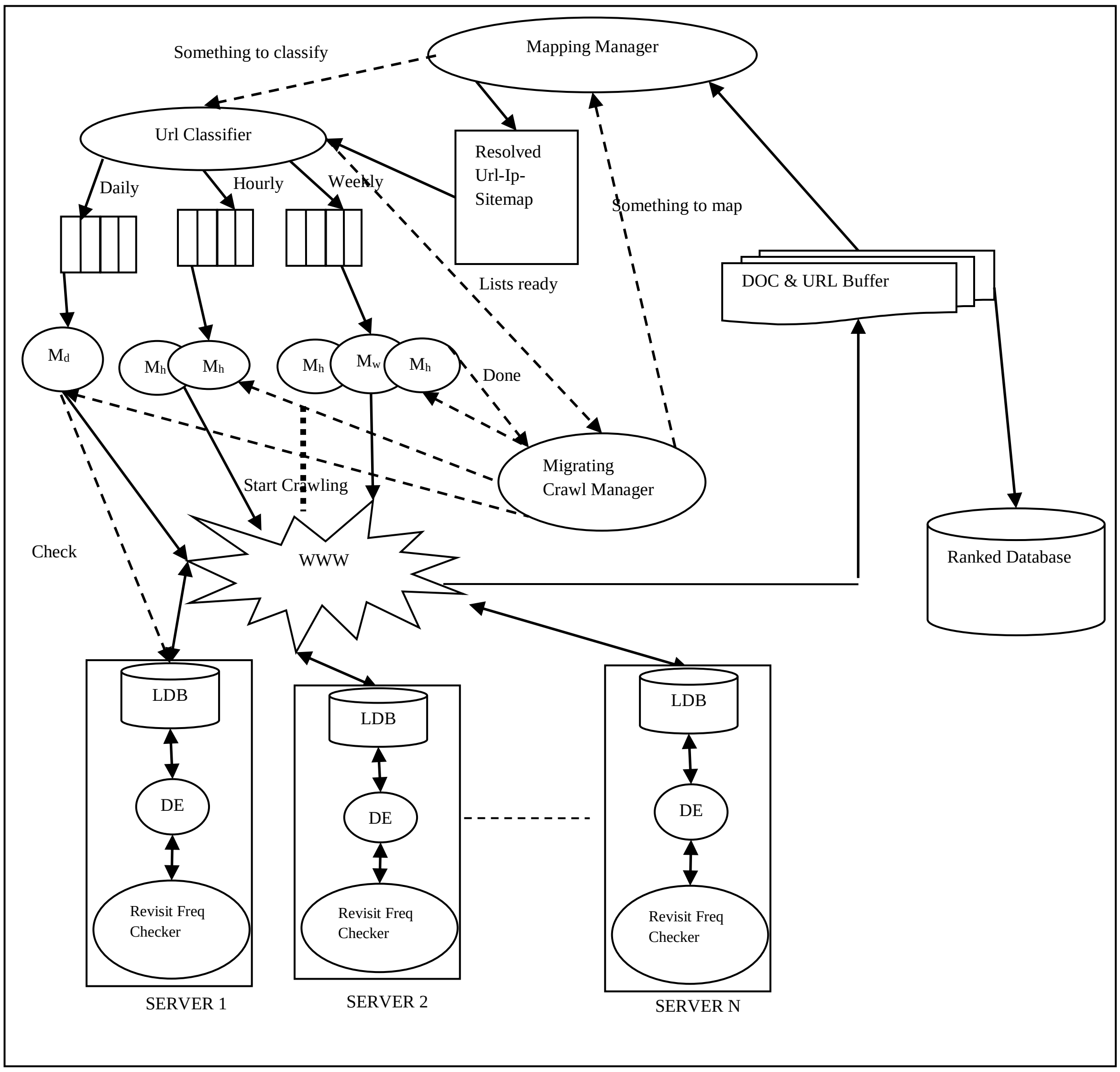


Figure 5.2 Proposed Architecture of Migrating Crawler with Sitemaps

The general algorithm of the system is given in figure 5.3.

```
Step1:URL_List_Sitemap= Mapping Manager ()
        Do Forever
      2: Pick URLs from URL_List_Sitemap.
      3: URLi=URL_classifier(s(i), URL_List).
         //URLi is the list of URL of i category
         Where i= daily, weekly, monthly
      4: Signal (agents_ready).
      5: LDB= Migrating_Crawl_manager(URLi).
         5.1: Doc_Buffer= DE(URLi).
      6: (URL,Doc)= Extract(doc_buffer).
      7: Take URLs and give it to URL_classifier.
       8: End
```

Figure 5.3: Algorithm: General Working

5.2.2 Mapping Manager

Mapping manager [64] provides resolved URL_IP pair. It gets an IP for the corresponding URL from DNS resolver and stored the pair in a Queue. In addition to this pair with the help of sitemap generator [65], a sitemap of every URL is also provided and stored with the same resolved URL_IP pair. For incorporating the above said Sitemap field, the structure of URL_IP pair has been modified and shown below in figure 5.4:-

URL	IP	SITEMAP

Figure 5.4: Modified URL_IP

After filling modified structure in the queue, mapping manager sends the signal to a URL classifier to start their work.

The Mapping Manager uses following data structures:

- **URL-IP Queue:** It consists of a queue of unique seed URL-IP pairs. The IP part may or may not be blank. It acts as an input to the mapping manager.
- **Resolved URL-IP-Sitemap Buffer:** It stores resolved URLs and also their corresponding sitemap. Sitemaps are generated with the help of Sitemap Generator. It acts as input to the URL Classifier.

The algorithm for Mapping Manager is given in figure 5.5.

```
Mapping Manager ()
Step1: Wait (Something to map)
      2: While (URL-IP Queue is not empty)
      3: Take a URL-IP pair from the Queue.
      4: If the IP is blank
          4.1 Call DNS resolver to resolve URL for IP.
          4.2 Store the Resolved URL in the Resolved URL Queue.
      5:  Call SiteMap Generator to create sitemap of every resolved
          URL.
      6:  Store the sitemap with each URL_IP pair.
      7: Signal (something to classify).
      8: End.
```

Figure 5.5: Algorithm: Mapping Manager

5.2.3 URL Classifier

After getting the signal from Mapping Manager, it picks the URLs and classifies them on the basis of their change frequency received from the sitemapS[i]. According to its change frequency whether it is daily, weekly, monthly etc changes, lists are maintained and corresponding URLs are added to them.

The algorithm for URL Classifier is given in figure 5.6.

```
URL_classifier()
Do forever
Step1: Wait (Something to classify)
        2: Pick a URL from URL_IP_SITEMAP queue.
        3: Check respective S(i) for frequency change
        3: If (changefreq==daily)
            3.1:  Add to Ld.
        4: If (changefreq==weekly)
            4.1: Add to Lw.
        5: If (changefreq==monthly)
            5.1: Add to Lm.
       6: Signal (list_ready)
       7: End.
Where,
 Ld: list of URLs changes daily
Lw: list of URLs change weekly
Lm: list of URLs changes monthly
```

Figure 5.6: Algorithm: URL Classifier

After maintaining the lists, it will send a signal to crawl manager to inform that lists are ready for Migrants to crawling.

5.2.4 Migrating Crawler Manager

It is the responsibility of the crawler manager to create multiple Migrants. After getting the signal (URLs_ready) from URL classifier, it will assign the Migrants to each list of URLs supplied by URL classifier. Then it will send a signal(crawl) to Migrants to start crawling to the WWW. After sending URLs to migrants. It will wait for downloading to be done so that new list will be provided to migrants and the process continues.

5.2.5 Migrant

Migrants are waiting for the signal (start crawling) from their crawl manager to start their work. They get their list of URLs and now they start downloading the data corresponding to each URL. After downloading the webpages, they send signal (check) to duplicate eliminator for checking duplicity and signal (done) to migrating crawler manager for more URLs to be crawled.

The Migrants during their working use the following data structures:

5.2.5.1 Local Buffer: It is a buffer used by the migrants for keeping the downloaded documents temporarily. Before storing the documents, Duplicate Checker Module checks whether the downloaded document is unique or the duplicate one.

5.2.5.2 Document and URL Buffer: This buffer is used to store the recently downloaded documents sent by migrants. From this buffer, URLs are sent back to the URL Classifier for classification.

5.2.6 Duplicate Eliminator (DE)

This module gets the web page from local buffer downloaded by Migrants. It is responsible for preventing duplicate web pages to be stored in a database [76, 94]. Figure 5.7 shows the component of duplicate eliminator module:

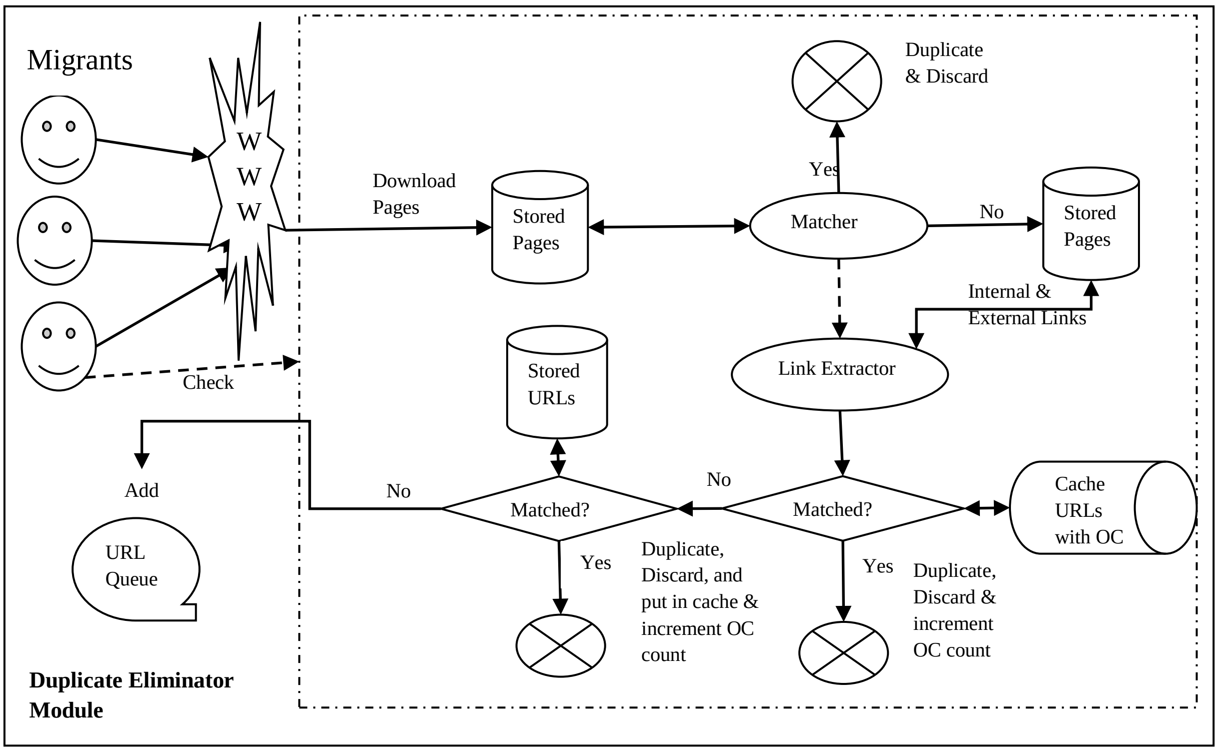


Figure 5.7: Duplicate Eliminator Module

Before storing the documents in the database, it sends the downloaded web page to the matcher module. After passing from matcher if the web page is not found matched, then stored in the *stored pages* database and passed to link extractor for extracting the links.

These extracted links are then matched with already stored URLs. First, the matching is done with URLs which are stored in cache memory. If found, check their lastmod value with the help of Sitemap. If found updated, the link/URL is added into the URL Queue for revisiting otherwise discarded as duplicate and its occurrence count (OC) is incremented by 1. Occurrence Count is the field that has integer value and used for maintaining Cache entries which are discussed later. If URL is not found in the cache, then it is searched in the database stored in secondary memory. If found, its change frequency and modification values is checked and it is added into the URL Queue for revisiting otherwise discarded as duplicate and its OC count is incremented by 1. If not found in secondary storage, it is added into the stored URLs database and URL Queue as a new entry.

The Duplicate Eliminator Module has the following components:-

1. Matcher
2. Link Extractor

1. Matcher

Matcher module matches the digest values of the downloaded web page with the pages available in stored page database. The digest value is calculated by applying hashing algorithm.

The algorithm for Matcher is given in figure 5.8.

```
Matcher ()
Step 1: Process Downloaded web pages and generate its SHA-1 Hash
      2: Match with existing Digest values of each web page available in stored page
      database.
     3: If (exists)
         3.1: Remove the web page
     4: Otherwise
         4.1: Store web page in stored pages database
         4.2: Call Link _Extractor ()
      5: End
```

Figure 5.8: Algorithm: Matcher

This matching is done by creating a SHA-1 hash of web document. The reason behind using SHA-1 hash is that it is more reliable and efficient hashing algorithm. The converted digest values are then matched with already existing digest value of web documents. These matching will filter out duplicate pages and stored only the unique web pages to the stored pages database.

2. Link Extractor

This module is responsible for checking the duplicacy at URL level. The extracted links get stored on cache and hard disk. The links which are frequently crawled are stored in cache.

The algorithm for Link extractor is given in figure 5.9.

```
Link Extractor ()
Do Forever
Step 1: Check the extracted links in stored cache URL list
        1.1: If Matched occurs
                1.1.1: Check lastmod value
                1.1.2: If Updated
                        Add to Queue
                        Else
                        Discard and increment its occurrence count by 1.
        1.2: Otherwise
                1.2.1: Check in stored URLs list
                        1.2.1.1: If found,
                                Check change frequency and last modified value
                                If so,
                                Add to Queue
                                Otherwise
                                Discard and increment its occurrence count by 1.
                        1.2.1.2: Otherwise
                                Add it to stored URLs and URL Queue
        2: End
```

Figure 5.9: Algorithm: Link Extractor

The advantage of Cache is to get the URLs frequently. It is helpful in accessing the URLs faster as compared to the URLs stored in secondary storage. Cache is maintained on the principle of Least Recent Used and implemented by having occurrence count values. The purpose of maintaining this occurrence count is to check which URL is frequently required and which is not like URL having high OC means it is used frequently whereas OC whose values is less than it is least used URL and can be removed from the cache.

5.3 PERFORMANCE ANALYSIS

While implementing the architecture of a migrating crawler the access permission from the website and the availability of the requisite environment always remain important issues; most of the present websites do not provide the access permission and suitable environment for the foreign agents/programs to run on their websites as it affects their performance. So far implementing the proposed Migrating Crawler, a virtual environment has been created where a copy of the respective target website was stored locally thereby making a virtual server for the websites. The implemented instance of migrating crawler is also provided the environment and the requisite data (sitemap) on the virtual server for the experimentation discussed in detail in the following sections.

The performance was compared with the conventional method of crawling i.e. the crawler without a sitemap. Various tests have been performed on different websites to observe the difference in crawling parameters e.g. web coverage, bandwidth preservation and time taken.

The Test1 was conducted by using Gautam Buddha University (GBU) website. The Website was crawled by both conventional and proposed new method of crawling. The results obtained are discussed in the next section. For simplicity, few links are shown here to verify the results.

The Test 1 was done on **www.gbu.ac.in/UserViewNews.aspx?NewsId=0** and results obtained are shown in table 5.1(a), 5.1(b) and 5.1(c).

Table 5.1(a): First Crawling Results (Both Crawler) Test 1

Anchor Text	**URL**	**Data Received (in Bytes)**
Home	http://www.gbu.ac.in/home.aspx	80476
ContactUs	http://www.gbu.ac.in/ReachUs.aspx	18171
University Grant Commission (UGC)	http://www.gbu.ac.in/UserViewNews.aspx?NewsId=117	46197
Ph.D. in Buddhist Studies & Civilization	http://www.gbu.ac.in/UserViewNews.aspx?NewsId=119	46454
Legal Aid Clinic	http://www.gbu.ac.in/UserViewNews.aspx?NewsId=160	45427
Academic Calendar 2016-17 Session	http://www.gbu.ac.in/UserViewNews.aspx?NewsId=298	96216
Boys/Girls Hostel allotment List for Academic Session 2016-2017	http://www.gbu.ac.in/UserViewNews.aspx?NewsId=301	49753

- The Total links crawled by Conventional crawler=723
- The Total links crawled by Proposed Crawler= 980

On revisiting proposed crawler has downloaded only the links which have undergone modifications as shown in table 5.1(b).

Table 5.1(b): Revisit Crawling Results (Proposed Crawler) Test 1

Anchor Text	**URL**	**Data Received (in Bytes)**
Boys/Girls Hostel allotment List for Academic Session 2016-2017	http://www.gbu.ac.in/UserViewNews.aspx?NewsId=301	49753
New Student Hostel allotment List for Academic Session 2016-2017	http://www.gbu.ac.in/UserViewNews.aspx?NewsId=302	46875

Whereas Conventional crawler has downloaded the all documents whether modified or not as shown in table 5.1(c).

Table 5.1(c): Revisit Crawling Results (Conventional Crawler) Test 1

Anchor Text	**URL**	**Data Received In Bytes**
Home	http://www.gbu.ac.in/home.aspx	80476
ContactUs	http://www.gbu.ac.in/ReachUs.aspx	18171
University Grant Commission (UGC)	http://www.gbu.ac.in/UserViewNews.aspx?NewsId=117	46197
Ph.D. in Buddhist Studies & Civilization	http://www.gbu.ac.in/UserViewNews.aspx?NewsId=119	46454
Legal Aid Clinic	http://www.gbu.ac.in/UserViewNews.aspx?NewsId=160	45427
Academic Calendar 2016-17 Session	http://www.gbu.ac.in/UserViewNews.aspx?NewsId=298	96216
Boys/Girls Hostel allotment List for Academic Session 2016-2017	http://www.gbu.ac.in/UserViewNews.aspx?NewsId=301	49753
New Student Hostel allotment List for Academic Session 2016-2017	http://www.gbu.ac.in/UserViewNews.aspx?NewsId=302	46875

On revisit:-

- The Total links crawled by Conventional crawler=734
- The Total links crawled by Proposed Crawler= 221

Summarized results of full website by Conventional Crawler and Proposed Crawler are shown in table 5.1 (d).

Table 5.1(d): Crawling Results Test 1

Parameters	**Conventional Crawling**	**Proposed Crawling with Sitemaps**
Number of Links	723	980
Total Data downloaded(Bytes)	5401300	7501233
Crawl Time(sec)	1051.35	890.23
Duplicate Links Identified	64	243
Total Pages Downloaded on revisit	734	221
Total Data Downloaded on revisit	5402387	897531

It is worth noting that the conventional crawling crawled less links i.e. 723 whereas proposed crawling crawled more links i.e. 980 as shown in table 5.1(d). But on revisiting, proposed crawler crawled only changed or modified links i.e.221 whereas conventional crawling crawled all unchanged and new i.e. 734 links as shown in table 5.1(d) and thus wasted network resources.

Similarly, the Test 2 was conducted by using YMCA website. Results obtained from **www.ymcaust.ac.in/computers** are shown in table 5.2 (a), 5.2 (b) and 5.2(c).

Table 5.2(a): First Crawling Results (Both Crawler) Test 2

Anchor Text	**URL**	**Data Received (in Bytes)**
Chairman message	http://www.ymcaust.ac.in/computers/index.php/chairman-s-message	13450
Faculty	http://www.ymcaust.ac.in/computers/index.php/faculty	10823

Anchor Text	**URL**	**Data Received (in Bytes)**
Labs	http://www.ymcaust.ac.in/computers/index.php/labs	14549
Courses	http://www.ymcaust.ac.in/computers/index.php/courses	15883
B.Tech Syllabus	http://www.ymcaust.ac.in/computers/index.php/b-tech-syllabus	96216
M.Tech Syllabus	http://www.ymcaust.ac.in/computers/index.php/m-tech-syllabus	11762
Updated Time Table	http://www.ymcaust.ac.in/computers/index.php/updated-time-table	11450
Notices	http://www.ymcaust.ac.in/computers/index.php/notices	11928

- The Total links crawled by Conventional crawler=141
- The Total links crawled by Proposed Crawler= 158

Whereas, on revisiting proposed crawler has downloaded the links which have undergone modifications only as shown in table 5.2(b).

Table 5.2(b): Revisit Crawling Results (Proposed Crawler) Test 2

Anchor Text	**URL**	**Data Received (in Bytes)**
Updated Time Table	http://www.ymcaust.ac.in/computers/index.php/updated-time-table	11450
Notices	http://www.ymcaust.ac.in/computers/index.php/notices	11728

And Conventional crawler has downloaded the all documents whether modified or not as shown in table 5.2(c).

Table 5.2(c): Revisit Crawling Results (Conventional Crawler) Test 2

Anchor Text	**URL**	**Data Received In Bytes**
Chairman message	http://www.ymcaust.ac.in/computers/index.php/chairman-s-message	13450
Faculty	http://www.ymcaust.ac.in/computers/index.php/faculty	10823
Labs	http://www.ymcaust.ac.in/computers/index.php/labs	14549
Courses	http://www.ymcaust.ac.in/computers/index.php/courses	15883
B.Tech Syllabus	http://www.ymcaust.ac.in/computers/index.php/b-tech-syllabus	96216
M.Tech Syllabus	http://www.ymcaust.ac.in/computers/index.php/m-tech-syllabus	11762
Updated Time Table	http://www.ymcaust.ac.in/computers/index.php/updated-time-table	11450
Notices	http://www.ymcaust.ac.in/computers/index.php/notices	11728

On revisit:-

- The Total links crawled by Conventional crawler=145
- The Total links crawled by Proposed Crawler= 40

It has been observed that in first visit of a specific link, both the crawlers have downloaded the same set of pages as shown in table 5.2(a). But on revisiting, proposed crawler crawled only changed or modified links whereas conventional crawling crawled all as shown in table 5.2(b) and 5.2(c).
Summarized results of full website by Conventional Crawler and Proposed Crawler are shown in table 5.2(d).

Table 5.2(d): Crawling Results Test 2

Parameters	**Conventional Crawling**	**Proposed Crawling with Sitemap**
Number of Links	141	158
Total Data Downloaded(Bytes)	1401300	1501233
Crawl Time(sec)	273.44	185.45
Total Pages Downloaded on revisit	145	40
Total Data Downloaded on revisit	1402387	397531

In this example also, it is observed that the conventional crawling crawled less links i.e. 141 whereas proposed crawling crawled more links i.e. 158 as shown in Table 5.2(d). But on revisiting, proposed crawler crawled only changed or modified links i.e.40 whereas conventional crawling crawled all unchanged and new i.e. 145 links as shown in table 5.2(d).

Likewise, The Test 3 was conducted on
www.ngfcet.in/computer_science_engineering and results obtained are summarized in table 5.3.

Table 5.3: Crawling Results Test 3

Parameters	**Conventional Crawling**	**Proposed Crawling with Sitemap**
Number of Links	423	503
Total Data Downloaded(Bytes)	1040104	1091000
Crawl Time(sec)	537.69	350
Total Pages Downloaded on revisit	442	154
Total Data Downloaded on revisit	1051123	303445

- The Total links crawled by Conventional crawler=423
- The Total links crawled by Proposed Crawler= 503

On revisit:-

- The Total links crawled by Conventional crawler=442
- The Total links crawled by Proposed Crawler= 154

Also, The Test 4 was conducted on **www.titsbhiwani.ac.in/departments/department-of-computer-engineering** and results obtained are summarized in table 5.4.

Table 5.4: Crawling Results Test 4

Parameters	**Conventional Crawling**	**Proposed Crawling with Sitemap**
Number of Links	174	223
Total Data Downloaded(Bytes)	1651256	1960107
Crawl Time(sec)	313.522	210
Total Pages Downloaded on revisit	180	34
Total Data Downloaded on revisit	1651598	30605

- The Total links crawled by Conventional crawler=174
- The Total links crawled by Proposed Crawler= 223

On revisit:-

- The Total links crawled by Conventional crawler=180
- The Total links crawled by Proposed Crawler= 34

The conventional method of crawling crawls to a limited number of links and it also downloads all documents, whether they have changed or not and thus, wasted network resources. On the contrary, this proposed new crawling method crawls the web with the help of sitemap. Now it will cover utmost URLs and downloads only that documents that have changed and thus utilize the network resources in an efficient manner.

5.4 EFFICIENCY WITH SITEMAP

There are many benefits of sitemap for both web users and web crawlers'. Following are the advantages while using a sitemap in the crawling process:

5.4.1 Web Coverage

By comparing the coverage area by Conventional Crawling and Proposed Crawling, it is observed that later one will have the better area of covering the web as it will visit more links. The results are shown in Table 5.5:

Table 5.5: Links Crawled

TESTS	Conventional	Proposed	Improvement%
TEST 1	723	980	](980-723)/723]*100=35.5%
TEST 2	141	158	12%
TEST 3	423	503	18%
TEST 4	174	223	28%

So, this coverage will increase if the size of website increases.

5.4.2 Bandwidth Preservation

It also preserves bandwidth by reducing the network traffic by downloading only modified pages on a revisit of a crawler. This can be seen from the above experiments where the total number of links on the first and second visit is same for conventional crawling whereas less for the new crawling method. In this simulation, links corresponding to each page has taken into the consideration for looking at the consumption of bandwidth by the conventional crawler. The improvement in bandwidth utilization is shown in table 5.6.

Table 5.6: Crawled Links on revisit

TESTS	**Conventional**	**Proposed**	**Saving%**
TEST 1	734	221	[(734-221)/734]*100=69.8%
TEST 2	145	40	72.4%
TEST 3	442	154	65.15%
TEST 4	180	34	81.1%

Thus, by getting information about the last modified date of the web page from the sitemap, the new crawling method now downloads only that pages that are modified or updated. This will help in saving bandwidth and also reduces network traffic.

5.4.3 Time Taken

The proposed crawling method crawled more links in less time as compared to the conventional crawling method. This will make proposed crawling method more efficient and time saving crawling. The time saving in all tests are shown in table 5.7.

Table 5.7: Time Saving

TESTS	**Conventional**	**Proposed**	**Saving%** **(approx)**
TEST 1	1051.35	890.23	[(1051.35-890.23)/1051.35]*100=15.3%
TEST 2	273.44	185.45	32.17%
TEST 3	537.69	350	34.9%
TEST 4	313.522	210	33.01%

The snapshots of crawling with the help of sitemap are shown in Appendix 5. After making database up to date by incorporating sitemap in crawling, database rich should be rich also i.e. contain almost all information. In maintaining richness of database, structure of web page is used which is discussed in next chapter.

CHAPTER VI

IMPROVING THE SEARCH RESULTS BASED ON USERS' BROWSING BEHAVIOUR

6.1 INTRODUCTION

With the advent increase in information over the web, people are now more interested and inclined towards the internet to get the information. Each user has its own interest and accordingly his expectations from search engine vary. Search engines use various ranking methods like HITS, PageRank etc. but these ranking methods do not consider user browsing behaviours on the web. In this work, a page rank mechanism has been proposed which considers users' browsing behaviour [73, 77, 78] to provide relevant pages from the web [102]. While browsing users perform various actions such as clicking, scrolling, opening an URL, searching text, refreshing etc. These actions can be used to perform an automatic evaluation of a web page and hence to improve search results. After storing these actions as events, Apriori algorithm has been applied to calculate the rank of a particular web page.

6.1.1 Introduction to Apriori Algorithm

Apriori algorithm [3] is an algorithm used in mining frequent itemsets for learning association rules. This algorithm is designed to operate on large databases containing transactions e.g. collection of items purchased by a customer. The whole point of an algorithm is to extract useful information from a large amount of data. This can be achieved by finding rules which satisfy both a minimum support threshold and a minimum confidence threshold.

The support and confidence can be defined as below:

- Support count of an itemset is a number of transactions that contain that itemset.
- The confidence value is the measure of certainty associated with discovered pattern.

Formally the working of Apriori algorithm can be defined by following two steps:-

i. Join Step

- Find the frequent itemsets i.e. Items whose occurrence in database are greater than or equal to the minimum support threshold
- Iteratively find frequent itemsets from 1 to k for k-itemsets.

ii. Prune Step

- The results are pruned to find the frequent itemsets.
- generate association rules from these frequent itemsets which satisfy minimum support and minimum confidence threshold.

6.2 IMPROVING THE SEARCH RESULTS BASED ON USERS' BROWSING BEHAVIOUR

Although many researchers have been worked in analyzing users' browsing behaviour pattern but relevancy is still lacking. One possible reason may be that not including every action of Users' behaviour for finding the interest. A module is proposed here that will try to give more relevant results by analyzing appropriate users' browsing behaviour. The description of
User Behaviour Analyzer module is explained in following section.

6.2.1 User Analysis Module

The general architecture of User Analysis module is shown in figure 6.1.

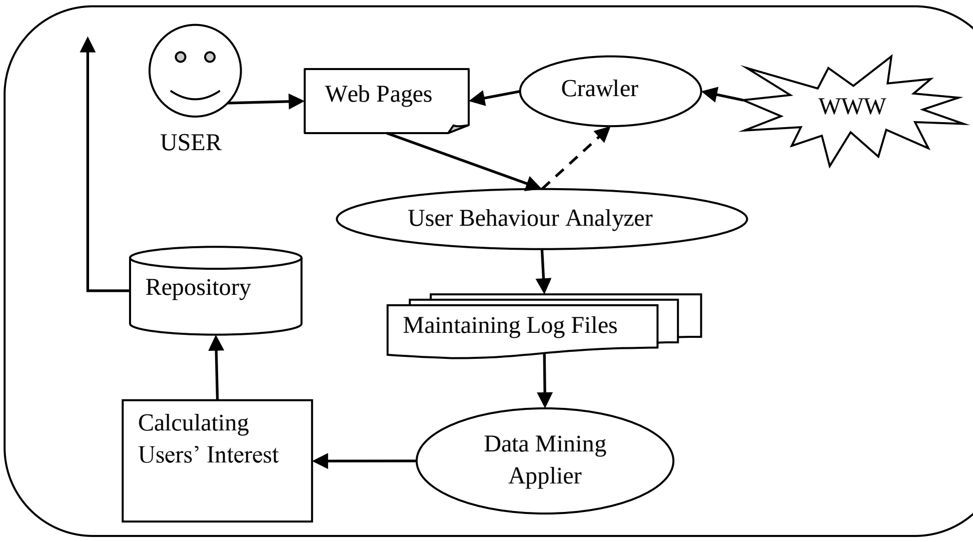


Figure 6.1 User Analysis Module

The User Analysis Module consists of following major components:

a. User Behaviour Analyzer
b. Data Mining Applier

Following are the data structures used:-

(i) Log Files
(ii) Repository

The details of components and data structures used are explained in following section.

a. User Behaviour Analyzer

From the past researches, it is concluded that browsing time play a vital role while analyzing the users' browsing behaviour. Other behaviour patterns like saving, printing, bookmarking, scrolling, copying etc also contribute to analyzing users' behaviour. But It has been observed that alone browsing time will not give the accurate idea about users' interest.

There can be several reasons of spending long time or short time on a particular web page such as:

For short time:-

- The content on the page is very less,
- The presentation of the page is not good

For long time,

- Doing another work along with browsing
- Loading of page takes time

So, along with browsing time other behaviour patterns should also consider. Moreover Kun Xing et al. [11] suggested that some actions like scrolling, mouse clicking should also be included in the total browsing time. The proposed algorithm for user behaviour Analyzer is given below in figure 6.2.

```
User behaviour analyzer ()
Step 1: Read visited pages;
     2: Analyze that page;
     3: Maintain log files of the browsing behaviours;
            3.1: How many mouse clicks?
            3.2: For how long user stay on page?
            3.3: Is user save that page?
            3.4: Is user print that page?
            3.5: Is user click on hyperlink?
            3.6: Is user click forward or backward button?
    4: Enter all these details in database;
    5: Signal (Mining);
    6: End
```

Figure 6.2 Algorithm: User Behaviour Analyzer

By considering all possibilities, browsing behaviour indicators that may relate to users' interest are listed below in table 6.1.

Table 6.1 Users' Browsing Behaviour Indicators

Pattern ID	**Actions**	**Purpose of actions**
PID1	Scroll	Moving page up & down
PID2	Copy Text	Copy page content
PID3	Search Text	Searching text in a page
PID4	Back	Go back one document
PID5	Open URL	Go to page via URL
PID6	Go Forward	Go forward one page
PID7	Stop Loading	Stop loading of page
PID8	Add to Favorite	Save URL for future use
PID9	Print	Take print out of a page
PID10	Save As	Save page to local disk
PID11	Hyperlink	Selection of hyperlink
PID12	Homepage	Go to home page
PID13	Mouse click	To read page

In above table, pattern ID is used for referencing patterns, actions indicates browsing behaviour pattern and purpose of actions describes the meaning of actions. For further analysis, some factors are taken into consideration that has a high frequency of occurrence. By combining contribution of browsing time along with above described user behaviour indicators, users' interest will be estimated more accurately.

b. Data Mining Applier

There are some indicators/behaviours that may contribute more while computing the user interest in comparison to other behaviour. These indicators are browsing time, bookmarking, printing, saving and copying.

