Design, Analysis and Implementation of Algorithms for Web Mining Support Functions

THESIS

Submitted in partial fulfilment
of the requirements for the degree of
DOCTOR OF PHILOSOPHY

by

K. JEYALATHA

Under the Supervision of
Dr. B. Vijayakumar

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

2012

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the thesis entitled **Design, Analysis and Implementation of Algorithms for Web Mining Support Functions** and submitted by **K. Jeyalatha** ID No **2008PHXF001U** for award of Ph.D. of the Institute embodies original work done by her under my supervision.

Signature of the Supervisor	B. Vijayakumar
Name in capital letters	Dr. B. VIJAYAKUMAR
Designation	Associate Professor

Date: 31/1/13

ACKNOWLEDGEMENTS

I would like to express my deepest sense of gratitude, to my Ph.D. Thesis Supervisor **Dr. B. Vijayakumar**, Associate Professor, Computer Science, Birla Institute of Technology and Science, Pilani, Dubai Campus, for his able guidance, valuable technical inputs and useful suggestions. I express my sincere thanks and gratitude to **Dr. R.K. Mittal**, Director and Senior Professor, BITS Pilani, Dubai Campus for his encouragement and continued support to pursue my research.

I sincerely thank the Doctoral Advisory Committee, BITS Pilani and also Doctoral Advisory Committee, BITS Pilani, Dubai Campus for its constant monitoring and advisory role that has really helped me very much throughout this programme.

I am very much grateful to Dr. B.N. Jain, Vice Chancellor, Registrar, Dean Research and Consultancy Division, BITS Pilani (Rajasthan), Dr. M. Ramachandran, Dr. G. Vijaya, Dean Research and Consultancy Division, Dr. K.K. Singh, Associate Professor, Dr. K. Kumar, Registrar BPDC for their constant support and encouragement given to me.

I thank Dr. S. Vadivel, HOD, CS and Dr. V. Santhosh Kumar, Assistant Professor, CS, BITS Pilani, Dubai Campus, for their valuable suggestions. I also thank my colleagues Dr. M. Madiajagan, Assistant Professor, Dr. A. Somasundaram, Associate Professor, Dr. Geetha, Assistant Professor, Dr. Suhel Ahmed Khan, Assistant Professor, Mr. K.N. Sheshadri, Librarian and Ms. S. Kavitha, Lecturer for their interaction.

I would like to thank the Information Technology Support Team, BITS, Pilani - Dubai Campus for providing me excellent facilities. I also would like to thank the lab instructors Mr. Zafar Ali Khan. N and Mr. Mohamed Anvar Haither for their kind cooperation.

I acknowledge Mr.Ehtesham A. Hazarika, Ms. Zainab A. Shikari, Ms. Anamika Datta, Mr. Sahil Dhingra, Mr. Sachin K.Thomas, Mr. Munawwar Firoz, Mr. M. Rohit, Mr. Gurpreet Singh Wadhwa, Ms. Aksheta Mukesh Mehta, Ms. Ashwini Annie Sam, Mr. Siddhartha Negi for their help, technical inputs and interactions.

I would like to place on record my deepest sense of gratitude to the **Research Board, BITS Pilani** for giving me an opportunity to do this research work and providing me guidance throughout the research programme.

I wish to thank every member of my family especially my spouse, children, parents, sister and in-laws for providing me all support and cooperation in pursuing this research programme.

Above all, I thank the Lord for giving me the strength to carry out this work to the best of my abilities.

K. Jeyalatha

2008PHXF001U

Abstract

This thesis draws upon web mining functions related to web search activities. Its main concern is the design, analysis and implementation of algorithms relating to web content, web structure and web usage support functions.

Web Content Mining involves data extraction, grouping, storage and indexing functions for web user. It helps the web users to access the web content in an organized way, by providing search assistance. The proposed approach reduces the time and effort in the search process. It uses the search engines Google and Yahoo at the back end. The web users can navigate the web content using a standard interface. The web administrator can perform a range of options for folders and files interactively.

Web Structure Mining has been carried out using two algorithms. The first algorithm involves modeling the web as an implicit graph using input keywords and traversal of the graph using Breadth First Search strategy. The second algorithm involves design of an interactive Page Rank calculator for a given universe of web pages, using a standard interface that considers web links attribute information.

The algorithm for the analysis of web usage log involves two phases: preprocessing and log analysis. The preprocessing phase formats the input search related data into text files with appropriate delimiters, between various attributes. The log analysis phase involves creation of web usage database, execution of SQL queries and analysis of web usage data pertaining to various users in an academic environment. The interestingness measures: t-weight and d-weight, relating to descriptive data mining are calculated.

The algorithms for web mining support functions can be effectively used by various categories of users in their search related activities and the current work considers an academic environment. The report also summarizes the work, highlights specific contributions and suggests some directions for future work.

Contents

Certificate ii

Acknowledgements iii

Abstract v

List of Tables xi

List of Figures xiii

List of Abbreviations xvi

1 Introduction 1

1.1 Main Functions of the Search application 3

1.1.1 Categories of Users 4

1.2 Motivation 6

1.3 Problem Description 6

1.4 Gap in Existing Research 7

1.5 Objectives, Scope and Limitations 8

1.6 Methodology 9

1.7 Organization of the Thesis 9

2 Literature Review 12

2.1 Introduction 12

2.2 Data Mining 12

2.3 Web Mining 13

2.4 Web Mining Categories 14

2.4.1 Data Warehouse . . . 17
2.4.2 Web Warehouse . . . 17
2.5 Web Structure Mining . . . 18
2.5.1 Web Structure Mining Algorithms . . . 19
2.5.2 Page Rank Algorithm . . . 20
2.5.3 The Random Surfer Model . . . 21
2.5.4 Hyperlink Induced Topic Search (HITS) Algorithm . . . 22
2.5.5 TrustRank Algorithm . . . 23
2.5.6 Selective HITS (Sel-HITS) Algorithm . . . 23
2.6 Web Usage Mining . . . 24
2.7 Summary . . . 27

3 Design and Implementation of Algorithm for Extraction and Storage of Web Content 28

3.1 Introduction . . . 28
3.2 Algorithm: Web Data Extraction and Storage (WDES) . . . 29
3.3 Time Complexity . . . 30
3.4 WDES Implementation using Google . . . 31
3.5 WDES Implementation using Yahoo . . . 36
3.6 Test Scenario . . . 37
3.7 Discussion . . . 40
3.7.1 Observation and Inference for WDES: Implementation using Google 41
3.7.2 Observation and Inference for WDES: Implementation using Yahoo 41
3.8 Summary . . . 42

4 Implementation of a Web based tool for Web user and Administrator 45

4.1 Introduction . . . 45
4.2 Web user . . . 46
4.2.1 Algorithm ACAD-SEARCH . . . 47
4.2.2 Database Design for Search Application . . . 48
4.2.3 User Interface Design for Web User . . . 50

4.2.4 Test Scenario 52
4.2.5 Time Complexity 53
4.3 Web Administrator 55
4.3.1 Algorithm MAINTAIN-REPOSITORY 56
4.3.2 User Interface Design for Web Administrator 57
4.3.3 Implementation 58
4.3.4 Time Complexity 59
4.4 Summary 59

5 Design and Implementation of Breadth First Search Strategy for Implicit Graph 61
5.1 Introduction 61
5.2 Problem Description 63
5.3 Algorithm: LINKS-TRAVERSAL 63
5.4 Time Complexity 65
5.5 User Interface Design 66
5.6 Test Scenario 67
5.7 Summary 71

6 Design of an Interface for Page Rank Calculation in Web Structure Mining 72
6.1 Introduction 72
6.2 Interface Description for Page Rank Algorithm 72
6.3 Algorithm: VIS-POS 73
6.4 Implementation 75
6.5 Test Scenario 78
6.6 Formal Verification using Hoare's logic 88
6.7 Summary 91

7 Web Usage Log Analysis using Descriptive Data Mining 92
7.1 Introduction 92
7.2 Problem Description 92

7.3 Algorithm: PRE-USE 94

7.4 Experimental Setup 97

7.4.1 Database Creation 97

7.5 Test Scenario 100

7.5.1 SQL Queries 101

7.6 Discussion 103

7.7 Creation of indexes using SQL 107

7.7.1 Index on Key and Link 107

7.7.2 Index on Key, Type and Size 107

7.7.3 Index on Key and Total download size 107

7.8 Interestingness measures : t-weight and d-weight 108

7.8.1 t-weight 108

7.8.2 d-weight 109

7.8.3 Significance of t-weight and d-weight in the current work 109

7.8.4 SQL queries for extracting summary information 110

7.9 Complexity 113

7.10 Summary 113

8 Conclusion, Contributions and Future Work 114

8.1 Contributions 115

8.2 Directions for Future Work 116

References 118

Appendix A System Configuration 125

Appendix B Functions and their Actions for WDES: Implementation using Google 127

Appendix C Functions and their Actions for WDES: Implementation using Yahoo 129

Appendix D Functions and their Actions for Links-Traversal Algorithm : Java Implementation 130

Appendix E Functions and their Actions for Maintain-Repository Algorithm : Java Implementation 132

Appendix F Awk Script for Pre-Use Algorithm in Web Usage Mining 133

Appendix G Layout of forms/menus for web based tool for Web user and Administrator 136

Appendix H List of Publications 144

Appendix I Biography of the Candidate 146

Appendix J Biography of the Supervisor 147

List of Tables

3.1 Time Complexity for WDES 30

3.2 User Categories and Tasks 39

3.3 Experimental Results for WDES using Google 40

3.4 Experimental Results for WDES using Yahoo 40

4.1 Database Schema Information 48

4.2 Snapshot of the Search Database 49

4.3 Time Complexity for Repository Operations 59

5.1 Time Complexity for Links-Traversal 65

5.2 Experimental Results for Web Link Traversal 71

6.1 Visibility of a link 79

6.2 Position of a link 79

6.3 Sample Inputs for 2 web pages 80

6.4 Generalized formulae for Evaluation of links 80

6.5 Evaluation of links for 2 web pages 80

6.6 Weighting quotients for 2 web pages 80

6.7 Evaluating factor for 2 web pages 81

6.8 Page Rank for 2 web pages 81

6.9 Sample Inputs for 3 web pages 82

6.10 Evaluation of links for 3 web pages 82

6.11 Weighting quotients for 3 web pages 82

6.12 Evaluating factor for 3 web pages 82

6.13 Page Rank for 3 web pages 83

6.14 Sample Inputs for 4 web pages 84
6.15 Evaluation of links for 4 web pages 84
6.16 Weighting quotients for 4 web pages 84
6.17 Evaluating factor for 4 web pages 84
6.18 Page Rank for 4 web pages 85
6.19 Sample Inputs for 5 web pages 86
6.20 Evaluation of links for 5 web pages 86
6.21 Weighting quotients for 5 web pages 87
6.22 Evaluating factor for 5 web pages 87
6.23 Page Rank for 5 web pages 87
6.24 Page Rank Calculator: Function and their Actions using Java 88

7.1 Database Schema Information: *filesummary* 99
7.2 Database Schema Information: *filedetails* 100
7.3 Results of Query 1 101
7.4 Results of Query 2 102
7.5 Results of Query 3 102
7.6 Experimental Results for Web log storing summary information: *filesummary*104
7.7 Experimental Results for Web log storing detailed information: *filedetails* . 104
7.8 Cross tab with different keywords 111
7.9 Cross tab for file size(count) and file extension 111
7.10 Cross tab with t-weight values associated for each class 112
7.11 Cross tab with d-weight values associated for each class 112
7.12 Complexity of SQL Operations 113

List of Figures

1.1 Categories of Search Application in an Academic environment 3

1.2 Block Schematic of Repository for Search Application in an academic environment 4

1.3 Categories of users for Search Application in an academic environment . . 5

2.1 Web Mining Categories 15

2.2 Structure Mining Algorithms 20

3.1 WDES: Implementation using Google 32

3.2 User Interface with various search options 35

3.3 Log Messages for WDES Algorithm using Google 38

3.4 Log Messages for WDES Algorithm using Yahoo 39

3.5 Performance Graph for WDES using Google 43

3.6 Performance Graph for WDES using Yahoo 44

4.1 Web User: Navigation Options 46

4.2 Web Administrator: Maintain Repository 55

5.1 Links-Traversal Flow Diagram 62

5.2 User Interface Design of Links-Traversal 66

5.3 Links-Traversal Test Case 1 68

5.4 Links-Traversal Test Case 2 68

5.5 Links-Traversal Test Case 3 69

5.6 Log Messages for Links-Traversal 70

6.1 Block Schematic of VIS-POS Interface 73

6.2 Mode Selection 76

6.3 Information Bar 76
6.4 The Page Rank Calculator 77
6.5 Inbound Links 78
6.6 Outbound Links 78
6.7 Web Universe with 2 web pages 79
6.8 Web Universe with 3 web pages 81
6.9 Web Universe with 4 web pages 83
6.10 Web Universe with 5 web pages 85
6.11 Sample Web with 4 web pages 90

7.1 Block Schematic of Web Usage Log Analysis algorithm 93
7.2 ER Diagram 93
7.3 Web log file with messages, status and log data 96
7.4 Preprocessed file layout 97
7.5 Performance Graph of the relation *filesummary* 105
7.6 Performance Graph of the relation *filedetails* 106

G.1 Main Menu for the Proposed Academic Search Application 136
G.2 Insert a Record(for Interactive mode) 136
G.3 Delete an existing Record 137
G.4 Search by Keyword 137
G.5 Search by File Type 137
G.6 Update an existing record 137
G.7 Record Insertion 138
G.8 Page after Creation 138
G.9 Enter Keyword 138
G.10 List of links by searching Keyword 138
G.11 Enter file type 139
G.12 List of links by searching through type 139
G.13 Maintain-Repository: Authentication page 139
G.14 Maintain-Repository: Home page 140
G.15 Maintain-Repository: HTML/XML/Text page 141

G.16 Maintain-Repository: Folder options 141

G.17 Maintain-Repository: File options 142

G.18 Maintain-Repository: Edit options 142

G.19 Maintain-Repository: Sorting options 143

G.20 Maintain-Repository: Search options 143

List of abbreviations

1. AJAX	Asynchronous JavaScript and XML
2. API	Application Programming Interface
3. CSS	Cascading Style Sheets
4. DB	Data Base
5. DBMS	Data Base Management System
6. DHTML	Dynamic Hyper Text Markup Language
7. DOC	Document (Microsoft Word File)
8. DOM	Document Object Model
9. DW	Data Warehouse
10. ER	Entity Relationship
11. FTP	File Transfer Protocol
12. GNU	Gnu's Not Unix
13. GUI	Graphical User Interface
14. HITS	Hyperlink Induced Topic Search
15. HTML	Hyper Text Markup Language
16. HTTP	Hyper Text Transfer Protocol
17. IDE	Integrated Development Environment
18. IP	Internet Protocol
19. IR	Information Retrieval

20. JavaSE6	Java Platform Standard Edition
21. JDBC	Java Data Base Connectivity
22. JDK	Java Development Kit
23. JSDK	Java Servlet Development Kit
24. KB	Kilo Bytes
25. KDD	Knowledge Discovery in Databases
26. LAN	Local Area Network
27. MB	Mega Bytes
28. MIME	Multipurpose Internet Mail Extension
29. MySQL	My Structured Query Language
30. NLP	Natural Language Processing
31. OEM	Object Exchange Model
32. OLAP	On Line Analytical Processing
33. OS	Operating System
34. PDF	Portable Document Format
35. PHP	Hypertext Preprocessor
36. PHPMA	PHP MyAdmin (Software)
37. PPT	Power Point
38. PR	Page Rank
39. PSQL	PostgreSQL

40. SAX	Simple API for XML
41. Sel HITS	Selective Hyper Link Induced Topic Search
42. SEO	Search Engine Optimization
43. SQL	Structured Query Language
44. TXT	Text File
45. UI	User Interface
46. URL	Uniform Resource Locator
47. WAN	Wide Area Network
48. WCM	Web Content Mining
49. WDES	Web Data Extraction and Storage
50. WICCAP	Web Information Collection Collaging And Programming System
51. WSM	Web Structure Mining
52. WUM	Web Usage Mining
53. WWW	World Wide Web
54. XHTML	Extensible Hyper Text Markup Language
55. XLS	Microsoft Excel Spreadsheet
56. XML	Extensible Markup Language
57. XPath	XML Path Language

Chapter 1

Introduction

The World Wide Web has resulted in the explosive growth of data in storage servers. The discovery and analysis of useful information from the World Wide Web becomes a practical necessity. The users make use of automated tools in finding the desired information resources, tracking and analyzing their usage patterns. These factors give rise to the need for creating server side and client side intelligent systems that can effectively mine for knowledge.