Many Data Mining Techniques are available such as association rules, decision tree, correlations, Apriori, rule-based, FP-tree, Support Vector Machines etc [70, 72]. Any of these methods may be used to find frequent patterns. In this work, Apriori technique has been used to identify frequent item sets. By applying Apriori technique on the above stated browsing indicators, an association is generated between the indicators. Hence an order of indicators contributing in users' interest as percentage has been calculated. A threshold value has been considered to decide which association is to be considered as of interest or of non-interest as for as users' interest is concerned. The algorithm of Data Mining Analyzer is given in figure 6.3.

```
Mining Analyzer ()
Step 1: Wait (Mining)
    2: Read Log files.
    3: Apply Apriori algorithm on the browsing behaviours mentioned in log files.
          3.1 Find out frequent patterns from the noted down browsing behaviours.
          3.2 Calculate confidence level of these patterns and remove those which are
          below specified threshold value of confidence.
          3.3 Add the Confidence level values calculated in above step.
          3.4 Added values are considered as weight. Larger the weight value, higher the
          importance of web page.
    4: Add the webpages in the database along with their weights.
    5: Signal (empty)
```

Figure 6.3: Algorithm: Data Mining Applier

The Data mining Analyzer access the log files on daily basis. Every day, different set of users on different set of webpages performs different set of actions. The set of webpages are maintained with corresponding to a particular query such as for query q1, a set of 7 users were accessing a set of 20 webpages. So, on these 20 webpages their actions are observed and the frequent occurring actions are found. On the basis these actions weight is calculated for each page and ranked them accordingly. In this work, log files were scanned and top 20 queries are picked for analyzing users' browsing behaviour. On these 20 queries, Apriori was applied and accordingly to their weight, pages added to the repository.

The description of used data structures are given below:-

(i) Log Files

Log files are used to analyze users' browsing behaviour. In these files, users' identity is recorded by Web server. Its structure is shown in figure 6.4 below:-

ID No	URL	Event	Frequency

Figure 6.4: LOG File Structure

These files contain the following information:

- **The ID No** : It is identification number of each user
- **URL** : It the website address which user access
- **Events :** These are browsing indicators that user performed on web pages
- **Frequency**: This is the frequency of indicators happened.

When User performs actions/events on webpages, these actions/events get stored in log file.

(ii) Repository

The repository is the collection of webpages. After applying data mining technique on these web pages, weight is assigned and these weight values are used as rank.

6.2.2 Working Methodology

The main purpose of proposed approach is to find relevant pages by estimating user's interest implicitly. The proposed technique starts by developing a web browser to record user's actions. Actions include duration on a web page, the number of mouse clicks, the number of key ups and down, save, print, the number of scrollbar clicks, reload, save, open URL, stop loading, add to favorites, back, Forward, Copy, Search Text, Hyperlink, Active time duration etc. The proposed browser also pops up a window at the time of closing the web page that asks the user to rate that web page. Whenever the user performs above mentioned actions, their details will be stored in the database. Apriori Algorithm will then be applied on above collected data. This algorithm will result in most frequent actions and confidence values of subsets of frequent actions .The values satisfying minimum confidence threshold will be used to calculate the weight of the web page. The flow of proposed work is shown in figure 6.5.

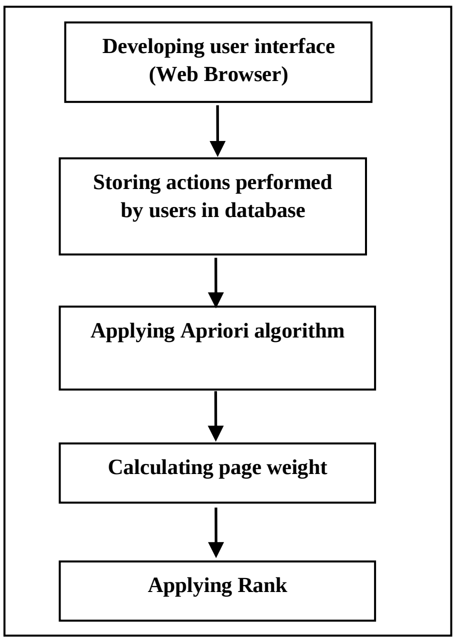

Developing user interface (Web Browser)
Storing actions performed by users in database
Applying Apriori algorithm
Calculating page weight
Applying Rank

Figure 6.5: Flow of Proposed Approach

The detailed description of each step is discussed below:-

Step 1: Developing User Interface: A web browser was proposed in the first step. The proposed web browser automatically stores various actions performed by different users. When a user opens this browser a unique id is generated and his/her actions get stored with this id.

Step 2: Storing Actions performed by the user in database: All actions performed by user get stored in the database. A user can view a summary of all actions by clicking on "*User Stats*" on the interface. "*User stats*" show the frequency of every action that is performed by different users.

Step 3: Applying Apriori: Apriori is applied on stored actions to get most frequent actions. For each page most frequent patterns are generated. These patterns are then used for calculating the Page Weight.

Step 4: Calculate Page Weight & Page Rank: After applying Apriori on actions, frequent patterns are obtained. Confidence values of each pattern are taken to calculate page weight. Confidence value can be calculated as per equation 6.1.

$$\textbf{Confidence value} = \frac{\textbf{support\_count (most frequent action)}}{\textbf{support\_count (subset of most frequent action)}} \qquad \textbf{(6.1)}$$

Where support count of an itemset is a number of transactions that contain that itemset.

Based on the confidence value of each subset the weight can be calculated by the formula given in equation 6.2.

$$\mathbf{P_{wt} = C_1 + C_2 + C_3 + \ldots\ldots\ldots\ldots + C_i = \sum C_i} \qquad \textbf{(6.2)}$$

Where C_1, C_2, C_3…. are the confidence values of subsets of frequent occurring actions which satisfy a minimum confidence threshold.

Step 5: Apply Rank: Higher the page weight, higher its rank. It means the weight of the page which is calculated from above step is considered for page rank. The page which has the highest weight is assigned higher rank.

6.2.3 Example

In this section, an example is taken to show the working of proposed work. For this, actions performed by 40 users on two pages Page1, Page2 were stored in the database. The database also stored number of times users visited those pages at different times. Users perform actions on those pages according to their needs. Their actions will be stored in a database. Apriori will be applied to those actions. For applying Apriori, minimum support of 20% and minimum confidence threshold of 60% were taken as it was used in previous researches [73]. The result of Apriori shows that most frequent actions on P1 were Save As, Add to Favorites, Number of scrollbar clicks and most frequent actions on another page P2 were Number of Mouseclicks and Print. With the help of this, the confidence values of pages were calculated.

Step 1: A Web browser is developed to store the user actions. It is developed using C# and .NET technology.

Step 2: All the actions performed by 15 users on two page P1 & P2 get stored in the database with the help of SQL. The actions on Page P1 are shown in table 6.2.

Table 6.2: Database of Actions

User ID	**List of Items**
U1	PID8, PID1,PID5
U2	PID10, PID8,PID1
U3	PID6
U4	PID7
U5	PID9
U6	PID11

User ID	List of Items
U7	PID12
U8	PID1, PID8,PID10
U9	PID4
U10	PID2
U11	PID8, PID3
U12	PID13
U13	PID8, PID10
U14	PID8
U15	PID1,PID5

Similarly, 15 users performed actions Page 2.

The actions for Page 1 and Page 2 with their support count more than 20% are shown in table 6.3.

Table 6.3: Users' Actions

Action ID	**Actions on Page 1**	**Support Count**	**Action ID**	**Actions on Page 2**	**Support Count**
PID1	Scroll	4	PID9	Print	6
PID8	Add to Favorites	6	PID13	Mouse Clicks	3
PID10	Save As	3	PID8	Add to Favorites	4

Step 3: After applying Apriori on stored actions, the frequency of actions can be calculated and then find out frequent patterns. Following are the frequent patterns with their support count as shown in table 6.4.

Table 6.4: Frequent Patterns

Frequent Pattern	**Support Count**
{Save As, Add to Favorites, Number of Scrollbar Clicks}	2
{Number of Mouseclicks, Print, Add to Favorites}	2

Step 4: The confidence value of a page is calculated by putting the values in equation (6.1) and using the values from table 6.3 and table 6.4.

First of all the confidence values of most frequent actions of page P1 has been calculated as shown below: -

C1=sc {Save As, Add to Favorites, Number of Scrollbar Clicks}/sc {Save As} =2/3=**67%**

C2= sc {Save As, Add to Favorites, Number of Scrollbar Clicks}/sc {Add to Favorites} =2/6=**33%**

C3= sc {Save As, Add to Favorites, Number of Scrollbar Clicks}/sc {Number of Scrollbar Clicks} =2/4=**50%**

C4= sc {Save As, Add to Favorites, Number of Scrollbar Clicks}/sc {Save As, Add to Favorites} =2/3=**67%**

C5= sc {Save As, Add to Favorites, Number of Scrollbar Clicks}/sc {Save as, Number of Scrollbar Clicks} =2/2=**100%**

C6= sc {Save As, Add to Favorites, Number of Scrollbar Clicks}/sc {Add to Favorites, Number of Scrollbar Clicks} =2/3=**67%**

Then, Confidence values of most frequent actions of page P2 has been calculated

C1'=sc {Number of Mouseclicks, Print, Add to Favorites}/sc {Number of mouseclicks} =2/3=**67%**

C2'=sc {Number of Mouseclicks, Print, Add to Favorites}/sc {Print} =2/6=**33%**

C3'=sc {Number of Mouseclicks, Print, Add to Favorites}/sc {Add to Favo -rites} =2/4=**50%**

C4'=sc {Number of Mouseclicks, Print, Add to Favorites}/sc {Number of Mouseclicks, Add to Favorites} =2/2=**100%**

C5'=sc {Number of Mouseclicks, Print, Add to Favorites}/sc {Number of Mouseclicks, Print} =2/4=**50%**

C6'=sc {Number of Mouseclicks, Print, Add to Favorites}/sc {Print, Add to Favorites} =2/3=**67%**

Now, Confidence values above the specified threshold value i.e. 60% were selected and others were rejected as shown in table 6.5.

Table 6.5 Confidence Values of Subsets of Most Frequent Actions

Confidence Values	Selection
C1=100%	√
C2=33%	×
C3=50%	×
C4=67%	√
C5=100%	√
C6=67%	√
C1'=67%	√
C2'=33%	×
C3'=50%	×
C4'=100%	√
C5'=50%	×
C6'=67%	√

Since minimum confidence threshold is 60%, confidence values C2, C3, C2', C3', C5' will be rejected in page weight calculation.

Therefore, weight of page P1 is calculated as per equation 6.2:-

P_{wt} = C1+C4+C5+C6=1+0.67+1+0.67=3.34

Similarly, Weight of page P2 is:-

P_{wt}' =C1'+C4'+C6' =0.67+1+0.67=2.34

Step 5: Since Page P1 has higher weight than Page P2, Rank of Page P1 will be higher.

6.2.4 Performance Evaluation

The proposed method is compared with Hit Count and User Rating method.

Page weight by Hit Count will be calculated by taking an average of the frequency of their visit on a particular page. Star rating is calculated by the user by explicitly asking while closing the web page.

Table 6.6: Comparison of Page Weight Calculation Methods

Pages	**Pwt (Apriori)**	**Pwt(Hit Count)**	**Star Rating(by User)**
Page1	3.34	1.34	3
Page2	2.6	2.2	2

By the use of above table 6.6, we can give a graphical representation in figure 6.6 which shows the comparison between page weights calculated by both Apriori and Hit Count.

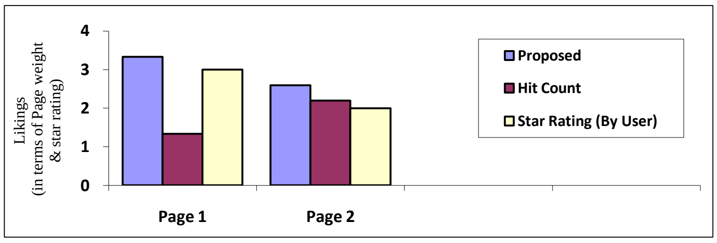


Figure 6.6: Comparison of Page Weight Calculation Methods

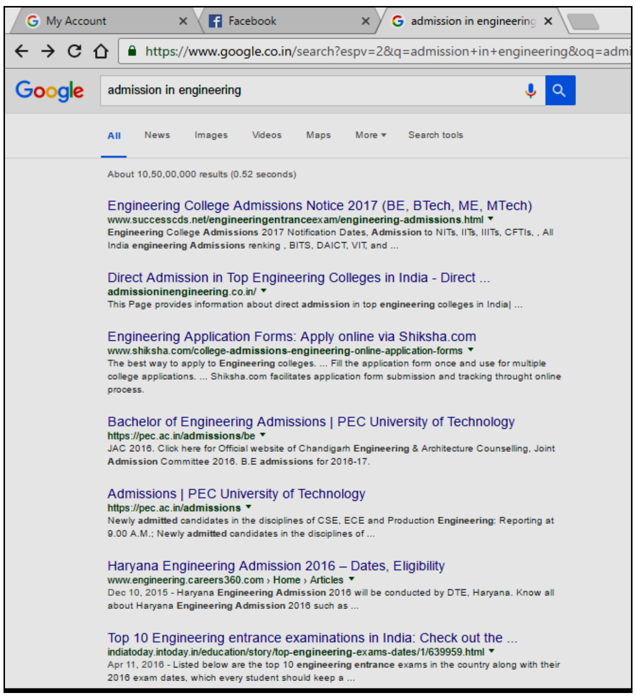
It is observed that Apriori is performing better than hit count as proposed work calculates higher relevance score. By explicitly asking users' interest also shows that proposed method predicts more accurate interest then Hit Count method.

CHAPTER VII

A NOVEL APPROACH FOR DOCUMENT STRUCTURE DRIVEN CRAWLING IN MIGRATING CRAWLER

7.1 INTRODUCTION

It has been observed that when a user fires a query, it gets the results on the basis of keywords matching i.e. the webpages as results which have these keywords are shown to the user. It may be the case that some webpages are not shown to the user as they didn't have all keywords. For example, a user searches for admission in engineering. In response to his query, search engine shows the results based on keyword matching and shows the links of colleges as shown in figure 7.1.



About 10,50,00,000 results (0.52 seconds)

Engineering College Admissions Notice 2017 (BE, BTech, ME, MTech)
www.successcds.net/engineeringentranceexam/engineering-admissions.html
Engineering College Admissions 2017 Notification Dates, Admission to NITs, IITs, IIITs, CFTIs, , All India engineering Admissions renking , BITS, DAICT, VIT, and ...

Direct Admission in Top Engineering Colleges in India - Direct ...
admissioninengineering.co.in/
This Page provides information about direct admission in top engineering colleges in India| ...

Engineering Application Forms: Apply online via Shiksha.com
www.shiksha.com/college-admissions-engineering-online-application-forms
The best way to apply to Engineering colleges. ... Fill the application form once and use for multiple college applications. ... Shiksha.com facilitates application form submission and tracking throught online process.

Bachelor of Engineering Admissions | PEC University of Technology
https://pec.ac.in/admissions/be
JAC 2016. Click here for Official website of Chandigarh Engineering & Architecture Counselling, Joint Admission Committee 2016. B.E admissions for 2016-17.

Admissions | PEC University of Technology
https://pec.ac.in/admissions
Newly admitted candidates in the disciplines of CSE, ECE and Production Engineering: Reporting at 9.00 A.M.; Newly admitted candidates in the disciplines of ...

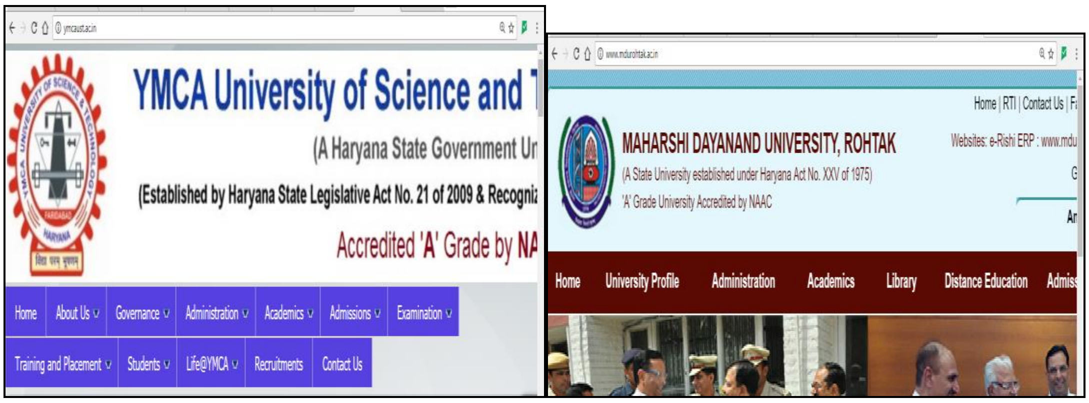
Haryana Engineering Admission 2016 – Dates, Eligibility
www.engineering.careers360.com › Home › Articles
Dec 10, 2015 - Haryana Engineering Admission 2016 will be conducted by DTE, Haryana. Know all about Haryana Engineering Admission 2016 such as ...

Top 10 Engineering entrance examinations in India: Check out the ...
indiatoday.intoday.in/education/story/top-engineering-exams-dates/1/639959.html
Apr 11, 2016 - Listed below are the top 10 engineering entrance exams in the country along with their 2016 exam dates, which every student should keep a ...

Figure 7.1: Snapshot User Query and Results

On searching Google ask for GPS location and shows the results of that particular location. In this example, results shown are nearby Haryana. As it is clearly visible from above given snapshot that User gets the link of colleges after 3 links. But it may be possible that User wants college links more and at starting positions. Moreover, results didn't show link of YMCA, MDU, GJU etc and these are the reputed universities of Haryana. So, there should be some other way of searching.

It has been observed that the web pages which are related to a common field such as Academic website posses' almost similar structure. For example, the site of two major universities of Haryana, MDU and YMCA UST displays similar structure as shown in figure 7.2.



YMCA University of Science and T
(A Haryana State Government Ur
(Established by Haryana State Legislative Act No. 21 of 2009 & Recogniz
Accredited 'A' Grade by NA
Home
About Us
Governance
Administration
Academics
Admissions
Examination
Training and Placement
Students
Life@YMCA
Recruitments
Contact Us
MAHARSHI DAYANAND UNIVERSITY, ROHTAK
(A State University established under Haryana Act No. XXV of 1975)
'A' Grade University Accredited by NAAC
Home | RTI | Contact Us | F
Websites: e-Rishi ERP : www.mdu
Home
University Profile
Administration
Academics
Library
Distance Education
Admis

Figure 7.2: Snapshot of Similar Structure Web Pages

In above figure 7.2, both websites have similar links like home, administration, admissions etc. Similarly, websites of Automobiles, websites of shopping etc have similar structure. So, it is observed that structure also plays a vital role in showing user interest webpages. Therefore, it is proposed that merely considering the content may not give as good as expected results while computing the rank before showing the results to the user. It is further suggested that crawler should also taken into consideration the structure as parameter while downloading the webpages from the web [98, 99]. At first, crawler searches its repository for similar structure pages. When such webpages are not available in its repository, it crawls the web and downloads those webpages whose structure is similar to those documents that have low IDF (Inverse document frequency) i.e. count is less in database.

7.2 DOM TREE

The internal structure of a web page is in the form of DOM tree i.e. Document object model [67, 101]. Dom is HTML representation of a web document. The composition of HTML documents consists of all nodes whether it is an element, attribute, text etc. In DOM tree, the start node is document node and its branches are extended till all text nodes covered. If the DOM trees of two documents are same then these web pages are treated to be same pages. In general, the structure of a document is expressed as in figure 7.3:-

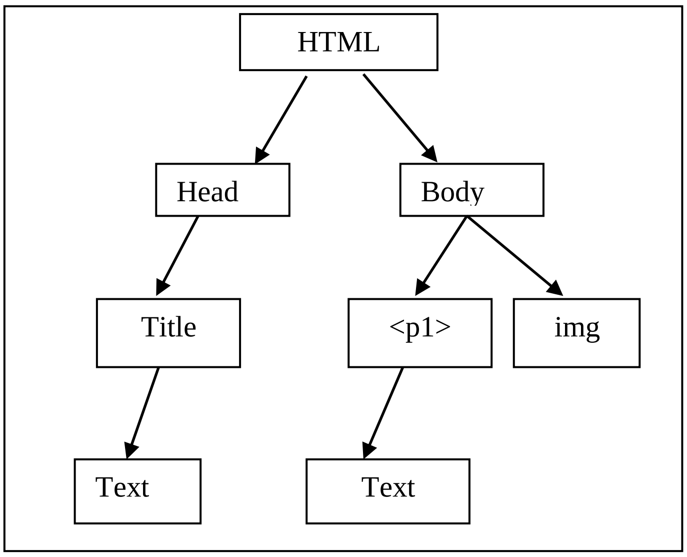


Figure 7.3: DOM Tree

The structure extractor extracts the DOM tree representation of webpage and then with the help of this DOM tree, HTML structure of web documents is used to check similarity between them. If the structures matched then web documents are also said to be similar in structure.

7.3 DOCUMENT’S STRUCTURE DRIVEN CRAWLING

In this work, a novel migrating crawler is proposed that has the capability that will create its Migrants [6] dynamically as per requirement. In this case, Migrants are created as per crawler manager decisions. The Crawler Manager takes the decision on behalf of User satisfaction signal. Every time when user retypes the query or closes the browser, a dialogue box is proposed to be opened which ask them whether the given information is sufficient or not. If user says yes then no action takes place otherwise the signal is sent to crawler manager. In response to this signal, crawler manager do two things as listed below:-

1. First searches the webpages with similar structure in its local repository

2. Searches the webpages with similar structure on the web

To search on web, it creates the Migrants. Now these Migrants crawl the web and send those pages only to the repository whose structure gets matched with supplied structure. These supplied structures are those web pages structure which is on top rank on users’ query but the user was not satisfied with the shown webpages as result. The architecture of proposed structure driver is given in figure 7.4.

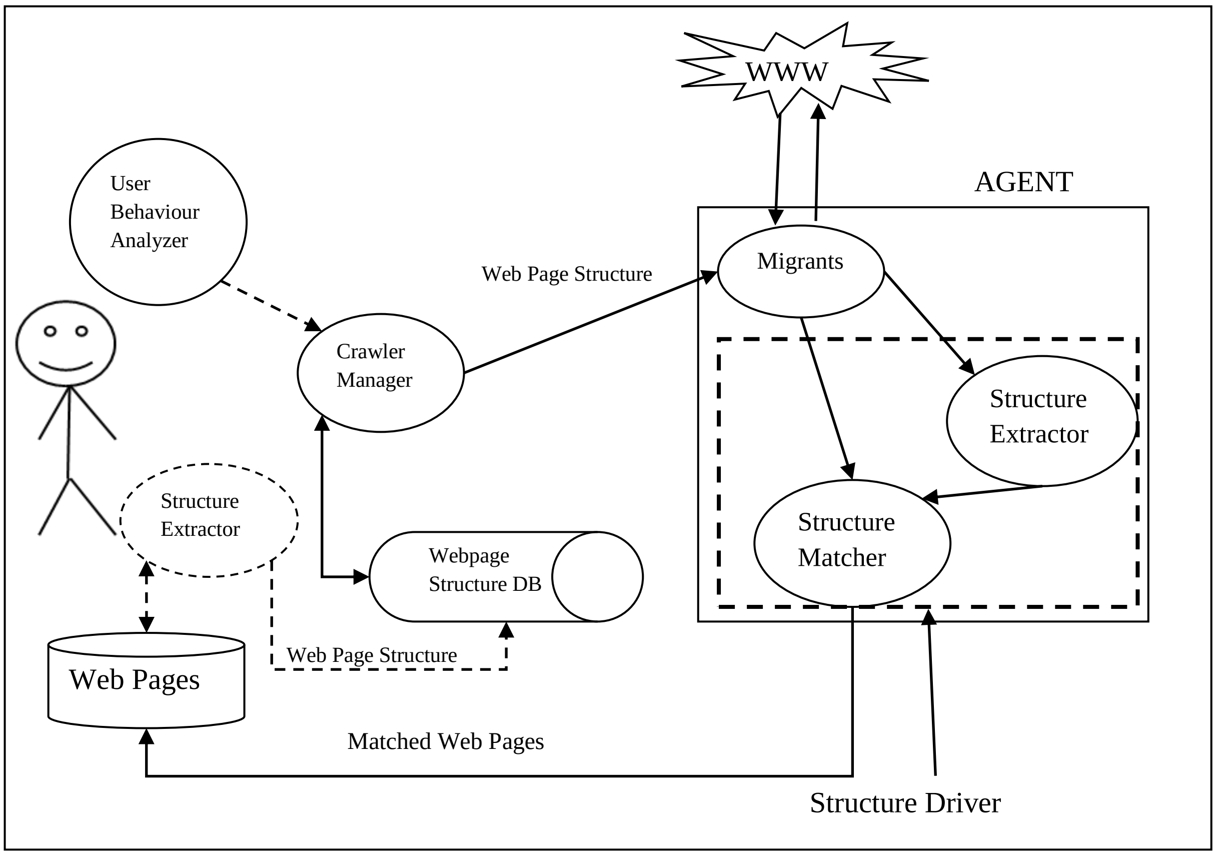


Figure 7.4: Proposed Architecture of Structure Driver

In this architecture, Structure Driver is the major module and it has following sub modules:-

a) **Structure Extractor:** This module is responsible for extracting the structure of a webpage.

b) **Structure Matcher:** It is responsible for matching the structure of webpage supplied by crawler manager and the webpage crawl by the Migrant.

The detailed explanation of these two modules is given below.

7.3.1 Structure Extractor

The matching of two or more documents on the basis of structure can be done by matching their DOM tree. The DOM tree of a document can be obtained by using different software. IE Dom Inspector is one of them and is used here.

7.3.2 Structure Matcher

After taking structure of web pages supplied by crawler manager and by migrants, structure matcher matched them. After getting DOM tree, Simple tree matching algorithm [66] is used here to calculate the similarity between two trees. The matching of nodes at each level occurs and continues till last level subtrees. The following are the steps:-

i. First, the roots of nodes are compared.
ii. If their roots contain different symbols, the comparison stops here and the tree does not match declared.
iii. Otherwise, the algorithm repeats, again and again, to find the maximum matching of subtrees at each level.
iv. The matching subtrees nodes get stored in matrix W.
v. Then, calculate the M according to W i.e. M[i, j] = max(M[i, j-1], M[i-1, j], M[i-1, j-1]) + W[i, j], where W[i, j] = SimpleTreeMatching (Ai, Bj);

The algorithm of Simple Tree Matching is given below in figure 7.5.

```
SimpleTreeMatching (A, B)
Step 1: if the roots of the two trees A and B contain distinct symbols then
        2: return 0;
        3: else m = the number of first-level sub trees of A;
        4: n = the number of first-level sub trees of B;
         5: Initialization: M[i, 0] = 0 for i = 0, … , m; M[0, j] = 0 for j = 0, …, n;
         6: for i = 1 to m do
         7: for j = 1 to n do
         8: M[i, j] = max(M[i, j-1], M[i-1, j], M[i-1, j-1]) + W[i, j]
                       where W[i, j] = SimpleTreeMatching (Ai, Bj);
         9: End for
         10: End for
         11: return (M[m, n]+1);
        12: End if
```

Figure 7.5: Algorithm: DOM Tree Matching

After finding matching number of nodes, similarity can be calculated using the formula given in equation 7.1.

$$\text{Similarity} = \frac{\text{STM (A\&B)}}{(\text{size (A)} + \text{size(B)})/2} \qquad (7.1)$$

Where, Size(A)= number of nodes in Tree A

Size(B)= number of nodes in Tree B

The value of Similarity lies in between 0 & 1. The value which is closer to 1 is treated as more similarity between two trees. Let's take an example by taking two trees as shown in figure 7.6.

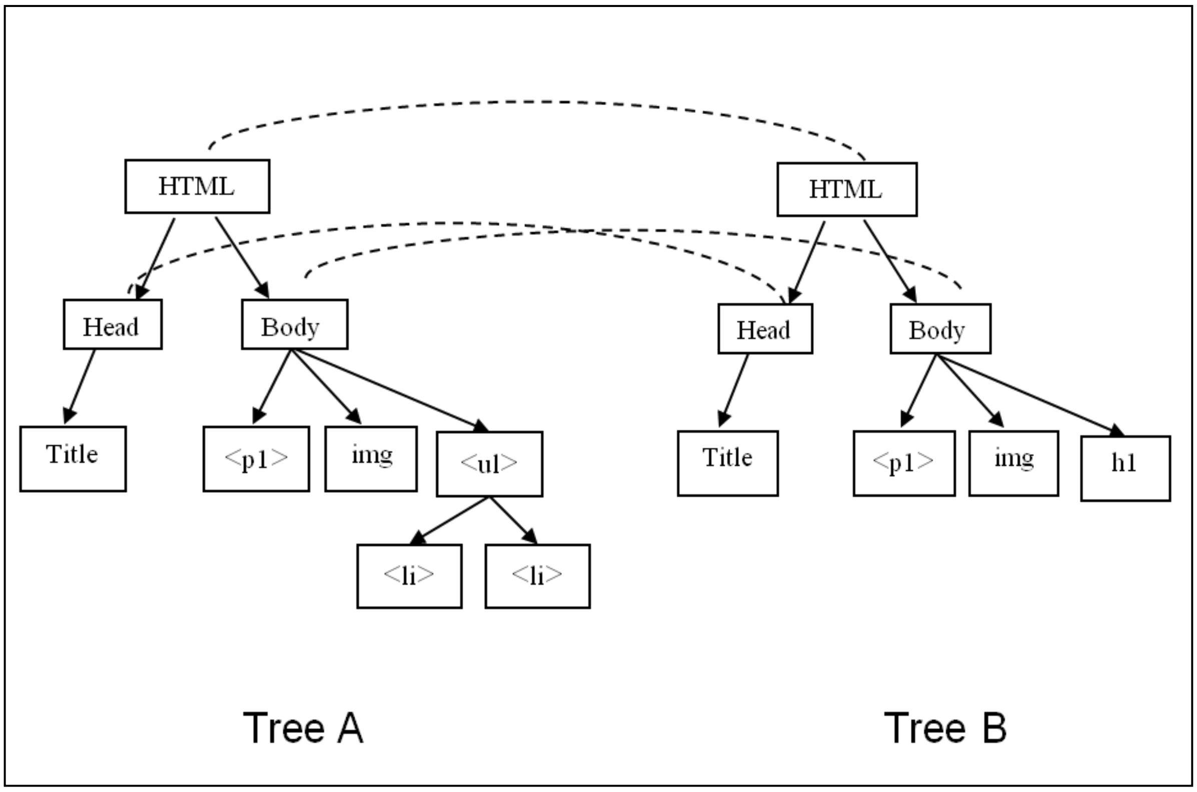


Figure 7.6: Tree Matching

In given above example, A and B are two trees. By applying above Simple Tree Matching algorithm on these two trees and then calculate their similarity.

Matching Nodes at each level as follows:

At level one: 1

At level second: 2

At level third: 3

At level fourth: 0

Therefore, Similarity by using equation 7.1 is given below:-

Similarity= [1+2+3+0]/[(9+7)/2]= 6/8 = .75

The similarity comes out to be .75 i.e. closer to 1. The similarity values show that two trees are similar. The threshold range for structure similarity is taken as 0.4 to 0.9. The similarity lies between this ranges is considered as similar trees.

7.4 PERFORMANCE EVALUATION

The performance of the proposed crawler can be compared with traditional crawler based on measures of accuracy, retrieved links attributes and user satisfaction.

7.4.1 Accuracy

To measure the accuracy of proposed crawler, two metrics are taken namely Precision and Recall. Precision is defined as ratio of relevant URLs crawled and Total URLs crawled and it is written as

$$\text{Precision} = \frac{\text{Relevant URLs crawled}}{\text{Total URLs crawled}}$$

Recall is the ratio of relevant URLs crawled and total number of available URLs. It is written as:-

$$\text{Recall} = \frac{\text{Relevant URLs crawled}}{\text{Total relevant URLs available}}$$

For experimental analysis of the proposed crawler, following ate the list of URLs taken:

- **www.ymcaust.ac.in**
- **www.ngfcet.in**
- **www.titsbhiwani.ac.in**
- **www.gbu.ac.in/UserViewNews.aspx?NewsId=0**

These lists are supplied to proposed crawler and their accuracy is calculated separately. Following are the measurement for each URL.

a)Test 1

The link **www.gbu.ac.in/UserViewNews.aspx?NewsId=0** is supplied to crawler and the following ate the data collected:

Total Number of Total URLs crawled= 980

Total Number of Relevant URLs crawled= 837

Total Number of URLs irrelevant= 143

Total number of relevant URLs not crawled= 126

Therefore, **Precision**= (837/980)*100=83.7%

Recall= (837/837+126)*100=86.9%

b) Test 2

The link **www.ngfcet.in** is supplied to crawler and the following are the data collected:-

Total Number of Total URLs crawled= 376

Total Number of Relevant URLs crawled= 308

Total Number of URLs irrelevant= 68

Total number of relevant URLs not crawled= 45

Therefore, **Precision**= (308/376)*100=81.9%

Recall= (308/308+45)*100=87.2%

c) Test 3

The link **www.titsbhiwani.ac.in** is supplied to crawler and the following ate the data collected:

Total Number of Total URLs crawled= 574

Total Number of Relevant URLs crawled= 480

Total Number of URLs irrelevant= 94

Total number of relevant URLs not crawled= 102

Therefore, **Precision**= (480/574)*100=83.6%

Recall= (480/480+102)=82.4%

d) Test 4

The link **www.ymcaust.ac.in** is supplied to crawler and the following ate the data collected:

Total Number of Total URLs crawled= 158

Total Number of Relevant URLs crawled= 136

Total Number of URLs irrelevant= 22

Total number of relevant URLs not crawled= 27

Therefore, **Precision**= (136/158)*100=86.0%

Recall= (136/136+27)*100=83.4%

Summarizing, the Precision (P) is found in the range of 81.9% to 86%, Recall (R) is found in the range of 82.4% to 87.2%. The table 7.1 shows the summarize result.