Web Mining is the extraction of interesting and potentially useful patterns and implicit information from artifacts or activity related to the World Wide Web [1]. **Web mining** can be defined as the automated discovery and analysis of useful information from web documents and services using data mining techniques [2]. It discovers potentially useful and previously unknown information or knowledge from web data.

There are several important issues, unique to the Web paradigm, that come into play if sophisticated types of analyses are to be done on server side data collections. These include the following items:

- integration of various data sources such as *server access logs*, *user registration* or *profile information.*
- resolving difficulties in the identification of users due to the *missing unique key attributes* in collected data.
- importance of identifying *user sessions* or transactions from usage data.

Web Mining methodologies can generally be classified into one of three distinct categories: **Web Content Mining, Web Structure Mining** and **Web Usage Mining** [3]. **Web Content Mining** is the process of extracting knowledge from the content of documents or their description, available on the World Wide Web [4]. **Web Structure Mining** is the process of inferring knowledge from the World Wide Web using hyperlinks and document structure [5]. **Web Usage Mining** also known as **Web Log Mining,** is the process of extracting interesting patterns in Web access logs. It analyzes navigational activities of web users [6].

The World Wide Web serves as a huge, widely distributed, global information service center for news, advertisements, consumer information, financial management, education, government, e-commerce and many other information services [7]. The web also contains a rich and dynamic collection of hyperlink information, web page access and usage information, providing rich sources for **data mining** [8]. However based on the following observations, the Web also poses great challenges for effective resource and knowledge discovery:

1. The Web comprises of voluminous information and hence it requires substantial efforts, to perform **data mining** [9].

2. **The complexity** of Web pages is far greater than that of any traditional text document collection.

3. The Web is a highly **dynamic** information source.

4. The Web serves a broad **diversity** of user communities.

5. Only a small portion of the information on the Web is truly **relevant** or useful.

There are many index-based Web search engines that search the Web, index Web pages, build and store huge keyword-based indices that help locate sets of Web pages containing certain keywords. The present work is intended to provide the necessary support for the web users in an academic environment and this is made possible by the design and implementation of algorithms for **Web mining**. A systematic analysis of the above **algorithms** has also been dealt with in this thesis report.

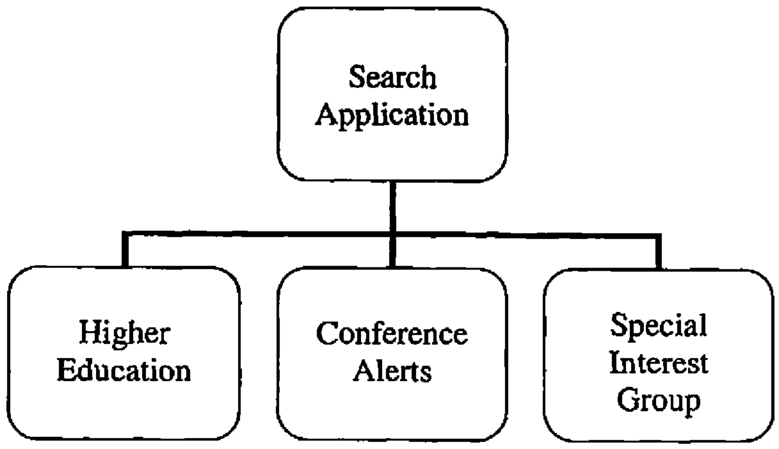


Figure 1.1: Categories of Search Application in an Academic environment

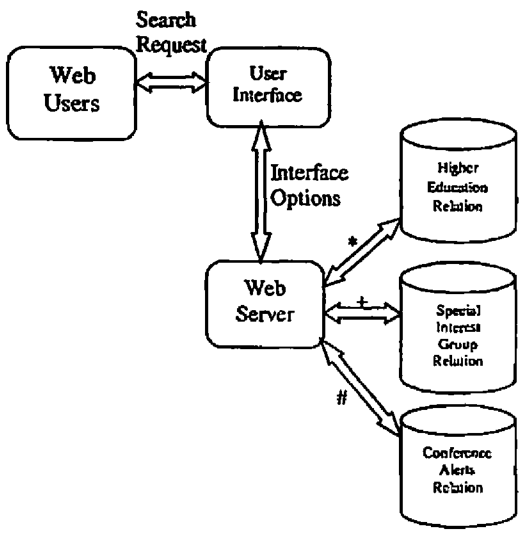
1.1 Main Functions of the Search application

The present work involves management of repository for Search Application in an academic environment. The repository is organized as a collection of files and directories. The files can be unstructured(text) or semi-structured(HTML) [10].

There is a need for the web users to query the repository using an interactive and user friendly interface. The extracted files stored in the repository have to be managed when the user wants to add, delete or modify the existing files. The user should also have an option to view the files in the directories sorted in the desired manner (by date of modification or name etc.)

Figure 1.1 shows the *Categories of Search Application in an academic environment* [11]. It involves search relating to higher education, conference alerts and special interest group.

Figure 1.2 shows the *Block Schematic of Repository for Search application in an Academic environment* . The different categories of web users access the web server via user interface. The academic database comprises of three relations each pertaining to one category [12].


Search Request
Web Users
User Interface
Interface Options
Web Server
Higher Education Relation
Special Interest Group Relation
Conference Alerts Relation
*
+
#

* represents Higher Education Details.

+ represents Special Interest Group Details.

represents Conference Alerts Details

Figure 1.2: Block Schematic of Repository for Search Application in an academic environment

1.1.1 Categories of Users

The different *Categories of web users in an Academic Environment* is shown in **Figure 1.3**. Web users are classified as Web Administrator, Faculty, General users and Students [13]. Each user has specific requirements while searching information from the web. The purpose of a repository in the present context is to serve the users and to analyze their interactions with the repository.

The main activities of these users are explained below:

1.1.1.1 Administrator:

Administrator will be interested in analyzing the log file of the application server to gather information on web usage data such as *keyword, title, link, MIME, movedto, size* and *time* corresponding to downloaded file. Administrator can monitor the frequency of interaction between the user and the web pages and perform necessary updates to the website.

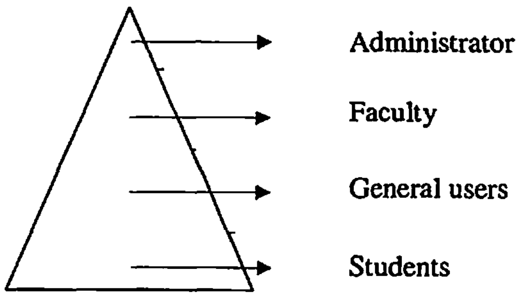


Figure 1.3: Categories of users for Search Application in an academic environment

1.1.1.2 Faculty:

Faculty will be interested in identifying workgroups and Special Interest Groups in different universities across the world. They can confine their search to a specific group like *Computer Graphics, Optimization, Data Mining* and so on. Faculty will also be interested in accessing information related to technical papers, conferences and articles.

1.1.1.3 General users:

General users will be interested in accessing the web to know the list of Conferences and upcoming events in a particular area such as Sofware Engineering, Operating Systems and so on. They can also perform search over the web using specific keywords.

1.1.1.4 Students:

Students will be interested in accessing information on higher education such as programs, courses, specialization, fee structure and stipend offered in various universities across the world. Web mining, facilitates easy access to higher education data from the updated repository. This will help students considerably in terms of saving time and effort in the search process.

An University consists of heterogeneous users such as Students, Faculty, System Administrator and Librarian. Each web user has specific requirements while searching information on the Web. Web mining helps greatly in achieving the user's preferences in less time and effort.

Web Log files record useful information about Web usage in an University's Web

Server [14]. The log file can be analyzed periodically. The time period can be specified on hourly, daily, weekly and monthly basis. The following information can be gathered from the log files of the web server:

- List of all records.
- Order by Keywords and Title.
- Order by Keywords and link.
- Order by Keywords, file type and file size.
- Error reports for identifying the problems and solving them.
- Report on file size, file type and download time.

1.2 Motivation

The World Wide Web acts as a large repository of data. It is very much necessary to analyze the navigational activities of users and also extract meaningful information from online internet sites. The present research work is motivated by the increasing importance of Web Mining in the contemporary internet applications and the main area of interest will be **Search Application** in an academic environment. It involves retrieving voluminous information from the web, analyzing the web usage patterns of different users, document structure and hyperlinks.

1.3 Problem Description

The proposed research work at the outset, would address itself to the *need* for Web Mining. The issues relating to Web Content, Structure and Usage will be identified and a class of algorithms pertaining to Web Mining will be designed, analyzed and implemented for **Search Application** in an academic environment. The proposed system will be very useful for different categories of users such as Students, Faculty, Librarian, General users and System Administrator.

1.4 Gap in Existing Research

With the fast increase in Web activities, **Web Mining** has recently become an important research topic and is receiving a significant amount of interest from both academic and industrial environments. While existing methods are efficient for the mining of frequent path traversal patterns from the access information contained in a log file, these approaches involve considerable overhead in evaluating associations in a given set of data [15].

The search engines have been successful for practical applications. However, there exists some problems relating to retrieval accuracy. Web server logs provide a rich source of information about how users access a site. When users access a sequence of web pages, they usually have a particular information seeking task in mind. The degree of accuracy is reduced when the user navigates through several links [16].

In the present days, the main problems encountered in web search are **Information mismatch** and **voluminous results.** The results often do not match the expectations of the user and moreover, they involve considerable amount of time. Most of the existing search engines perform searches that are keyword based. They get the search query from the users, search for the presence of keywords in the web documents, rank them and display the results to the user. However, it is also essential to consider the semantics of the user query and precise requirements of the user to yield reasonably good results.

The user either browses or uses the search service when he or she want to find specific information on the web. The user usually specifies a simple keyword query and the response from a web search engine is a list of pages, ranked based on their similarity to the query. However, today's search tools have the following problems:

1. **Low precision** : This is due to the irrelevance of many of the search results. The user may get many pages of information which are not really relevant to the input query.

2. **Low recall** : This is due to the inability to index all the information available on the web. Because some of the relevant pages are not properly indexed, the user may not get those pages through any of the search engines.

Some of the problems encountered by the users when interacting with the web are finding relevant information, creating new knowledge out of the information available on the web and personalization of the information. The above problem can be termed as a query triggered process (retrieval oriented) [17]. Web mining techniques address the above mentioned issues.

The current work addresses the following issues relating to Web mining:

1. **Provide** an environment for extraction and storage of web content in an organized way, associate the keywords to the extracted content using indexes, facilitate easy access by web user and provide an estimate of *<size of the downloaded files, execution time>* for the experimental setup.

2. **Model** the web as an implicit graph using input keywords and traverse the graph in a breadth first order. This reduces the search space for user specific keywords.

3. **Provide** an interface to calculate page rank using web links attribute information. This will help a web administrator to identify different types of web pages.

4. **Formulate** and implement a strategy to analyze the application specific web usage logs. It is also necessary to arrive at interestingness measures: t-weight and d-weight relating to descriptive data mining.

1.5 Objectives, Scope and Limitations

The present work is intended to meet the following objectives:

- **Specify** the functions of web content, structure and usage mining.

- **Identify** the search application where web mining can be applied effectively.

- **Extract** useful academic information from web content involving text and semi-structured data. The information pertains to research articles from journals, conference proceedings and websites.

- **Analyze** the structure information from the web (Hyperlinks, Document Structure).

- **Study and Analyze** the navigation patterns of the different types of web users. This will enhance the adaptability of the web site.

In general, web mining is applicable to all forms of data available in the web. However, the present work is limited to mining the following forms of data only: *html, pdf, ppt, doc, txt, xml* and *xls.* It considers only the web data corresponding to http protocol. Secure information handling(https:) is not included in the scope of the present work.

1.6 Methodology

The thesis has been carried out as per the following phases:

- Literature Review.
- The core algorithms pertaining to web mining support functions.
- Summary of findings.

The methodology adopted in the current work involves the following elements: problem description, algorithm, experimental setup, implementation, test scenario, analysis and discussion.

1.7 Organization of the Thesis

The **thesis** is organized into **eight chapters** followed by the **References, Appendices** and the **List of Publications.**

- **Chapter 2** gives an overview of existing Literature on *Web Mining.* Web Mining can be broadly divided into three categories: *Web Content Mining, Web Structure Mining* and *Web Usage Mining.* Web Content Mining involves exploration of unstructured and semi-structured data. In Web Structure Mining, an overview of existing algorithms: Page Rank, HITS, Trust Rank and Sel-HITS are discussed. *Web Usage Mining* primarily involves analysis of Web log files to extract meaningful information relating to access patterns of web users. The relevance of data warehouse functions in web mining is also discussed.

- **Chapter 3** deals with design and implementation of WDES Algorithm. The algorithm performs data extraction, grouping and storage functions for web user. Web users can access the web content in an organized way, by providing search assistance and thereby reduce the time and effort in the search process. The algorithm uses the search engines Google and Yahoo at the back end. The performance of the above algorithm is analyzed. The algorithm has been successfully tested for the search application in academic environment.

- **Chapter 4** deals with implementation of a web based tool for web user and Administrator. A web user can make use of navigation options for accessing web content. An academic search database is mainly used to store attributes such as *Keyword, Title, Link, File extension, File moved to* and *File size.* The database is hosted on the web server, to facilitate easier access by academic users. A Web Administrator can maintain the repository relating to data on *Higher Education, Conference Alerts* and *Special Interest Group.* The repository is organized as a collection of folders and file types, stored in the web server. The operations for the folders include *create, rename, move* and *delete.* The operations for the files include *view, upload, delete, search* and *sort.*

- **Chapter 5** deals with **Links-Traversal** algorithm for implicit graph. It makes use of Breadth First Search Strategy for traversing the hyperlinks. The timing statistics are gathered from the log messages, during the search process.

- **Chapter 6** focuses on interactive Page Rank Calculation algorithm for Web Structure Mining. It considers two main factors namely *Visibility of a link* and *Position of a link within a document.* A web user can view the page ranks for a given web universe of web pages using a standard interface.

- **Chapter 7** presents Web Usage Log Analysis algorithm using Descriptive data mining. The algorithm works in two phases namely, *Preprocessing* and *Log Analysis.* The web usage log files are first converted into text files with appropriate delimiters between various attributes, in the preprocessing phase. In the Log Analysis phase, the web usage database is created using the preprocessed text files and SQL queries are run to provide output results and summary data on web usage by various users in an academic environment. This will help web administrators in their activities relating to analysis and maintenance of web usage logs. The relation between two entities *filesummary* and *filedetails* are drawn using ER diagram. The interestingness measures: t-weight and d-weight values for the input classes, have been calculated.

- **Chapter** 8 summarizes the work carried out in this thesis, highlights specific contributions and suggests some directions for future work.

The **Appendices** consist of the following supplementary information:

- The **System Configuration.**
- Functions and their Actions for WDES: Implementation using Google.
- Functions and their Actions for WDES: Implementation using Yahoo.
- Functions and their Actions for Links-Traversal Algorithm: Java Implementation.
- Functions and their Actions for Maintain-Repository Algorithm: Java Implementation.
- Awk Script for Pre-Use Algorithm in Web Usage Log Analysis.
- Layout of forms / menus for web based tool for web user and Administrator.

The relevant articles, books and web sites are listed in the **References**. The **List of Publications** related to the thesis are given at the end.

Chapter 2

Literature Review

2.1 Introduction

The World Wide Web (WWW) continues to grow at an astounding rate in terms of the sheer volume of traffic and the complexity of web sites. The complexity of tasks such as web site design, web server design and web site navigation have increased along with this growth [18]. It is necessary to improve the efficiency and accuracy of information retrieved from the web. An important input to these design tasks is the analysis of how a web site is being used.

The World Wide Web provides every internet user with access to an abundance of information, but it becomes increasingly difficult to identify the relevant pieces of information. Research in web mining tries to address this problem by applying techniques from data mining and machine learning to Web data and documents. When a user visits a website, in the background the user leaves his impressions, usage patterns and also access patterns in the web servers log file [19]. This chapter gives an overview of literature pertaining to Web Content Mining, Web Structure Mining and Web Usage Mining.

2.2 Data Mining

Data Mining is the extraction of patterns or knowledge from large volumes of data. It is also known as Knowledge Discovery in Databases [7]. It is a multidisciplinary field from different areas like Database technology, Statistics, Visualization, Machine Learning, Pattern Recognition and so on. Knowledge discovery process consists of an iterative sequence of the steps like data cleaning, data integration, data selection, data transformation, data mining, pattern evaluation and knowledge presentation. The current work considers measures for descriptive data mining. They include t-weight, d-weight and SQL based aggregate functions.

2.3 Web Mining

The World Wide Web (WWW) is a popular and interactive medium to disseminate information today. This results in information overload. To be able to cope with the abundance of available information, users of the Web need assistance of intelligent software agents for finding, sorting and filtering the available information [20]. A Web mining system can be viewed as the use of data mining techniques to automatically *retrieve, extract, generalize,* and *analyse* information on the Web.

Web mining can be decomposed into these subtasks, namely:

1. *Resource finding:* the task of retrieving intended Web documents.

2. *Information selection* and *pre-processing:* automatically selecting and pre-processing specific information from retrieved Web resources.

3. *Generalization:* automatically discovers general patterns at individual Web sites as well as across multiple sites.

4. *Analysis:* validation and interpretation of the mined patterns.

Resource finding is the process of retrieving data that is either online or offline from the web sources like text and semi structural data like HTML/XML. The *information selection* and *pre-processing step* is any kind of transformation process of the original data, retrieved in the IR process. Machine learning or data mining techniques are used for generalization. Humans play an important role in the information or *knowledge discovery process* on the Web since the web is an interactive medium. This is especially important for validation and interpretation.

Web content mining is related to data mining as many data mining techniques can be applied in WCM. Data mining generally uses structured data, but web content mining uses semi-structured and unstructured data. Web content mining is also related to text mining as most of the web content is text. It differs from text mining based on the types of data available in the web. Text mining focuses on unstructured text [65].

2.4 Web Mining Categories

Web Mining can be broadly classified into three categories namely **Web Content Mining, Web Structure Mining** and **Web Usage Mining** as shown in **Figure 2.1**. The following sections deal with the literature pertaining to these three categories.

Web Content Mining [21] is the process of discovering useful information from the content of a web page. Web contents encompass a very broad range of data. Basically, the Web content consists of several types of data such as *textual, image, audio, video, metadata* as well as *hyperlinks.* Web content mining can be viewed from two different points of view: *IR(Information Retrieval)* and *DB (Database)* views. The goal of web content mining from the IR view is mainly to assist or to improve information finding or filtering the information to the users, usually based on either inferred or solicited user profiles. The goal of Web content mining from the DB view is mainly to model the data on the Web and to integrate them so that more sophisticated queries other than the keywords based search could be performed.

Web Content Mining involves extracting useful information from web pages. It is important to understand how data is presented on the web. Initially all data was represented in static HTML tabular form using rows and columns. With the advent of XHTML and CSS, more stylized and less strict means are used. This has made it difficult for code developers to identify data and extract them into offline tables. Images and audio-visual data can be treated separately under the broad area of multimedia data mining. In the following paragraphs, the different types of data and the techniques used to extract them are discussed.

1. **Unstructured Data**

   An *unstructured data* contains loose text and does not follow any particular grammar

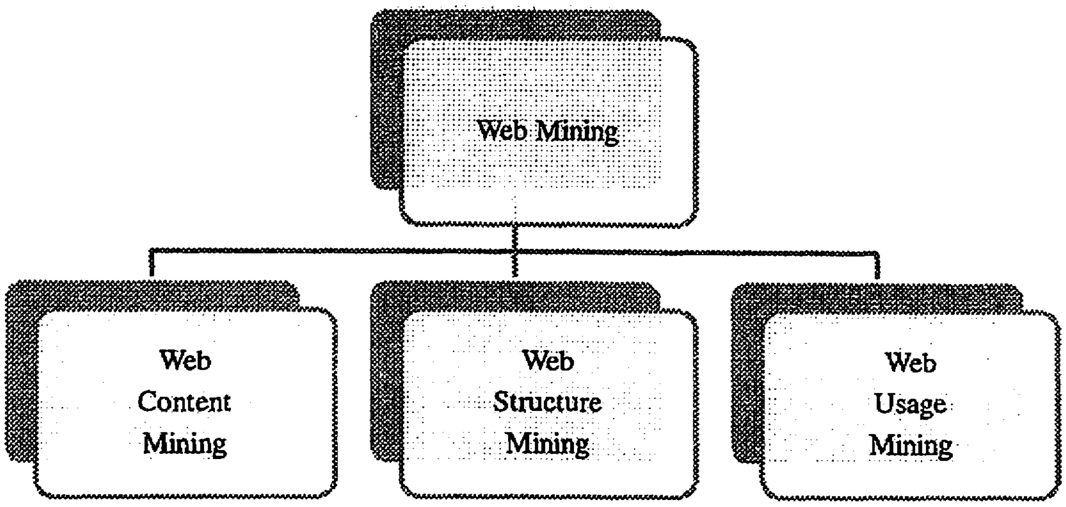


Figure 2.1: Web Mining Categories

for representation on a web page. The most common examples are text and pdf files, memos and e-mails. These information are generated through inter-organization communication [22].

2. **Semi Structured Data**

   Semi Structured data follows a very loose grammar that is not uniform across the web. The uniformity may exist within a website or within a subset of the website [23]. In Semi Structured format, the data is surrounded by identical or similar combination of tags and meta-data, thus giving it some level of structure and hierarchy. It is not possible to structure all the data available on the web. This is because the web is no longer a portal just for displaying and accessing information. It has become a marketing tool wherein every organization aims to relay its products and services with sophisticated style and animation.

There are multiple methods to extract the data available in a Semi Structured Format. OEM (Object Exchange Model) and Schema Knowledge Mining [24] allows the connection of multiple atomic OEM objects through a textual description. This method assumes previously acquired knowledge of the schema pertaining to the web page. Top Down Extraction [25] extracts all the required data by first generating a string pattern based on user specification and then matching and storing the data from the remaining web pages. WICCAP [26] (Web Information Collection, Collaging and Programming System) is a system IDE that converts the data into a tree structure which could later be modified using an XML document.

Clustering of the search results in small groups of data, based on its similarity to the search query, greatly improving the web mining process as a whole [27]. It allows discarding of a large amount of the data which is potentially useless for the input query with specified keywords. The transformation of HTML standard to XHTML as the defacto web display standard is dealt with in detail in the literature[28]. It is necessary to have a generic parser to handle both XML and HTML documents.

SAX (Simple API for XML) is a serial access parser API for XML [29]. It is an API that obtains data and analyses the text from that particular document in dynamically created web pages or web pages with interactive content. A SAX parser functions as a stream parser and it requires less memory. It traverses the document once and can quickly stop over parts not of interest. In SAX, processing of XML documents is fast and memory efficient. It is used to parse and process XML documents. DOM (Document Object Model) is a convention that is completely independent of a specific language. It is compatible with multiple platforms. DOM provides support for navigating and modifying XML documents. DOM parsers are well suited for small documents. Java language provides rich support for web content mining.

The information from the parsed documents should be presentable in a form that the user wants [30]. In order to arrange the information into a table or graph, the data must be clustered and categorized into groups of similar data. The possible representations are *tabular form* and *graphical output*. These can be accomplished by a suitable design for User Interface.

Google Scholar is a freely accessible Web search engine that indexes the full text of scholarly literature across an array of publishing formats and disciplines. It includes most peer-reviewed online journals of the world's largest scientific publishers. It allows the users to search for digital copies of online articles. It is a subset of the larger Google search index, consisting of full-text journal articles, technical reports, preprints, theses, books, and other documents, including selected Web pages that are deemed to be *scholarly* [31].

2.4.1 Data Warehouse

A **data warehouse** consists of a database (online or offline) responsible for the compilation and storage of information for a particular organization. This collection of data is then used to handle information efficiently and analyze the collected data. The data provided by the data warehouse for analysis provides information on a specific subject.

Data warehousing is commonly used by companies to analyze trends over time. In other words, companies may very well use data warehousing to view day-to-day operations, but its primary function is facilitating strategic planning resulting from long-term data synopsis.

- A **data warehouse** provides a common data model for all data of interest regardless of the data's source. This makes it easier to report and analyze information than it would be if multiple data models were used to retrieve information such as *sales invoices, order receipts, general ledger charges etc.*

- Prior to loading data into the data warehouse, inconsistencies are identified and resolved. This greatly simplifies reporting and analysis.

- Information in the data warehouse is under the control of data warehouse users so that even if the source system data is removed over time, the information in the warehouse can be stored safely for extended periods of time.

2.4.2 Web Warehouse

Web Warehouse is a repository of web based data in the form of *text, html, xml* and *relational tables.* Web warehousing is an approach to the building of computer systems which have primary functions like *identification, cataloging, retrieval, storage* and *analysis of information (in the form of text, html, xml* and *relational tables)* through the use of Web technology.

The model associated with web warehouse consists of five well-defined activities:

1. **extracting** the related data from the web pages.

2. **integrating** web information from various sources and types.

3. **transforming** or mapping the information into the appropriate schema.

4. **assisting** in the refinement of data and information schema.

5. **loading** the refined data into web warehouse.

2.5 Web Structure Mining

Web Structure Mining [32] is the process of inferring knowledge from the World Wide Web organization and links between references and referents in the web. The structure of a typical web graph consists of web pages as nodes, and hyperlinks as edges connecting related pages. Web Structure mining is the process of using graph theory to analyze the node and connection structure of a web site. It is used to discover structure information from the web and it can be divided into two kinds based on the kind of structure information used. They are *Hyperlinks* and *Document Structure*.

The first kind of web structure mining is extracting patterns from hyperlinks in the web. A hyperlink is a structural component that connects the web page to a different location. The other kind of web structure mining is mining the document structure. It is using the tree-like structure to analyze and describe the HTML (Hyper Text Markup Language) [33] or XML (eXtensible Markup Language) tags within the web page.

Web Structure Mining tries to discover the model underlying the link structures of the Web. The model is based on the topology of the hyperlinks with or without the description of the links. This model can be used to categorize web pages and is useful to generate information such as the similarity and relationship between different web sites. Web structure mining could be used to discover authority sites for the subjects (authorities) and overview sites for the subjects that point to many authorities (hubs).

Web topology has been modeled using algorithms such as HITS (Hyperlink Induced Topic Search) [34], Page Rank [35] and CLEVER [34]. These models are mainly applied as a method to calculate the page rank or relevancy of each web page. Some applications of web structure mining include the following:

- Measure the completeness of web sites by measuring the frequency of local links that reside on the same server.

- Measure the replication of web documents across the web warehouse.
- Identify mirrored sites.
- Discover the nature of the *links hierarchy* in the web sites of a particular domain.
- Study how the flow of information affects their design.

Web Structure Mining uses graph hypothesis to explore nodes and various connection structures of a web page. A usual *web graph* has *web page* that is represented as a node, and hyperlinks are represented as edges, that are connected between two associated pages.

The Web structure terminologies, used in this thesis are listed below:

- *Web-graph*: A directed graph representing a web.
- *Node*: Web pages are nodes of the web graph.
- *Link*: A hyperlink is a directed edge of web graph.
- *In-degree*: No. of different links that end at a node p is the in-degree of the node p.
- *Out-degree*: No. of different links that are originating from node p to other nodes.

2.5.1 Web Structure Mining Algorithms

The two best known algorithms for web structure mining are **HITS** and **Page Rank**. Page Rank is used in the highly successful Google search engine. The heuristic underlying both of these approaches is that pages with many inlinks are more likely to be of high quality than pages with few inlinks, given that the author of a page will presumably include in it links to pages that he believes are of high quality [36].

Given a query(set of words or the query terms) **Page Rank** computes a single measure of quality for a page at crawl time and it greatly improves the results of web search by taking into account the link structure of the Web [37].

The following sections give an overview of the existing **Web Structure Mining** algorithms:

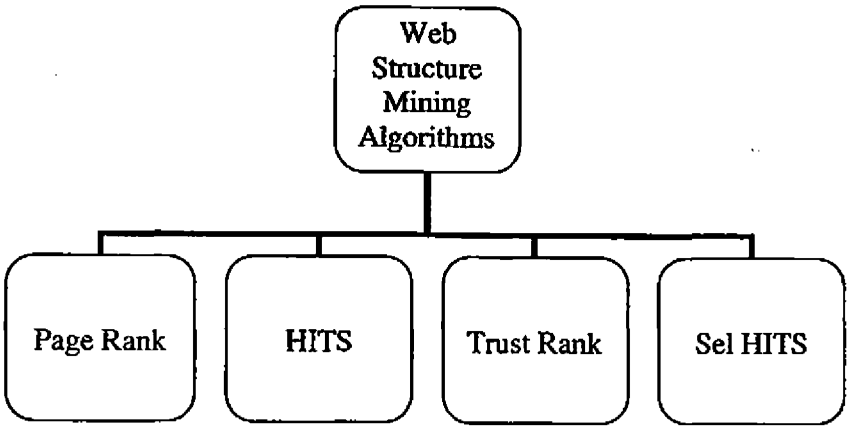


Figure 2.2: Structure Mining Algorithms

- PageRank.
- Hyperlink Induced Topic Search(HITS).
- TrustRank.
- Selective HITS(Sel-HITS).