Table 7.1: Accuracy Measure

Test #	**P (in %)**	**R (in %)**
1	83.7	86.9
2	81.9	87.2
3	83.6	82.4
4	86.0	83.4
Average	83.8	84.9

The value of Precision and Recall for each of the four tests of the proposed crawler is depicted graphically in figure 7.7:

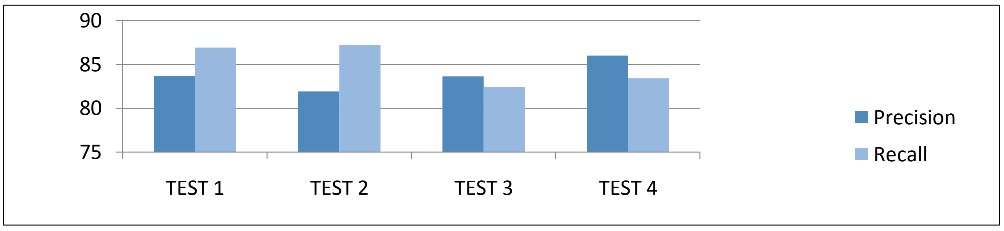


Figure 7.7: P and R values for each of the four test

7.4.2 Attribute of Retrieved links comparison

The performance can also be calculated based on the given below attributes. In given experiment total 70 documents are retrieved by all crawlers. The following attributes gives the performance of proposed crawler and traditional and shown in table 7.2:

a) Retrieved number of dead links: These are the links that doesn't contain any data i.e. pointed to blank webpage.

b) Retrieved number of redundant links: These are links that point to the same downloaded webpage i.e. duplicate link.

Table 7.2: Identified Attributes

Attributes	Conventional	Proposed
Dead Links	9	15
Redundant Links	4	29

The figure 7.8 given below compared the performance of proposed crawler and conventional in terms of above mentioned attributes identified:

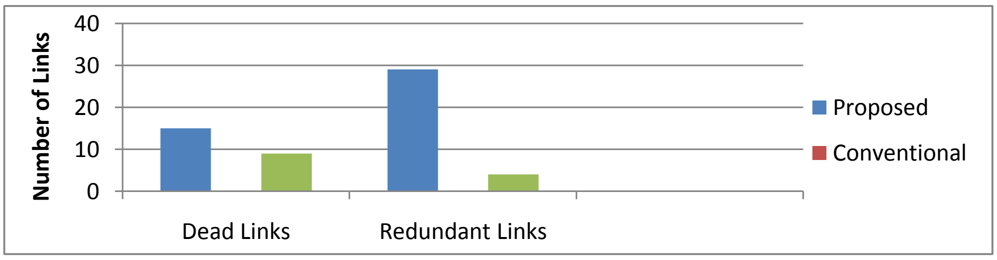


Figure 7.8: Performance on the basis of Identified Attributes

This graph depicts that the relevancy is higher in our proposed mechanism as compared to other search engines and redundancy is also completely removed through this methodology.

7.4.3 User Satisfaction

After maintain the database, with the help of all proposed modules such as URL ordering, Migrant Scheduling, Users' Browsing Behaviour Analyzer, Structure driven crawling etc, was examined by User. On closing the browser, a pop up window was open on which user rates his satisfaction for the available information. A 5-point scale is taken to rate the satisfaction level by 10 user. The table 7.3 given below shows the result.

Table 7.3: User Satisfaction

USER	Proposed Crawler	Traditional Crawler
USER 1	4	2
USER 2	4	1
USER 3	5	3
USER 4	4	1
USER 5	5	3
USER 6	5	2
USER 7	3	2
USER 8	4	2
USER 9	4	1
USER 10	4	3
Average	4.2	2

From the above table, it is clearly observed that proposed crawler has better average rating as compared to traditional crawler. The conclusion and future scope is discussed in next chapter.

CHAPTER VIII

CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION

In this thesis, a novel architecture of Structure Driven Cooperative Migrating Crawler for Retrieving Quality data has been developed and implemented. After a widespread study of existing crawling techniques, limitations were identified which became the basis for the objectives of the work carried out in this thesis.

The proposed work meets the following objectives:-

- **Web Coverage & Scalability**
  The proposed architecture of migrating crawler designed in this work creates the migrants on the basis of the available load and also as per requirement dynamically. These migrants crawls the web effectively and cover the web as maximum as possible. Their dynamic creation property makes them to handle any amount of load and thus make system scalable.

- **Load Distribution and cooperation**
  It should not be the case that some migrants are overloaded and some are idle. This distribution is efficiently balanced with the help of proposed URL Scheduling Module. All the migrants are supplied with the list of URLs for downloading in such a way that load is uniformly distributed over all available migrants. A novel Scheduling method is proposed in this work that schedules the URLs on the basis of the migrants and URL criterions. A multi variant decision making technique based on AHP is used for designing of scheduling policy.

- **Unique Database**
  In order to make the database unique, redundancy removal both at URL level as well as at document level is necessary. Here, normalization steps are applied to identify syntactically similar URLs. A duplicate removal module is proposed to

identify duplicacy at the document level before storing them to ranked database. To save the time hashing technique is applied so that less time is taken while matching the duplicate documents. To further speedup the matching process, the concept of cache is also used. With the help of this proposed module, duplicate documents are identified and only the unique one are stored in Ranked database.

- **Volatile Information Updation**

  Once the webpages get stored in database, there is need to revisit them so that if there is any change then that changed webpages can be stored in place of old ones. The concept of Sitemap is incorporated here. It contains the information of website like its change frequency i.e. daily, weekly, hourly, etc, last modify date, number of links etc. With the help of this information proposed migrants visit the site accordingly and download the updated information only.

- **Relevant Results**

  The results obtained from search engine in response to user query should be of user interest. A module called as User Behaviour Analyzer is proposed. It analyzes the user interest from their browsing behaviour on the web page. The browsing behaviour indicators are mouse click, scroll, print, save, bookmark, key up and down, hyperlink click etc. With the help of these indicators users' interest are identified and the webpages are ranked accordingly.

- **Structure Driven Crawling**

  In general, search results are provided by the technique of content matching. But it has been observed that structure of webpage should also be taken into consideration for better results. A structure driven crawling is proposed here which works on the basis of the feedback supplied by the user. If user is not satisfied with available links then more webpages are incorporated in the database. The identification of such documents is done with the help of their document structure. The structure of document along with content is taken as the criteria of crawling the web. The dynamic creation property of proposed migrants is used to crawl the web and store only those documents that are structurally similar in order to make database rich and relevant.

The proposed architecture of structure driven cooperative migrating crawler has been implemented using Aglets. The following is the summary of results obtained thereof:

- **COVERAGE**

  The web coverage is calculated in terms of number of URLs crawled. The web coverage is high in all four tests as compared to conventional migrating crawler. The increase in coverage lies between 12% to 35% in different tests.

- **NETWORK UTILIZATION**

  The network bandwidth utilized by the proposed migrating crawler is found less in all four tests done so far. The bandwidth calculation is based on duplicate elimination and revisit frequency of crawler. In this work, the saving in bandwidth lies between 65% to 81% in different tests.

- **ACCURACY**

  To measure the accuracy of proposed crawler, two metrics are taken namely Precision and Recall. The results of four tests are given in table 8.1.

Table 8.1: Accuracy Measure

Test	**Precision (in %)**	**Recall (in %)**
1	83.7	86.9
2	81.9	87.2
3	83.6	82.4
4	86.0	83.4
Average	83.8	84.9

The average *Precision* and *Recall* as shown in Table 8.1 is found to be more than 80% suggesting a high performance of the proposed crawler design

8.2 FUTURE WORK

In this dissertation various design issues of crawler has been addressed. But still there is a scope of improvement in few areas as discussed below:

- Fault Tolerance at database level

  The number of migrants is sufficient here to manage the load in case of any failure. But still there should also be some mechanism that will handle fault which may be at database level.

- Hidden Web

  The proposed migrating crawler has been designed for general web. However, the same can be modified for hidden web also. Design issues regarding hidden web must be incorporated while extending this work.

REFERENCES

[1] A. Arasu, J. Cho, and H. G. Molina, “Searching the Web,” *ACM Transactions on Internet Technology*, vol. 1, no. 1, pp. 2–43, Aug. 2001.

[2] A. Terrence Brooks, “Web Search: How the Web has changed information retrieval”, Information Research, April 2003.

[3] S. Brin and L. Page, “The anatomy of a large-scale hypertextual Web search engine,” *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[4] J. Cho and H. Garcia-Molina, “Parallel crawlers,” *Proceedings of the eleventh international conference on World Wide Web - WWW 02*, 2002.

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

[6] N. Singhal, A. Dixit, R. P. Agarwal, A. K. Sharma, (2012), “Using Migrating Agents in Designing Web Search Engines and Property Analysis of Available Platforms”, International Journal of Advancements in Technology (IJOAT), ISSN: 0976-4860, Vol. 3, No. 4, December 2012, pp.254-269.

[7] S. Raghavan and H. Garcia-Molina, "Crawling the Hidden Web", VLDB conference, 2001.

[8] O. Papapetrou, S. Papastavrou, and G. Samaras, “UCYMICRA: Distributed Indexing of the Web Using Migrating Crawlers,” *Advances in Databases and Information Systems Lecture Notes in Computer Science*, pp. 133–147, 2003.

[9] V. Shkapenyuk and T. Suel, “Design and implementation of a high-performance distributed Web crawler,” *Proceedings 18th International Conference on Data Engineering*.

[10] F. Liu, F.-Y. Ma, Y.-M. Ye, M.-L. Li, and J.-D. Yu, “IglooG: A Distributed Web Crawler Based on Grid Service,” *Web Technologies Research and Development - APWeb 2005 Lecture Notes in Computer Science*, pp. 207–216, 2005.

[11] T. Berners-Lee, R. Fielding, and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” 2005.

[12] S. Lawrence, “Searching the World Wide Web,” *Science*, vol. 280, no. 5360, pp. 98–100, Mar. 1998.

[13] B. E. Brewington and G. Cybenko, “How dynamic is the Web?,” *Computer Networks*, vol. 33, no. 1-6, pp. 257–276, 2000.

[14] O. Mcbryan, “GENVL and WWWW: Tools for taming the Web,” *Computer Networks and ISDN Systems*, vol. 27, no. 2, p. 308, 1994.

[15] C. Franklin, "How Internet Search Engines Work", Sep 2002.

[16] B. Grossan, "Search Engines: What they are, how they work, and practical suggestions for getting the most out of them," February 1997.

[17] A. Dixit, N. Singhal, "Retrieving Information from the Web and Search Engine'^ Application, National conference on Emerging trends in Software and Networking Technologies (ETSNT09), Amity University, Noida, India, April 17- 18,2009.

[18] A. Dixit, N. Singhal, "Need of Search Engines and Role of a Web Crawler", National Conference on Recent Trends in Computer and Information Technologies Century (RTCIT-2009), Panipat, Haryana, India, April 2009.

[19] M. Najork and J. L. Wiener, “Breadth-first crawling yields high-quality pages,” *Proceedings of the tenth international conference on World Wide Web - WWW 01*, May 2001.

[20] P. De Bra, G.-J. Houben, Y. Kornatzky, R. Post, "Information Retrieval in distributed hypertexts", Proc. of RIAO'94, Intelligent Multimdia, Information Retrieval systems and management, New York, 1994

[21] M. Hersovici, M. Jacovi, Y. S. Maarek, D. Pelleg, M. Shtalhaim, and S. Ur, “The shark-search algorithm. An application: tailored Web site mapping,” *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 317–326, 1998.

[22] J. M. Kleinberg, “Hubs, authorities, and communities,” *ACM Computing Surveys*, vol. 31, no. 4es, Jan. 1998.

[23] J. C. Miller, G. Rae, F. Schaefer, L. A. Ward, T. Lofaro, and A. Farahat, “Modifications of Kleinbergs HITS algorithm using matrix exponentiation and web log records,” *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR 01*, 2001.

[24] M. Abukausar, V. S. Dhaka, and S. K. Singh, “Web Crawler: A Review,” *International Journal of Computer Applications*, vol. 63, no. 2, pp. 31–36, 2013.

[25] S. Chakrabarti, M. V. D. Berg, and B. Dom, “Focused crawling: a new approach to topic-specific Web resource discovery,” *Computer Networks*, vol. 31, no. 11-16, pp. 1623–1640, 1999.

[26] U. Schonfeld and N. Shivakumar, “Sitemaps,” *Proceedings of the 18th international conference on World wide web - WWW 09*, 2009.

[27] M. Najork and J. L. Wiener, “Breadth-first crawling yields high-quality pages,” *Proceedings of the tenth international conference on World Wide Web - WWW 01*, May 2001.

[28] S. S. Vishwakarma , A. Jain, and A. K. Sachan. "A Novel Web Crawler Algorithm on Query based Approach with Increases Efficiency." vol 46 2011, pp: 34-37.

[29] O. Brandman, J. Cho, H. Garcia-Molina, and N. Shivakumar, “Crawler-Friendly Web Servers,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 28, no. 2, pp. 9–14, Jan. 2000.

[30] D. Lefortier, L. Ostroumova, E. Samosvat, and P. Serdyukov, “Timely crawling of high-quality ephemeral new content,” *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management - CIKM 13*, 2013.

[31] A. Tripathy and P. K. Patra, “A Web Mining Architectural Model of Distributed Crawler for Internet Searches Using PageRank Algorithm,” *2008 IEEE Asia-Pacific Services Computing Conference*, 2008.

[32] M. Shoaib and A. K. Maurya, “URL ordering based performance evaluation of Web crawler,” *2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014)*, 2014.

[33] A. Guerriero, F. Ragni, and C. Martines, “A dynamic URL assignment method for parallel web crawler,” *2010 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, 2010.

[34] Y. Wan and H. Tong, “URL Assignment Algorithm of Crawler in Distributed System Based on Hash,” *2008 IEEE International Conference on Networking, Sensing and Control*, 2008.

[35] D. Ge and Z. Ding, “A Task Scheduling Strategy Based on Weighted Round-Robin for Distributed Crawler,” *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, 2014.

[36] A. Chandramouli, S. Gauch, and J. Eno, “A popularity-based URL ordering algorithm for crawlers,” *3rd International Conference on Human System Interaction*, 2010.

[37] T. Berners-Lee, R. Fielding, and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” 2004.

[38] S. H. Lee, S. J. Kim, and S. H. Hong, “On URL Normalization,” *Computational Science and Its Applications – ICCSA 2005 Lecture Notes in Computer Science*, pp. 1076–1085, 2005.

[39] U. Schonfeld, Z. Bar-Yossef, and I. Keidar, “Do not crawl in the DUST,” *Proceedings of the 15th international conference on World Wide Web - WWW 06*, 2006.

[40] A. Agarwal, H. S. Koppula, K. P. Leela, K. P. Chitrapura, S. Garg, P. K. Gm, C. Haty, A. Roy, and A. Sasturkar, “URL normalization for de-duplication of web pages,” *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM 09*, 2009.

[41] F. Qureshi & A. A. Khan. Improving the Performance of Crawler Using Body Text Normalization. Compusoft, 2(7), 2013, pp215.

[42] L. Leitão, P. Calado, and M. Herschel, “Efficient and Effective Duplicate Detection in Hierarchical Data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 5, pp. 1028–1041, 2013.

[43] W. Li, J.-Y. Liu, and C. Wang, “Web Document Duplicate Removal Algorithm Based on Keyword Sequences,” *2005 International Conference on Natural Language Processing and Knowledge Engineering*.

[44] H. L. Yu, L. Bingwu, and Y. Fang, “Similarity Computation of Web Pages of Focused Crawler,” *2010 International Forum on Information Technology and Applications*, 2010.

[45] M. C. A. L. A. Vidal, A. S. D. Silva, E. S. D. Moura, and J. M. B. Cavalcanti, “GoGetIt!,” *Proceedings of the 15th international conference on World Wide Web - WWW 06*, 2006.

[46] H. Wang and Y. Zhang, “Web Data Extraction Based on Simple Tree Matching,” *2010 WASE International Conference on Information Engineering*, 2010.

[47] C. Y. Kang, “DOM-Based Web Pages to Determine the Structure of the Similarity Algorithm,” *2009 Third International Symposium on Intelligent Information Technology Application*, 2009.

[48] Nierman and H. V. Jagadish. “Evaluating structural similarity in XML documents”. In Proceedings of the 5th International Workshop on the Web and Databases (WebDB2002), Madison, Wisconsin, USA, June 2002.

[49] E. Bertino, G. Guerrini, and M. Mesiti, “A matching algorithm for measuring the structural similarity between an XML document and a DTD and its applications,” *Information Systems*, vol. 29, no. 1, pp. 23–46, 2004.

[50] M. Morita and Y. Shinoda, “Information Filtering Based on User Behavior Analysis and Best Match Text Retrieval,” *Sigir ’94*, pp. 272–281, 1994.

[51] B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl, “Using filtering agents to improve prediction quality in the GroupLens research collaborative filtering system,” *Proceedings of the 1998 ACM conference on Computer supported cooperative work - CSCW 98*, 1998.

[52] M. Claypool, P. Le, M. Wased, and D. Brown, “Implicit interest indicators,” *Proceedings of the 6th international conference on Intelligent user interfaces - IUI 01*, 2001.

[53] H. Weinreich, H. Obendorf, E. Herder, and M. Mayer, “Off the beaten tracks,” *Proceedings of the 15th international conference on World Wide Web - WWW 06*, 2006.

[54] J. Goecks and J. Shavlik, “Learning users interests by unobtrusively observing their normal behavior,” *Proceedings of the 5th international conference on Intelligent user interfaces - IUI 00*, 2000.

[55] H. Lieberman, “Letizia, an agent that assists web browsing,” In:Burke, R., ed. Proceedings of the International Joint Conference on Artificial Intelligence. Menlo Park, CA: AAAI Press, 1995, pp. 924-929.

[56] X. Ying. The Research on User Modeling for Internet Personalized Services. National University of Defense Technology, 2003.

[57] K. Xing, B. Zhang, B. Zhou, and Y. Liu, "Behavior Based User Interests Extraction Algorithm," *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, 2011.

[58] Z. Jingling, X. Wang, and Y. Zhou. "Study and implementation of user behaviour analysis." Advanced Communication Technology (ICACT), 2010 The 12th International Conference on. Vol. 1. IEEE, 2010.

[59] Y. Qinghong, H. Hao, and X. Neng, "The research on user interest model based on quantization browsing behavior," *2012 7th International Conference on Computer Science & Education (ICCSE)*, 2012.

[60] T. L. Saaty, Analytic Hierarchy process, McGraw Hill 1980.

[61] D. Punj and A. Dixit, "Design of an Efficient Migrating Crawler based on Sitemaps," *International Journal of Advanced Science and Technology*, vol. 83, pp. 1–12, 2015.

[62] D. Punj and A. Dixit, "Capturing User Browsing Behaviour Indicators," *Electrical & Computer Engineering: An International Journal*, vol. 4, no. 2, pp. 22–30, 2015.

[63] D. Punj & A. Dixit. Web Crawler Design Issues: A Review. International Journal of Management, IT and Engineering, 2(8), 2012, pp394-404.

[64] A.K. Sharma, J.P. Gupta, D. P. Agarwal, PARCAHYD: An Architecture of a Parallel Crawler based on Augmented Hypertext Documents. Ph.D. Thesis, HIT & M, Gwalior, Aug. 2003.

[65] https://www.xml-sitemaps.com/

[66] H. Wang and Y. Zhang, "Web Data Extraction Based on Simple Tree Matching," *2010 WASE International Conference on Information Engineering*, 2010.

[67] C. Y. Kang, "DOM-Based Web Pages to Determine the Structure of the Similarity Algorithm," *2009 Third International Symposium on Intelligent Information Technology Application*, 2009.

[68] The MD5 Message-Digest Algorithm, available at: http://tools.ietf.org/html/rfc132.

[69] G. Jeh and J. Widom, "SimRank," *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 02*, 2002.

[70] S.-H. Liao, P.-H. Chu, and P.-Y. Hsiao, “Data mining techniques and applications – A decade review from 2000 to 2011,” *Expert Systems with Applications*, vol. 39, no. 12, pp. 11303–11311, 2012.

[71] S. Shinde, M. R Bidkar, M. N. Deore, M. N. Salunke & M. Neelay. Detection of Distinct URL and Removing DUST Using Multiple Alignments of Sequences. International Research Journal of Engineering and Technology (IRJET), 03(1), 2016.

[72] Chandel, G. S., Patidar, K., & Mali, M. S. (2016). A Result Evolution Approach for Web usage mining using Fuzzy C-Mean Clustering Algorithm. International Journal of Computer Science and Network Security (IJCSNS),16 (1), 135.

[73] G. Wang, X. Zhang, S. Tang, H. Zheng, and B. Y. Zhao, “Unsupervised Clickstream Clustering for User Behavior Analysis,” *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI 16*, 2016.

[74] K. Rodrigues, M. Cristo, E. S. D. Moura, and A. D. Silva, “Removing DUST Using Multiple Alignment of Sequences,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 8, pp. 2261–2274, Jan. 2015.

[75] L. Pamulaparty, C. G. Rao, and M. S. Rao, “XNDDF: Towards a Framework for Flexible Near-Duplicate Document Detection Using Supervised and Unsupervised Learning,” *Procedia Computer Science*, vol. 48, pp. 228–235, 2015.

[76] Y. Yu, Z. Hu, and Y. Zhang, “Rearch on Large Scale Documents Deduplication Technique based on Simhash Algorithm,” *Proceedings of the First International Conference on Information Sciences, Machinery, Materials and Energy*, 2015.

[77] M. Narvekar and S. S. Banu, “Predicting Users Web Navigation Behavior Using Hybrid Approach,” *Procedia Computer Science*, vol. 45, pp. 3–12, 2015.

[78] A. Ladekar, P. Pawar, D. Raikar, and J. Chaudhari, “Web Log based Analysis of User's Browsing Behavior,” *International Journal of Computer Applications*, vol. 115, no. 11, pp. 5–8, 2015.

[79] D. Anupama and S. D. Gowda, “Clustering of Web User Sessions to Maintain Occurrence of Sequence in Navigation Pattern,” *Procedia Computer Science*, vol. 58, pp. 558–564, 2015.

[80] K. Filipowski, “Comparison of Scheduling Algorithms for Domain Specific Web Crawler,” *2014 European Network Intelligence Conference*, 2014.

[81] W. R. Bhaginath, S. Shingade, and M. Shirole, “Virtualized dynamic URL assignment web crawling model,” *2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014)*, 2014.

[82] G. H. Agre and N. V. Mahajan, “Keyword focused web crawler,” *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, 2015.

[83] Q. Pu, “The Design and Implementation of a High-Efficiency Distributed Web Crawler,” *2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, 2016.

[84] M. Kumar and R. Bhatia, “Design of a mobile Web crawler for hidden Web,” *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, 2016.

[85] A. Lawankar and N. Mangrulkar, “A review on techniques for optimizing web crawler results,” *2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)*, 2016.

[86] S. Gaikwad and N. Bogiri, “A survey analysis on duplicate detection in Hierarchical Data,” *2015 International Conference on Pervasive Computing (ICPC)*, 2015.

[87] P. E. D., P., P. Anandhakumar, G. D. Raj, and T. Rajendran, “Efficient Priority Queue algorithm and Strainer mode Technique for identification and eradication of duplications in XML records,” *2013 Fifth International Conference on Advanced Computing (ICoAC)*, 2013.

[88] Y. Takano and R. Miura, “FARIS: Fast and Memory-Efficient URL Filter by Domain Specific Machine,” *2016 6th International Conference on IT Convergence and Security (ICITCS)*, 2016.

[89] C. Kumari, D. Joshi, and S. N. Singh, “Canonization rules for detecting different URLs,” *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*, 2016.

[90] Y. Yang, L. Zhang, G. Liu, and E. Chen, “UPCA: An efficient URL-Pattern based algorithm for accurate web page classification,” *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, 2015.

[91] B. Mehta and M. Narvekar, “DOM tree based approach for Web content extraction,” *2015 International Conference on Communication, Information & Computing Technology (ICCICT)*, 2015.

[92] P. Zeng, Q. Tan, W. Cao, and T. Huang, “High Efficient Multi-pattern URL Matching Algorithm Based on HTTP Protocol,” *2015 Fifth International Conference on Communication Systems and Network Technologies*, 2015.

[93] D. Punj & A. Dixit. URL Ordering Policies for Distributed Crawlers: A Review. Journal of Network Communications and Emerging Technologies (JNCET) Volume 5, Special Issue 2.2015.

[94] L. Leitão, P. Calado, and M. Herschel, “Efficient and Effective Duplicate Detection in Hierarchical Data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 5, pp. 1028–1041, 2013.

[95] G. Tummarello, “A sitemap extension to enable efficient interaction with large Quantity of linked data”, Presented at W3C Workshop on RDF Access to Relational Databases 2007.

[96] R. Cyganiak, H. Stenzhorn, R. Delbru, S. Decker, and G. Tummarello, “Semantic Sitemaps: Efficient and Flexible Access to Datasets on the Semantic Web,” *Lecture Notes in Computer Science The Semantic Web: Research and Applications*, pp. 690–704, 2008.

[97] G. S. Bedi, A. Singh, “Analysis of Search Engine Optimization (SEO) Techniques”, published in International Journal of Advanced Research in Computer Science and Software Engineering in Vol; 4, March 2014.

[98] D. Punj and A. Dixit “Document structure based Filtering in Migrating Crawler- A Review”, presented in International Conference Paradigms shift in Management and Technology at Faridabad in 9-10 March 2015.

[99] D. Punj and A. Dixit “A Novel Approach for Document Structure based Filtering in Migrating Crawler”, published in International Journal of YMCA UST Faridabad in 2014.

[100] S.-H. Tan, M. Chen, and G.-H. Yang, “User behavior mining on large scale web log data,” *The 2010 International Conference on Apperceiving Computing and Intelligence Analysis Proceeding*, 2010.

[101] D. A. D’Mello and V. S. Ananthanarayana , “A Tree Structure for Efficient Web Service Discovery” Department of Computer Science & Engineering, St. Joseph Engineering College, Mangalore, INDIA – 575 028.

[102] D. Punj, S. Juneja and A. Dixit “Improving Search Results Based On Users’ Browsing Behaviour Using Apriori Algorithm”, 50th Golden jubilee International annual convention of Computer Society of India (CSI-2015) theme Digital Life, organised by BVICAM New Delhi, 2nd to 5th December 2015.

[103] D. Punj and A. Dixit, “Design of a Migrating Crawler Based on a Novel URL Scheduling Mechanism using AHP,” *International Journal of Rough Sets and Data Analysis*, vol. 4, no. 1, pp. 95–110, Jan. 2017.

[104] A. Dixit and A. K. Sharma "Self Adjusting Refresh Time Based Architecture for Incremental Web Crawler" International Journal of Computer Science and Network Security (IJCSNS) Korea. Vol. 8 No. 12 pp. 349-354. ISSN: 1738-7906. 2008.

[105] Malav, M. Prasad, and A. Rasool. "Variations of Wu-Manber String Matching Algorithm." International Journal of Engineering Research and Technology. Vol. 3. No. 4. ESRSA Publications.2014.

[106] D. K. Sharma and A. K. Sharma, “Search Engine,” *ICT Influences on Human Development, Interaction, and Collaboration*, pp. 117–131, 2012.

[107] D. K. Sharma and A. K. Sharma, “A Novel Architecture for Deep Web Crawler,” *Network and Communication Technology Innovations for Web and IT Advancement*, pp. 106–129.

[108] H. Agrawal and S. Yadav, “Search Engine Results Improvement -- A Review,” *2015 IEEE International Conference on Computational Intelligence & Communication Technology*, 2015.

[109] J. A. Khan, “Comparative study of information retrieval models used in search engine,” *2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014),* 2014.

[110] D. K. Sharma and A. K. Sharma, “Search Engine: A Backbone for Information,” *ICT Influences on Human Development, Interaction, and Collaboration*, pp. 117–131, 2012.

[111] A. Surya and D. K. Sharma, "An approach for web page ordering using user session," *2013 Ieee Conference On Information And Communication Technologies*, 2013.

[112] A. Gupta, A. Dixit, and T. Kumari, "A Novel Link and Prospective terms Based Page Ranking Technique," *International Journal of Engineering Trends and Technology*, vol. 27, no. 6, pp. 292–299, 2015.

[113] Manvi, K. K. Bhatia, and A. Dixit, "A Novel Design of Hidden Web Crawler using Ontology," *International Journal of Engineering Trends and Technology*, vol. 26, no. 1, pp. 14–20, 2015.

[114] P. Devi, A. Gupta, A. Dixit, "Comparative Study of HITS and PageRank Link Based Ranking Algorithms", International Journal of Advanced Research in Computer and Communication Engineering" , Vol-3,Issue 2. ISSN: 2278-1021. Feb-2014.

[115] K. Sachdeva and A. Dixit, "Estimating Page Importance based on Page Accessing Frequency," *International Journal of Computer Applications*, vol. 86, no. 5, pp. 1–6, 2014.

[116] N. Singhal, A. Dixit and R. P. Agarwal, (2012), "User Perception based Inverted Index for Web Search Engines", International Journal of Contemporary Research in Engineering and Technology, ISSN: 2250-0510, Vol. 2, No. 2, pp. 39-44, 2012.

[117] N. Singhal, A. Dixit, A. K. Sharma, and R. P. Aggarwal "Regulating Frequency of a Migrating Web Crawler based on Users Interest" International Journal of Engineering and Technology (IJET) Vol 4 No 4 .ISSN: 2319-8613, EISSN: 0975-4024 Impact Factor (IF)=.33) Aug-Sep 2012. pp 246-253.

[118] A. Dixit and U. C. Pant "A Novel and Efficient Mechanism for Securing Migrating Crawler Data" International Journal of Contemporary Research in Engineering & Technology (IJCREAT) Vol. 1, No. 1, ISSN: 2250-0510. July-Dec 2011. pp. 1-8.

[119] N. Singhal, A. Dixit, and A. K. Sharma, "Design of a Priority Based Frequency Regulated Incremental Crawler," *International Journal of Computer Applications*, vol. 1, no. 1, pp. 42–47, 2010.

[120] A. Dixit, and J. K. Seth "A migrating parallel exponential crawling approach to search engine" International Journal of Computer Science & Emerging

Technologies (IJCSET) Volume 1. Issue 2. ISSN: 2044-6004.August, 2010. pp. 83-90.

[121] A. Dixit PhD thesis MDU Rohtak, 2010.

[122] A. Gupta, A. Dixit and A.K. Sharma, "A Novel Web Page Change Detection Technique For Migrating Crawlers", 50th Golden jubilee international annual convention of Computer Society of India (CSI-2015) theme Digital Life, organised by BVICAM New Delhi, 2nd to 5th December 2015.

[123] A. Gupta, A. Dixit, P. Devi, "A novel user preference and feedback based Page Ranking technique", IEEE 2nd International Conference on Computing for Sustainable Global Development (INDIACom), Organised by BVICAM New Delhi, Print ISBN: 978-9-3805-4415-1. Mar, 11-13 2015. pp 1335- 1340.

[124] A. Kaur, A. Dixit and P. S. Grover, "Quantitative evaluation of proposed maintainability model using AHP method", IEEE 2nd International Conference on Computing for Sustainable Global Development (INDIACom), Organised by BVICAM New Delhi.Print ISBN: 978-9-3805-4415-1. Mar 2015.pp 1367-1371.

[125] A. Dixit and A. Sharma, "Security System for Migrating Crawlers," *2011 International Conference on Computational Intelligence and Communication Networks*, 2011.

[126] N. Singhal, R. Agarwal, A. Dixit, and A. Sharma, "Information Retrieval from the Web and Application of Migrating Crawler," *2011 International Conference on Computational Intelligence and Communication Networks*, 2011.

[127] A. Dixit and A. K. Sharma, "A mathematical model for crawler revisit frequency," *2010 IEEE 2nd International Advance Computing Conference (IACC)*, 2010.

[128] A. K. Sharma, A. Dixit, and A. Arora "Relavance Improvement by Selective Retrieval Strategies in Meta Search Engines" Accepted in National conference on Recent Developments in Applicable Mathematics & Information Technology JIET Guna Oct 2009.

[129] A. Dixit, N. Singhal, "Need of Search Engines and Role of a Web Crawler", National Conference on Recent Trends in Computer and Information Technologies Century (RTCIT-2009), Panipat, Haryana, India, April 2009.

[130] A. Dixit, N. Singhal "Web Crawling Techniques: A Review" accepted in National Conference at ITS MohanNagar, Ghaziabad on Information Security: Emerging Threats and Innovations in the 21stCentury to be held on 4th Apr 2009.

[131] Z.-K. Wei and J.-P. Du, "Research on several key issues about Search Engines," *2008 International Conference on Machine Learning and Cybernetics*, 2008.