Figure 2.2 shows the Structure Mining Algorithms.

2.5.2 Page Rank Algorithm

PageRank is a link analysis algorithm, named after Larry Page. The Internet Search Engine Google, assigns a numerical weighting to each element of a hyperlinked set of web pages. The numerical weight that it assigns to any given element E is also called the PageRank of E and is denoted by PR(E). The page rank within the set measures the relative importance of web pages.

PageRank can be explained as a "ballot" for all the webpages in the world. It indicates the importance or relevance of a web page. A hyperlink to a page is taken as a vote of support. The PageRank of a page is defined recursively and depends on the number, as well as the rank of all the pages that link to it (in-degree). A page that is pointed at by many other pages with high ranks receives a high rank itself. If there are no or very few links to a web page, then there is no support for that page and it ends up getting a low PageRank [38] [39].

The formula used for the calculation of a page using PageRank is given by eqn[1]:

$$PR(A)=(1-d)+d\ (PR(T_1)/C(T_1)+...+PR(T_n)/C(T_n)) \quad\text{............}\quad [1]$$

where,

PR(A) is the PageRank of page A.

$PR(T_i)$ is the PageRank of pages T_i which link to page A.

$C(T_i)$ is the number of outbound links on page T_i.

d is a damping factor and can be set between 0 and 1.

The rank of a certain page A depends on the ranks of all the pages T_i which have outgoing links pointing to page A, divided by their total number of outgoing links in those pages. *d* is known as the *damping factor.* The PageRank algorithm gives individual ranks to all the pages and not to websites as a whole, i.e. each page of a certain website has its own Page Rank. Also, referring to the formula, page ranks of pages T_i do not affect the page rank of page A uniformly. This is because the ranks are divided by the total number of outgoing links in each page. These weighted page ranks are then added up and multiplied to the damping factor. At any step, the probability that the person will continue is given by *damping factor d.* Various studies have tested different damping factors, but it is generally assumed that the damping factor will be set around 0.85 [4]. The Page Rank Algorithm finds applications in *Searching, Traffic Estimation* and *User Navigation.* Some additional factors influence PageRank algorithm [31]. They are **Visibility of a link, Position of a link within a document, Distance between Web pages, Importance of a linking page and up-to-dateness of a linking page.** Google utilizes many factors to rank search results including standard IR measures, proximity and anchor text.

2.5.3 The Random Surfer Model

PageRank was developed based on a model known as the **Random Surfer Model** [40]. Here, the probability that a person may arrive at a certain page by randomly clicking on hyperlinks on every page he visits, is a very good measure of a page importance or rank. Hence, PageRank of a page can also be interpreted as a probabilistic value. Higher the chances of a person arriving at a certain page (by random clicking), higher is the page's PageRank.

2.5.4 Hyperlink Induced Topic Search (HITS) Algorithm

Hyperlink Induced Topic Search is an iterative algorithm for mining the Web graph to identify topic **hubs** and **authorities**. **Authorities** are highly ranked pages for a given topic. **Hubs** are pages with links to authorities [40]. The algorithm takes as input search results returned by traditional text indexing techniques, and filters these results to identify hubs and authorities. The number and weight of hubs pointing to a page determine the page's authority. The algorithm assigns weight to a hub based on the authoritativeness of the pages it points to.

HITS Algorithm has been developed by John Kleinberg. It views a website as having content and links in their bodies. It gives two scores to each web page, **Authority Score** and **Hub Score. Authority Score** is taken to be a measure of the importance or relevance of the content of the web page. **Hub Score** is taken to be a measure of the importance of the outbound links provided in the webpage.

In this algorithm, the first step is to get the set of results to the search query. Further computation is performed only on the results, not across all the Web pages [41]. Authority and Hub scores are calculated in terms of one-another in a mutual recursion. An authority score of a certain page is calculated as the sum of the hub values that point to that particular page. A hub value of a page is the sum of the authority values of the outbound pages. The result of this idea is that after all the iterations are done, a page with a very high authority score would mean that its content is very important and relevant to user query as many pages point to it. This in turn increases the hub scores of the pages which point to it. Similarly, a very high hub score would mean that the links embedded in a page are of great importance (they point to pages which contain important information/content). This in turn increases the Authority scores of the outbound pages.

2.5.5 TrustRank Algorithm

TrustRank Algorithm [13] is built up on the concepts of PageRank but is a little more reliable. It semi-automatically separates useful pages from spam, hence making the results more relevant and reliable. Many websites often have very high PageRanks which they do not really deserve. The designers take to certain tricks to achieve higher-than-deserved rankings. Manual examination of all the pages in the world is not practical.

This algorithm involves selecting a small set of seed pages which are manually examined or analyzed by an expert. After analysis, if they are marked as reliable, then they qualify as the seed set. After recognition, a crawl extends outwards (through links to other neighbouring pages) to other pages which also can be deemed as trustworthy. As it moves further and further away from the seed set, the reliability of the pages diminish. The advantage of this algorithm is that it returns results which are more reliable and relevant to user query. It also avoids the appearance of spammed pages in the list of results.

2.5.6 Selective HITS (Sel-HITS) Algorithm

The Sel-HITS Algorithm [42] is an upgraded version of the original HITS Algorithm developed by Kleinberg. It involves two data sets known as:

- **Root Set**: Set of relevant pages from user query using some existing search algorithm.
- **Base Set**: Set including all pages in one-link neighbourhood of root set.

Unlike in HITS, it involves selective expansion of the root set after a small modified Hub and Authority Score calculation. The main difference lies in the fact that in HITS algorithm, the expansion is not selective and hence, there is no guarantee of irrelevant pages being included in the results. On the contrary, in Sel-HITS, due to the selective expansion process, topic relevance is maintained [43].

The selective expansion process can be described as follows:

- After the Hub and Authority values of the root-set are computed, about 20 hubs and 20 authorities are selected for expansion.

- This selective-expansion procedure drastically reduces size of the base set and avoids topic drift as irrelevant pages are not added to the root set.

- The results are consistent with regard to the interpretation of the query.

It is a process that is used to extract important information from history of users (server logs). It is mainly used for finding what users search on the internet. The data depends upon user to user. For example, most of the users would be looking for text data, but some would be interested in multimedia data.

The typical sources of data include the following:

- Server Access Logs, Referrer logs and client-side cookies.

- User profiles.

- **Meta data**: page attributes, content attributes and usage data.

2.6 Web Usage Mining

Web Usage Mining, [21] also known as **Web Log Mining**, is the process of extracting interesting patterns in web access logs. Web Usage Mining is the application of data mining techniques to usage logs of large web data repositories in order to produce results that can be used in the design tasks.

In web usage mining, an application uses data mining to analyze and discover interesting patterns of user's usage data on the web. The usage data records the user's behavior when the user browses or makes transactions on the web site. It is an activity that involves the automatic discovery of patterns from one or more Web servers. Organizations often generate and collect large volumes of data. Most of this information is usually generated automatically by web servers and collected in a server log. Analyzing such data can help these organizations to determine the value of particular customers, cross marketing strategies across products and the effectiveness of promotional campaigns. The first level web analysis tools simply provided mechanisms to report user activity as recorded in the servers. Using such tools, it was possible to determine information such as the number of accesses to the servers, the times or time intervals of visits, the domain names and the URLs of users of the Web server. However, in general these tools provide little or no analysis of data relationships among the accessed fields and directories within the web space. Now more sophisticated techniques for discovery and analysis of patterns are emerging. These tools fall into two main categories: **Pattern Discovery Tools** and **Pattern Analysis Tools** [44].

Web usage mining tries to analyze the sessions and access patterns of web users. While web content and structure mining utilize the real or primary data on the web, Web usage mining mines the secondary data derived from the interactions of the users while interacting with the web. The web usage data includes data from *web server access logs, proxy server logs, browser logs, user profiles, registration data, user sessions or transactions, cookies, user queries, book mark data, mouse clicks* and *scrolls* and any other data such as the result of interactions.

Web usage mining facilitates the discovery of *web access patterns.* With web usage mining, the user log can be analyzed. Some patterns about user behavior can be obtained from usage logs and then these patterns can be turned into a user profile. The user profiles are then utilized to filter incoming articles for the individual. The profiles can be constructed using a variety of learning techniques including the vector space model, genetic algorithm and the probabilistic model or clustering. A System Administrator can extract and analyze data recorded in web server log files by means of a script or program. There are a lot of commercial tools and programs intended to extract and analyze data recorded in web server log files.

Web usage mining can be classified depending on the kind of usage data [45]. They are:

1. **Web Server Data** User logs are collected by the web server. They typically include IP address, page reference and access time.

2. **Application Server Data** Commercial application servers such as Weblogic and StoryServer have significant features to enable e-commerce applications to be built on top of them with little effort. A key feature is the ability to track various kinds of business events and log them in application server logs.

3. **Application Level Data** New kinds of events can be defined in an application and

logging can be turned on for them generating histories of these events. It must be noted, however, that many end applications require a combination of one or more of the techniques applied in the above categories.

The main steps in web usage mining comprises of *data preparation, pattern discovery, pattern analysis* and *visualization* [46]. While extracting simple information from web logs is easy, mining complex structural information is very challenging. Data cleaning and preparation constitute a very significant effort before mining can even be applied. The relevant data challenges include the following items: elimination of irrelevant information such as image files and cgi scripts, user identification, user session formation, and incorporating temporal windows in user modeling. After all this pre-processing, one is ready to mine the resulting database [47].

Most of the existing web analysis tools provide mechanisms for reporting user activity in the servers and various forms of data filtering. Using such tools, it is possible to determine the number of accesses to the server and the individual files within the organization's web space, the times or time intervals of visits, domain names and the URLs of users of the web server.

The application of web usage mining can be classified into two main categories;

- Learning a *user profile*, or user modeling in adaptive interfaces (personalized).
- Learning user *navigation patterns* (impersonalized).

Web based learning environments are now extensively used as integral components of course delivery in tertiary education. To provide an effective learning environment, it is important that educators understand how these environments are used by their students [48].

A Web server access log contains a complete history of files accessed by clients. Most WWW access logs follow the Common Log Format specified as part of the HTTP protocol. A log entry, following this standard, contains the client *IP address, user id, access time, request method,* the *URL of the page accessed*, the protocol used for data transmission, an error code and the number of bytes transmitted. The above is the background against which the proposed research work will investigate the objectives listed earlier.

2.7 Summary

This chapter dealt with an overview of existing literature with respect to web mining and the three categories namely Web Content Mining, Web Structure Mining and Web Usage Mining. Web Content Mining involves exploration of unstructured and semi-structured data. In Web Structure Mining, an overview of existing algorithms Page Rank, HITS, Trust Rank and Sel-HITS are discussed. *Web Usage mining*, primarily involves analysis of web log files to extract meaningful information relating to access patterns of web users. The relevance of data warehouse functions in web mining has been discussed.

Chapter 3

Design and Implementation of Algorithm for Extraction and Storage of Web Content

3.1 Introduction

The amount of information available in the web increases continuously and therefore requires an accurate textual representation of the investigated web pages. Advertisements and navigational elements complicate the task of extracting the web page's content. Web content extraction is concerned with extracting the relevant text from web pages by removing unrelated textual noise like advertisements, navigational elements, contact and copyright notes [49]. This chapter focuses on the design, analysis and implementation of algorithm for extraction and storage of web content.

Most web extraction tools include a web crawler. Web crawling refers to the recursive process of downloading web pages from a set of URL's, extracting the link found within them and adding them to the set of links waiting to undergo the same process. The crawled pages are then indexed according to content. These files are then served to the users based on the user's keywords or query.

Web crawling involves searching a very large solution space which requires a lot of time, hard disk space and lot of usage of resources in general. Hence using existing search engines as the back-end is a more practical approach to web content extraction. It gives the opportunity for the user to store the search results in a local server or public directory. This way, searches involving the same keywords can be performed locally rather than by repeated searches and frequent downloads. This saves time and resources to a great extent.

Web content mining extends the work performed by the basic search engines. The focus is on extraction and storage of web content. Both Google and Yahoo were chosen for back end operations since they practically used by many web users [67]. Google is fast, reliable uses Page Rank algorithm to search and rank the web pages. It also provides additional facility like Google Scholar which is used widely in academics. Yahoo has been used by many web users.

3.2 Algorithm: Web Data Extraction and Storage (WDES)

This section deals with formulation of the algorithm for extraction and storage of web data. The types of data considered include the following *html, pdf, ppt, txt* and *doc* formats.

Inputs:

Keywords, Number of links.

Output:

Downloaded files in respective folders, search log file.

Procedure:

1. Start.
2. Initialization step involves reading the configuration details.
3. The user specifies the search engine (Google / Yahoo) for backend operations.
4. The input parameters can be specified using GUI / text file.
5. For keywords varying from 1 to n do
6. For links varying from 1 to m do
    (a) Download the contents from (keyword, link) and save the contents in appropriate folder.

(b) Save output messages for (keyword, link) to the log file.

7. If any more queries are to be searched, go to step 4.

8. Stop.

3.3 Time Complexity

The worst case time complexity for WDES algorithm is shown in **Table 3.1**. The total download time can be calculated as:

Table 3.1: Time Complexity for WDES

Steps	Computing Time	Explanation
1 and 2	O(c_1)	c_1 is a constant and it corresponds to initialization procedure
3	O(1)	Choosing a browser involves a single basic operation
4	O(c_2)	Reading of user inputs. c_2 is a constant and it corresponds to the reading of user parameters.
5-8	O(X)	corresponds to total download time

$$Total\ download\ time = Transfer\ time + Local\ Storage\ time$$

For all the files pertaining to the given set of keywords, the total download time can be calculated as a summation of the transfer time and local storage time.

$$O(X) = n * \sum_{i=1}^{m} \left(\frac{size(i)}{rate(i)} + storage(i) \right)$$

where,

$size(i)$ = Size of the i^{th} file that is transferred.

$rate(i)$ = Data transfer rate, for i^{th} file that is transferred.

$storage(i)$ = Local storage time, for the i^{th} file that is transferred.

n = number of keywords given as input by the user.

m = number of links.

The expression corresponding to the worst case time complexity is

$$O(c_1) + O(1) + O(c_2) + O(X)$$

The significant term in the above expression is $O(X)$ and is regarded as the worst case time complexity.

3.4 WDES Implementation using Google

The *Implementation of WDES using Google* is shown in **Figure 3.1**. The keywords are taken as input from the user, either from the file or GUI. It generates URL from the search query given and the search options selected by the user. Using Wget software, the files are downloaded [50]. GNU Wget is a free software package for retrieving files using HTTP, FTP. It is a non-interactive command line tool that is called from other programs [51]. The downloaded search results in HTML form is converted to XML using HTML Cleaner [52]. HTML on the web is not well formed and not suitable for further processing because it contains missing quotes and unclosed tags in the document. HTML Cleaner is an open source HTML parser written in Java which accepts HTML document and corrects individual elements and produces well formed XML [53]. XPath tools are used to find the links in the XML document [54] [60]. The initial link is chosen and Wget is used to download the document corresponding to specified link and save the output messages to a file. The file extension of the downloaded file is determined from the MIME type [55] [56]. The file is placed in appropriate folder according to file extension. The keywords can be classified into four categories of Higher Education, Conference Alerts, Special Interest Group and general users. For each of the above categories the downloaded files are stored in their corresponding folders separately. The downloaded files can be viewed by the user from the appropriate folders either directly in the interactive mode or by using the Navigation option search as discussed in Chapter-4.