APPENDIX I

The following figures show the snapshots of implementation of the proposed URL Organizer. An example of www.google.com is taken here:-

Web Browser Interface

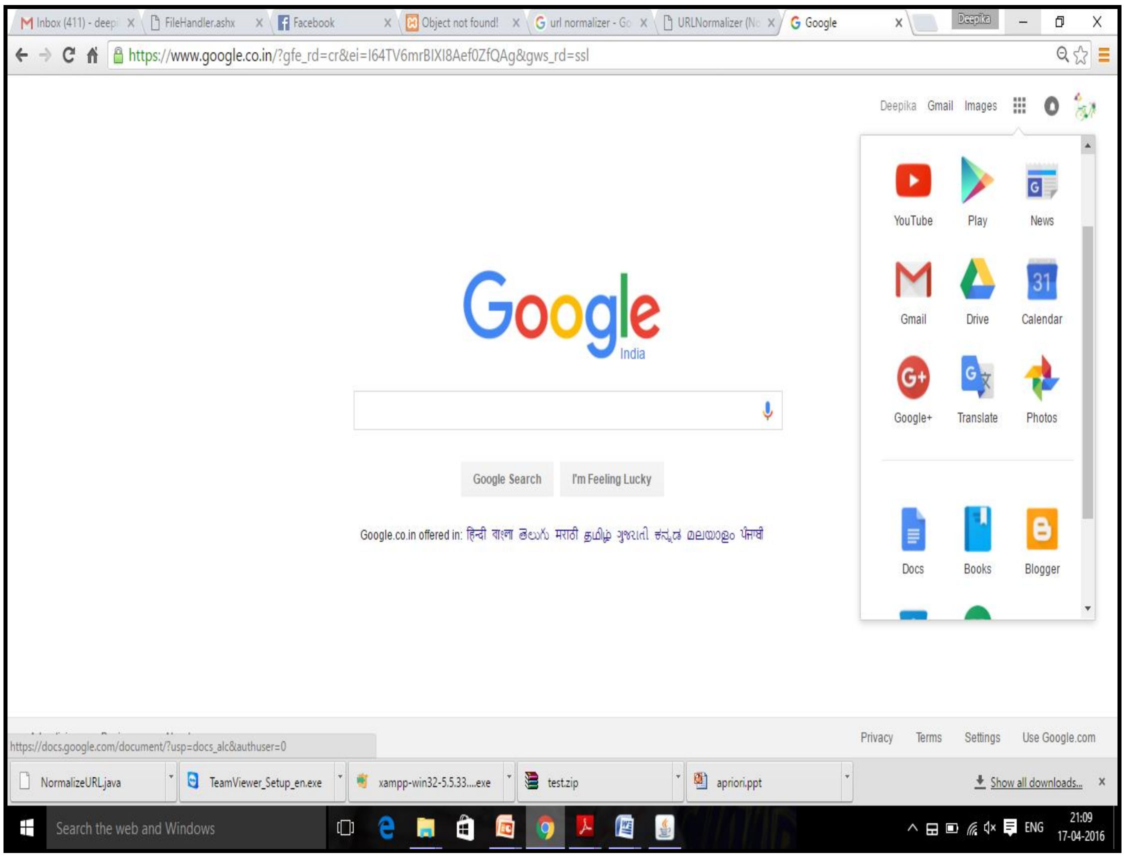
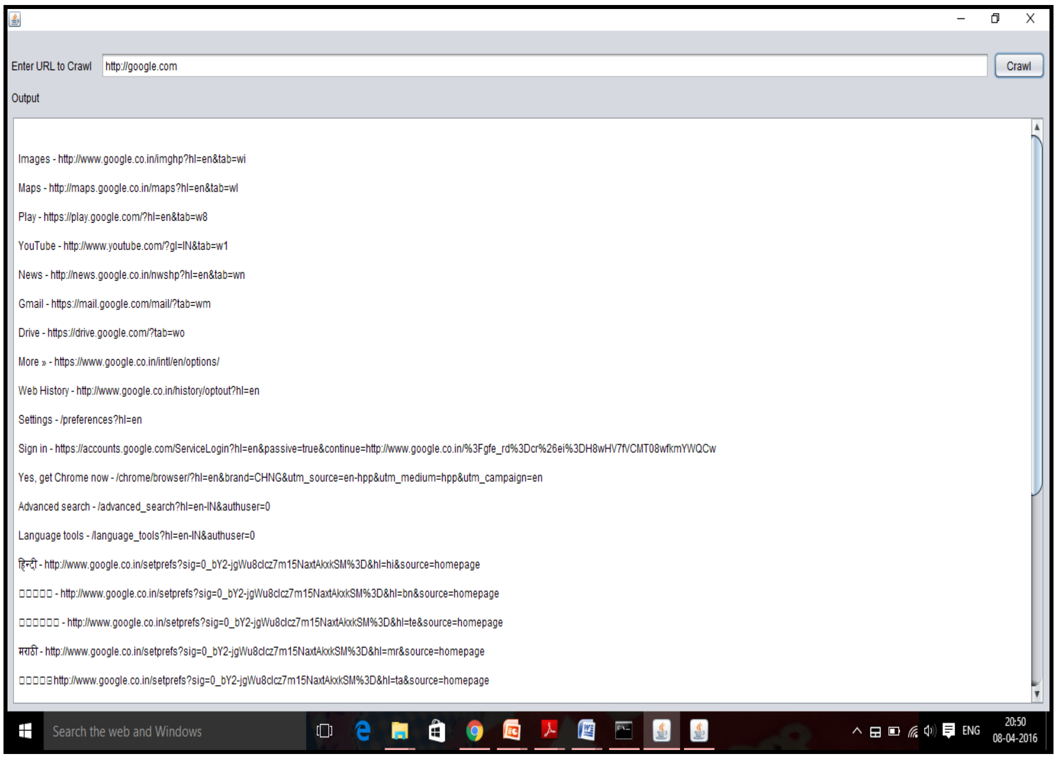


Figure A1.1: Google Home page and its links

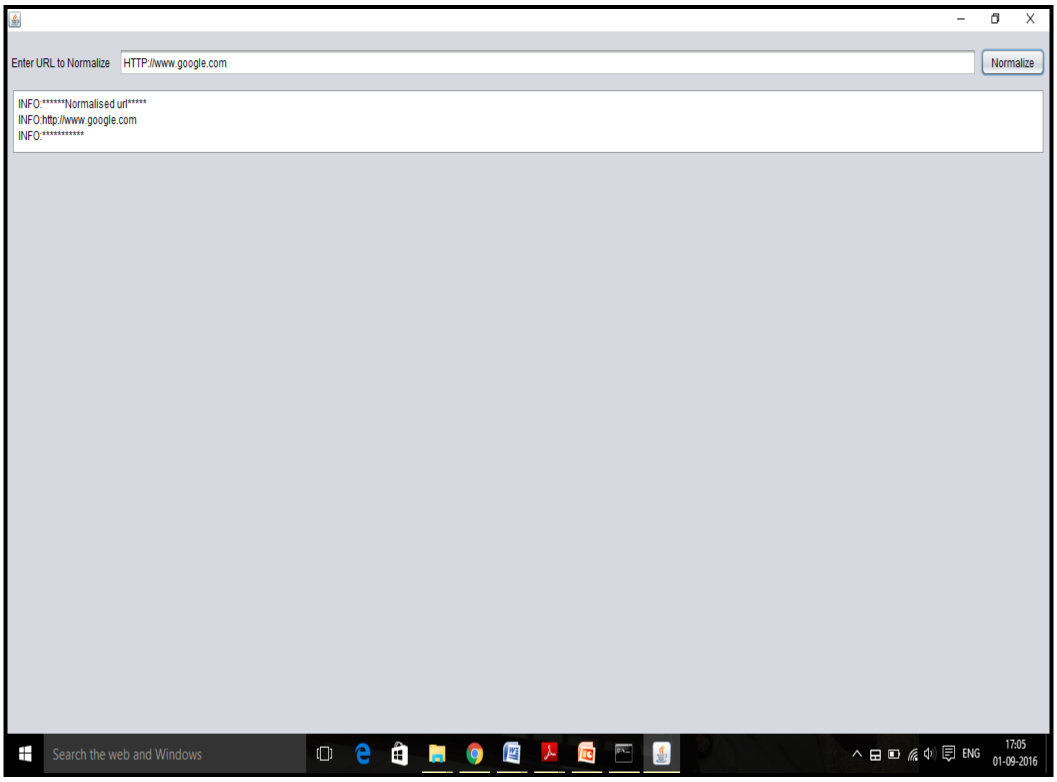
Link Extraction by Proposed Crawler



```
Images - http://www.google.co.in/imghp?hl=en&tab=wi

Maps - http://maps.google.co.in/maps?hl=en&tab=wl

Play - https://play.google.com/?hl=en&tab=w8

YouTube - http://www.youtube.com/?gl=IN&tab=w1

News - http://news.google.co.in/nwshp?hl=en&tab=wn

Gmail - https://mail.google.com/mail/?tab=wm

Drive - https://drive.google.com/?tab=wo

More » - https://www.google.co.in/intl/en/options/

Web History - http://www.google.co.in/history/optout?hl=en

Settings - /preferences?hl=en

Sign in - https://accounts.google.com/ServiceLogin?hl=en&passive=true&continue=http://www.google.co.in/%3Fgfe_rd%3Dcr%26ei%3DH8wHV7fVCMT08wfkmYWQCw

Yes, get Chrome now - /chrome/browser/?hl=en&brand=CHNG&utm_source=en-hpp&utm_medium=hpp&utm_campaign=en

Advanced search - /advanced_search?hl=en-IN&authuser=0

Language tools - /language_tools?hl=en-IN&authuser=0

हिन्दी - http://www.google.co.in/setprefs?sig=0_bY2-jgWu8clcz7m15NaxtAkxkSM%3D&hl=hi&source=homepage

□□□□□ - http://www.google.co.in/setprefs?sig=0_bY2-jgWu8clcz7m15NaxtAkxkSM%3D&hl=bn&source=homepage

□□□□□□ - http://www.google.co.in/setprefs?sig=0_bY2-jgWu8clcz7m15NaxtAkxkSM%3D&hl=te&source=homepage

मराठी - http://www.google.co.in/setprefs?sig=0_bY2-jgWu8clcz7m15NaxtAkxkSM%3D&hl=mr&source=homepage

□□□□□http://www.google.co.in/setprefs?sig=0_bY2-jgWu8clcz7m15NaxtAkxkSM%3D&hl=ta&source=homepage
```

Figure A1.2: Extracted Links

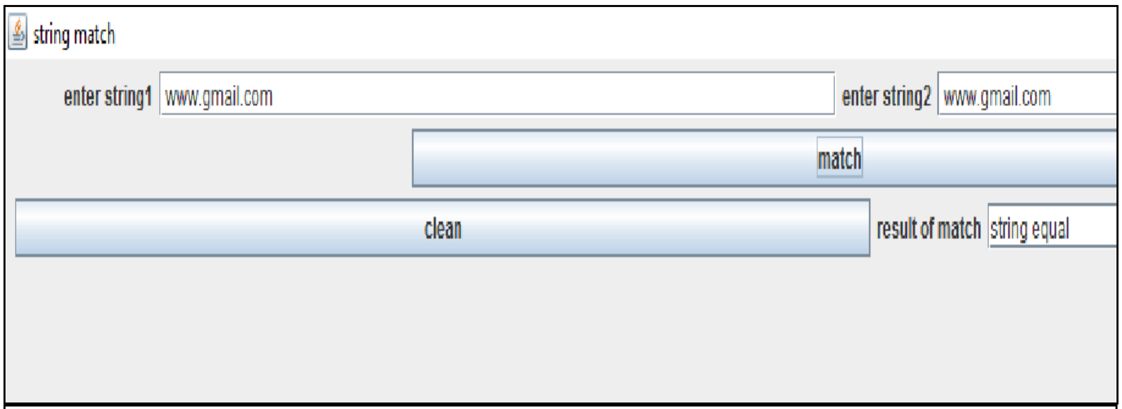
URL Normalization Step



```
INFO:******Normalised url*****
INFO:http://www.google.com
INFO:***********
```

Figure A1.3: Normalized URL

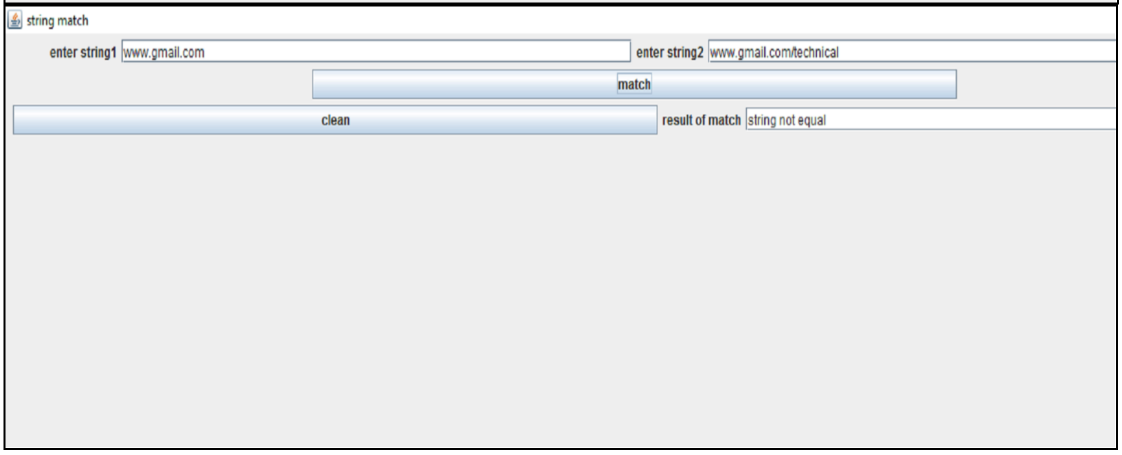
After Normalization, duplicate matching result


string match

enter string1 www.gmail.com

enter string2 www.gmail.com

result of match string equal


string match

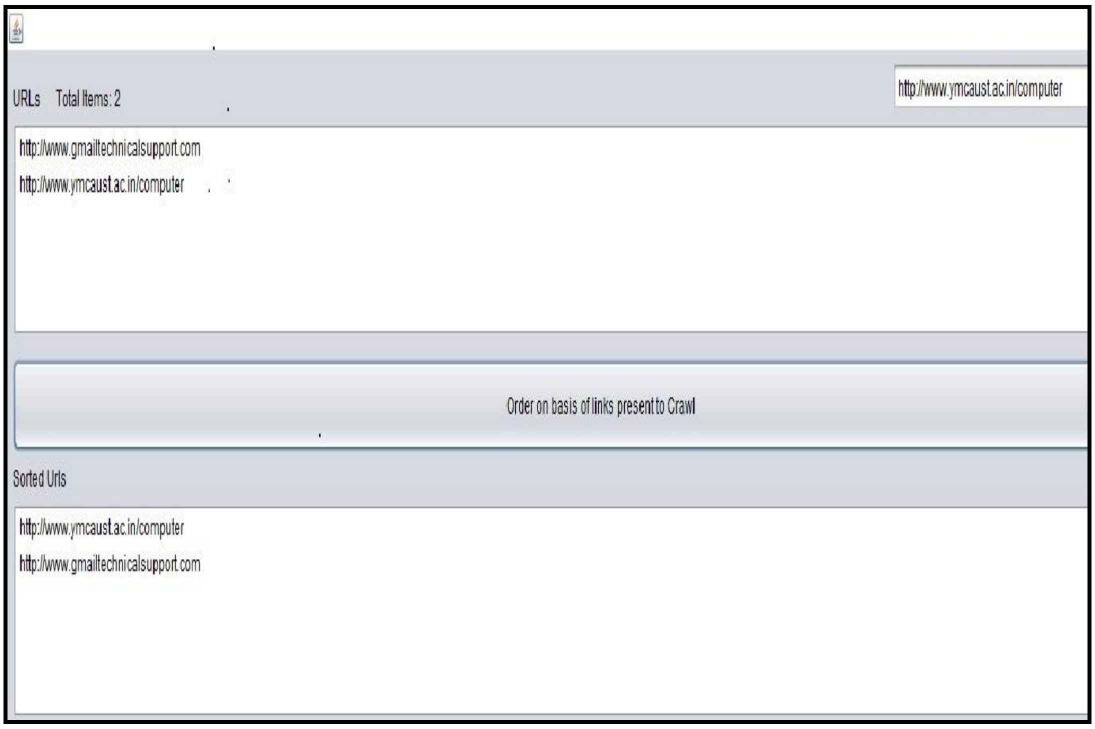
enter string1 www.gmail.com

enter string2 www.gmail.com/technical

result of match string not equal

Figure A1.4: URL Matching Process

URL Ordering on the basis of embedded links


http://www.ymcaust.ac.in/computer

URLs Total Items: 2

http://www.gmailtechnicalsupport.com
http://www.ymcaust.ac.in/computer

Sorted Urls

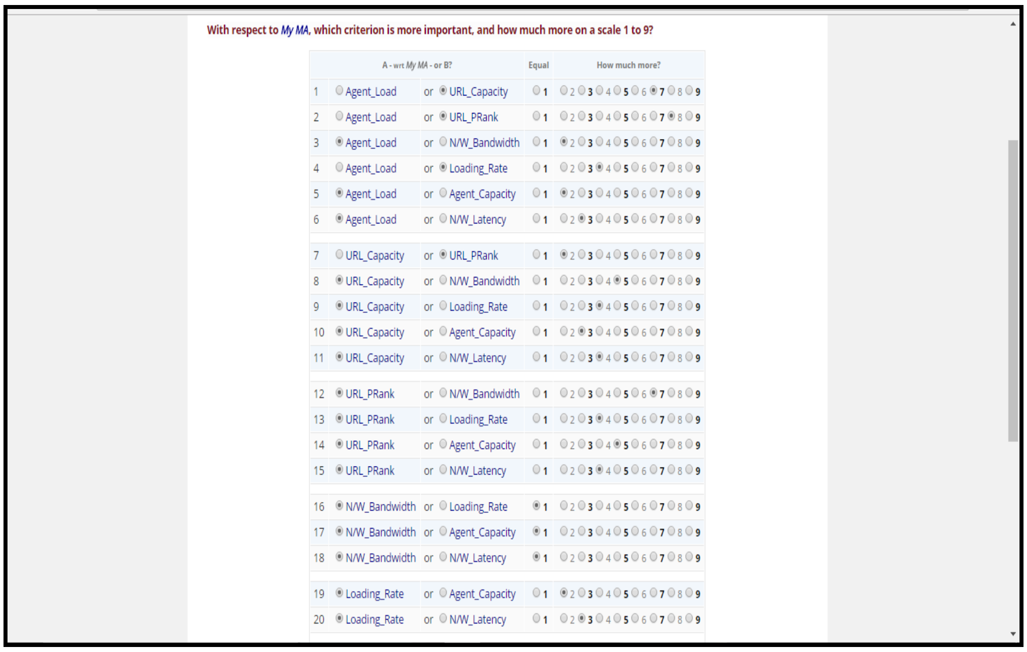
http://www.ymcaust.ac.in/computer
http://www.gmailtechnicalsupport.com

Figure A1.5: URL Ordering Process

APPENDIX II

Snapshots of URL Scheduling Process

Scheduling Criterions and their relative importance

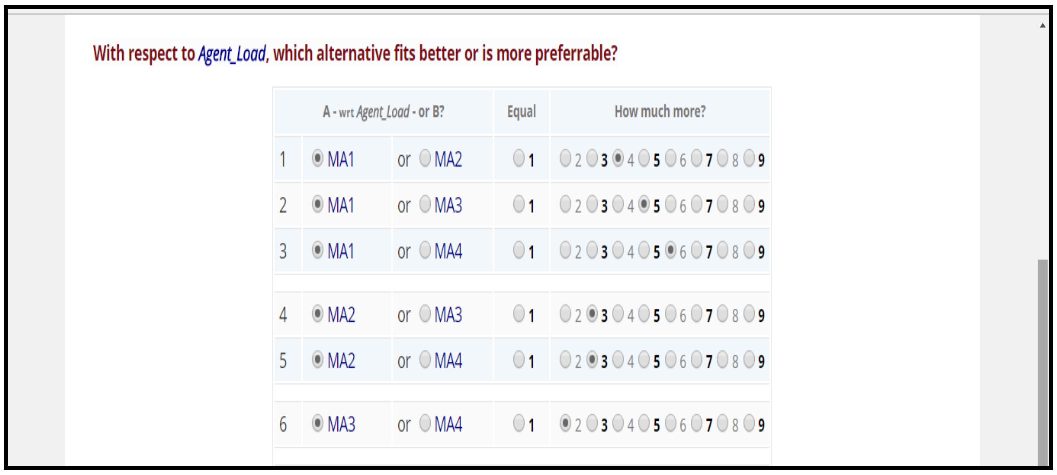


With respect to *My MA*, which criterion is more important, and how much more on a scale 1 to 9?

	A - wrt *My MA* - or B?			Equal	How much more?
1	Agent_Load	or	[✓ URL_Capacity]	1	2 3 4 5 6 [✓ 7] 8 9
2	Agent_Load	or	[✓ URL_PRank]	1	2 3 4 5 6 7 [✓ 8] 9
3	[✓ Agent_Load]	or	N/W_Bandwidth	1	[✓ 2] 3 4 5 6 7 8 9
4	Agent_Load	or	[✓ Loading_Rate]	1	2 3 [✓ 4] 5 6 7 8 9
5	[✓ Agent_Load]	or	Agent_Capacity	1	[✓ 2] 3 4 5 6 7 8 9
6	[✓ Agent_Load]	or	N/W_Latency	1	2 [✓ 3] 4 5 6 7 8 9
7	URL_Capacity	or	[✓ URL_PRank]	1	[✓ 2] 3 4 5 6 7 8 9
8	[✓ URL_Capacity]	or	N/W_Bandwidth	1	2 3 4 [✓ 5] 6 7 8 9
9	[✓ URL_Capacity]	or	Loading_Rate	1	2 3 [✓ 4] 5 6 7 8 9
10	[✓ URL_Capacity]	or	Agent_Capacity	1	2 [✓ 3] 4 5 6 7 8 9
11	[✓ URL_Capacity]	or	N/W_Latency	1	2 3 [✓ 4] 5 6 7 8 9
12	[✓ URL_PRank]	or	N/W_Bandwidth	1	2 3 4 5 6 [✓ 7] 8 9
13	[✓ URL_PRank]	or	Loading_Rate	1	2 3 [✓ 4] 5 6 7 8 9
14	[✓ URL_PRank]	or	Agent_Capacity	1	2 3 4 [✓ 5] 6 7 8 9
15	[✓ URL_PRank]	or	N/W_Latency	1	2 3 [✓ 4] 5 6 7 8 9
16	[✓ N/W_Bandwidth]	or	Loading_Rate	[✓ 1]	2 3 4 5 6 7 8 9
17	[✓ N/W_Bandwidth]	or	Agent_Capacity	[✓ 1]	2 3 4 5 6 7 8 9
18	[✓ N/W_Bandwidth]	or	N/W_Latency	[✓ 1]	2 3 4 5 6 7 8 9
19	[✓ Loading_Rate]	or	Agent_Capacity	1	[✓ 2] 3 4 5 6 7 8 9
20	[✓ Loading_Rate]	or	N/W_Latency	1	2 [✓ 3] 4 5 6 7 8 9

Figure A2.1: Criterions Comparison

Migrants comparison w.r.t Agent Load

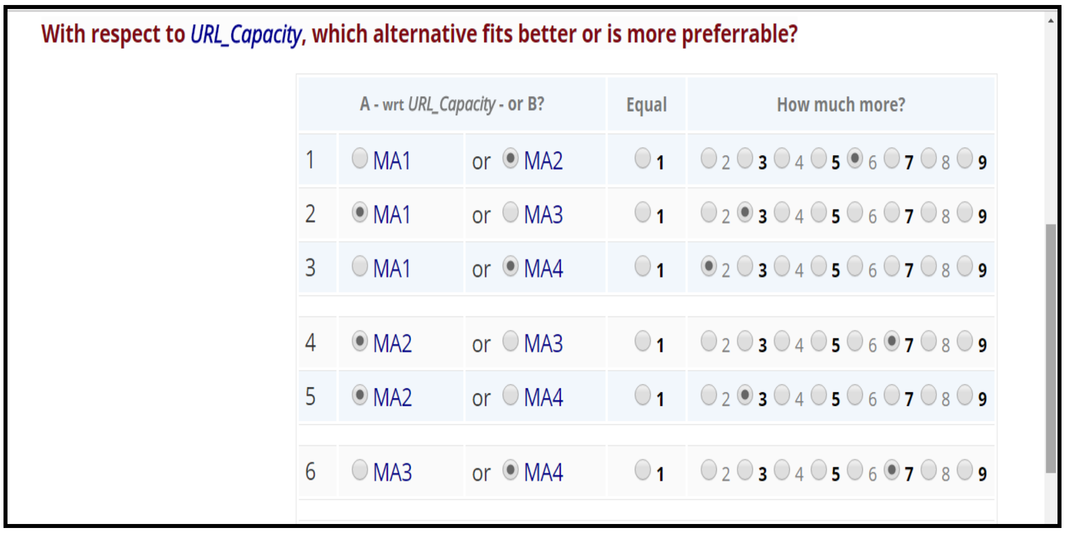


With respect to *Agent_Load*, which alternative fits better or is more preferrable?

	A - wrt *Agent_Load* - or B?			Equal	How much more?
1	[✓ MA1]	or	MA2	1	2 3 [✓ 4] 5 6 7 8 9
2	[✓ MA1]	or	MA3	1	2 3 4 [✓ 5] 6 7 8 9
3	[✓ MA1]	or	MA4	1	2 3 4 5 [✓ 6] 7 8 9
4	[✓ MA2]	or	MA3	1	2 [✓ 3] 4 5 6 7 8 9
5	[✓ MA2]	or	MA4	1	2 [✓ 3] 4 5 6 7 8 9
6	[✓ MA3]	or	MA4	1	[✓ 2] 3 4 5 6 7 8 9

Figure A2.2: Migrants Comparison

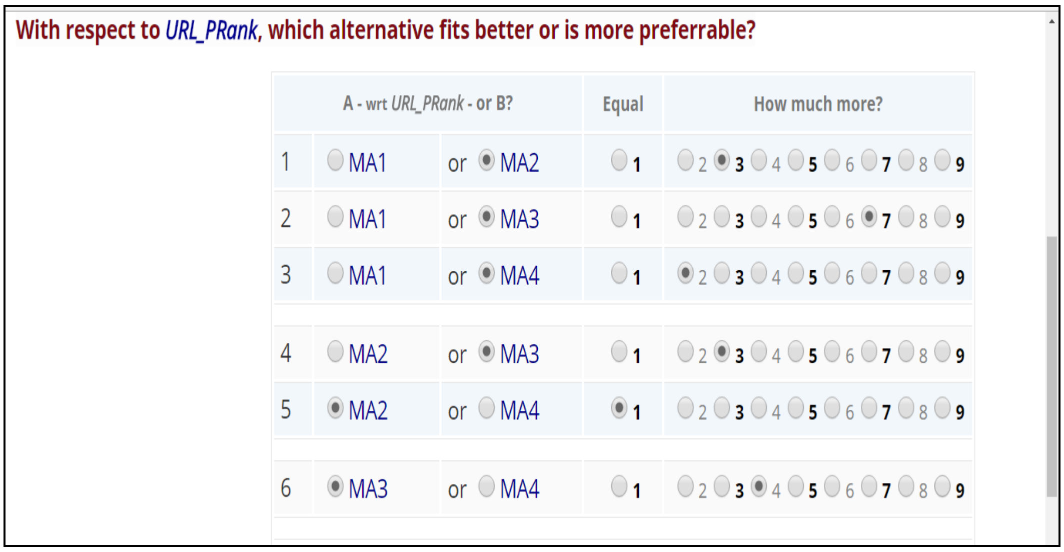
Migrants comparison w.r.t URL Capacity


With respect to *URL_Capacity*, which alternative fits better or is more preferrable?

	A - wrt *URL_Capacity* - or B?		Equal	How much more?
1	MA1	or [✓ MA2]	1	2 3 4 5 [✓ 6] 7 8 9
2	[✓ MA1]	or MA3	1	2 [✓ 3] 4 5 6 7 8 9
3	MA1	or [✓ MA4]	1	[✓ 2] 3 4 5 6 7 8 9
4	[✓ MA2]	or MA3	1	2 3 4 5 6 [✓ 7] 8 9
5	[✓ MA2]	or MA4	1	2 [✓ 3] 4 5 6 7 8 9
6	MA3	or [✓ MA4]	1	2 3 4 5 6 [✓ 7] 8 9

Figure A2.3: Alternative Comparison

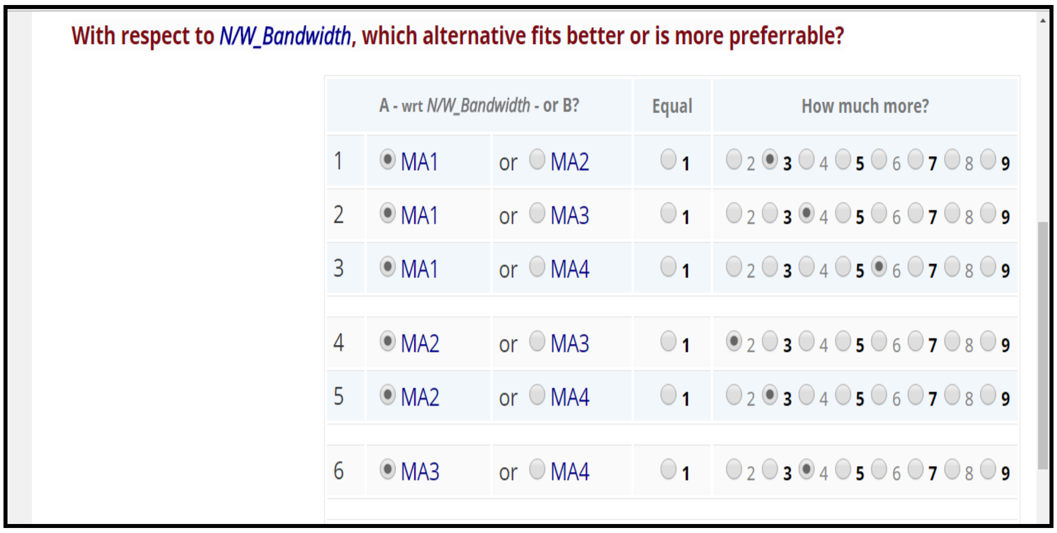
Migrants comparison w.r.t URL Parent Rank


With respect to *URL_PRank*, which alternative fits better or is more preferrable?

	A - wrt *URL_PRank* - or B?		Equal	How much more?
1	MA1	or [✓ MA2]	1	2 [✓ 3] 4 5 6 7 8 9
2	MA1	or [✓ MA3]	1	2 3 4 5 6 [✓ 7] 8 9
3	MA1	or [✓ MA4]	1	[✓ 2] 3 4 5 6 7 8 9
4	MA2	or [✓ MA3]	1	2 [✓ 3] 4 5 6 7 8 9
5	[✓ MA2]	or MA4	[✓ 1]	2 3 4 5 6 7 8 9
6	[✓ MA3]	or MA4	1	2 3 [✓ 4] 5 6 7 8 9

Figure A2.4: Alternative Comparison

Migrants comparison w.r.t N/W Bandwidth

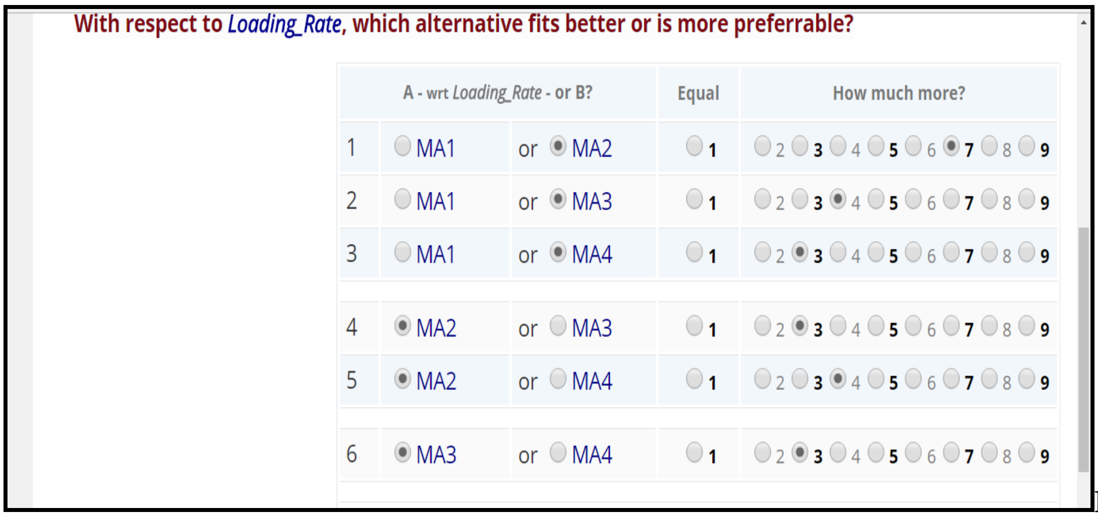


With respect to *N/W_Bandwidth*, which alternative fits better or is more preferrable?

A - wrt *N/W_Bandwidth* - or B?				Equal	How much more?
1	[✓ MA1]	or	MA2	1	2 [✓ 3] 4 5 6 7 8 9
2	[✓ MA1]	or	MA3	1	2 3 [✓ 4] 5 6 7 8 9
3	[✓ MA1]	or	MA4	1	2 3 4 5 [✓ 6] 7 8 9
4	[✓ MA2]	or	MA3	1	[✓ 2] 3 4 5 6 7 8 9
5	[✓ MA2]	or	MA4	1	2 [✓ 3] 4 5 6 7 8 9
6	[✓ MA3]	or	MA4	1	2 3 [✓ 4] 5 6 7 8 9

Figure A2.5: Alternative Comparison

Migrants comparison w.r.t Loading Rate



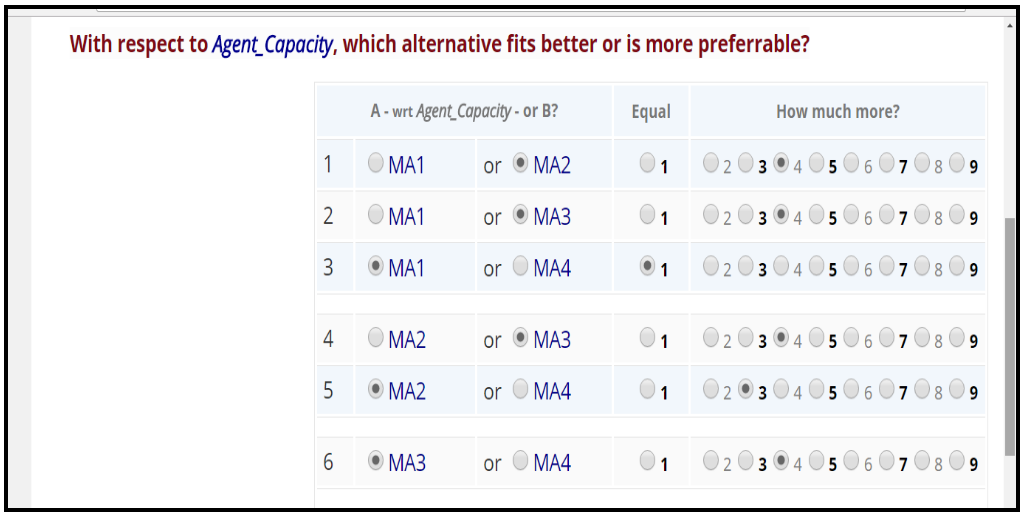
With respect to *Loading_Rate*, which alternative fits better or is more preferrable?

A - wrt *Loading_Rate* - or B?				Equal	How much more?
1	MA1	or	[✓ MA2]	1	2 3 4 5 6 [✓ 7] 8 9
2	MA1	or	[✓ MA3]	1	2 3 [✓ 4] 5 6 7 8 9
3	MA1	or	[✓ MA4]	1	2 [✓ 3] 4 5 6 7 8 9
4	[✓ MA2]	or	MA3	1	2 [✓ 3] 4 5 6 7 8 9
5	[✓ MA2]	or	MA4	1	2 3 [✓ 4] 5 6 7 8 9
6	[✓ MA3]	or	MA4	1	2 [✓ 3] 4 5 6 7 8 9

Fi

gure A2.6: Alternative Comparison

Migrants comparison w.r.t Agent Capacity

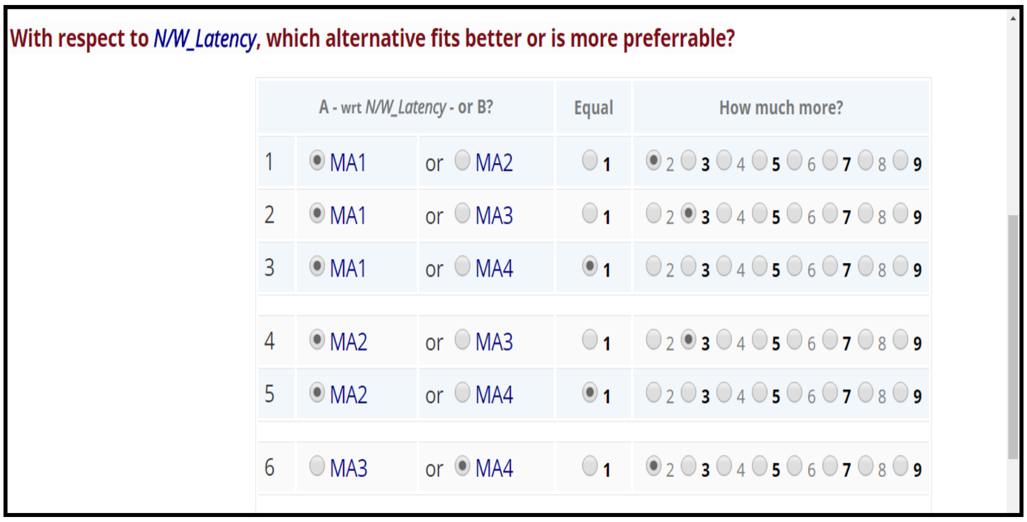


With respect to *Agent_Capacity*, which alternative fits better or is more preferrable?

	A - wrt *Agent_Capacity* - or B?		Equal	How much more?
1	MA1	or [✓ MA2]	1	2 3 [✓ 4] 5 6 7 8 9
2	MA1	or [✓ MA3]	1	2 3 [✓ 4] 5 6 7 8 9
3	[✓ MA1]	or MA4	[✓ 1]	2 3 4 5 6 7 8 9
4	MA2	or [✓ MA3]	1	2 3 [✓ 4] 5 6 7 8 9
5	[✓ MA2]	or MA4	1	2 [✓ 3] 4 5 6 7 8 9
6	[✓ MA3]	or MA4	1	2 3 [✓ 4] 5 6 7 8 9

Figure A2.7: Alternative Comparison

Migrants comparison w.r.t Network Latency



With respect to *N/W_Latency*, which alternative fits better or is more preferrable?

	A - wrt *N/W_Latency* - or B?		Equal	How much more?
1	[✓ MA1]	or MA2	1	[✓ 2] 3 4 5 6 7 8 9
2	[✓ MA1]	or MA3	1	2 [✓ 3] 4 5 6 7 8 9
3	[✓ MA1]	or MA4	[✓ 1]	2 3 4 5 6 7 8 9
4	[✓ MA2]	or MA3	1	2 [✓ 3] 4 5 6 7 8 9
5	[✓ MA2]	or MA4	[✓ 1]	2 3 4 5 6 7 8 9
6	MA3	or [✓ MA4]	1	[✓ 2] 3 4 5 6 7 8 9

Figure A2.8: N/W latency

Overall Priority of each Criterion

	CRITERIONS	Priority	Rank
1	Agent_Load	7.4%	4
2	URL_Capacity	27.3%	2
3	URL_Prank	37.3%	1
4	N/W_Bandwidth	5.6%	6
5	Loading_Rate	11.6%	3
6	Agent_Capacity	6.2%	5
7	N/W_Latency	4.7%	7

Figure A2.9: Overall Priority

Consolidated Weights and Rank of each Migrant

Participants	MA1	MA2	MA3	MA4
Weights	16.4%	35.0%	31.5%	17.1%
Rank	4	1	2	3

Figure A2.10: Migrant's Rank

Graphical Representation of Alternatives Weights

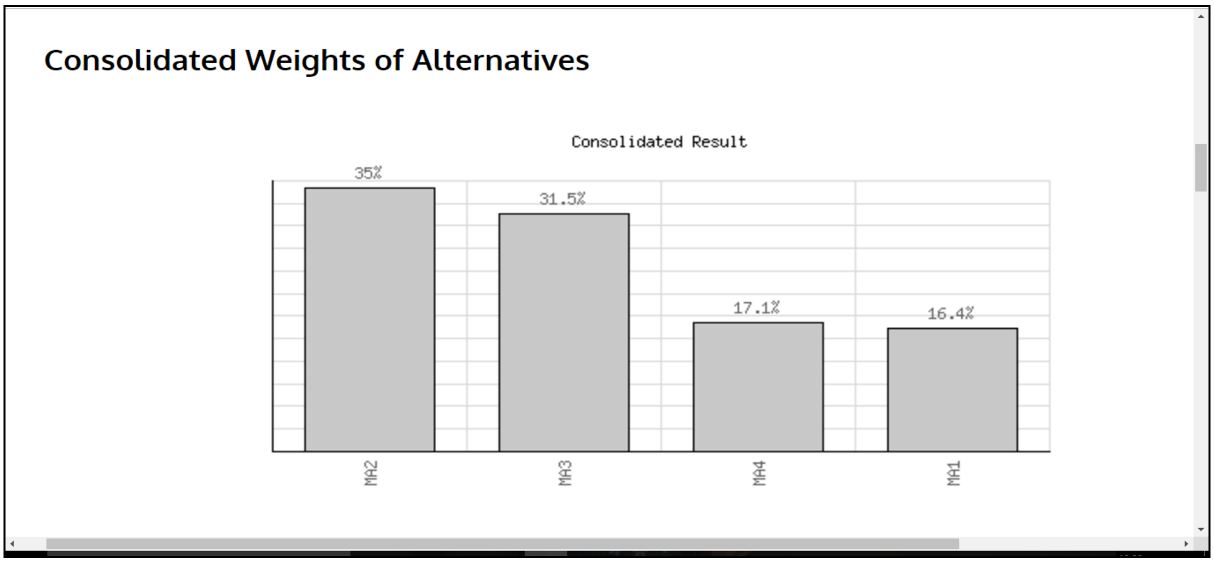


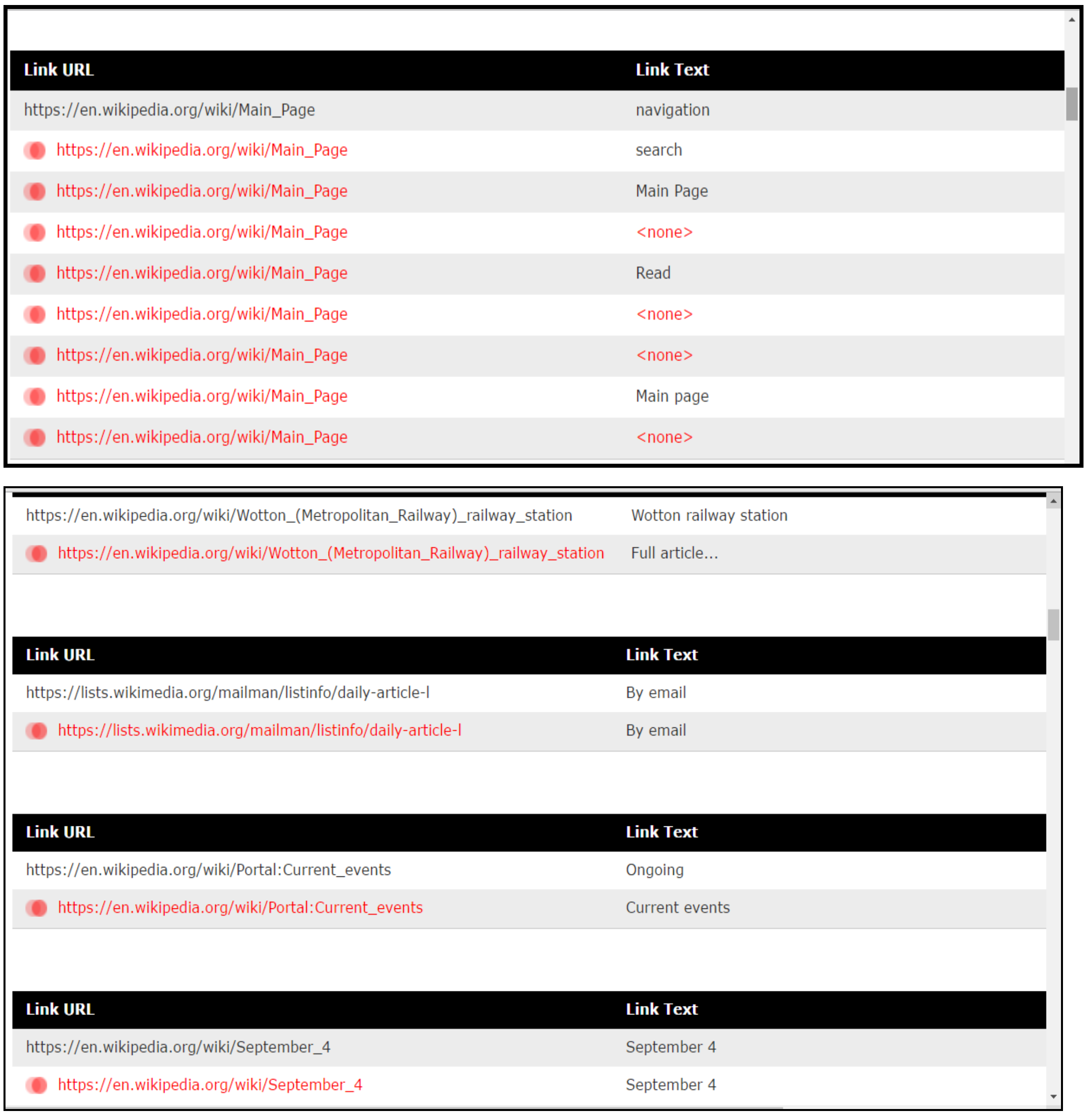
Figure A2.11: Consolidated Weights

APPENDIX III

Snapshots of duplicate URLs and elimination of such URLs

For example, Input URL: - https://www.wikipedia.org

List of Duplicate URLs



Link URL	Link Text
https://en.wikipedia.org/wiki/Main_Page	navigation
● https://en.wikipedia.org/wiki/Main_Page	search
● https://en.wikipedia.org/wiki/Main_Page	Main Page
● https://en.wikipedia.org/wiki/Main_Page	<none>
● https://en.wikipedia.org/wiki/Main_Page	Read
● https://en.wikipedia.org/wiki/Main_Page	<none>
● https://en.wikipedia.org/wiki/Main_Page	<none>
● https://en.wikipedia.org/wiki/Main_Page	Main page
● https://en.wikipedia.org/wiki/Main_Page	<none>

https://en.wikipedia.org/wiki/Wotton_(Metropolitan_Railway)_railway_station	Wotton railway station
● https://en.wikipedia.org/wiki/Wotton_(Metropolitan_Railway)_railway_station	Full article...

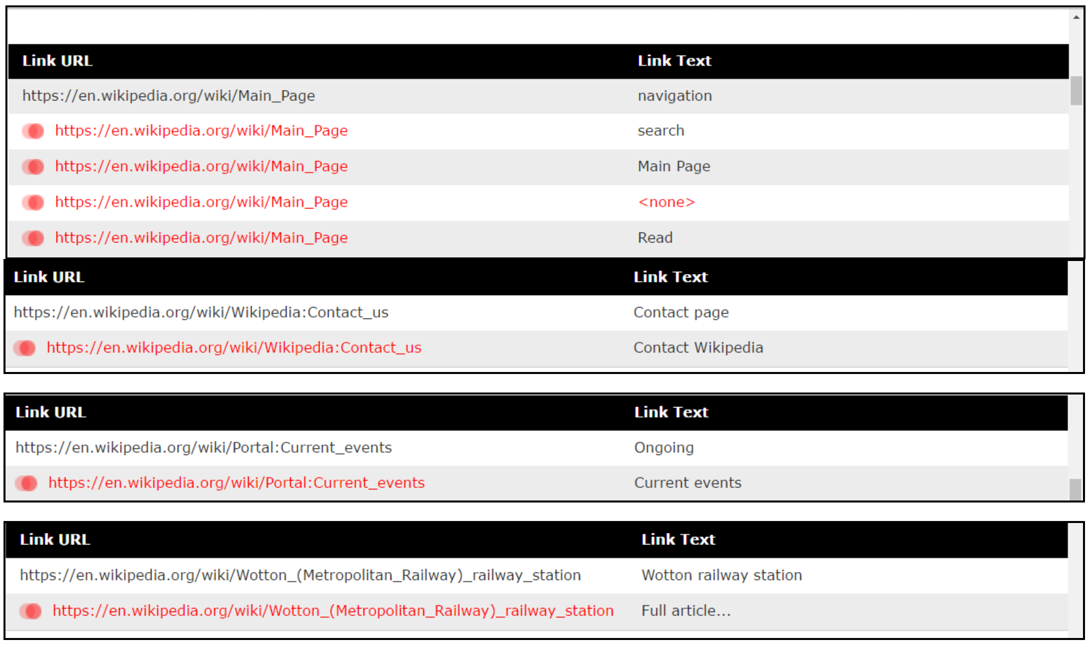
Link URL	Link Text
https://lists.wikimedia.org/mailman/listinfo/daily-article-l	By email
● https://lists.wikimedia.org/mailman/listinfo/daily-article-l	By email

Link URL	Link Text
https://en.wikipedia.org/wiki/Portal:Current_events	Ongoing
● https://en.wikipedia.org/wiki/Portal:Current_events	Current events

Link URL	Link Text
https://en.wikipedia.org/wiki/September_4	September 4
● https://en.wikipedia.org/wiki/September_4	September 4

Figure A3.1: Snapshot of duplicate URLs

List of Different Link_name pointed to same page



Link URL	Link Text
https://en.wikipedia.org/wiki/Main_Page	navigation
● https://en.wikipedia.org/wiki/Main_Page	search
● https://en.wikipedia.org/wiki/Main_Page	Main Page
● https://en.wikipedia.org/wiki/Main_Page	<none>
● https://en.wikipedia.org/wiki/Main_Page	Read

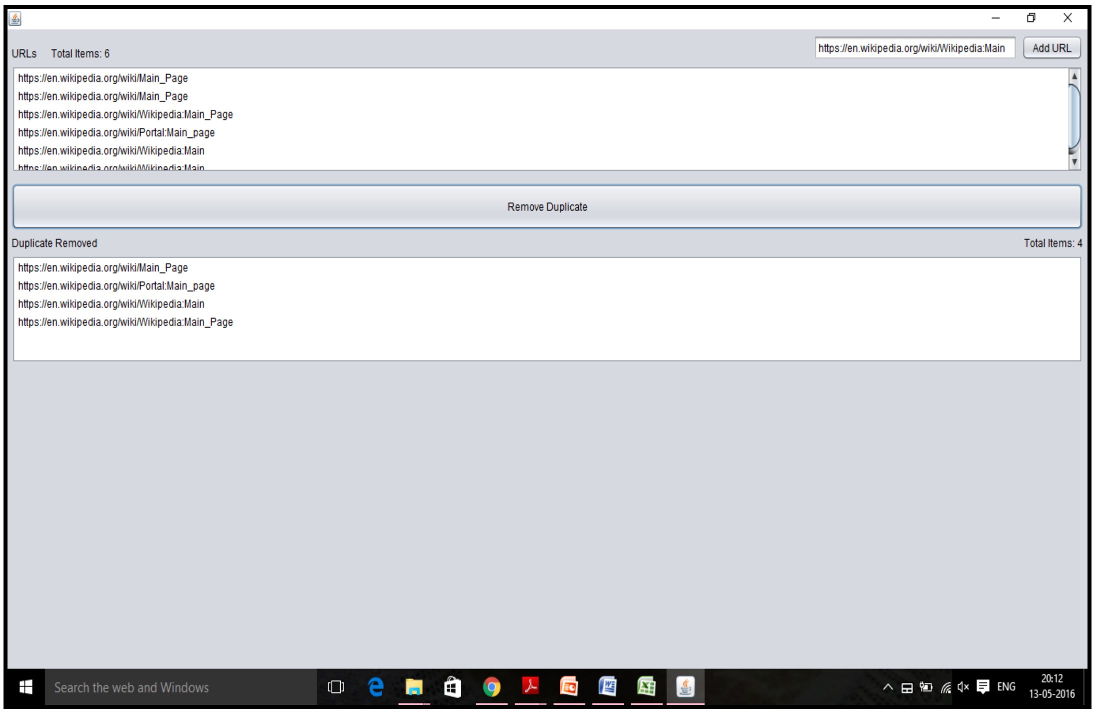
Link URL	Link Text
https://en.wikipedia.org/wiki/Wikipedia:Contact_us	Contact page
● https://en.wikipedia.org/wiki/Wikipedia:Contact_us	Contact Wikipedia

Link URL	Link Text
https://en.wikipedia.org/wiki/Portal:Current_events	Ongoing
● https://en.wikipedia.org/wiki/Portal:Current_events	Current events

Link URL	Link Text
https://en.wikipedia.org/wiki/Wotton_(Metropolitan_Railway)_railway_station	Wotton railway station
● https://en.wikipedia.org/wiki/Wotton_(Metropolitan_Railway)_railway_station	Full article...

Figure A3.2: Snapshots of Different Link_Name point to same page

Eliminate Duplicate Entries



```
URLs    Total Items: 6                         https://en.wikipedia.org/wiki/Wikipedia:Main   Add URL
https://en.wikipedia.org/wiki/Main_Page
https://en.wikipedia.org/wiki/Main_Page
https://en.wikipedia.org/wiki/Wikipedia:Main_Page
https://en.wikipedia.org/wiki/Portal:Main_page
https://en.wikipedia.org/wiki/Wikipedia:Main
https://en.wikipedia.org/wiki/Wikipedia:Main

                         Remove Duplicate

Duplicate Removed                                                            Total Items: 4
https://en.wikipedia.org/wiki/Main_Page
https://en.wikipedia.org/wiki/Portal:Main_page
https://en.wikipedia.org/wiki/Wikipedia:Main
https://en.wikipedia.org/wiki/Wikipedia:Main_Page
```

Figure A3.3: Snapshot of Eliminator Module

APPENDIX IV

Snapshots of Users' Browsing Behaviour Based Ranking

Web Browser

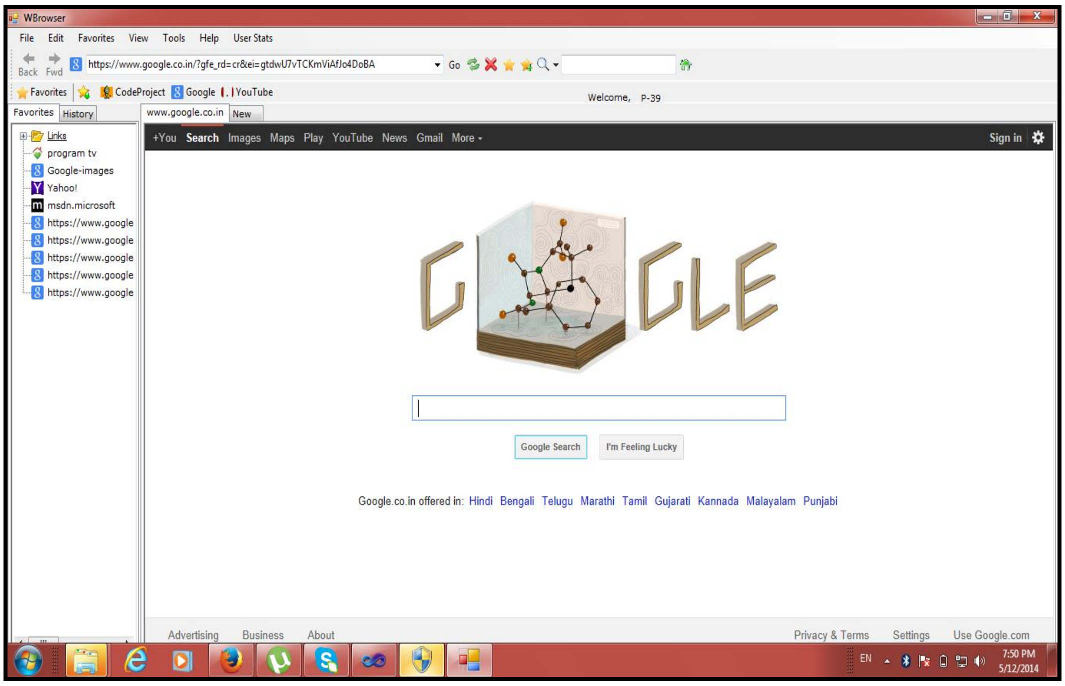
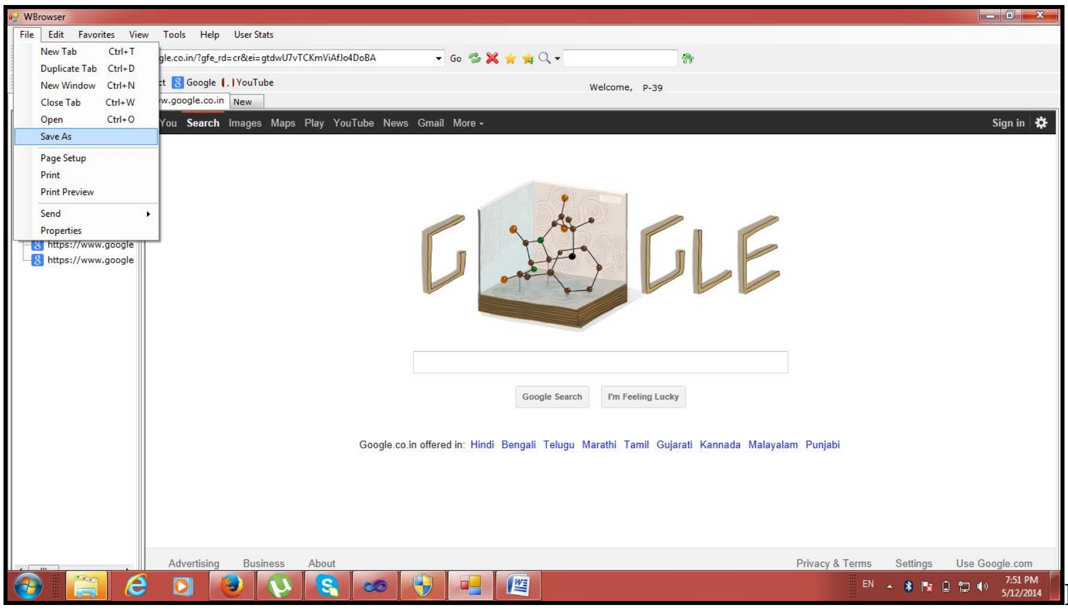

WBrowser
File Edit Favorites View Tools Help User Stats
Back Fwd
https://www.google.co.in/?gfe_rd=cr&ei=gtdwU7vTCKmViAfJo4DoBA
Go
Favorites CodeProject Google YouTube
Welcome, P-39
Favorites History
www.google.co.in New
Links
program tv
Google-images
Yahoo!
msdn.microsoft
https://www.google
https://www.google
https://www.google
https://www.google
https://www.google
+You Search Images Maps Play YouTube News Gmail More
Sign in
Google Search I'm Feeling Lucky
Google.co.in offered in: Hindi Bengali Telugu Marathi Tamil Gujarati Kannada Malayalam Punjabi
Advertising Business About
Privacy & Terms Settings Use Google.com
EN 7:50 PM 5/12/2014

Figure A4.1: Web Browser Interface

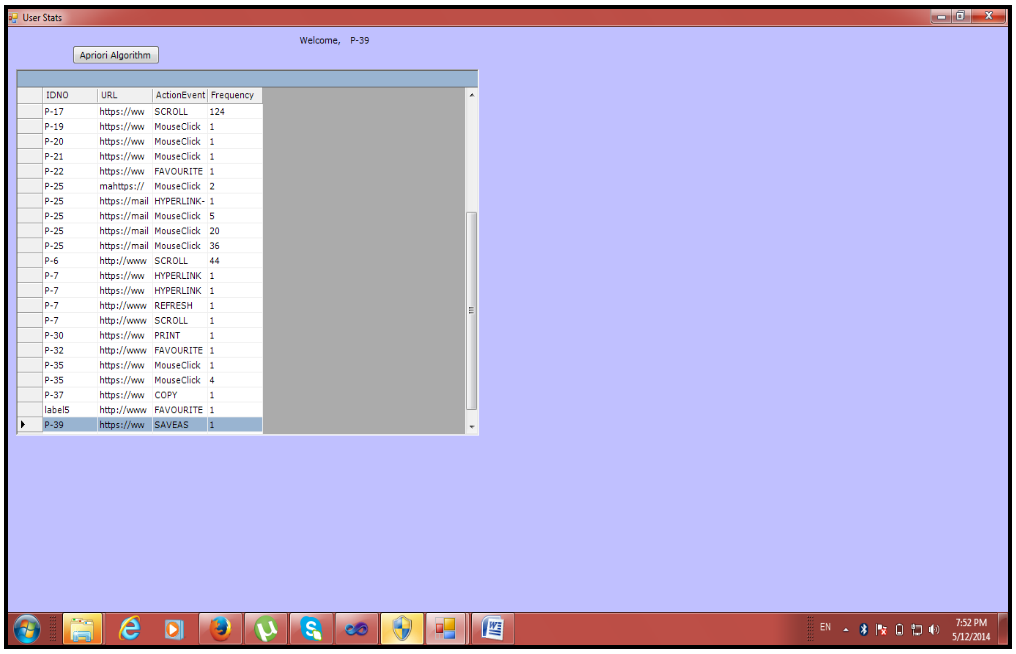
Actions performed by users


WBrowser
File Edit Favorites View Tools Help User Stats
New Tab Ctrl+T
Duplicate Tab Ctrl+D
New Window Ctrl+N
Close Tab Ctrl+W
Open Ctrl+O
Save As
Page Setup
Print
Print Preview
Send
Properties
https://www.google
https://www.google
gle.co.in/?gfe_rd=cr&ei=gtdwU7vTCKmViAfJo4DoBA
Go
Google YouTube
Welcome, P-39
w.google.co.in New
You Search Images Maps Play YouTube News Gmail More
Sign in
Google Search I'm Feeling Lucky
Google.co.in offered in: Hindi Bengali Telugu Marathi Tamil Gujarati Kannada Malayalam Punjabi
Advertising Business About
Privacy & Terms Settings Use Google.com
EN 7:51 PM 5/12/2014

Figf

Figure A4.2: Save As action

Save As Action



IDNO	URL	ActionEvent	Frequency
P-17	https://ww	SCROLL	124
P-19	https://ww	MouseClick	1
P-20	https://ww	MouseClick	1
P-21	https://ww	MouseClick	1
P-22	https://ww	FAVOURITE	1
P-25	mahttps://	MouseClick	2
P-25	https://mail	HYPERLINK-	1
P-25	https://mail	MouseClick	5
P-25	https://mail	MouseClick	20
P-25	https://mail	MouseClick	36
P-6	http://www	SCROLL	44
P-7	https://ww	HYPERLINK	1
P-7	https://ww	HYPERLINK	1
P-7	http://www	REFRESH	1
P-7	http://www	SCROLL	1
P-30	https://ww	PRINT	1
P-32	http://www	FAVOURITE	1
P-35	https://ww	MouseClick	1
P-35	https://ww	MouseClick	4
P-37	https://ww	COPY	1
label5	http://www	FAVOURITE	1
P-39	https://ww	SAVEAS	1

Figure A4.3: Save As action in Database

Copy Action

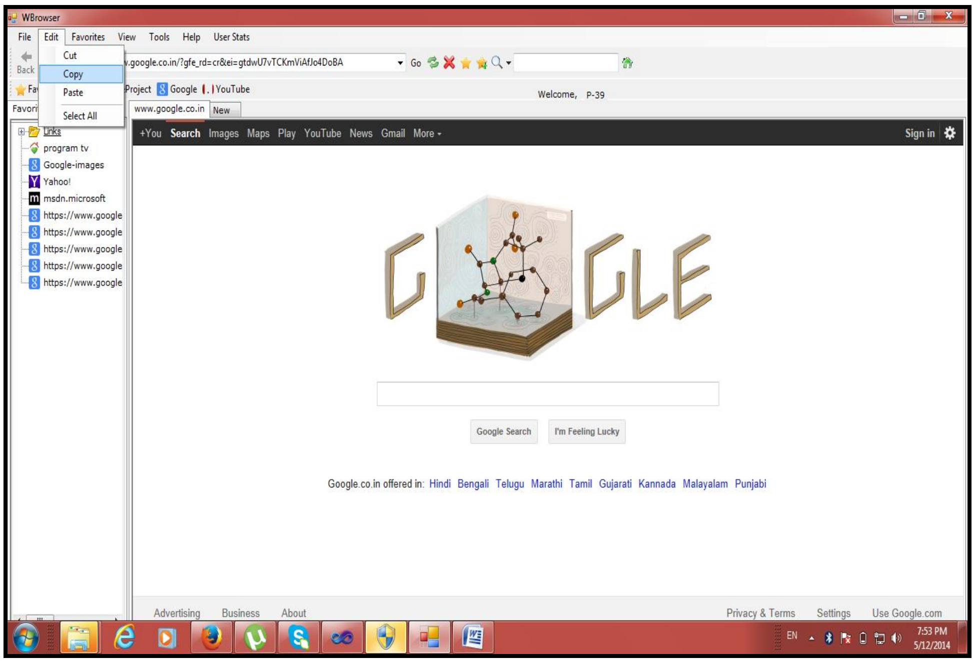
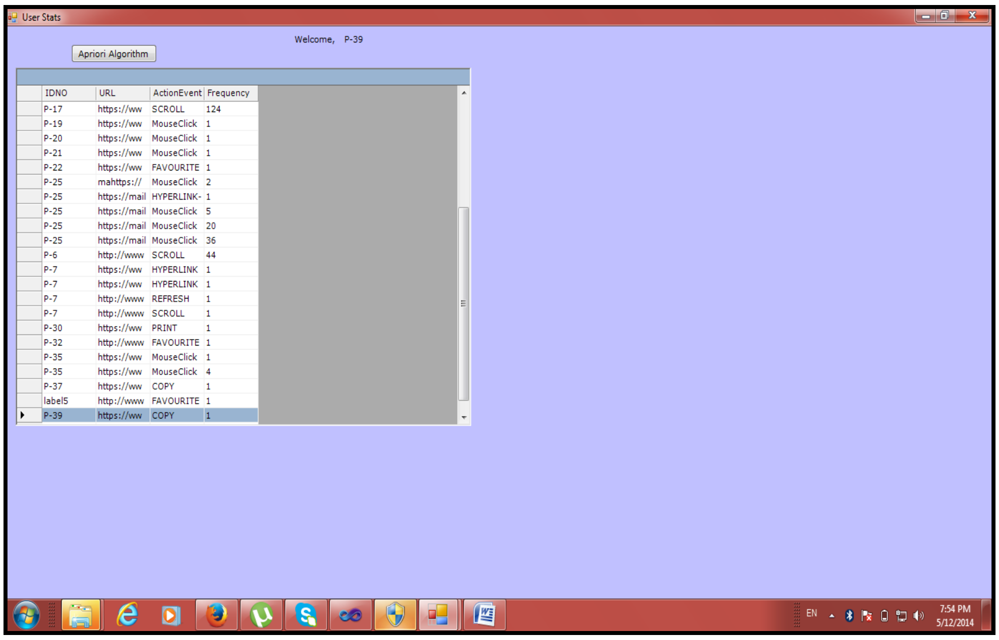


Figure A4.4: Copy action

Copy Action Stored in Database



	IDNO	URL	ActionEvent	Frequency
	P-17	https://ww	SCROLL	124
	P-19	https://ww	MouseClick	1
	P-20	https://ww	MouseClick	1
	P-21	https://ww	MouseClick	1
	P-22	https://ww	FAVOURITE	1
	P-25	mahttps://	MouseClick	2
	P-25	https://mail	HYPERLINK-	1
	P-25	https://mail	MouseClick	5
	P-25	https://mail	MouseClick	20
	P-25	https://mail	MouseClick	36
	P-6	http://www	SCROLL	44
	P-7	https://ww	HYPERLINK	1
	P-7	https://ww	HYPERLINK	1
	P-7	http://www	REFRESH	1
	P-7	http://www	SCROLL	1
	P-30	https://ww	PRINT	1
	P-32	http://www	FAVOURITE	1
	P-35	https://ww	MouseClick	1
	P-35	https://ww	MouseClick	4
	P-37	https://ww	COPY	1
	label5	http://www	FAVOURITE	1
▶	P-39	https://ww	COPY	1