This algorithm is used for extraction and storage of web content and its details are given below:

Inputs:

Keywords, Number of links.

Output:

Downloaded files in respective folders.

Procedure:

1. Generate the search URL from the keywords and various options provided by the user.

   Action: User enters keyword and presses search button. A thread is spawned to

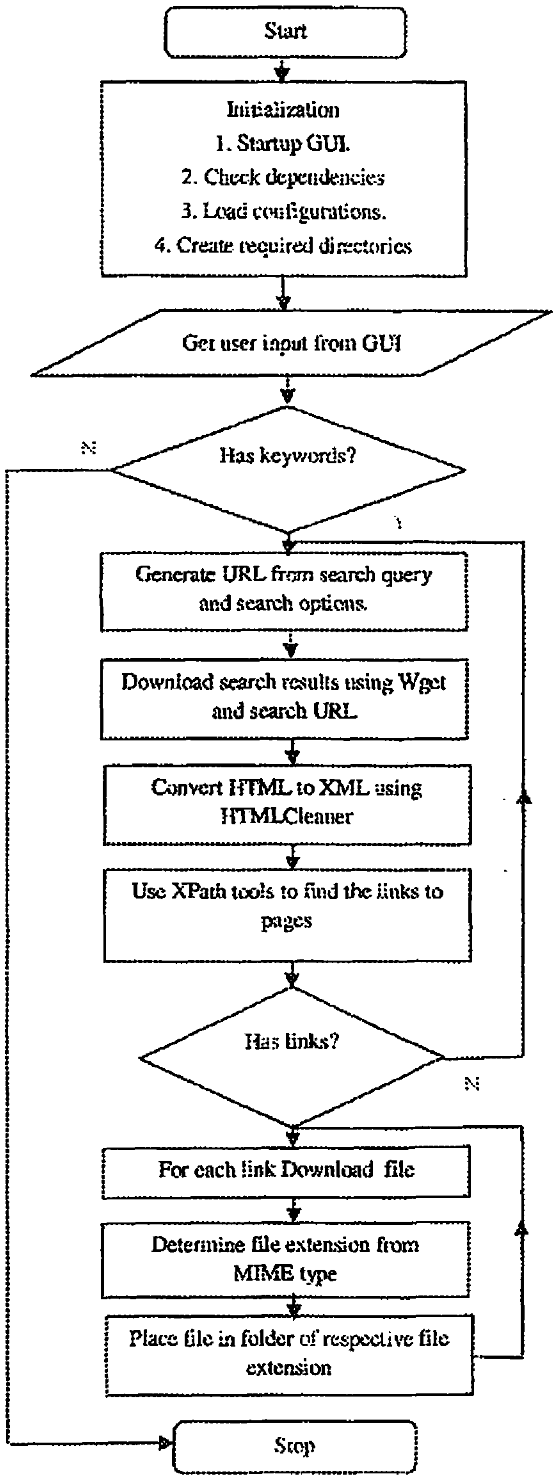

Start
Initialization
1. Startup GUI.
2. Check dependencies
3. Load configurations.
4. Create required directories
Get user input from GUI
Has keywords?
N
Y
Generate URL from search query and search options.
Download search results using Wget and search URL.
Convert HTML to XML using HTMLCleaner
Use XPath tools to find the links to pages
Has links?
N
For each link Download file
Determine file extension from MIME type
Place file in folder of respective file extension
Stop

Figure 3.1: WDES: Implementation using Google

do the search. All the steps here onwards happen within the thread.

2. Use Wget to retrieve the search results as HTML file using the search URL.

   Action: From the GUI options, a link to search engine google with the search options is generated. For example the link would be:

   `http://www.google.com/search?hl=en&safe=active&=web+content+mining&num=10&start=0`

   where

   - hl = en denotes the language preference is English.
   - safe = active sets safe search.
   - q = keywords.
   - num = number of search results to show.
   - start = indicates from which search result to start from.

   To display the links from 10 to 30, num has to be set to 20.

3. Use HTMLCleaner (an external party utility), to convert the HTML to a clean XML document.

   Action: Send the link to **Wget** and Wget will extract the resulting html page which contains the search results.

4. Use XML XPath tools to find the links in the document and store them.

   Action: HTMLCleaner is used to convert HTML to **XML**.

5. Choose each link and use Wget to download the documents. Also save the output messages to a file.

   Action: Use **XPath** to extract the links (search results) from the Google search page.

6. Determine the MIME type of the downloaded file from the output file. And then check if the type is associated with a file extension from req or mime.txt.

   Action: This file contains the mapping between common MIME types and their file extension. Some of them are HTML, PDF, PPT, DOC, TXT, XML, XLS.

7. If file extension is found, create a folder for the extension (if it does not exist) and place them in the folder. If file extension was not recognized, place the downloaded file into *Unrecognized* folder.

   Action: Download the link with Wget and retrieve the **MIME** type of the file from the Wget output (MIME type is given along with the **HTTP** header while downloading the file. The Wget output is saved onto a log file which is retrieved, after download, by WDES).

8. Repeat the steps 5 to 7.

   Action: Determine the **file extension** of the downloaded file from the MIME type. MIME-to-file extension mapping has been pre-written in a file. Only a few, and most important types are currently recognized.

9. If keywords have been specified by file input, repeat this process for other keywords (Steps 1 to 7).

   Action: Rename the file with the extension and move the file to a **folder** named after the respective extension. If the folder does not exist, it is created on the run.

10. Goto step 1 and check if any more queries are to be searched. When no more queries remain, quit the thread.

Log messages are shown during the entire search process and the time taken for the process to complete is also shown. When the application is closed, the log messages of the session are saved onto a file for later use (Web Usage Log Analysis: Chapter 7). Thus, the complete history of all the searched and downloaded links can be tracked. The log messages can be grouped into two categories: *detailed log information* and *summary log information*. The detailed log information comprises of the following fields *Key, Title, Link, Mime, Extension, Moved* and *Size*. The summary log information comprises of the following fields: *Key, Time* and *Downsize*.

The WDES algorithm has been implemented using JavaSE Runtime 6 and GNU Wget 1.12 [50]. The details of the Java classes along with their descriptions is shown in **Appendix B.**

Figure 3.2 shows the *User Interface with various search options.* There are two

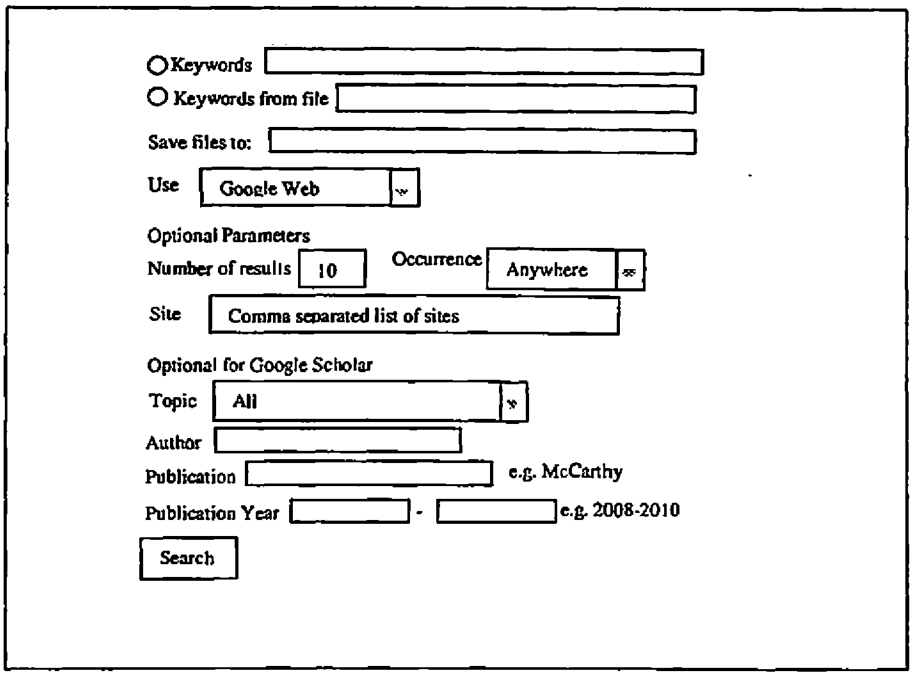

Keywords
Keywords from file
Save files to:
Use Google Web
Optional Parameters
Number of results 10 Occurrence Anywhere
Site Comma separated list of sites
Optional for Google Scholar
Topic All
Author
Publication e.g. McCarthy
Publication Year - e.g. 2008-2010

Figure 3.2: User Interface with various search options

types of parameters namely **Mandatory** and **Optional**. The various GUI options are explained in the following section.

Mandatory Parameters

1. **Keywords** (either by typing the query in the text field provided or by specifying text file containing queries where each query is on a new line): This query is sent to Google as *q* URL parameter.

2. **Save files to:** User specifies a path to a directory on the local machine. All files (log files, downloaded files etc) are saved into this directory.

3. **Use:** User is given an option to choose between Google Web search and Google Scholar.

Optional Parameters

1. **Number of Results:** Number of search results to be returned. Default is 10. This parameter is sent as *num* Google URL parameter. Even though this parameter is optional, the program always sends this parameter to Google.

2. **Occurrence:** User can specify where to find occurrences of the search query. There are 4 options, namely, *On Page Title*, *Page Text*, *On Links To Page* and *Anywhere*. *Anywhere* is the default and it includes occurrences on page title, text and links to pages.

3. **Site:** User can specify the web sites that must be searched. Each web domain is separated by a comma. By default, all and any websites are included in the search.

Options for Google Scholar

1. **Topic:** User can search for a specific field of interest, say mathematics or medical science. The Google URL Parameter for this option is *as-subj*.

2. **Author:** User can specify the author of the book or article which he is searching for.

3. **Publication:** Publication of the book.

4. **Publication Year:** Search for publications between two specified years. Example, 2008 to 2010.

3.5 WDES Implementation using Yahoo

This section deals with implementation of WDES algorithm using Yahoo. The algorithm is outlined in the following steps.

Inputs:

A text file containing keywords to be searched.

Output:

Saved search results, search log file.

Procedure:

1. Create a text file containing the keywords to be searched.

2. Keywords are read one by one from the file.

3. Each keyword is passed to search function.

4. The keyword is integrated into Yahoo URL by the search function.
5. The resulting address is passed into getLinks() function.
6. The function creates a document object using Jsoup library. It is an HTML parser for Java.
7. Web links corresponding to the search results are extracted from DOM.
8. Each result link is identified using id.
9. Links are stored in a queue.
10. The function SavePage() takes a link at a time and creates a URL object out of it.
11. The page is read block by block and saved in a file.
12. Size of the file created is calculated.
13. Details of transaction is written onto log file.
14. Steps 3 - 13 are repeated for every keyword in the text file.

The WDES algorithm has been implemented using Yahoo as a back end search engine. The details of the Java classes along with their descriptions are shown in **Appendix C**.

3.6 Test Scenario

The Test Scenario involved many test inputs involving keywords. The keywords considered here include the following: *Higher Education*, *Special Interest Group* and *Conference Alerts*.

Figure 3.3 illustrates the *Log messages for WDES algorithm using Google* recorded during the entire search process and the total time taken for extraction and storage process. When the application is closed, the log messages of the session are saved onto a file for later use. Thus the complete history of all the searched and downloaded links can be tracked. The downloaded file size is also displayed in log file.

```
#Wed Mar 14 13:39:08 GST 2012
Current Directory: C:\Documents and Settings\sjeyalatha\Desktop\SearchAssist2
************Starting download************
Search URL:
http://www.google.com/search?hl=en&safe=active&q=introduction+to+graphic+programming&num=50&s
tart=0
1. Title: Notes for a Computer Graphics Programming Course
Link: http://www.cs.csustan.edu/~rsc/CS3600F00/Notes.pdf
-----------------
mime: application/pdf
File extension: pdf
File moved to 'K:\Users\Mogral\Desktop\search data\pdf\Notes for a Computer Graphics Programming
Course.pdf'
File size ~ 3444 KB
2. Title: Introduction to SAS IML Graphics Programming
Link: http://javeeh.net/sasintro/intro110.html
-----------------
mime: text/html
File extension: html
File moved to 'K:\Users\Mogral\Desktop\search data\html\Introduction to SAS IML Graphics
Programming.html'
File size ~ 5 KB
3. Title: CS 112 Introduction to Computer Graphics Programming Assignment ...
Link: http://www.ics.uci.edu/~majumder/CG/assn/W11-PA3.pdf
-----------------
mime: application/pdf
File extension: pdf
File moved to 'K:\Users\Mogral\Desktop\search data\pdf\CS 112 Introduction to Computer Graphics
Programming Assignment ....pdf'
File size ~ 16 KB
************Done************
Total time taken (in milliseconds)=189807
Total download size=17982 KB
```

Figure 3.3: Log Messages for WDES Algorithm using Google

Figure 3.4 illustrates the *Log messages for WDES algorithm using Yahoo* recorded during the entire search process and the total time taken for extraction and storage process. The log messages contain the following details like Title, Link, mime, File extension, File moved to, File size. It also displays the total time taken (in milli seconds) for a particular keyword and total download size in kB.

Table 3.2 shows the *Applicability of WDES Algorithm* in an University Environment. It specifies the tasks carried out by various categories of users.

```
Keyword      Links      Time Taken      Total Size
#Thu Mar 29 09:03:05 GST 2012
web content mining      10      93118ms      4994 KB
************End of session************
***************Session****************
#Thu Mar 29 09:03:34 GST 2012
web structure mining      10      29879ms      533 KB
************End of session************
***************Session****************
#Thu Mar 29 09:04:04 GST 2012
web content management      10      29099ms      507 KB
************End of session************
***************Session****************
#Thu Mar 29 09:04:43 GST 2012
database management      10      39433ms      963 KB
************End of session************
```

Figure 3.4: Log Messages for WDES Algorithm using Yahoo

Table 3.2: User Categories and Tasks

User Categories	Tasks
Student	Literature Review
Faculty	Lecture Notes
Librarian	Building of Academic Search Repository
Web Administrator	Usage Analysis from Log files
Staff	Search on specific topics

3.7 Discussion

The experimental setup and the observed results for the implementation of WDES algorithm is dealt with in this section. The back end search engines considered in the present work are Google and Yahoo. These two search engines have been chosen due to their efficient operations for informational queries and also their widespread usage of web user community. The relevance of information for the downloaded content is taken care of using Search Engine Optimization technique of backend search engines. The keywords for the search process using the above search engines are shown in **Tables 3.3** and **3.4** respectively. The number of links considered in the search process for each keyword has been 10. The internet connection used in the experimental setup has been 100Mbps, as available at BITS Pilani, Dubai Campus.