Figure A4.5: Copy action in Database

Add to Favourite Action

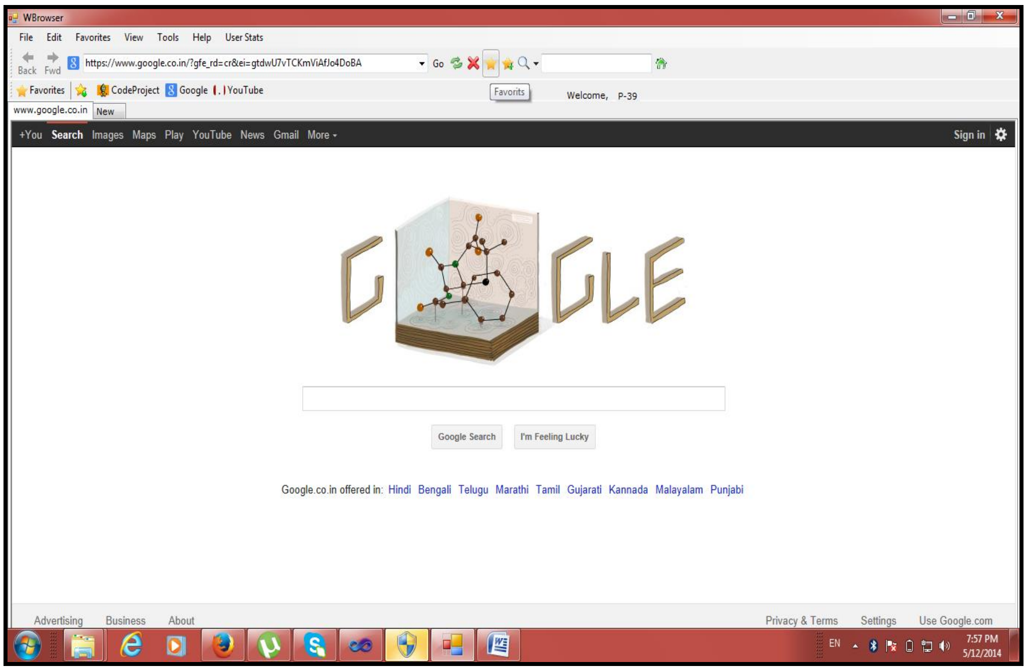
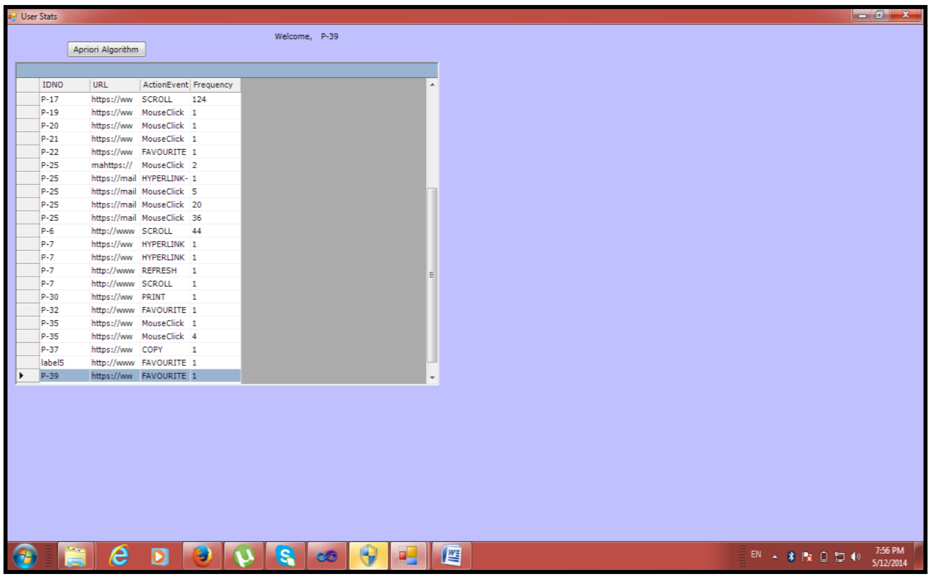

WBrowser
File Edit Favorites View Tools Help User Stats
Back Fwd
https://www.google.co.in/?gfe_rd=cr&ei=gtdwU7vTCKmViAfJo4DoBA
Go
Favorites CodeProject Google YouTube
Favorits
Welcome, P-39
www.google.co.in New
+You Search Images Maps Play YouTube News Gmail More
Sign in
Google Search I'm Feeling Lucky
Google.co.in offered in: Hindi Bengali Telugu Marathi Tamil Gujarati Kannada Malayalam Punjabi
Advertising Business About
Privacy & Terms Settings Use Google.com

Figure A4.6: Add to Favourite action

Favourite Action stored in Database



IDNO	URL	ActionEvent	Frequency
P-17	https://ww	SCROLL	124
P-19	https://ww	MouseClick	1
P-20	https://ww	MouseClick	1
P-21	https://ww	MouseClick	1
P-22	https://ww	FAVOURITE	1
P-25	mahttps://	MouseClick	2
P-25	https://mail	HYPERLINK-	1
P-25	https://mail	MouseClick	5
P-25	https://mail	MouseClick	20
P-25	https://mail	MouseClick	36
P-6	http://www	SCROLL	44
P-7	https://ww	HYPERLINK	1
P-7	https://ww	HYPERLINK	1
P-7	http://www	REFRESH	1
P-7	http://www	SCROLL	1
P-30	https://ww	PRINT	1
P-32	http://www	FAVOURITE	1
P-35	https://ww	MouseClick	1
P-35	https://ww	MouseClick	4
P-37	https://ww	COPY	1
label5	http://www	FAVOURITE	1
P-39	https://ww	FAVOURITE	1

Figure A4.7: Add to Favorites action in database

Print Action

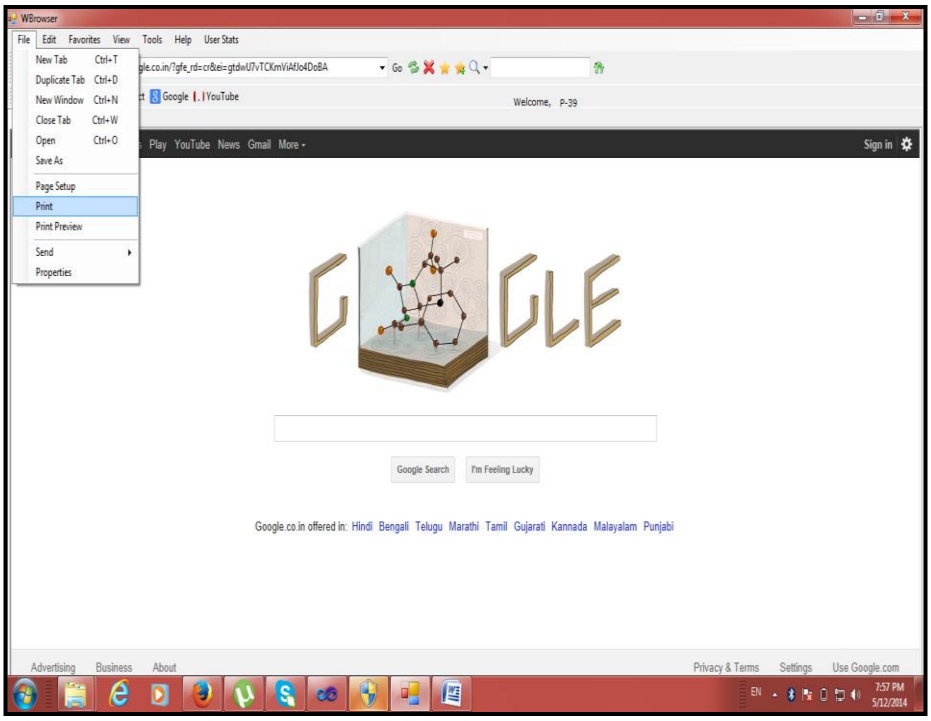


Figure A4.8: Print action

Print Action with printer window

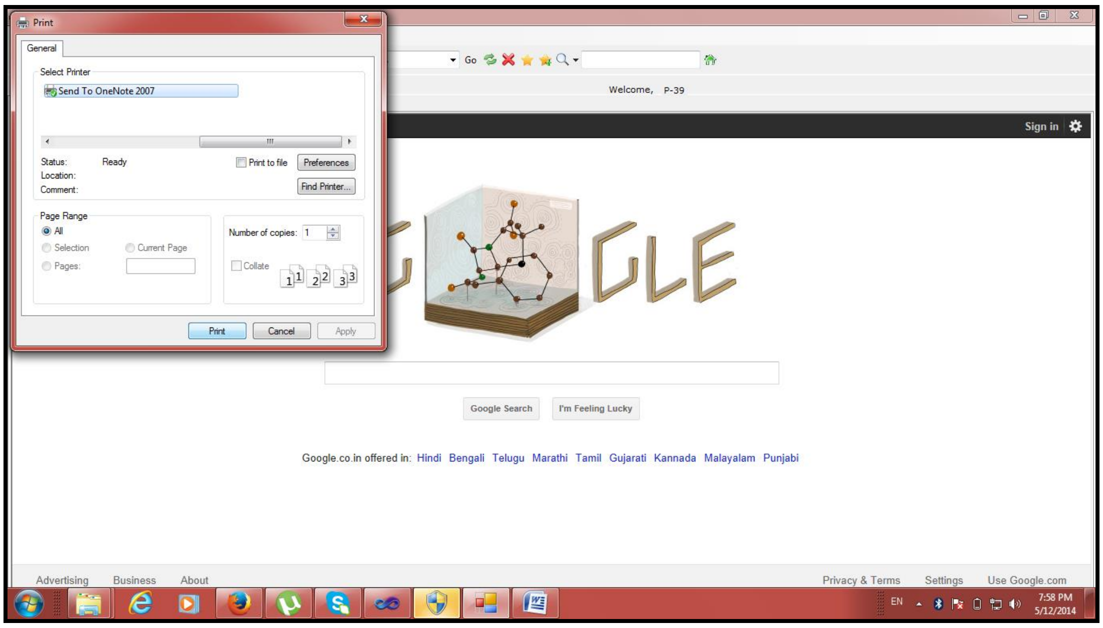
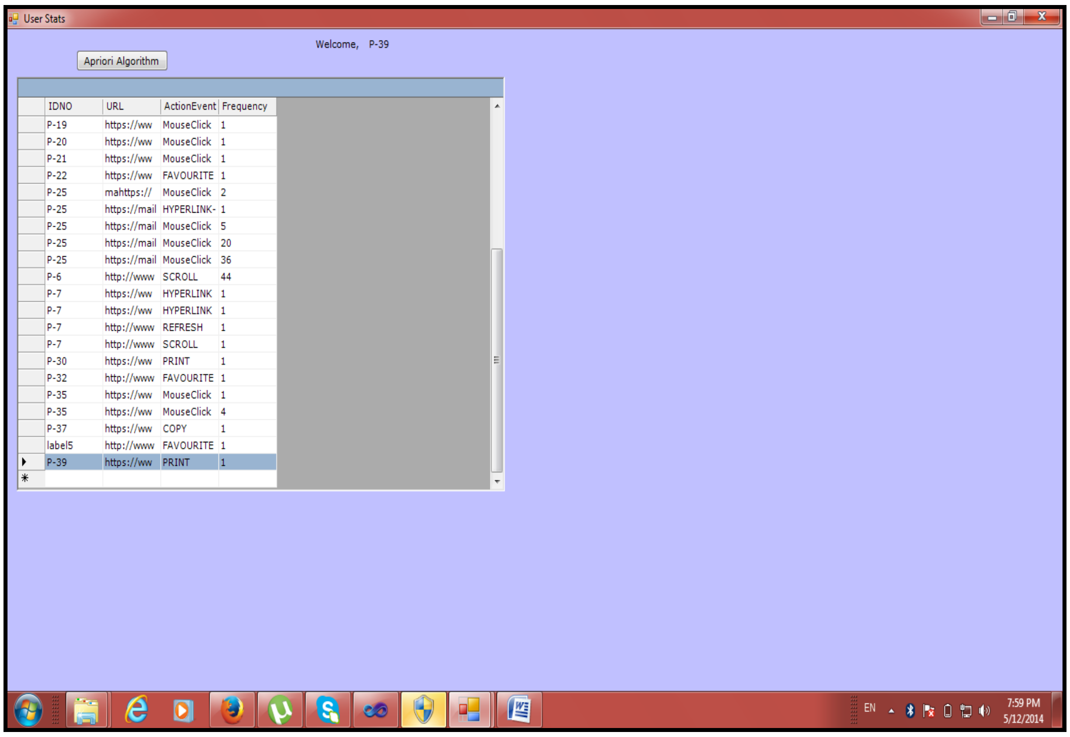


Figure A4.9: Print action

Print Action in Database



	IDNO	URL	ActionEvent	Frequency
	P-19	https://ww	MouseClick	1
	P-20	https://ww	MouseClick	1
	P-21	https://ww	MouseClick	1
	P-22	https://ww	FAVOURITE	1
	P-25	mahttps://	MouseClick	2
	P-25	https://mail	HYPERLINK-	1
	P-25	https://mail	MouseClick	5
	P-25	https://mail	MouseClick	20
	P-25	https://mail	MouseClick	36
	P-6	http://www	SCROLL	44
	P-7	https://ww	HYPERLINK	1
	P-7	https://ww	HYPERLINK	1
	P-7	http://www	REFRESH	1
	P-7	http://www	SCROLL	1
	P-30	https://ww	PRINT	1
	P-32	http://www	FAVOURITE	1
	P-35	https://ww	MouseClick	1
	P-35	https://ww	MouseClick	4
	P-37	https://ww	COPY	1
	label5	http://www	FAVOURITE	1
▶	P-39	https://ww	PRINT	1
*				

Figure A4.10: Print action in user statistics

Refresh Action

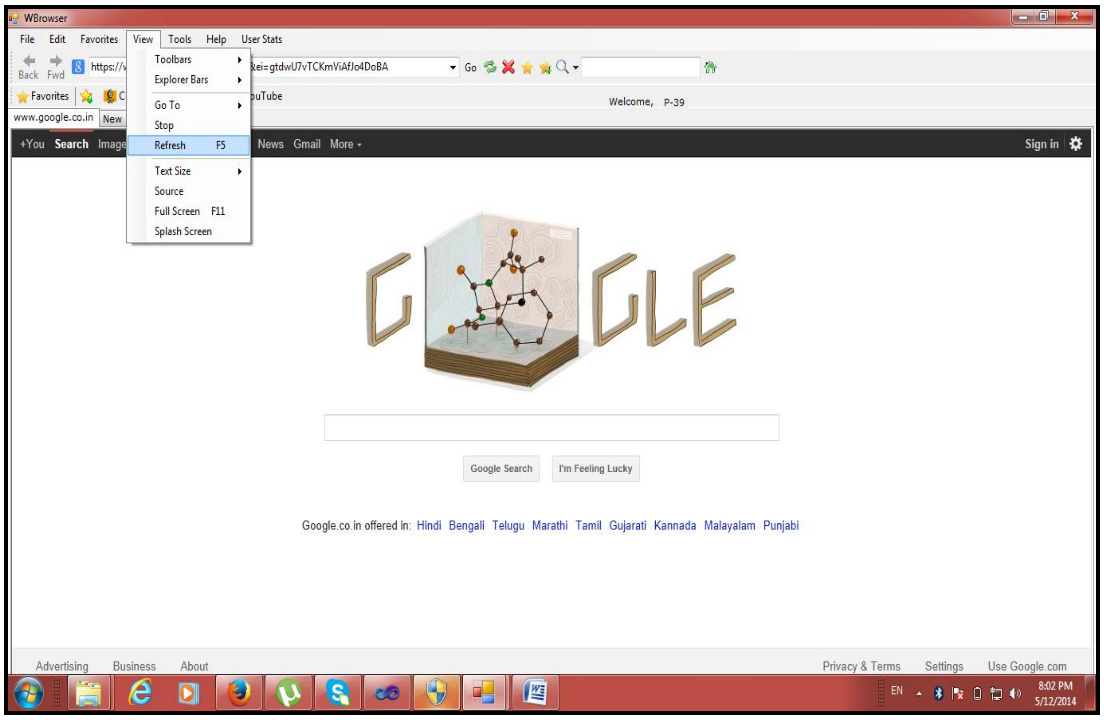
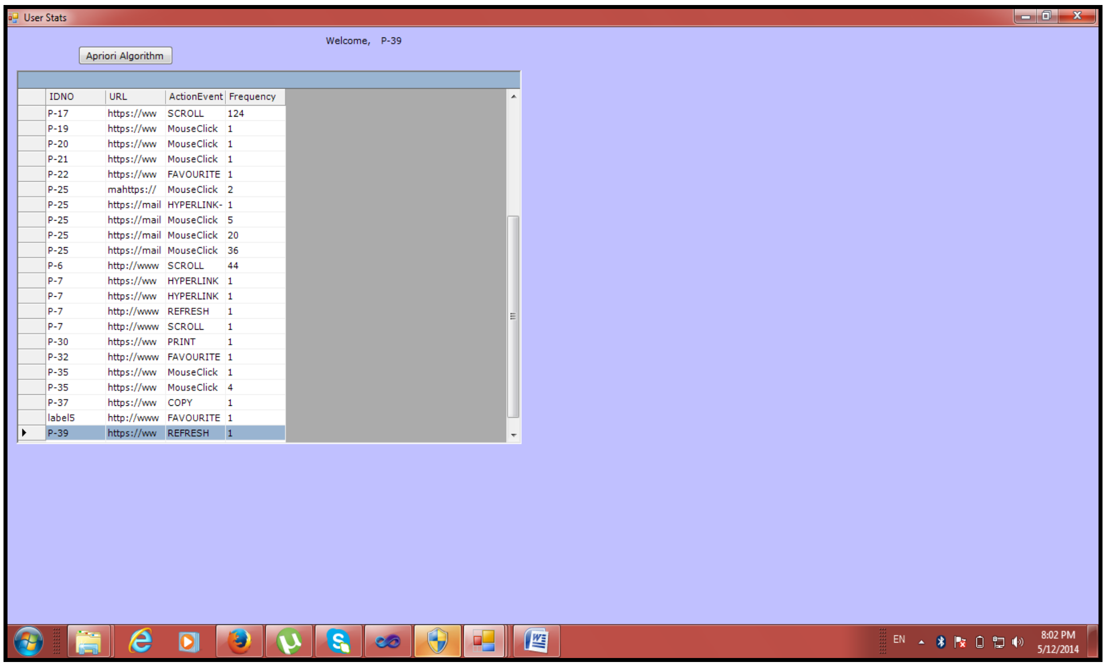


Figure A4.11: Refresh action

Refresh Action in Database



IDNO	URL	ActionEvent	Frequency
P-17	https://ww	SCROLL	124
P-19	https://ww	MouseClick	1
P-20	https://ww	MouseClick	1
P-21	https://ww	MouseClick	1
P-22	https://ww	FAVOURITE	1
P-25	mahttps://	MouseClick	2
P-25	https://mail	HYPERLINK-	1
P-25	https://mail	MouseClick	5
P-25	https://mail	MouseClick	20
P-25	https://mail	MouseClick	36
P-6	http://www	SCROLL	44
P-7	https://ww	HYPERLINK	1
P-7	https://ww	HYPERLINK	1
P-7	http://www	REFRESH	1
P-7	http://www	SCROLL	1
P-30	https://ww	PRINT	1
P-32	http://www	FAVOURITE	1
P-35	https://ww	MouseClick	1
P-35	https://ww	MouseClick	4
P-37	https://ww	COPY	1
label5	http://www	FAVOURITE	1
P-39	https://ww	REFRESH	1

Figure A4.12: Refresh action in user statistics

Stop Action

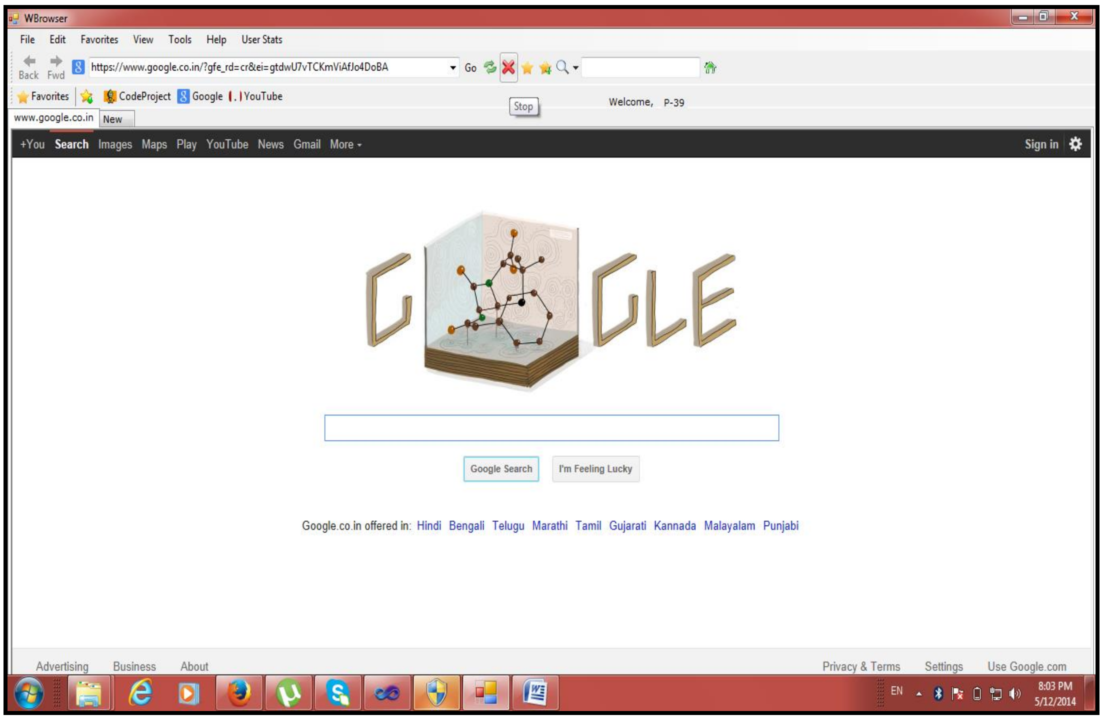
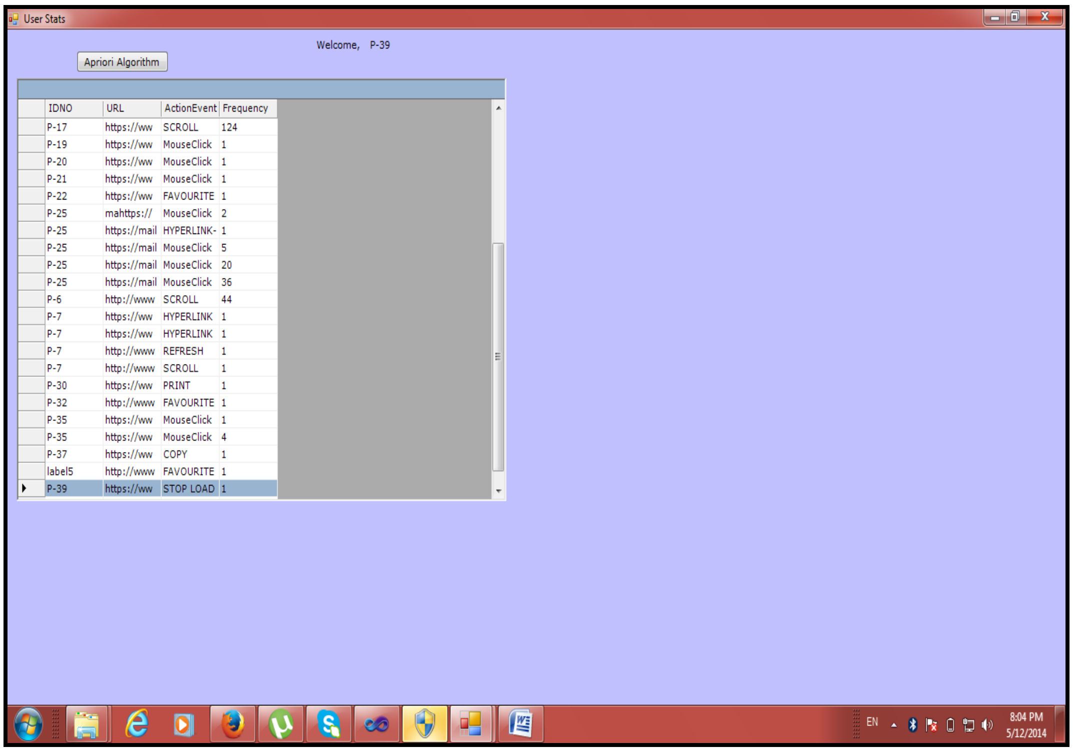

WBrowser
File Edit Favorites View Tools Help User Stats
Back Fwd
https://www.google.co.in/?gfe_rd=cr&ei=gtdwU7vTCKmViAfJo4DoBA
Go
Favorites CodeProject Google YouTube
Stop
Welcome, P-39
www.google.co.in New
+You Search Images Maps Play YouTube News Gmail More
Sign in
Google Search
I'm Feeling Lucky
Google.co.in offered in: Hindi Bengali Telugu Marathi Tamil Gujarati Kannada Malayalam Punjabi
Advertising Business About
Privacy & Terms Settings Use Google.com
8:03 PM 5/12/2014

Figure A4.13: Stop Loading action

Stop Action in Database


User Stats

Welcome, P-39

Apriori Algorithm

IDNO	URL	ActionEvent	Frequency
P-17	https://ww	SCROLL	124
P-19	https://ww	MouseClick	1
P-20	https://ww	MouseClick	1
P-21	https://ww	MouseClick	1
P-22	https://ww	FAVOURITE	1
P-25	mahttps://	MouseClick	2
P-25	https://mail	HYPERLINK-	1
P-25	https://mail	MouseClick	5
P-25	https://mail	MouseClick	20
P-25	https://mail	MouseClick	36
P-6	http://www	SCROLL	44
P-7	https://ww	HYPERLINK	1
P-7	https://ww	HYPERLINK	1
P-7	http://www	REFRESH	1
P-7	http://www	SCROLL	1
P-30	https://ww	PRINT	1
P-32	http://www	FAVOURITE	1
P-35	https://ww	MouseClick	1
P-35	https://ww	MouseClick	4
P-37	https://ww	COPY	1
label5	http://www	FAVOURITE	1
P-39	https://ww	STOP LOAD	1

8:04 PM 5/12/2014

Figure A4.14: Stop loading action in user statistics

Back Action

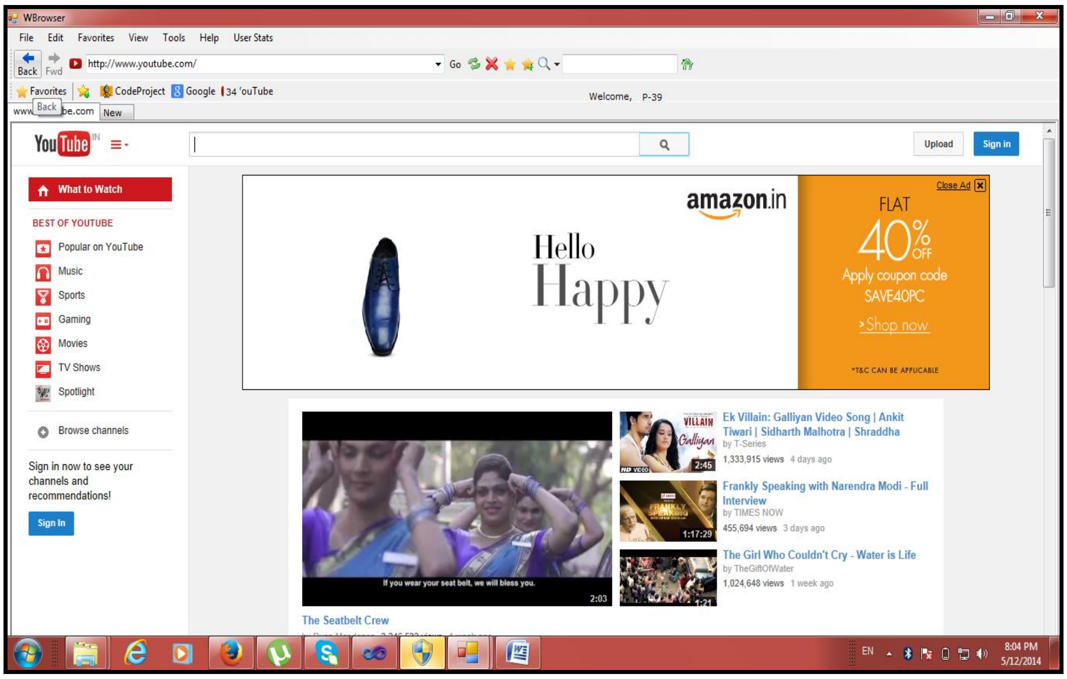
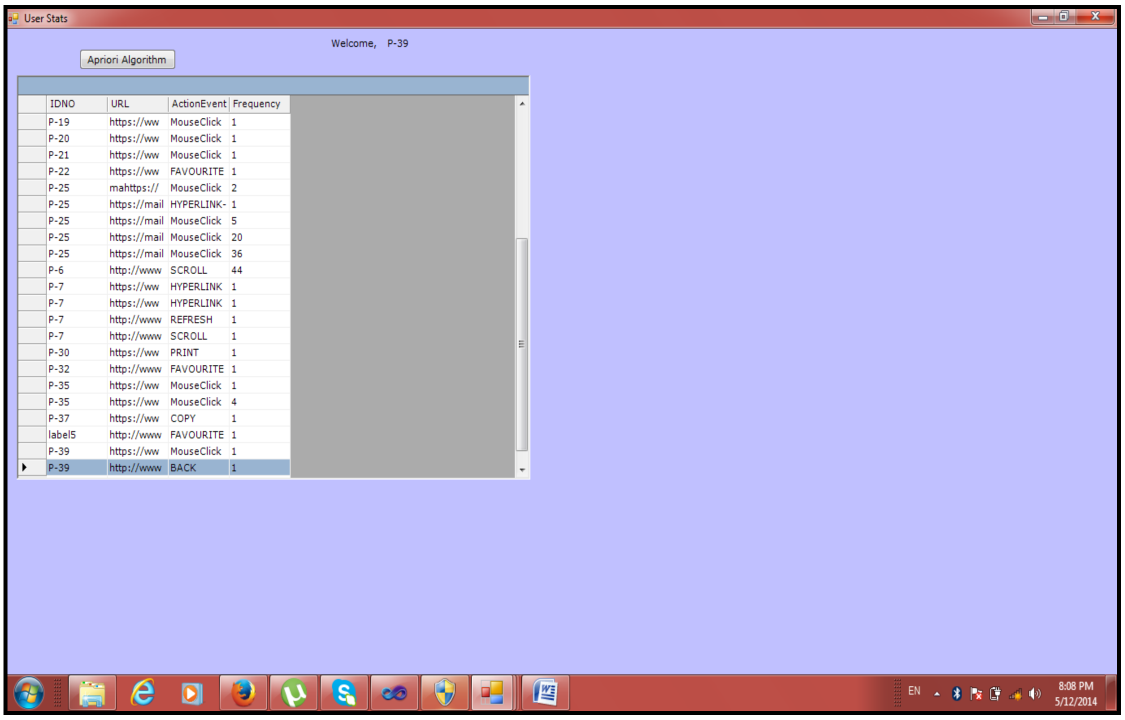


Figure A4.15: Back action

Back Action in Database



IDNO	URL	ActionEvent	Frequency
P-19	https://ww	MouseClick	1
P-20	https://ww	MouseClick	1
P-21	https://ww	MouseClick	1
P-22	https://ww	FAVOURITE	1
P-25	mahttps://	MouseClick	2
P-25	https://mail	HYPERLINK-	1
P-25	https://mail	MouseClick	5
P-25	https://mail	MouseClick	20
P-25	https://mail	MouseClick	36
P-6	http://www	SCROLL	44
P-7	https://ww	HYPERLINK	1
P-7	https://ww	HYPERLINK	1
P-7	http://www	REFRESH	1
P-7	http://www	SCROLL	1
P-30	https://ww	PRINT	1
P-32	http://www	FAVOURITE	1
P-35	https://ww	MouseClick	1
P-35	https://ww	MouseClick	4
P-37	https://ww	COPY	1
label5	http://www	FAVOURITE	1
P-39	https://ww	MouseClick	1
P-39	http://www	BACK	1

Figure A4.16: Back action in user statistics

Forward Action

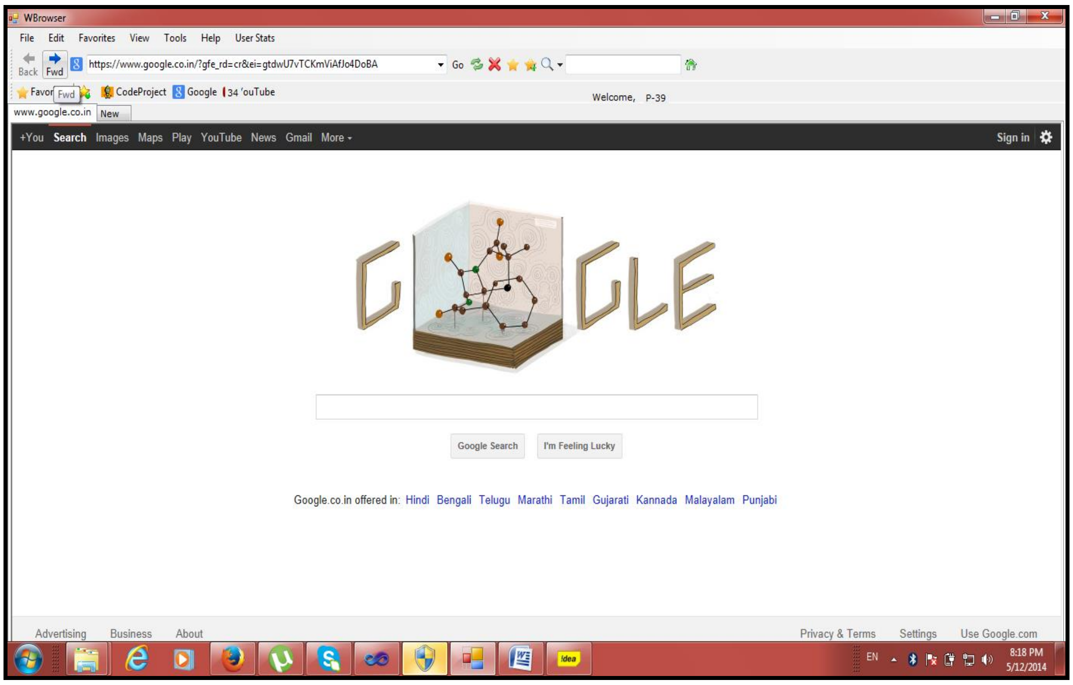
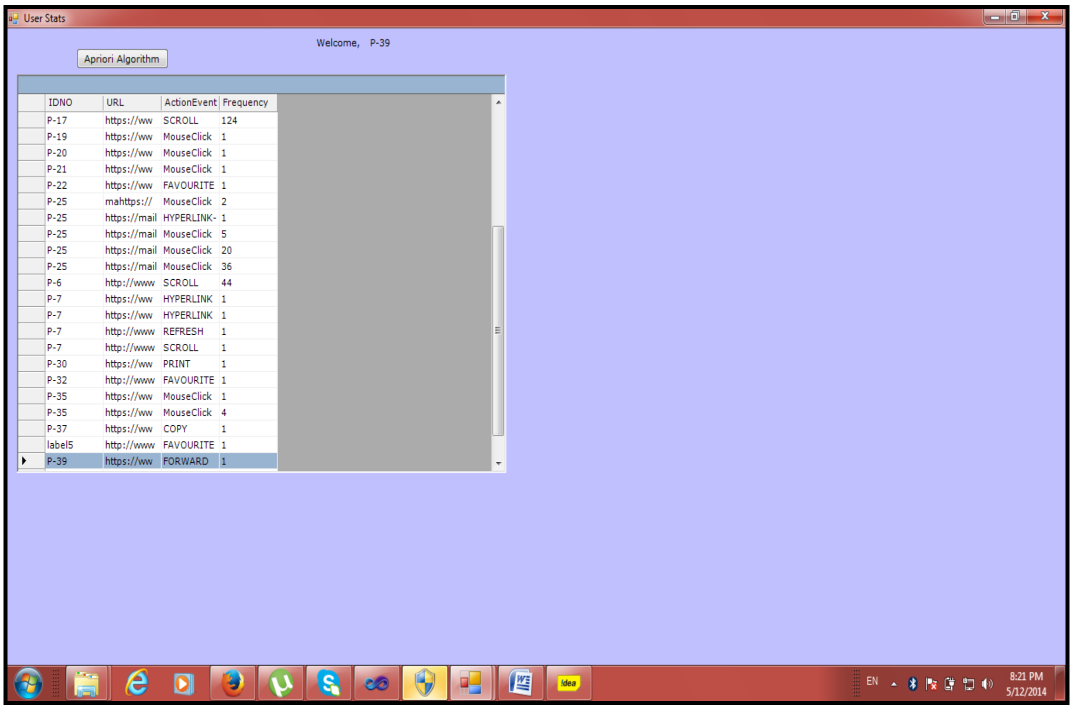

WBrowser
File
Edit
Favorites
View
Tools
Help
User Stats
Back
Fwd
https://www.google.co.in/?gfe_rd=cr&ei=gtdwU7vTCKmViAfJo4DoBA
Go
Favor
CodeProject
Google
YouTube
Welcome, P-39
www.google.co.in
New
+You Search Images Maps Play YouTube News Gmail More
Sign in
Google Search
I'm Feeling Lucky
Google.co.in offered in: Hindi Bengali Telugu Marathi Tamil Gujarati Kannada Malayalam Punjabi
Advertising Business About
Privacy & Terms Settings Use Google.com
8:18 PM
5/12/2014

Figure A4.17: Forward action

Forward Action in Database



IDNO	URL	ActionEvent	Frequency
P-17	https://ww	SCROLL	124
P-19	https://ww	MouseClick	1
P-20	https://ww	MouseClick	1
P-21	https://ww	MouseClick	1
P-22	https://ww	FAVOURITE	1
P-25	mahttps://	MouseClick	2
P-25	https://mail	HYPERLINK-	1
P-25	https://mail	MouseClick	5
P-25	https://mail	MouseClick	20
P-25	https://mail	MouseClick	36
P-6	http://www	SCROLL	44
P-7	https://ww	HYPERLINK	1
P-7	https://ww	HYPERLINK	1
P-7	http://www	REFRESH	1
P-7	http://www	SCROLL	1
P-30	https://ww	PRINT	1
P-32	http://www	FAVOURITE	1
P-35	https://ww	MouseClick	1
P-35	https://ww	MouseClick	4
P-37	https://ww	COPY	1
label5	http://www	FAVOURITE	1
P-39	https://ww	FORWARD	1

Figure A4.18: Forward action in user statistics

Scroll Action

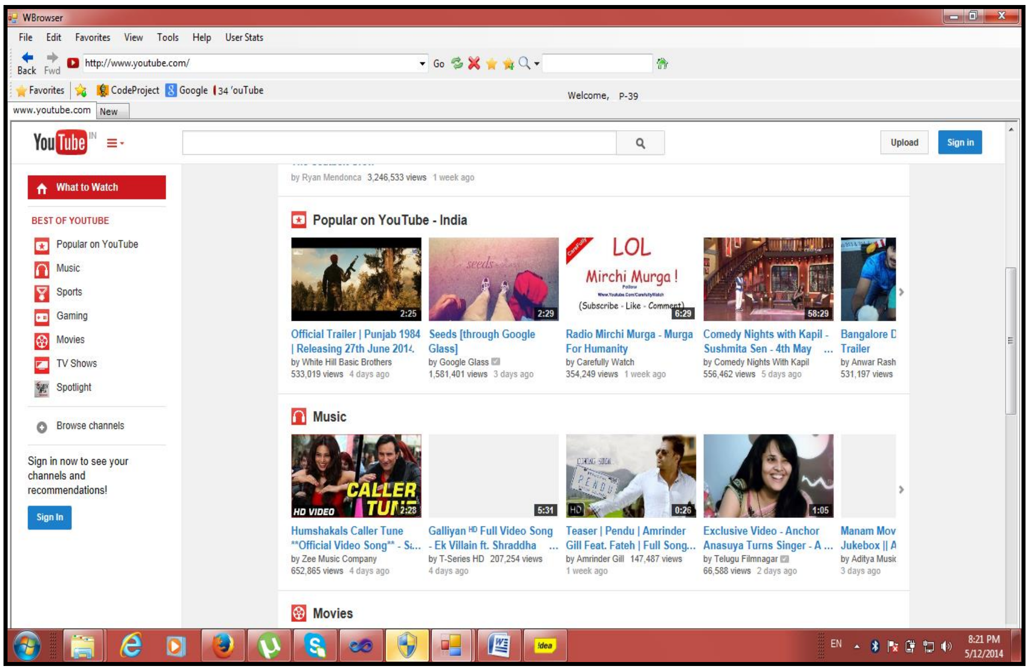
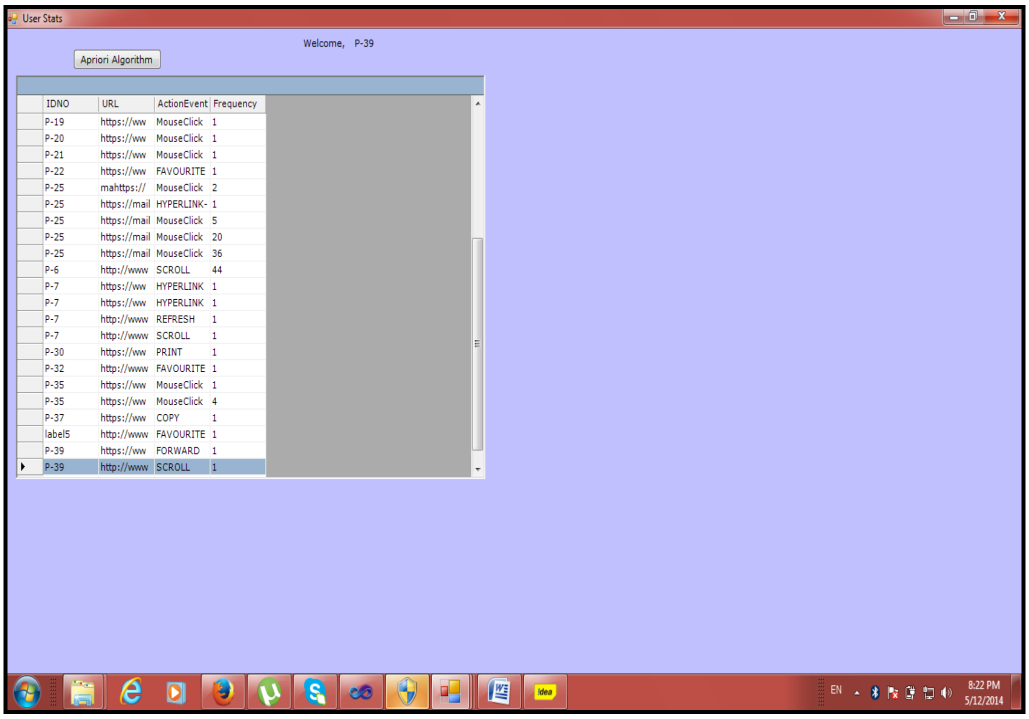

WBrowser
File Edit Favorites View Tools Help User Stats
http://www.youtube.com/
Welcome, P-39
What to Watch
Popular on YouTube - India
Music
Movies

Figure A4.19: Scroll action

Scroll Action in Database



IDNO	URL	ActionEvent	Frequency
P-19	https://ww	MouseClick	1
P-20	https://ww	MouseClick	1
P-21	https://ww	MouseClick	1
P-22	https://ww	FAVOURITE	1
P-25	mahttps://	MouseClick	2
P-25	https://mail	HYPERLINK-	1
P-25	https://mail	MouseClick	5
P-25	https://mail	MouseClick	20
P-25	https://mail	MouseClick	36
P-6	http://www	SCROLL	44
P-7	https://ww	HYPERLINK	1
P-7	https://ww	HYPERLINK	1
P-7	http://www	REFRESH	1
P-7	http://www	SCROLL	1
P-30	https://ww	PRINT	1
P-32	http://www	FAVOURITE	1
P-35	https://ww	MouseClick	1
P-35	https://ww	MouseClick	4
P-37	https://ww	COPY	1
label5	http://www	FAVOURITE	1
P-39	https://ww	FORWARD	1
P-39	http://www	SCROLL	1

Figure A4.20: Scroll action in user statistics

HyperLink Action

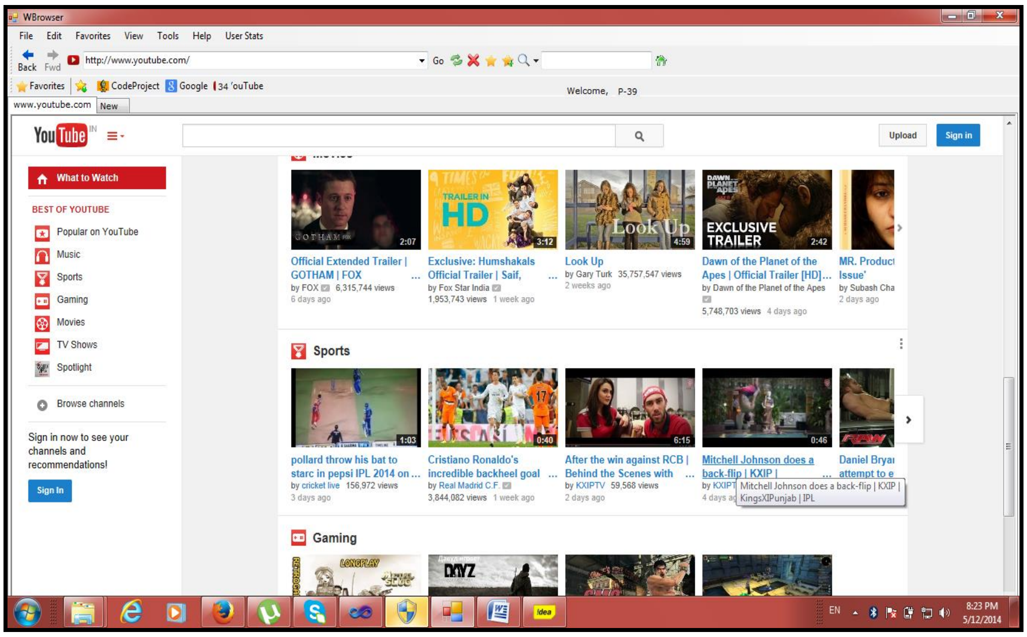
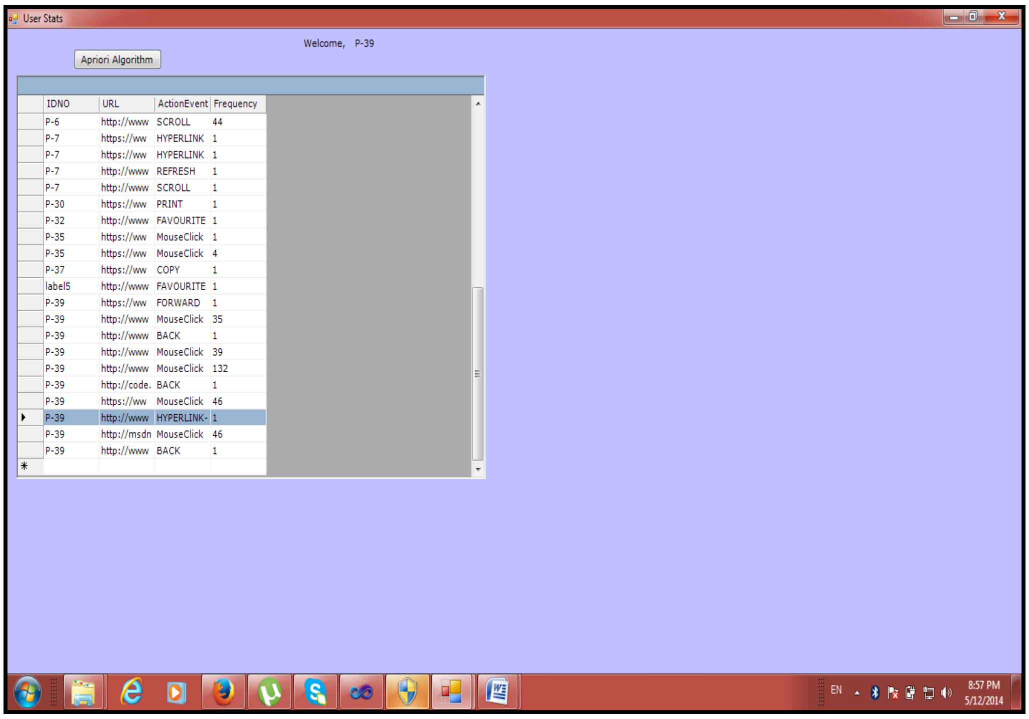


Figure A4.21: HyperLink action

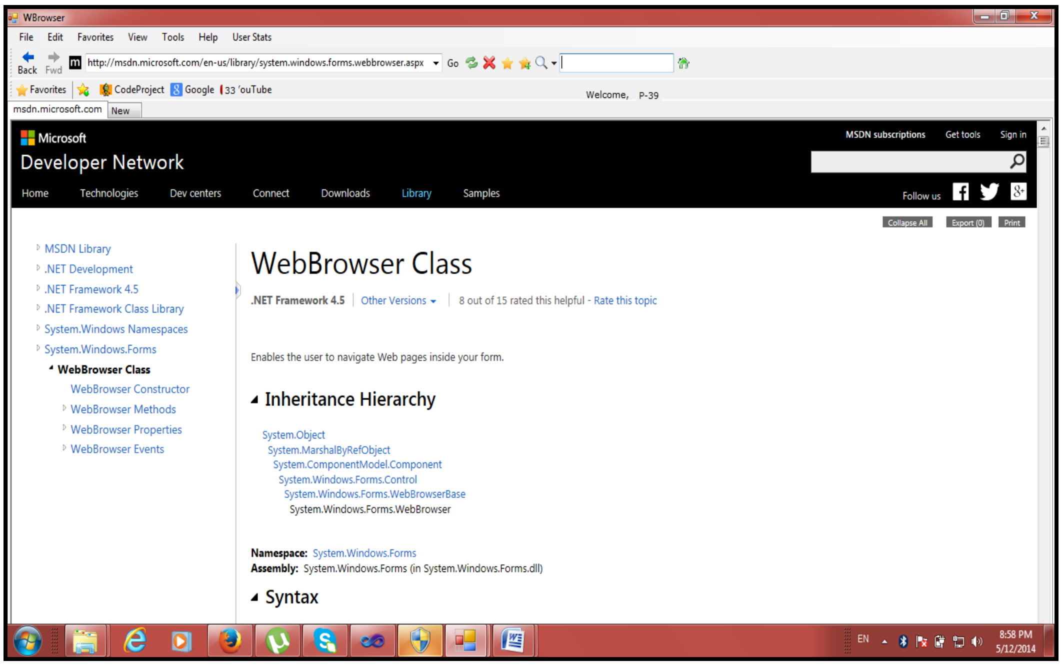
HyperLink Action in Database



IDNO	URL	ActionEvent	Frequency
P-6	http://www	SCROLL	44
P-7	https://ww	HYPERLINK	1
P-7	https://ww	HYPERLINK	1
P-7	http://www	REFRESH	1
P-7	http://www	SCROLL	1
P-30	https://ww	PRINT	1
P-32	http://www	FAVOURITE	1
P-35	https://ww	MouseClick	1
P-35	https://ww	MouseClick	4
P-37	https://ww	COPY	1
label5	http://www	FAVOURITE	1
P-39	https://ww	FORWARD	1
P-39	http://www	MouseClick	35
P-39	http://www	BACK	1
P-39	http://www	MouseClick	39
P-39	http://www	MouseClick	132
P-39	http://code.	BACK	1
P-39	https://ww	MouseClick	46
P-39	http://www	HYPERLINK-	1
P-39	http://msdn	MouseClick	46
P-39	http://www	BACK	1

Figure A4.22: Hyperlink action in user statistics

Search Action


WBrowser
File Edit Favorites View Tools Help User Stats
Back Fwd
http://msdn.microsoft.com/en-us/library/system.windows.forms.webbrowser.aspx
Go
Favorites CodeProject Google 33 'ouTube
Welcome, P-39
msdn.microsoft.com New
Microsoft
Developer Network
MSDN subscriptions Get tools Sign in
Home Technologies Dev centers Connect Downloads Library Samples
Follow us
Collapse All Export (0) Print
MSDN Library
.NET Development
.NET Framework 4.5
.NET Framework Class Library
System.Windows Namespaces
System.Windows.Forms
WebBrowser Class
WebBrowser Constructor
WebBrowser Methods
WebBrowser Properties
WebBrowser Events
EN
8:58 PM
5/12/2014

WebBrowser Class

.NET Framework 4.5 | Other Versions | 8 out of 15 rated this helpful - Rate this topic

Enables the user to navigate Web pages inside your form.

Inheritance Hierarchy

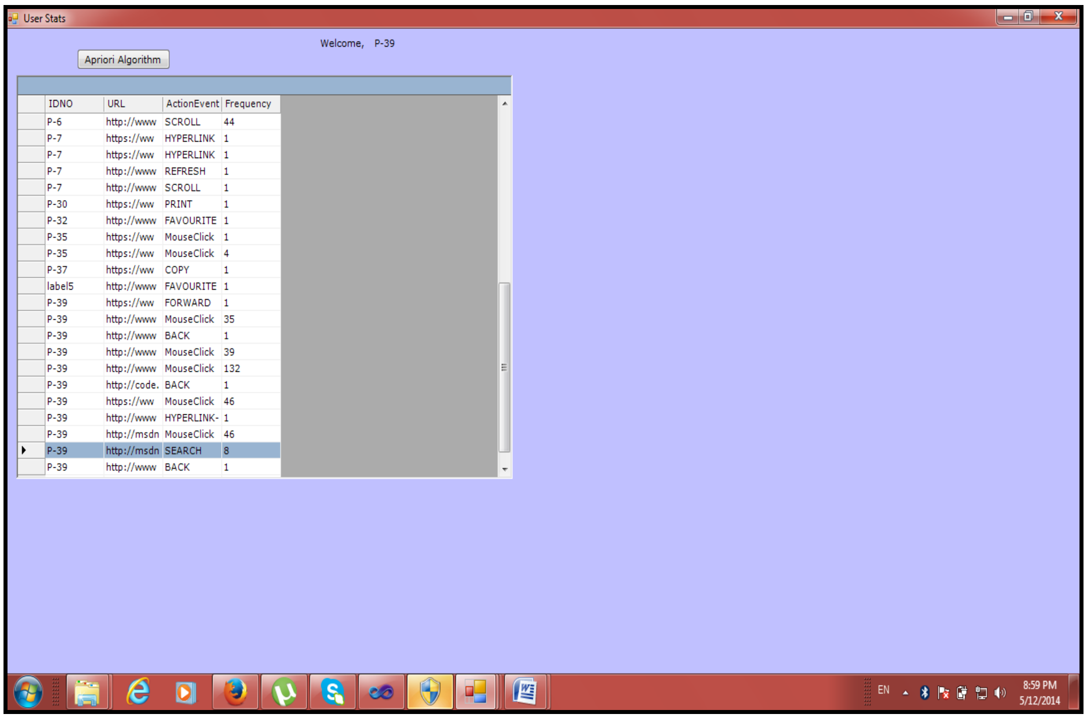
System.Object
System.MarshalByRefObject
System.ComponentModel.Component
System.Windows.Forms.Control
System.Windows.Forms.WebBrowserBase
System.Windows.Forms.WebBrowser

Namespace: System.Windows.Forms
Assembly: System.Windows.Forms (in System.Windows.Forms.dll)

Syntax

Figure A4.23: Search action

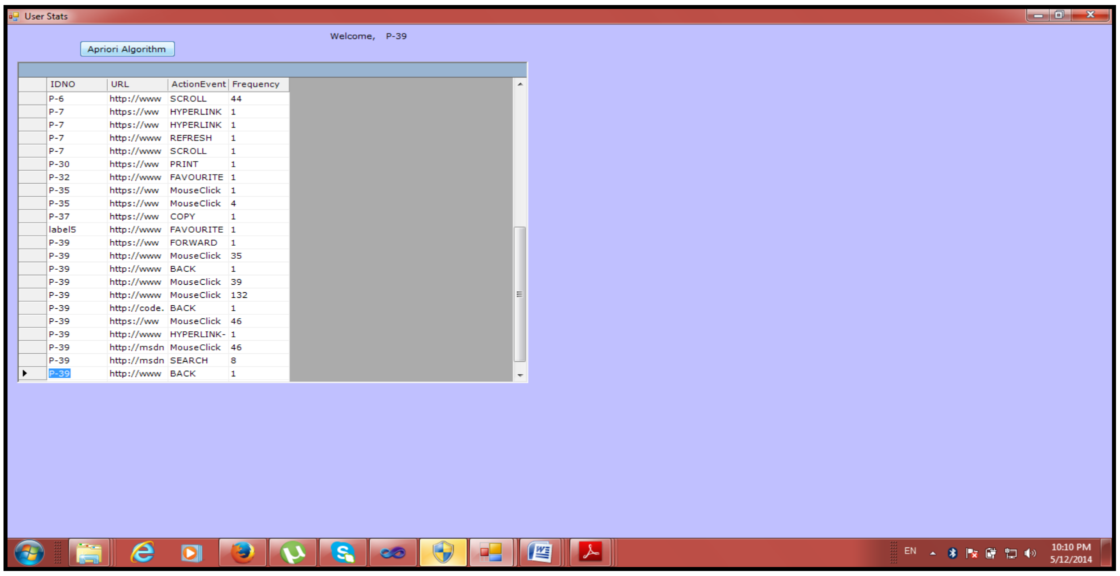
Search Action in Database



IDNO	URL	ActionEvent	Frequency
P-6	http://www	SCROLL	44
P-7	https://ww	HYPERLINK	1
P-7	https://ww	HYPERLINK	1
P-7	http://www	REFRESH	1
P-7	http://www	SCROLL	1
P-30	https://ww	PRINT	1
P-32	http://www	FAVOURITE	1
P-35	https://ww	MouseClick	1
P-35	https://ww	MouseClick	4
P-37	https://ww	COPY	1
label5	http://www	FAVOURITE	1
P-39	https://ww	FORWARD	1
P-39	http://www	MouseClick	35
P-39	http://www	BACK	1
P-39	http://www	MouseClick	39
P-39	http://www	MouseClick	132
P-39	http://code.	BACK	1
P-39	https://ww	MouseClick	46
P-39	http://www	HYPERLINK-	1
P-39	http://msdn	MouseClick	46
P-39	http://msdn	SEARCH	8
P-39	http://www	BACK	1

Figure 4.24: Search action in user statistics

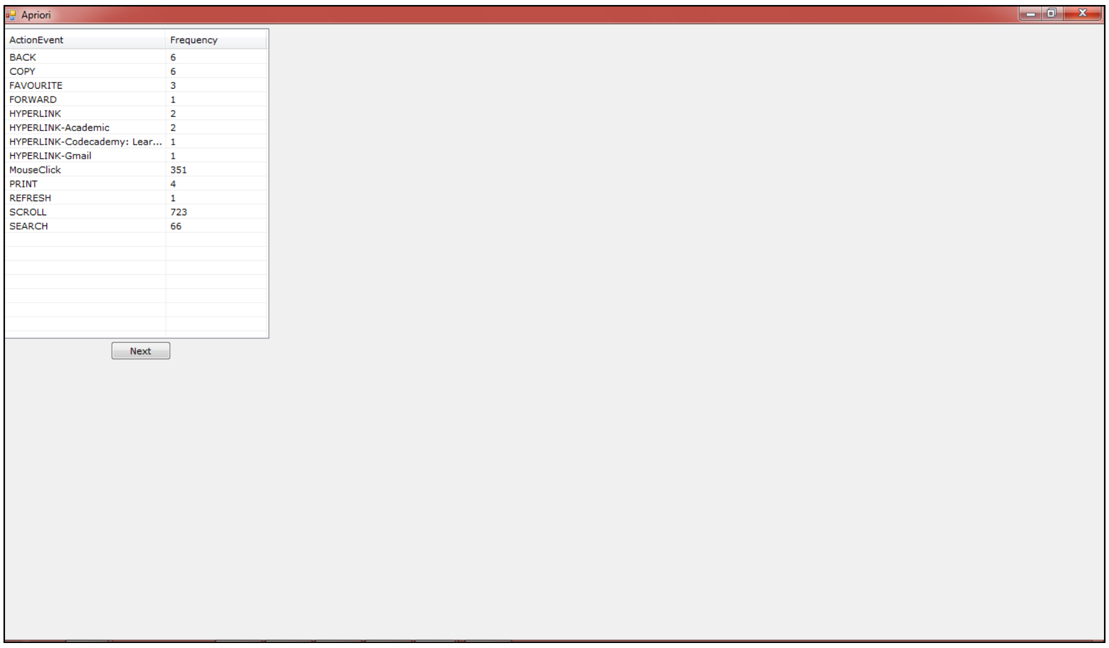
Applying Apriori Algorithm



IDNO	URL	ActionEvent	Frequency
P-6	http://www	SCROLL	44
P-7	https://ww	HYPERLINK	1
P-7	https://ww	HYPERLINK	1
P-7	http://www	REFRESH	1
P-7	http://www	SCROLL	1
P-30	https://ww	PRINT	1
P-32	http://www	FAVOURITE	1
P-35	https://ww	MouseClick	1
P-35	https://ww	MouseClick	4
P-37	https://ww	COPY	1
label5	http://www	FAVOURITE	1
P-39	https://ww	FORWARD	1
P-39	http://www	MouseClick	35
P-39	http://www	BACK	1
P-39	http://www	MouseClick	39
P-39	http://www	MouseClick	132
P-39	http://code.	BACK	1
P-39	https://ww	MouseClick	46
P-39	http://www	HYPERLINK-	1
P-39	http://msdn	MouseClick	46
P-39	http://msdn	SEARCH	8
P-39	http://www	BACK	1

Figure A4.25: Applying Apriori algorithm

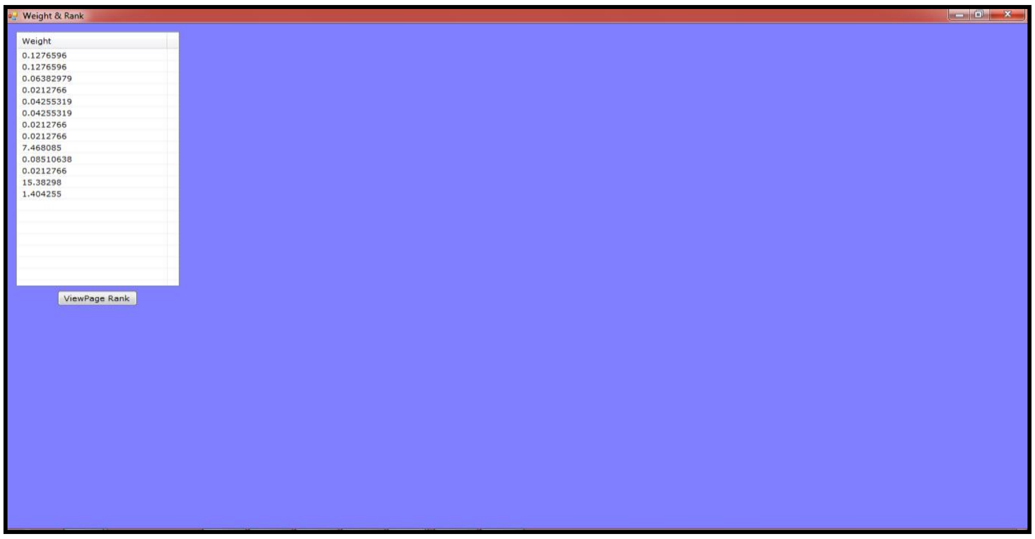
Summary of user's actions with frequency



ActionEvent	Frequency
BACK	6
COPY	6
FAVOURITE	3
FORWARD	1
HYPERLINK	2
HYPERLINK-Academic	2
HYPERLINK-Codecademy: Lear...	1
HYPERLINK-Gmail	1
MouseClick	351
PRINT	4
REFRESH	1
SCROLL	723
SEARCH	66

Figure A4.26: Frequency of actions performed by users

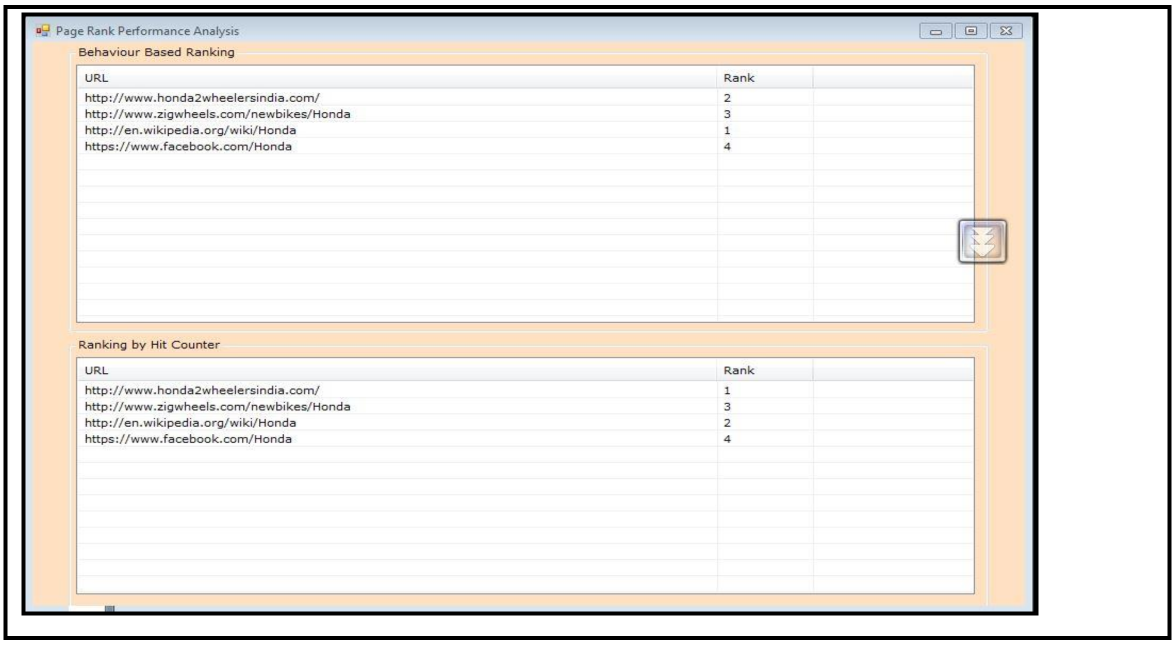
Calculation of weight



Weight
0.1276596
0.1276596
0.06382979
0.0212766
0.04255319
0.04255319
0.0212766
0.0212766
7.468085
0.08510638
0.0212766
15.38298
1.404255

Figure A4.27: Weight Calculation

Page Rank



Behaviour Based Ranking

URL	Rank
http://www.honda2wheelersindia.com/	2
http://www.zigwheels.com/newbikes/Honda	3
http://en.wikipedia.org/wiki/Honda	1
https://www.facebook.com/Honda	4