Table 3.3: Experimental Results for WDES using Google

Keywords	No. of links	Size of downloaded files in kB	Total time taken in milliseconds
Web Mining	10	2917	431343
Database Management	10	884	300317
Web Content Management	10	592	256646
Web Content Mining	10	864	259767
Web Structure Mining	10	5585	494028

Table 3.4: Experimental Results for WDES using Yahoo

Keywords	No. of links	Size of downloaded files in kB	Total time taken in milliseconds
Web Mining	10	843	36645
Database Management	10	4944	91416
Web Content Management	10	534	29173
Web Content Mining	10	566	24362
Web Structure Mining	10	963	37771

The experiment has been conducted at different time intervals. The number of iterations considered has been twenty. The average values are considered for analysis, for the parameters (size of the downloaded files, total time taken for download).

3.7.1 Observation and Inference for WDES: Implementation using Google

The experimental results for Web Extraction and Storage functions using Google search engine as the backend is shown in **Table 3.3**. The above results are significant in terms of providing an estimate of *<size of the downloaded files, execution time>* for the experimental setup. The corresponding performance graph is shown in **Figure 3.5**, where X axis represents the *Size of downloaded files in kB* and Y axis represents the *Total time taken in milliseconds.* The graph is nearly linear and it is bounded by function O(X) as explained in **Section 3.3** (Time complexity). The graph has been plotted using the options *smooth, sbezier* in GNU plot.

For size of dowloaded files in kB varying from 560kB to 1890kB, the obtained values for total time increases from 281,250 milliseconds to 337,600 milliseconds, i.e. it increases at the rate of 21184 milliseconds for every 500kB. For size of dowloaded files in kB varying from 1890kB to 4100kB, the obtained values for total time increases from 337,600 milliseconds to 440,000 milliseconds, i.e. it increases at the rate of 23552 milliseconds for every 500kB. For size of dowloaded files in kB varying from 4100kB to 5600kB, the obtained values for total time increases from 440,000 milliseconds to 490,000 milliseconds, i.e. it increases at the rate of 16666 milliseconds for every 500kB. From the graph, it can be inferred that the total time increases in near linear fashion, with the size of the downloaded files corresponding to the given set of keywords for the given experimental setup and parameters.

3.7.2 Observation and Inference for WDES: Implementation using Yahoo

The experimental results for Web Extraction and Storage functions using Yahoo search engine as the backend is shown in **Table 3.4**. The above results are significant in terms of providing an estimate of *<size of the downloaded files, execution time>* for the experimental setup. The corresponding performance graph is shown in **Figure 3.6**, where X axis represents the *Size of downloaded files in kB* and Y axis represents the *Total time* *taken in milliseconds.* The graph is nearly linear and it is bounded by function O(X) as explained in **Section 3.3** (Time complexity). The graph has been plotted using the options *smooth,sbezier* in GNU plot. For size of dowloaded files in kB varying from 566kB to 2850kB, the obtained values for total time increases from 24362 milliseconds to 60500 milliseconds, i.e. it increases at the rate of 7911 milliseconds for every 500kB. For size of dowloaded files in kB varying from 2850kB to 4944kB, the obtained values for total time increases from 60500 milliseconds to 91416 milliseconds, i.e. it increases at the rate of 7382 milliseconds for every 500kB. From the graph, it can be inferred that the total time increases in near linear fashion, with the size of the downloaded files corresponding to the given set of keywords for the given experimental setup and parameters.

3.8 Summary

This chapter dealt with the design and implementation of WDES algorithm for data extraction, grouping and storage functions. The above algorithm uses the search engines Google and Yahoo at the back end. Web users can access the web content in an organized way, by providing search assistance and thereby reducing the time and effort in the search process. The performance for the above algorithm is analyzed and it has been successfully tested for the search application. The User Interface with search options has been implemented using Java. Files with different extensions have been downloaded and stored in their respective folders. The experimental results for web extraction and storage functions have been tabulated and plotted.

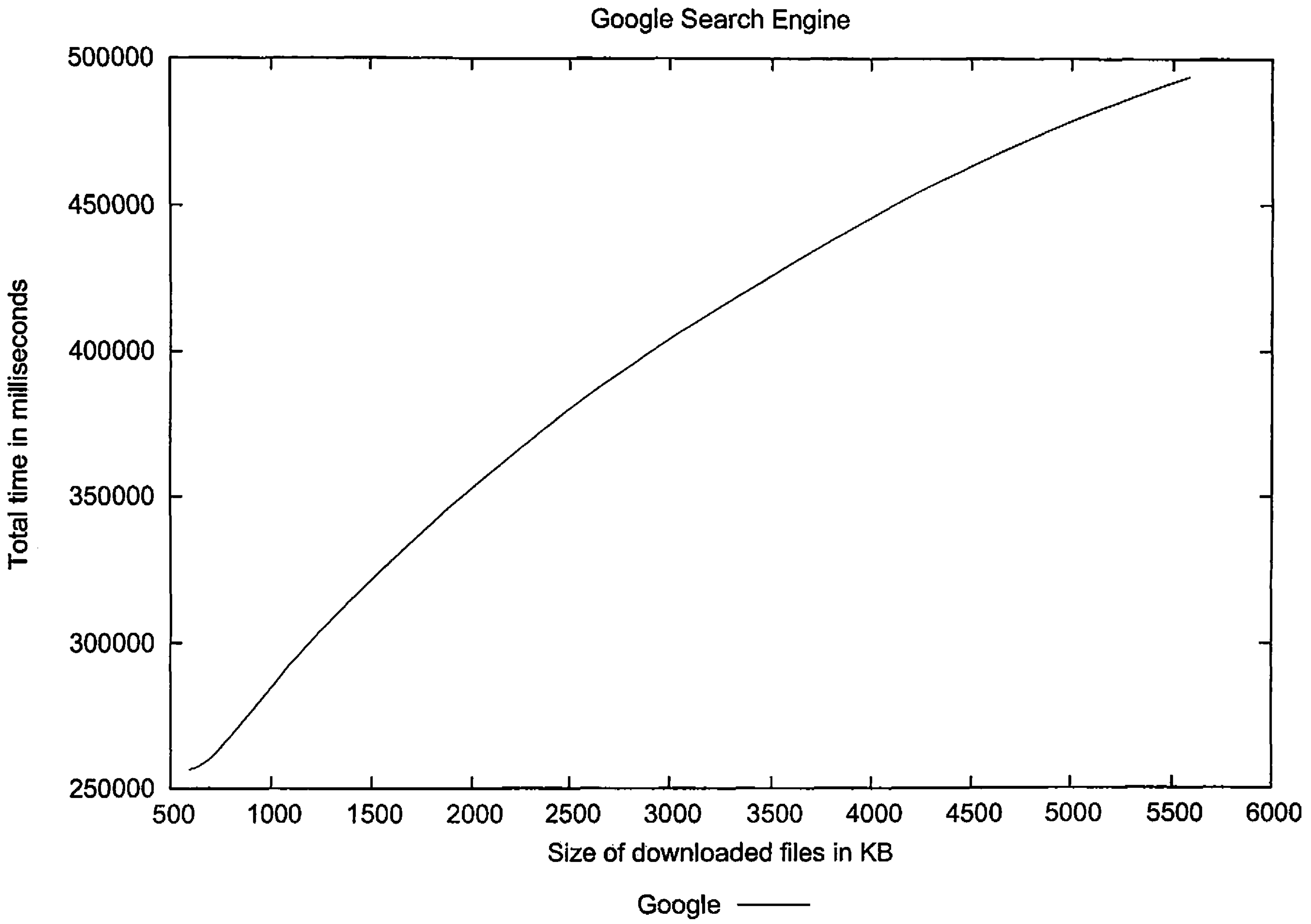


Figure 3.5: Performance Graph for WDES using Google

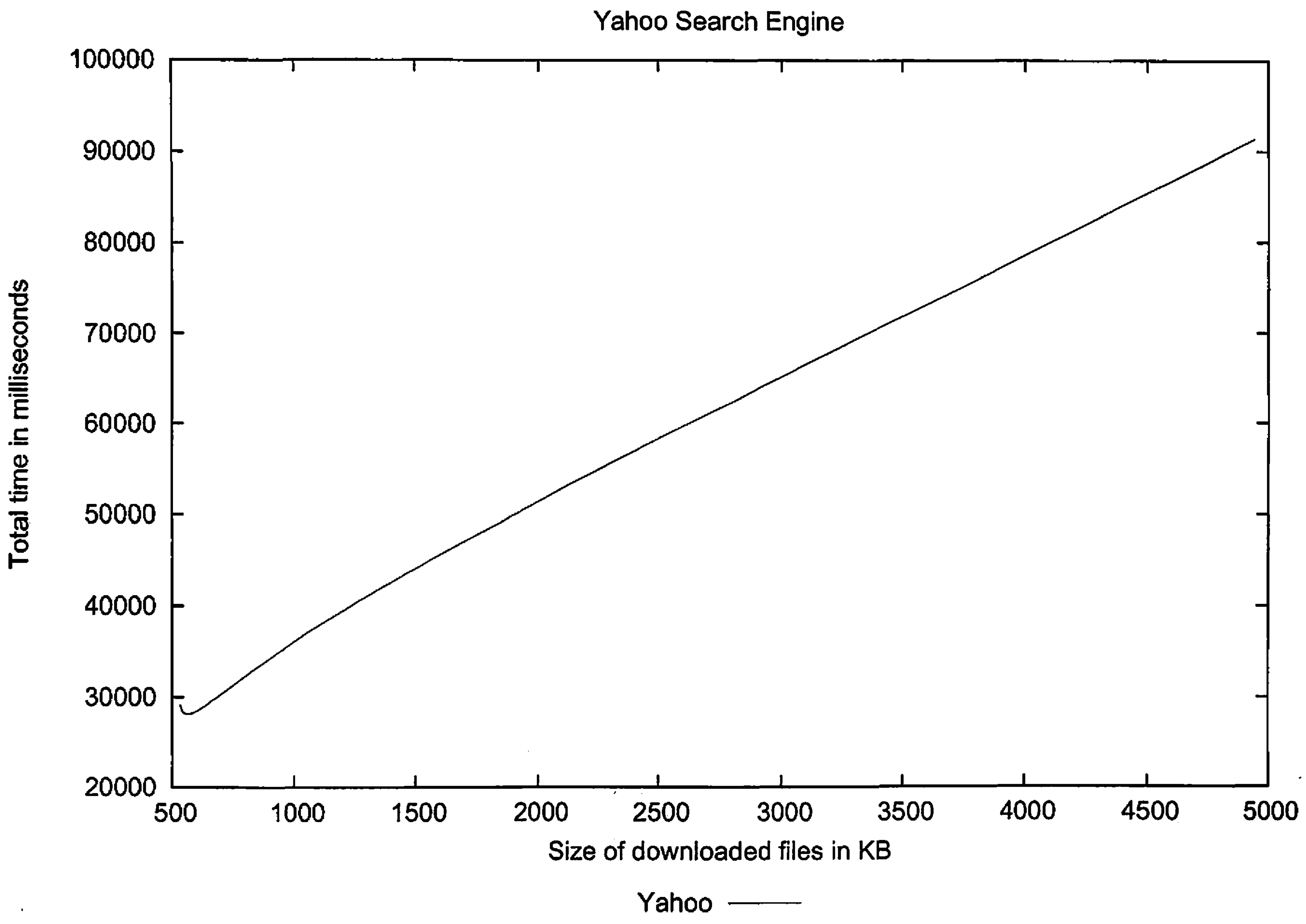


Figure 3.6: Performance Graph for WDES using Yahoo

Chapter 4

Implementation of a Web based tool for Web user and Administrator

4.1 Introduction

This chapter deals with designing and implementing a web based tool for web user and administrator using two algorithms namely **ACAD-SEARCH** and **MAINTAIN-REPOSITORY**. It implements the interface for queries relating to **Higher Education**, **Conference Alerts** and **Special Interest Group**. Web based application is a computer software application that is coded in a programming language that can be run on a browser environment. It is used for the following reasons:

- Web Browser can be used as a client.
- It is easy to update and maintain web applications than to install or distribute software on many computer systems.
- It does not require space on the client computer.

Interfaces used for developing web applications are Java, JavaScript, DHTML, Flash, Silverlight, PHP, and Ajax [57]. As mentioned earlier in chapter 3, the downloads for various file types has been carried out using the command line tool wget. This chapter deals with a web based interface for *text, xml* and *html* data maintenance operations. The present work is intended to provide an efficient way to organize and retrieve web

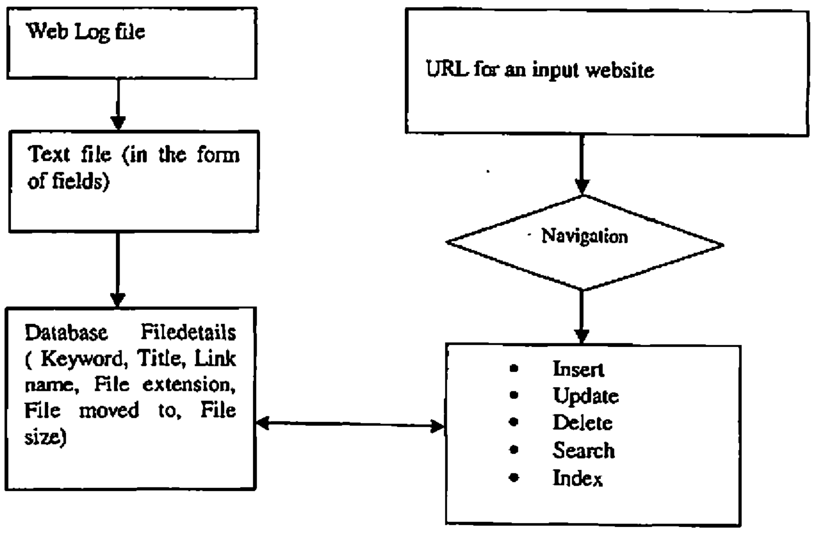

Web Log file
Text file (in the form of fields)
Database Filedetails (Keyword, Title, Link name, File extension, File moved to, File size)
URL for an input website
Navigation
Insert
Update
Delete
Search
Index

Figure 4.1: Web User: Navigation Options

content in an University environment. It considers web based data pertaining to *Higher Education, Conference Alerts* and *Special Interest Group.*

4.2 Web user

The *Block Schematic of the Web based tool for Navigation options* for the web user is shown in **Figure 4.1**. The web log file is taken as input and converted into a text file in the form of fields in MySQL separated by a comma [58]. The text file has to be uploaded in PHPMyAdmin database table. The database table is used to store the commonly used keywords and related information (Name, Link, File Type and Keyword used to search the link). A website domain named **www.bpdcthesis.com/data/index.php** has been created to use the database table records to search for queries based on **Higher Education, Special Interest Group** and **Conference Alerts.** In the website there is a main menu named Navigation which gives the options like *create a record, delete a record, update a record, search by keyword,* and *search by link type.* The proposed model improves retrieval accuracy since frequently accessed links relating to search application are indexed and stored over a relational table in web hosting server.

4.2.1 Algorithm ACAD-SEARCH

The present work deals with implementation of algorithm **ACAD-SEARCH**. The algorithm has been implemented using PHP as the scripting language and PHPMA as database management tool. The interface is created using HTML.

Inputs:

Search database, user specified URL.

Output:

The website: **www.bpdcthesis.com/data/index.php** is created, which the user can use for the various navigation functions.