Ranking by Hit Counter

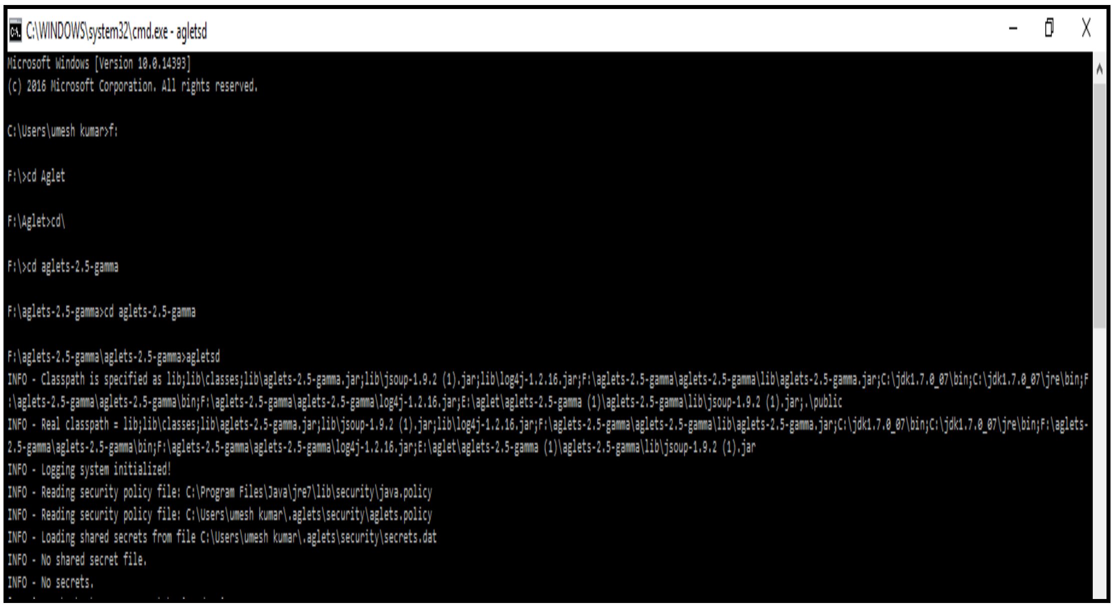
URL	Rank
http://www.honda2wheelersindia.com/	1
http://www.zigwheels.com/newbikes/Honda	3
http://en.wikipedia.org/wiki/Honda	2
https://www.facebook.com/Honda	4

Figure A4.28: Page Rank

APPENDIX V

Snapshots of implementation Migrating Crawler using Aglets

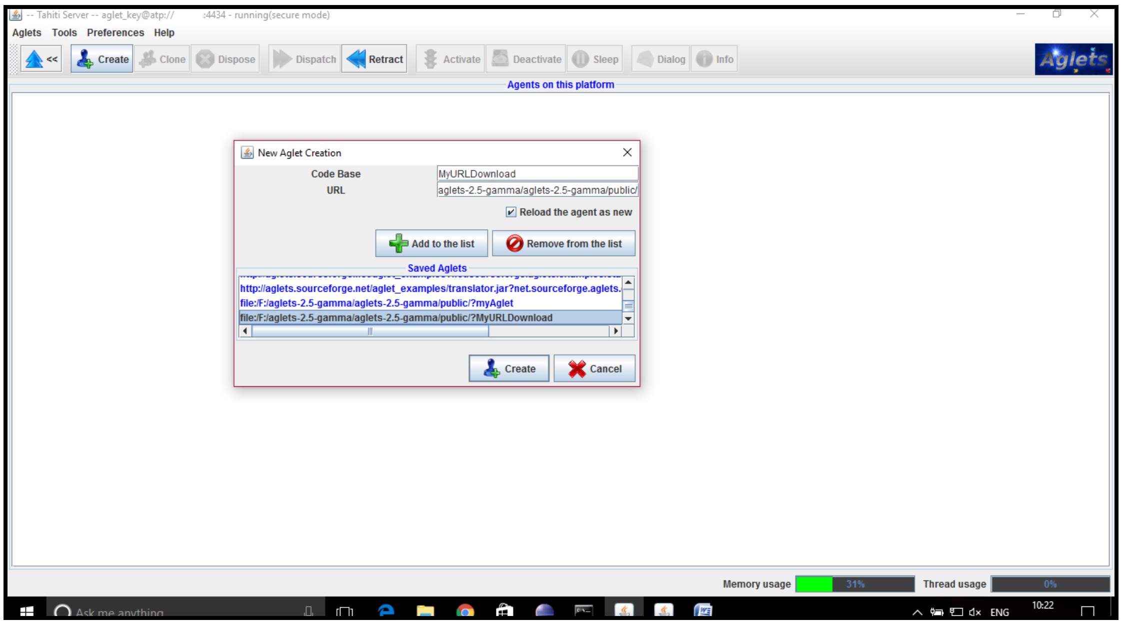
Server Starts



```
C:\WINDOWS\system32\cmd.exe - agletsd
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\umesh kumar>f:

F:\>cd Aglet

F:\Aglet>cd\

F:\>cd aglets-2.5-gamma

F:\aglets-2.5-gamma>cd aglets-2.5-gamma

F:\aglets-2.5-gamma\aglets-2.5-gamma>agletsd
INFO - Classpath is specified as lib;lib\classes;lib\aglets-2.5-gamma.jar;lib\jsoup-1.9.2 (1).jar;lib\log4j-1.2.16.jar;F:\aglets-2.5-gamma\aglets-2.5-gamma\lib\aglets-2.5-gamma.jar;C:\jdk1.7.0_07\bin;C:\jdk1.7.0_07\jre\bin;F
:\aglets-2.5-gamma\aglets-2.5-gamma\bin;F:\aglets-2.5-gamma\aglets-2.5-gamma\log4j-1.2.16.jar;E:\aglet\aglets-2.5-gamma (1)\aglets-2.5-gamma\lib\jsoup-1.9.2 (1).jar;.\public
INFO - Real classpath = lib;lib\classes;lib\aglets-2.5-gamma.jar;lib\jsoup-1.9.2 (1).jar;lib\log4j-1.2.16.jar;F:\aglets-2.5-gamma\aglets-2.5-gamma\lib\aglets-2.5-gamma.jar;C:\jdk1.7.0_07\bin;C:\jdk1.7.0_07\jre\bin;F:\aglets-
2.5-gamma\aglets-2.5-gamma\bin;F:\aglets-2.5-gamma\aglets-2.5-gamma\log4j-1.2.16.jar;E:\aglet\aglets-2.5-gamma (1)\aglets-2.5-gamma\lib\jsoup-1.9.2 (1).jar
INFO - Logging system initialized!
INFO - Reading security policy file: C:\Program Files\Java\jre7\lib\security\java.policy
INFO - Reading security policy file: C:\Users\umesh kumar\.aglets\security\aglets.policy
INFO - Loading shared secrets from file C:\Users\umesh kumar\.aglets\security\secrets.dat
INFO - No shared secret file.
INFO - No secrets.
```

Figure A5.1: Server Starts

Client 1 Creation at Server End



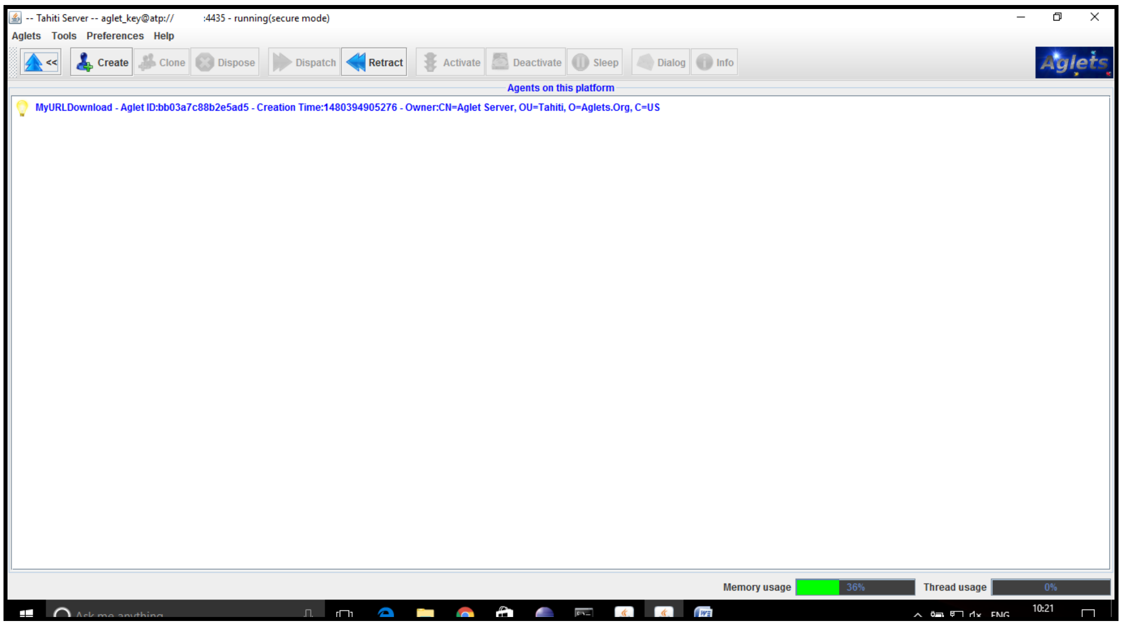
```
-- Tahiti Server -- aglet_key@atp://          :4434 - running(secure mode)
```

Code Base: MyURLDownload
URL: aglets-2.5-gamma/aglets-2.5-gamma/public/
[✓] Reload the agent as new

Saved Aglets:
http://aglets.sourceforge.net/aglet_examples/translator.jar?net.sourceforge.aglets.
file:/F:/aglets-2.5-gamma/aglets-2.5-gamma/public/?myAglet
file:/F:/aglets-2.5-gamma/aglets-2.5-gamma/public/?MyURLDownload

Figure A5.2: Client 1 Created

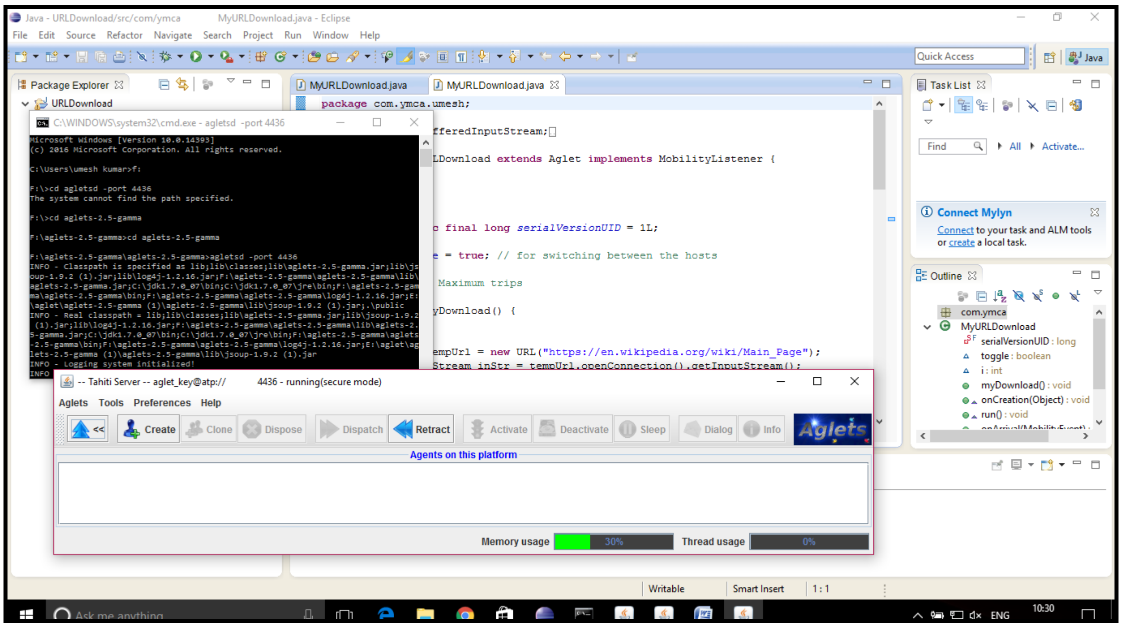
Client 1 running


-- Tahiti Server -- aglet_key@atp://   :4435 - running(secure mode)
Aglets Tools Preferences Help
Create Clone Dispose Dispatch Retract Activate Deactivate Sleep Dialog Info
Aglets
Agents on this platform
Memory usage
Thread usage

```
MyURLDownload - Aglet ID:bb03a7c88b2e5ad5 - Creation Time:1480394905276 - Owner:CN=Aglet Server, OU=Tahiti, O=Aglets.Org, C=US
```

Figure A5.3: Client 1 Interface

Client 2 Creation at port 4436

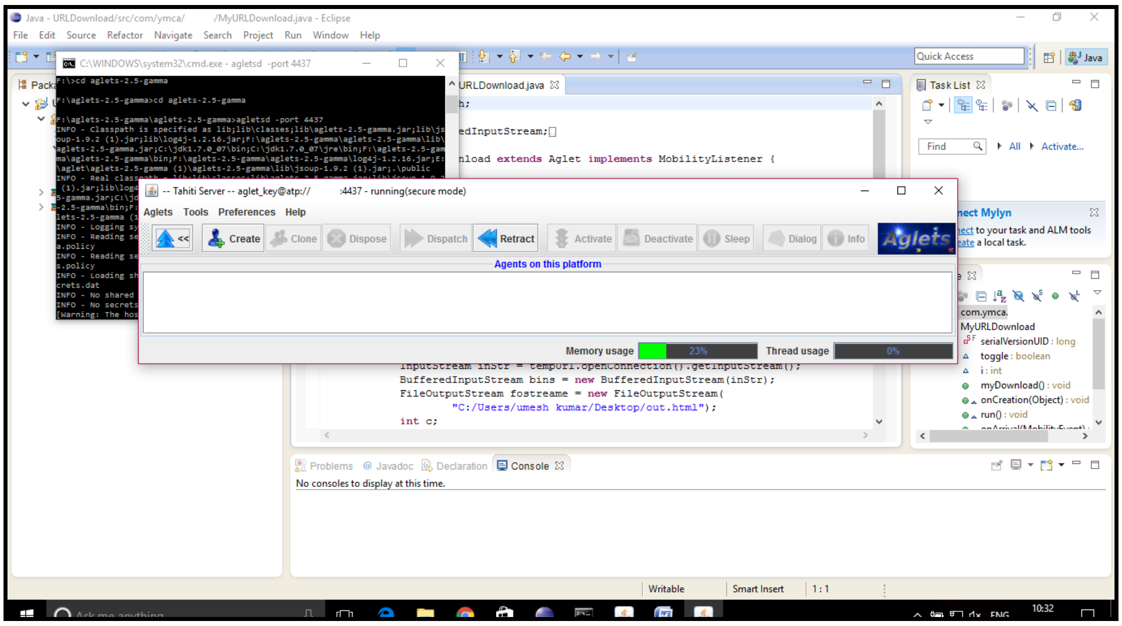

Java - URLDownload/src/com/ymca   MyURLDownload.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
URLDownload
MyURLDownload.java
C:\WINDOWS\system32\cmd.exe - agletsd -port 4436
Task List
Connect Mylyn
Outline
-- Tahiti Server -- aglet_key@atp://   4436 - running(secure mode)
Aglets Tools Preferences Help
Create Clone Dispose Dispatch Retract Activate Deactivate Sleep Dialog Info
Agents on this platform
Memory usage
Thread usage
Writable
Smart Insert
1 : 1

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\umesh kumar>f:

F:\>cd agletsd -port 4436
The system cannot find the path specified.

F:\>cd aglets-2.5-gamma

F:\aglets-2.5-gamma>cd aglets-2.5-gamma

F:\aglets-2.5-gamma\aglets-2.5-gamma>agletsd -port 4436
INFO - Classpath is specified as lib;lib\classes;lib\aglets-2.5-gamma.jar;lib\js
oup-1.9.2 (1).jar;lib\log4j-1.2.16.jar;F:\aglets-2.5-gamma\aglets-2.5-gamma\lib\
aglets-2.5-gamma.jar;C:\jdk1.7.0_07\bin;C:\jdk1.7.0_07\jre\bin;F:\aglets-2.5-gam
ma\aglets-2.5-gamma\bin;F:\aglets-2.5-gamma\aglets-2.5-gamma\log4j-1.2.16.jar;E:
\aglet\aglets-2.5-gamma (1)\aglets-2.5-gamma\lib\jsoup-1.9.2 (1).jar;.\public
INFO - Real classpath = lib;lib\classes;lib\aglets-2.5-gamma.jar;lib\jsoup-1.9.2
 (1).jar;lib\log4j-1.2.16.jar;F:\aglets-2.5-gamma\aglets-2.5-gamma\lib\aglets-2.
5-gamma.jar;C:\jdk1.7.0_07\bin;C:\jdk1.7.0_07\jre\bin;F:\aglets-2.5-gamma\aglets
-2.5-gamma\bin;F:\aglets-2.5-gamma\aglets-2.5-gamma\log4j-1.2.16.jar;E:\aglet\ag
lets-2.5-gamma (1)\aglets-2.5-gamma\lib\jsoup-1.9.2 (1).jar
INFO - Logging system initialized!
INFO
```

```
package com.ymca.umesh;

fferedInputStream;

LDownload extends Aglet implements MobilityListener {


c final long serialVersionUID = 1L;

e = true; // for switching between the hosts

 Maximum trips

yDownload() {


empUrl = new URL("https://en.wikipedia.org/wiki/Main_Page");
Stream inStr = tempUrl.openConnection().getInputStream();
```

Figure A5.4: Client 2 Created

Client 3 Creation at port 4437



```
C:\WINDOWS\system32\cmd.exe - agletsd  -port 4437
F:\>cd aglets-2.5-gamma

F:\aglets-2.5-gamma>cd aglets-2.5-gamma

F:\aglets-2.5-gamma\aglets-2.5-gamma>agletsd -port 4437
INFO - Classpath is specified as lib;lib\classes;lib\aglets-2.5-gamma.jar;lib\js
oup-1.9.2 (1).jar;lib\log4j-1.2.16.jar;F:\aglets-2.5-gamma\aglets-2.5-gamma\lib\
aglets-2.5-gamma.jar;C:\jdk1.7.0_07\bin;C:\jdk1.7.0_07\jre\bin;F:\aglets-2.5-gam
ma\aglets-2.5-gamma\bin;F:\aglets-2.5-gamma\aglets-2.5-gamma\log4j-1.2.16.jar;E:
\aglet\aglets-2.5-gamma (1)\aglets-2.5-gamma\lib\jsoup-1.9.2 (1).jar;.\public
INFO - Real class
 (1).jar;lib\log4
5-gamma.jar;C:\jd
-2.5-gamma\bin;F:
lets-2.5-gamma (1
INFO - Logging sy
INFO - Reading se
a.policy
INFO - Reading se
s.policy
INFO - Loading sh
crets.dat
INFO - No shared
INFO - No secrets
[Warning: The hos
```

```
-- Tahiti Server -- aglet_key@atp://          :4437 - running(secure mode)
Memory usage 23%    Thread usage 0%
```

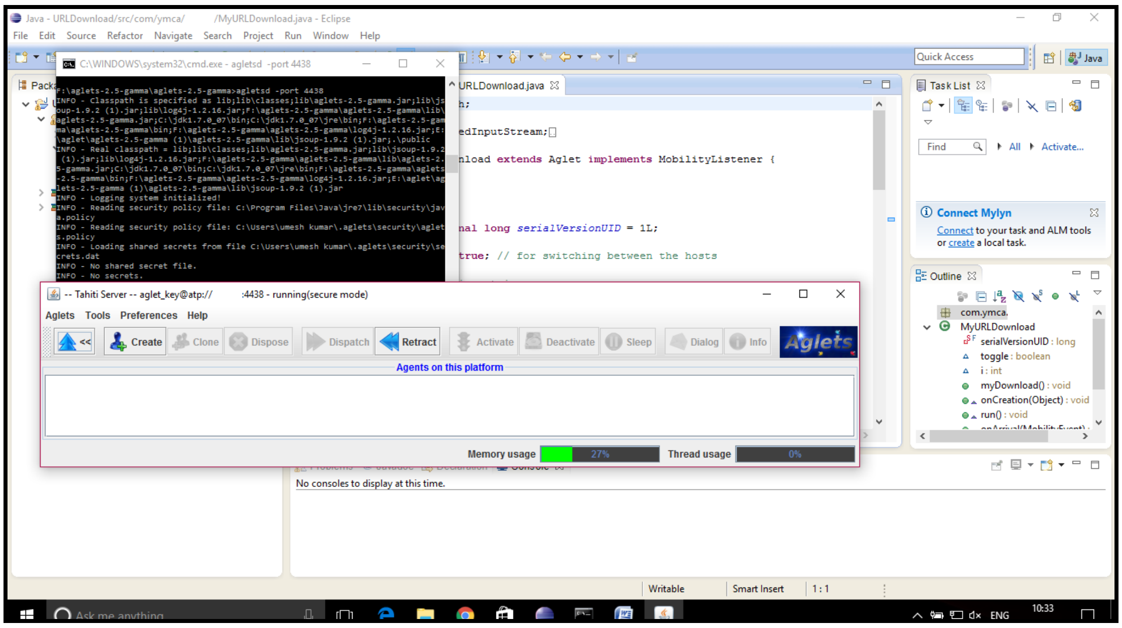
```java
            InputStream inStr = tempUrl.openConnection().getInputStream();
            BufferedInputStream bins = new BufferedInputStream(inStr);
            FileOutputStream fostreame = new FileOutputStream(
                    "C:/Users/umesh kumar/Desktop/out.html");
            int c;
```

No consoles to display at this time.

Java - URLDownload/src/com/ymca/ /MyURLDownload.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Aglets Tools Preferences Help
Create Clone Dispose Dispatch Retract Activate Deactivate Sleep Dialog Info
Agents on this platform
Task List
Outline
Problems Javadoc Declaration Console
Writable Smart Insert 1 : 1

Figure A5.5: Client 3 Created

Client 4 Creation at Port 4438



```
C:\WINDOWS\system32\cmd.exe - agletsd  -port 4438
F:\aglets-2.5-gamma\aglets-2.5-gamma>agletsd -port 4438
INFO - Classpath is specified as lib;lib\classes;lib\aglets-2.5-gamma.jar;lib\js
oup-1.9.2 (1).jar;lib\log4j-1.2.16.jar;F:\aglets-2.5-gamma\aglets-2.5-gamma\lib\
aglets-2.5-gamma.jar;C:\jdk1.7.0_07\bin;C:\jdk1.7.0_07\jre\bin;F:\aglets-2.5-gam
ma\aglets-2.5-gamma\bin;F:\aglets-2.5-gamma\aglets-2.5-gamma\log4j-1.2.16.jar;E:
\aglet\aglets-2.5-gamma (1)\aglets-2.5-gamma\lib\jsoup-1.9.2 (1).jar;.\public
INFO - Real classpath = lib;lib\classes;lib\aglets-2.5-gamma.jar;lib\jsoup-1.9.2
 (1).jar;lib\log4j-1.2.16.jar;F:\aglets-2.5-gamma\aglets-2.5-gamma\lib\aglets-2.
5-gamma.jar;C:\jdk1.7.0_07\bin;C:\jdk1.7.0_07\jre\bin;F:\aglets-2.5-gamma\aglets
-2.5-gamma\bin;F:\aglets-2.5-gamma\aglets-2.5-gamma\log4j-1.2.16.jar;E:\aglet\ag
lets-2.5-gamma (1)\aglets-2.5-gamma\lib\jsoup-1.9.2 (1).jar
INFO - Logging system initialized!
INFO - Reading security policy file: C:\Program Files\Java\jre7\lib\security\jav
a.policy
INFO - Reading security policy file: C:\Users\umesh kumar\.aglets\security\aglet
s.policy
INFO - Loading shared secrets from file C:\Users\umesh kumar\.aglets\security\se
crets.dat
INFO - No shared secret file.
INFO - No secrets.
```

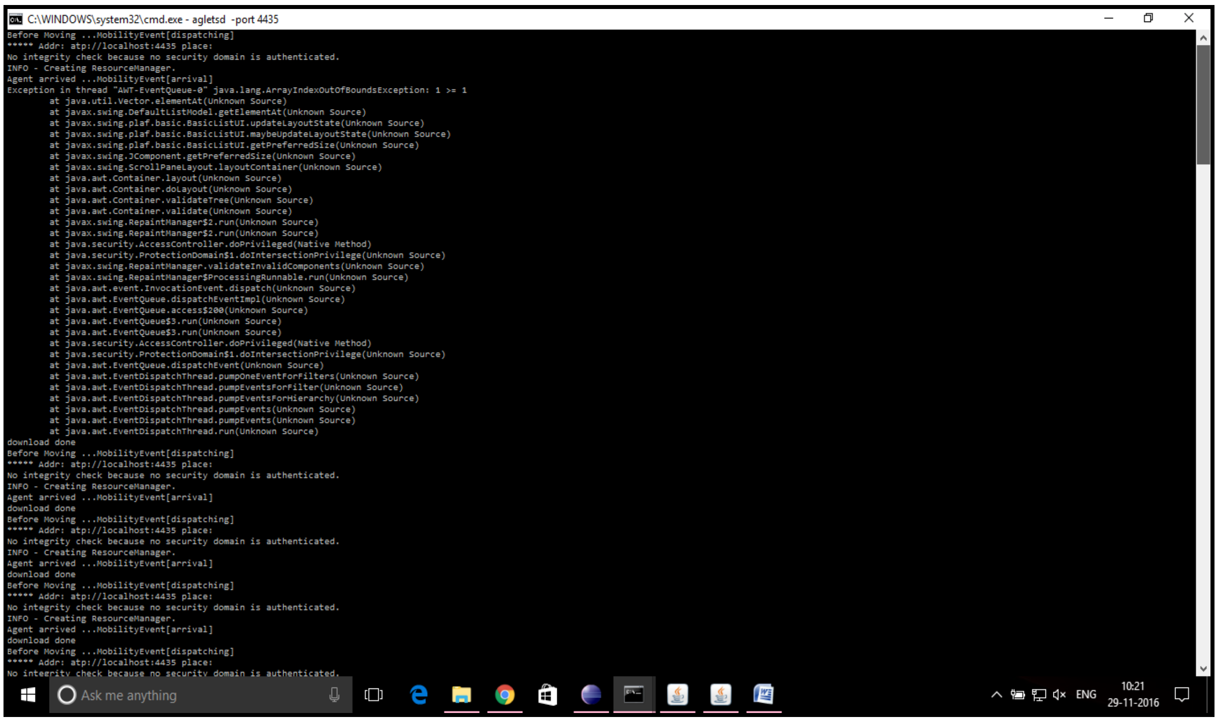
```
-- Tahiti Server -- aglet_key@atp://          :4438 - running(secure mode)
Memory usage 27%    Thread usage 0%
```

No consoles to display at this time.

Java - URLDownload/src/com/ymca/ /MyURLDownload.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Aglets Tools Preferences Help
Create Clone Dispose Dispatch Retract Activate Deactivate Sleep Dialog Info
Agents on this platform
Task List
Connect Mylyn
Outline
Writable Smart Insert 1 : 1

Figure A5.6: Client 4 Created

Downloading at Client 1



```
Before Moving ...MobilityEvent[dispatching]
***** Addr: atp://localhost:4435 place:
No integrity check because no security domain is authenticated.
INFO - Creating ResourceManager.
Agent arrived ...MobilityEvent[arrival]
Exception in thread "AWT-EventQueue-0" java.lang.ArrayIndexOutOfBoundsException: 1 >= 1
        at java.util.Vector.elementAt(Unknown Source)
        at javax.swing.DefaultListModel.getElementAt(Unknown Source)
        at javax.swing.plaf.basic.BasicListUI.updateLayoutState(Unknown Source)
        at javax.swing.plaf.basic.BasicListUI.maybeUpdateLayoutState(Unknown Source)
        at javax.swing.plaf.basic.BasicListUI.getPreferredSize(Unknown Source)
        at javax.swing.JComponent.getPreferredSize(Unknown Source)
        at javax.swing.ScrollPaneLayout.layoutContainer(Unknown Source)
        at java.awt.Container.layout(Unknown Source)
        at java.awt.Container.doLayout(Unknown Source)
        at java.awt.Container.validateTree(Unknown Source)
        at java.awt.Container.validate(Unknown Source)
        at javax.swing.RepaintManager$2.run(Unknown Source)
        at javax.swing.RepaintManager$2.run(Unknown Source)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.security.ProtectionDomain$1.doIntersectionPrivilege(Unknown Source)
        at javax.swing.RepaintManager.validateInvalidComponents(Unknown Source)
        at javax.swing.RepaintManager$ProcessingRunnable.run(Unknown Source)
        at java.awt.event.InvocationEvent.dispatch(Unknown Source)
        at java.awt.EventQueue.dispatchEventImpl(Unknown Source)
        at java.awt.EventQueue.access$200(Unknown Source)
        at java.awt.EventQueue$3.run(Unknown Source)
        at java.awt.EventQueue$3.run(Unknown Source)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.security.ProtectionDomain$1.doIntersectionPrivilege(Unknown Source)
        at java.awt.EventQueue.dispatchEvent(Unknown Source)
        at java.awt.EventDispatchThread.pumpOneEventForFilters(Unknown Source)
        at java.awt.EventDispatchThread.pumpEventsForFilter(Unknown Source)
        at java.awt.EventDispatchThread.pumpEventsForHierarchy(Unknown Source)
        at java.awt.EventDispatchThread.pumpEvents(Unknown Source)
        at java.awt.EventDispatchThread.pumpEvents(Unknown Source)
        at java.awt.EventDispatchThread.run(Unknown Source)
download done
Before Moving ...MobilityEvent[dispatching]
***** Addr: atp://localhost:4435 place:
No integrity check because no security domain is authenticated.
INFO - Creating ResourceManager.
Agent arrived ...MobilityEvent[arrival]
download done
Before Moving ...MobilityEvent[dispatching]
***** Addr: atp://localhost:4435 place:
No integrity check because no security domain is authenticated.
INFO - Creating ResourceManager.
Agent arrived ...MobilityEvent[arrival]
download done
Before Moving ...MobilityEvent[dispatching]
***** Addr: atp://localhost:4435 place:
No integrity check because no security domain is authenticated.
INFO - Creating ResourceManager.
Agent arrived ...MobilityEvent[arrival]
download done
Before Moving ...MobilityEvent[dispatching]
***** Addr: atp://localhost:4435 place:
No integrity check because no security domain is authenticated.
```

Figure A5.7: URL Downloaded

BRIEF PROFILE OF THE RESEARCH SCHOLAR

Deepika Punj did her B.Tech (Computer Science & Engineering) from Maharashi Dayananad University, Rohtak in 2004 and M.Tech (Computer Engineering) in 2008. Currently, She is working as Assistant Professor in Department of Computer Engineering at YMCA University of Science and Technology, Faridabad, India. She is having total 11 years of experience in teaching. Her areas of interest are Programming Language C, Operating System and Internet technologies. She has authored papers in various national and international journals.

LIST OF PUBLISHED PAPERS

International Journals:

S. No	Title of the paper along with volume, Issue No, year of publication	Publisher	Impact factor	Refereed or Non-Referred	Whether you paid any money or not for the publication	Remarks
1.	Design of a Migrating Crawler Based on a Novel URL Scheduling Mechanism using AHP. IJRSDA: Volume 4, Issue 1, Article 6	IGI GLOBAL		Refereed	No	ACM, DBLP, Google Scholar
2.	Capturing User Browsing Behaviour Indicators. Electrical & Computer Engineering: An International Journal (ECIJ) Volume 4, Number 2, 2015	Wireilla Scientific Publications, Australia		Refereed	No	CiteSeer, Google Scholar, Scibd
3.	Design of an Efficient Migrating Crawler based on Sitemaps. International Journal of Advanced Science and Technology Vol.83, 2015	SERSC, Australia		Refereed	No	EBSCO, ProQuest, DOAJ, J-Gate
4.	URL Ordering Policies for Distributed Crawlers: A Review. International Journal of Journal of Network Communications and Emerging Technologies (JNCET) Volume 4, Special Issue 1, 2015.	EverScience		Refereed	No	Google Scholar, Cite Factor
5.	A Novel Approach for Document Structure based Filtering in Migrating Crawler. YMCAUST International Journal of Research, Vol. 2, Issue 2, 2014	YMCAUST		Refereed	No	
6.	Web Crawler Design Issues: A Review. International Journal of Management, IT and Engineering, Vol. 2, Issue 8, 2012	IJMRA		Refereed	YES	Google Scholar, Cornell University Library, Pro Quest, EBSCO
7.	Document structure similarity methods: a review. Published in International Journal of Multidisciplinary Research Studies (IJMRS) in 2012.	IJMRS		Refereed	YES	DOAJ, CiteSeer, Google Scholar

International & National Conferences

S.No	Title of the paper along with volume, Issue No, year of publication	Publisher	Impact factor	Refereed or Non-Refereed	Whether you paid any money or not for the publication	Remarks
1.	Improving Search Results Based On Users' Browsing Behaviour Using Apriori Algorithm". Presented in International Conference CSI-2015	SPRINGER			YES	CSI Sponsored
2.	Document structure based Filtering in Migrating Crawler- A Review. Presented in International Conference Paradigms shift in Management and Technology at Faridabad in 9-10 March 2015				YES	Best Paper Award
3.	Design of an User Browsing Behaviour Tracking Tool". Presented in International Conference on Advanced Information Communication Technologies in Engineering (ICAICTE-2K13) in 2013				YES	AICTE & CSI Sponsored
4.	Security issues in mobile agent system: A review. Presented in National Conference NEIET-2012 in April 2012.				YES	