Procedure:

1. Start.
2. If the mode of operation is
   (a) **Batch,** then Accept input URL from Search Database in sequential mode. This mode is used for handling bulk insertions.
   (b) **Interactive,** then Accept any input URL from the user. This mode is used for handling fewer insertions. This is needed when the user wishes to add records into the search database for the available information or entry of information for the accidentally deleted record.
3. Create the PHP scripts for the navigation facilities for the input URL.
   Action: The navigation includes the following functions over database for the table *filedetails*.
   - Insert a record.
   - Delete a record.
   - Update an existing record.
   - Search by Keyword.
   - Search by File Type.
   - View Status.

Table 4.1: Database Schema Information

Field	Type
Sn	int(11)
Name	varchar(30)
Link	varchar(30)
Type	varchar(30)
Search	varchar(30)

- Create Index.

4. Upload the PHP scripts on the webserver.

   Action: The PHP scripts for the navigational options are hosted on the web server.

5. Link the PHP scripts.

   Action: Link all the scripts for navigation with index.php.

6. Run the PHP script.

   Action: The PHP script can be run by the user using following link.

   www.bpdcthesis.com/data/index.php

7. End.

   Action: End the program.

4.2.2 Database Design for Search Application

The user interface has been created keeping the user convenience in mind.

Table 4.1 shows the *Structure of the Search Database.*

Database Name: *project.db*

The details of the fields are given below:

- **sn**(primary key): sn is the serial number that is used to show how many records exist in the table.

- **Name**: Name is a title given to a particular link.

Table 4.2: Snapshot of the Search Database

S.no	Name	Link	Type	Search
1	Scottish Ruby Conference 2010	/ca/html/Scottish Ruby Conference 2010 Arduino Ru	html	arduino conference
2	Arduino Forum -AT commandsfo	/ca/html/Arduino Forum-At commend afor Call Con	html	arduino conference
3	Arduino Blog web n stuff	/ca/html/Arduino Blog web n stuff.html	html	arduino conference
4	Miniconf's Arduino LCA2010	/ca/html/Miniconfs Arduino LCA2010.html	html	arduino conference
5	Arduino Day Conference	/ca/html/Arduino Day Conference Architecture	html	arduino conference
6	RubyConf 2010	/ca/html/RubyConf 2010.html	html	ruby conference
7	Los Angeles Ruby Conference 2010	/ca/html/LosAngelsRuby Conference2	html	ruby conference
8	IEEE Conferences and Events	/ca/html/IEEE Conferences and Events.html	html	ieee conference
9	NetBeans Sessions at the Java	/ca/html/NetBeansattheJAX 07 Conference in Germany	html	netbeans conference

- **Link**: It is the link of the particular web page as it is stored in the web server. For example: **http://www.bpdc/data/ca/html/arduinoconferene2011.html** will be the link that is stored in the link field.

- **Search**: This field is used to store the keywords through which a user can search.

There are two indexes for uniquely accessing the records.

- Sn - (serial number).

- SeaLin - It is an index on 2 fields *Search* and *Link*.

In PhpMyAdmin, the software automatically selects the best index out of the available options to be used for a query.

Table 4.2 shows the *Snapshot of the Search Database.*

4.2.3 User Interface Design for Web User

The layout of forms / menus for web based tool for web user and Administrator are shown in **Appendix G**. **Figure G.1** (Appendix G) shows the *Main Menu for the Proposed Search Application.* The link of the main page is:

http://www.bpdcthesis.com/data/index.php

The web application main page contains the following options for navigation:

- **Insert a Record:** Inserting a record that will be added to the MySQL database system.
- **Delete an existing record:** Deleting a record from database.
- **Index Page:** All the records in the database system are displayed using the index on two attributes: search keyword and link.
- **Search by Keyword:** By giving a particular keyword, all the records matching the keyword will give the link to the file.
- **Search by Link type:** By giving the link type, all the links of files under the given link type will be shown.
- **Update an existing record:** Modify an existing record in the database.

4.2.3.1 Insert a record

If the user has selected *Insert a Record (Option 1 of Main Menu)*, then the user will be redirected to the page with the link.

http://www.bpdcthesis.com/data/Insert a record.php

where the layout of *Insert a record* form will be displayed. The user has to enter all the information and click Submit button. **Figure G.2** (Appendix G) shows the Layout of *Insert a record* form. The details to be given by the user are include *Link name, Link, Type of file* and *Keyword.*

4.2.3.2 Delete an existing record

If the user has selected *Delete an existing record (Option 2 of Main Menu)*, then the user will be redirected to the page with the link.

http://www.bpdcthesis.com/data/Delete a record.php

where the layout of *Delete an existing record* form will be displayed. The user has to enter the link to delete a record because it is unique. This will cause the record to be deleted from the PhpMyAdmin. **Figure G.3** (Appendix G) shows the Layout of *Delete an existing record* form.

4.2.3.3 Index Page

If the user has selected *Index page (Option 3 of Main Menu)*, then the user will be redirected to the page with the link.

http://www.bpdcthesis.com/data/Status.php

where the page containing the records will be displayed using the index SeaLin (Search keyword and Link).

4.2.3.4 Search by Keyword

If the user has selected *Search by Keyword (Option 4 of Main Menu)*, then the user will be redirected to the page with the link.

http://www.bpdcthesis.com/data/Search by Keyword.php

where the layout of *Search by Keyword* form will be displayed. The user has to enter the keyword that will display all the links matching with the given keyword from the database. **Figure G.4** (Appendix G) shows the Layout of *Search by Keyword* form.

4.2.3.5 Search by File Type

If the user has selected *Search by File Type (Option 5 of Main Menu)*, then the user will be redirected to the page with the link.

http://www.bpdcthesis.com/data/Search by file type.php

where the layout of *Search by File type* form will be displayed. The user has to enter the file type that will display all the links under that type of file in the database. **Figure**

G.5 (Appendix G) shows the Layout of *Search by File Type* form.

4.2.3.6 Update an existing record

If the user has selected *Update an existing record (Option 6 of Main Menu)*, then the user will be redirected to the page with the link.

http://www.bpdcthesis.com/data/Update.php

where the layout of *Update an existing record* form will be displayed. The query takes Link name as the unique field and changes the record in the database where it matches the Link name with name in table. Accordingly, the current records gets updated with the new information.

Figure G.6 (Appendix G) shows the Layout of *Update an Existing Record form.*

4.2.4 Test Scenario

The application has been implemented using PHP as the language and MySQL as the database. The user interface has been designed using PHP scripts. The Test Scenario involved various test cases involving keywords. The data stored in the database comes under *Higher Education, Special Interest Groups* and *Conference Alerts.* The following subsections deal with the test cases.

4.2.4.1 Insert a Record

Figure G.7 (Appendix G) shows the Layout of *Record Insertion* form. The user has to fill in all the information and click submit button.

Figure G.8 (Appendix G) shows the Layout of *Page after Creation.* The message *Your info has been added* will be displayed.

4.2.4.2 Search by Keyword

Figure G.9 (Appendix G) shows the Layout of *Search by Keyword* form. The user has to enter the keyword and click submit button. The keyword entered by the user for the test scenario has been *Arduino Conference.*

Figure G.10 (Appendix G) shows the *List of links* for the keyword *Arduino Conference.* All the links under the particular keyword will be displayed in a link format. User can view all the links by opening the link in a new tab.

4.2.4.3 Search by File Type

Figure G.11 (Appendix G) shows the Layout of *Search by File* form. The user has to enter the file type and click submit button. The Type entered by the user for the test scenario has been *pdf.*

After entering the type and submitting the form, all the links under the file type pdf will be displayed in a link format as shown in **Figure G.12** (Appendix G). User can view all the links by clicking them. Similarly the file types like *html, ppt, doc* can be given as input and all the corresponding files may be displayed.

4.2.5 Time Complexity

The time complexity for various database operations are discussed in the following sub-sections. The parameters [66] considered for analysis are:

B - No. of data pages on the disk.

R - No. of records in a data page on the disk.

D - Time to read or write a disk page.

C - Time to perform a single basic operation on a record.

4.2.5.1 Insert a record

Inserting a record takes place at the end of the table [66]. The table is organized as a heap file (unsorted file). Insertion requires fetching the last page in the file, add the record and write back the page into the disk. The time involved is D+C+D = 2D+C. (i.e. read + insert + write)

4.2.5.2 Delete a record

The following actions are performed to delete a record [66];

1. Search a record with equality selection on a given attribute. This involves a time of B(D+RC) in the worst case (since B pages are retrieved, each page involves D time units; R records are processed in each page; C refers to time to process a record).

2. Remove a record from the data page on the disk. This involves a time of C; i.e time to delete a record.

3. Write the modified page back. This involves a time of D. Hence, the total time to delete a record in the worst case is

   B(D+RC)+C+D (i.e from Steps 1,2,3)

   Time for delete operation = D(B+1) + C(R+1)

4.2.5.3 Update a record

The following actions are performed to update a record [66];

1. Search a record with equality selection on a given attribute. This involves a time of B(D+RC) in the worst case, since B pages are retrieved, each page involves D time units; R records are processed in each page; C refers to time to process a record.

2. Update a record from the data page on the disk. This involves a time of C; i.e time to update a record.

3. Write the modified page back. This involves a time of D. Hence, the total time to update a record in the worst case is

   B(D+RC)+C+D (i.e from Steps 1,2,3)

   Time for update operation = D(B+1) + C(R+1)

4.2.5.4 Search a record from a heap file

Here, the input table is not indexed on the primary key. Hence the worst case time complexity is B(D+RC) (since B pages are retrieved, each page involves D time units; R records are processed in each page; C refers to time to process a record).

4.2.5.5 Search a record using an index on primary key

Let *n* be the total number of keys present in the index file. Then, each search operation on a primary key value involves O(log n) computing time.

4.2.5.6 Creating an index on primary key

Let *n* be the total number of keys to be inserted into the index file. Insertion of a single entry in index file is O(log n). Since creation involves insertion of n key values, the worst case time complexity is O(n log n).

4.3 Web Administrator

The Block Schematic of the web based tool for repository maintenance by the web administrator is shown in **Figure 4.2.**

The repository is organized as collection of files and directories in the web server [59]. The html page is taken as input. The user interface for **Repository** has three different types of data like *text, xml* and *html.* The functions like *view, upload, delete, search* and *sort* can be applied to the repository. The Repository consists of data related to **Higher Education, Conference Alerts and Special Interest Group.** By clicking on the exit option, the web administrator should be able to exit from the main menu. The interface has the options to perform the following tasks:

- Create a folder.

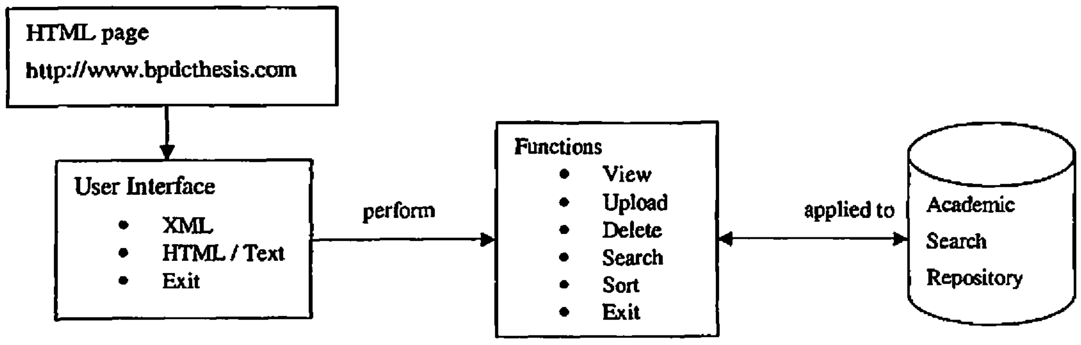


Figure 4.2: Web Administrator: Maintain Repository

- Upload a file.
- Rename, move and delete folders.
- Rename, move, delete, duplicate and edit file.
- Search a file based on first few characters of filename.
- Sort by alphabetical order.
- Sort by file size.
- Sort by date of creation or last modification.
- Sort by file type.
- Exit (from the main menu).

4.3.1 Algorithm MAINTAIN-REPOSITORY

The present work deals with implementation of algorithm **MAINTAIN-REPOSITORY**. The algorithm has been implemented using PHP as the scripting language and PHPMA as database management tool. The interface is created using HTML.

Inputs:

Startup HTML page, Data type (*xml, html, text*) selected by the user.

Output:

The intended options for files and folders.

Procedure:

1. Start.

   Action: Start the program.

2. Create the PHP scripts for the navigation facilities for the input URL.

   Action: The basic functions of the repository include *View, Upload, Delete, Search, Sort* and *Exit.*

3. Upload the PHP scripts on the webserver.

   Action: The PHP scripts for the navigational options are hosted on the web server.

4. Link the PHP scripts.

   Action: Link all the scripts for navigation with the main script.

5. Run the PHP script.

   Action: The PHP script can be run by the user using following link.

   www.bpdcthesis.com/data/index.php

6. End.

   Action: End the program.

4.3.2 User Interface Design for Web Administrator

The following menus and screens are presented in the interface and their actions are discussed subsequently. The link of the main page is:

http://www.bpdcthesis.com/data/index.php

4.3.2.1 Maintain-Repository: Authentication page

Figure G.13 (Appendix G) shows the *Maintain-Repository: Authentication page.*

It is a login page where *user name* and *password* are entered. Clicking on *login* button will take the users to the home page.

4.3.2.2 Maintain-Repository: Home page

Figure G.14 (Appendix G) shows the *Maintain-Repository: Home Page.*

On the right hand side of home page, the options namely *html, xml* and *text* are displayed with their corresponding number of files. Users can view the files by clicking on them. The Home page contains options to upload and search files and to create folders.

Figure G.15 (Appendix G) shows a typical *HTML page* along with input files of the test configuration. The web application main page contains the following options for navigation:

- **Upload files**
- **Create folder**

- **Search files and folders**

Using the option *Upload files*, the user can click *browse* button to select a particular file to be uploaded. Clicking on *upload* button uploads the selected file from the folder. Using the option *Create Folder*, the user can create folder or file according to his / her choice. Clicking on *OK* button will help the users to create folder or file. Using the option *Search files and Folder option*, the user has to type the keyword in the search box. A list of folders and files associated with keywords will be displayed. The user can select a particular folder or file from the dropdown list.

4.3.2.3 Maintain-Repository: Folder options

Figure G.16 (Appendix G) shows the *Maintain-Repository: Folder options.* Using this option, the user can click the circle shape to the left of the folder. The options include *Rename, Move* and *Delete.*

4.3.2.4 Maintain-Repository: Files options

For the files, the following options are provided: *Rename, Move, Delete, Duplicate* and *Edit.*

Figure G.17 (Appendix G) shows the *Maintain-Repository: File options.* **Figure G.18** (Appendix G) shows the *Maintain-Repository: Edit options.*

4.3.2.5 Maintain-Repository: Sorting options

In Maintain-Repository: Sorting options, the user has the following options like *Name, Size, Date* and *Type.*

Figure G.19 (Appendix G) shows the *Maintain-Repository: Sorting options.* **Figure G.20** (Appendix G) shows the *Maintain-Repository: Search options.*

4.3.3 Implementation

The implementation of ACAD-SEARCH has been carried out in PHP and MySQL to make an organized search and facilitate easy downloads based on keywords. The web application stores information in a web server and provides the user with a range of

Table 4.3: Time Complexity for Repository Operations

Repository Operation	Time Complexity	Explanation
View/Upload	O(c_1) + O(X)	O(c_1)=time to locate the file/include a directory entry O(X)=O(file size/data transfer rate over internet)
Delete	O(Y)+O(c_2)	O(Y)=O(Locate filename in web server's directory) c_2 = time to remove the entry from the directory
Search	O(logn)	uses binary search to locate the directory entry
Sort	O(nlogn)	uses tree sort for sorting n file entries in the repository

options. It uses PHP and MySQL to provide different navigation options to access the database. The application **MAINTAIN-REPOSITORY** has been implemented using PHP and Javascript as the scripting languages and Apache 2.2 as the web server. The interface is created using HTML page. The application helps the web administrator to make an organized search and facilitate easy maintenance of the repository. Functions and their Actions for Maintain-Repository Algorithm using Java implementation are shown in **Appendix E.**

4.3.4 Time Complexity

The worst case time complexity for various Repository Operations are shown in Table 4.3. Here n represents the number of file entries in the repository.

4.4 Summary

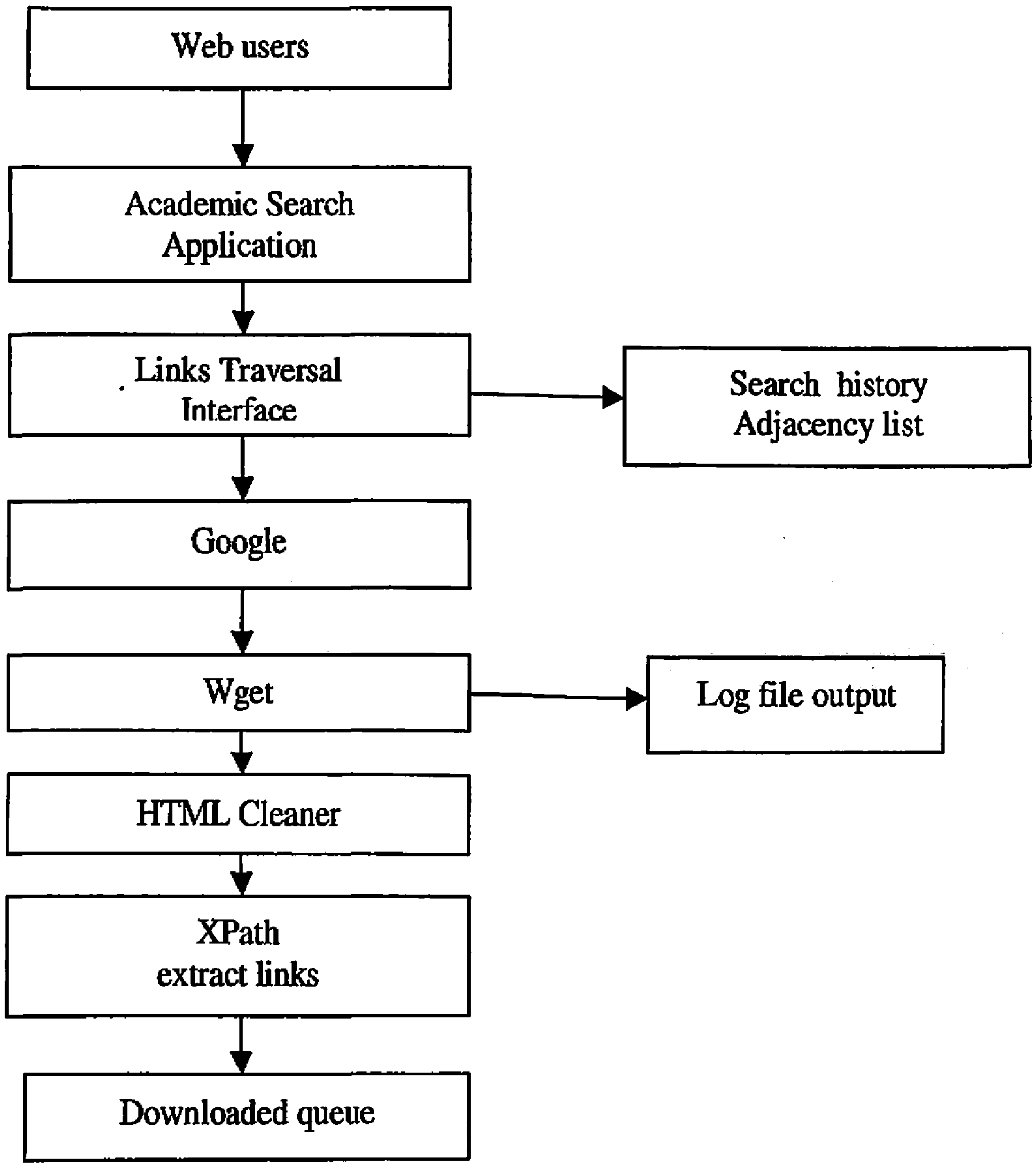
This chapter dealt with design of a web based tool for web user and administrator. The algorithm **ACAD-SEARCH** is for web user. The algorithm **MAINTAIN-REPOSITORY** is for Web Administrator. The algorithm **ACAD-SEARCH** has been successfully deployed in a web server using a standard interface for the web user. The Search database mainly stores the various fields in a relational table and it is hosted on the web server. The web users can access the database through web browser using a standard interface. This helps in saving considerable time and effort in the search process. The application has been implemented in PHP and MySQL to make an organized search and facilitate easy download based on keywords. The algorithm **MAINTAIN-REPOSITORY** has been successfully deployed in a Web server using a standard interface for web administrator. The web administrator can perform a range of options for folders and files, in the interactive mode. It has been tested successfully for the three categories of web data namely *Special Interest Group, Conference Alerts and Higher Education.* The application has been implemented using PHP and Javascript and it helps the web administrator, for easy maintenance of repository.

Chapter 5

Design and Implementation of Breadth First Search Strategy for Implicit Graph

5.1 Introduction

The World Wide Web contains abundant information and acts as a large data repository. The main challenge of a web user while searching the web is to find relevant information in an efficient and effective manner with less time and effort and this can be accomplished to a greater extent by making use of the structure information of the web. This chapter deals with the design and implementation of the breadth first search strategy for the links traversal and this can be effectively deployed for academic search activities. Breadth First Search technique is a systematic search strategy which begins at an initial node (state) and from the initial node search actions are applied downward on a tree structure in a breadthwise order. BFS has been implemented using a queue data structure.



Web users
Academic Search Application
Links Traversal Interface
Search history Adjacency list
Google
Wget
Log file output
HTML Cleaner
XPath extract links
Downloaded queue

Figure 5.1: Links-Traversal Flow Diagram

5.2 Problem Description

The *Flow diagram* for the proposed system *Links-Traversal* is shown in **Figure 5.1**. The proposed system uses Breadth First Search strategy in extracting web links. The user can limit the number of links (to avoid recursive calls). This significantly reduces search space (instead of going through each link interactively in the browser).

The keywords for the **Search Application** are taken as input from the user through GUI. Initial URL is generated from the search query and the HTML files are downloaded using Wget software tool. The HTML files downloaded are not well formed and are not suitable for processing. Hence, they are converted by HTML Cleaner into XML. XPath tools are used to extract all the links and they are added to the download queue. Each link is chosen by processing the queue and is crawled. After downloading, the log file is checked to ensure that the file downloaded is of text or HTML MIME. Keyword occurrence count is also done. The steps are repeated till the maximum number of links specified is reached. At each stage, appropriate output is tabulated on the GUI. When the process ends, a log file of the output is saved into a text file.

5.3 Algorithm: LINKS-TRAVERSAL

This section describes an algorithm named Links-Traversal, for the implicit graph function relating to Search Application.

A. Reading of Inputs:

Keywords, Number of links to crawl (m), Number of start links from Google.

B. Generation of Output:

Downloaded links in a text file, corresponding to the given set of keywords.

C. Procedure:

1. Enter keyword in GUI.

   Action: User enters the keyword in Links-Traversal interface and clicks the search button. A search query is generated and passed to the search engine Google, using Wget software.

2. Use Wget to retrieve the search results as HTML file.

Action: Wget software is used to download the file. The downloaded HTML file is converted to XML using HTML Cleaner. HTML found on the web is ill formed and not suitable for further processing because it contains unclosed tags and missing quotes in the document. HTML Cleaner is an open source HTML parser written in Java. It accepts HTML documents and produces well-formed XML.

3. Extract the links.

   Action: XPath is used to extract the links from the search page and each link is added to download queue. XPath is a language for finding information in an XML document.

4. Construct the implicit graph.

   Action: The implicit graph is constructed based on the input keywords. Let the number of nodes in the graph be n. These nodes refer to the actual web pages returned during the search process.

5. The extracted links are added to the queue.

   Action: All the internal links are extracted from the search page using Xpath tools. These links are added to the download queue. Duplicate entries are removed.

6. Calculate the keyword occurrences.

   Action: Number of occurrences of keywords is calculated and the user interface is updated to reflect the keyword count for each link.

7. The link is marked as *'crawled'.*

   Action: If the links are crawled successfully, it is marked as 'crawled', else any one of the following error messages like *XML Parse error, Unknown error* or *Crawl error* is displayed.

8. Repeat Steps 6 through 8, till maximum number of links is reached (i.e. m) for each of the n nodes.

   Action: Repeat Steps 6 to 8 till the user defined *'maximum number of links'* is reached for every node of the implicit graph.

Table 5.1: Time Complexity for Links-Traversal

Steps	Computing Time	Explanation
1	O(1)	Accepting keyword involves a single basic operation
2	O(c_1)	c_1 is a constant and it corresponds to the time to retrieve search results
3,4,5	O(c_2)	c_2 is a constant and it corresponds to the time to extract the links from html page, build the implicit graph using queue
6,7,8	O(1) + O(|n|) + O(|m|)	marking the status as crawled involves a single operation; moreover every vertex(node) and every edge will be explored in the worst case [64], Hence O(|n|) + O(|m|) are also included

The Links-Traversal algorithm has been implemented using JavaSE Runtime 6 and GNU Wget 1.12. The user interface has been designed using Javascript. The details of the Java classes along with their descriptions are shown in **Appendix D.**

The search engine Google does not explicitly show the keyword count, although it considers it in its SEO strategy. The proposed algorithm provides the keyword count information to the web user. This provides a flexible option for the web user to download the content only if necessary based on prior knowledge of keyword count. This feature is desirable for pattern matching applications.

5.4 Time Complexity

Let **m** represent the maximum number of links for each keyword.

Let **n** represent the number of nodes in the implicit graph.

The expression corresponding to the worst case time complexity is

$$\text{Computing time} = O(1)+O(c_1)+O(c_2)+O(1)+O(|n|)+O(|m|),$$

as shown in **Table 5.1**. The significant term in the above expression is O(|n|)+O(|m|) and is regarded as the worst case time complexity.

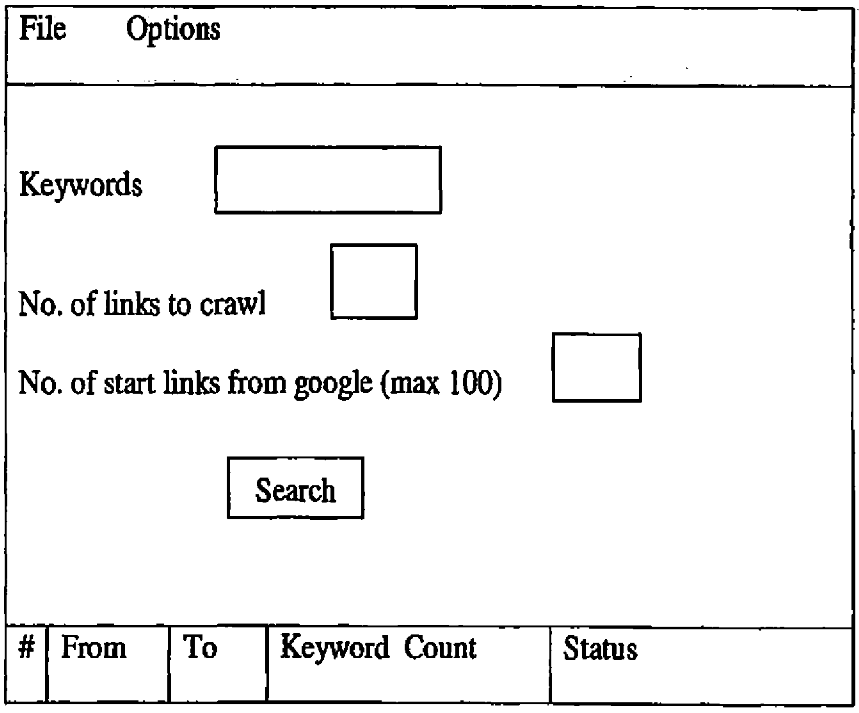


Figure 5.2: User Interface Design of Links-Traversal

5.5 User Interface Design

Figure 5.2 shows the *User Interface Design of Links-Traversal.* The Main Menu has two choices namely *File* and *Options.* The sub menu *File* contains *About* and *Exit.* The sub menu *Options* contains *Settings.*

The user has to enter all the information like **Keywords, Number of links to crawl, Number of start links from Google** and click *Search* button.

The maximum number of links is limited to 100.

The option **Keyword Count** displays the number of occurrences of keyword.

The option **Status** displays any one of the following messages.

1. Crawled without errors.

2. Crawl error.

3. Malformed URL.

4. XML Parsing error.

5. Ignored. Not an HTML file.

6. Unknown error.

The link is marked as *crawled* if it downloads link successfully. In case of an error, appropriate error message is displayed.

5.6 Test Scenario

The Test Scenario involved many test inputs involving **keywords**. The keywords considered here include the following: *Higher Education, Web Mining, Academic Search Application, Web Content Management, Web Structure Mining, Web Log Mining, Web Link Extraction and Breadth First Search, Conference Alerts and Special Interest Group.*

The following test cases are considered for implementation of **Links-Traversal algorithm**. The Test Scenario has been carried out in a LAN connection of 100 Mbps.

A. Links Traversal: Test Case 1

Figure 5.3 shows *Test Case 1.*

The keyword entered is **Higher Education.**

Number of links to crawl is **3**.

Number of start links from google (max 100) is **2**.

The total time taken for execution is **14765ms**.

Three links are extracted from Google and the keyword count for the links are displayed. All the links are crawled without errors.

B. Links Traversal: Test Case 2

Figure 5.4 shows *Test Case 2.*

The keyword entered is **Conference Alerts**.

Number of links to crawl is **5**.

Number of start links from google (max 100) is **3**.

The total time taken for execution is **7703ms**. Three links are crawled successfully without errors and two links are ignored because of *XML Parsing error.*

5. Ignored. Not an HTML file.

6. Unknown error.

The link is marked as *crawled* if it downloads link successfully. In case of an error, appropriate error message is displayed.

5.6 Test Scenario

The Test Scenario involved many test inputs involving **keywords**. The keywords considered here include the following: *Higher Education, Web Mining, Academic Search Application, Web Content Management, Web Structure Mining, Web Log Mining, Web Link Extraction and Breadth First Search, Conference Alerts and Special Interest Group.*

The following test cases are considered for implementation of **Links-Traversal algorithm.** The Test Scenario has been carried out in a LAN connection of 100 Mbps.

A. Links Traversal: Test Case 1

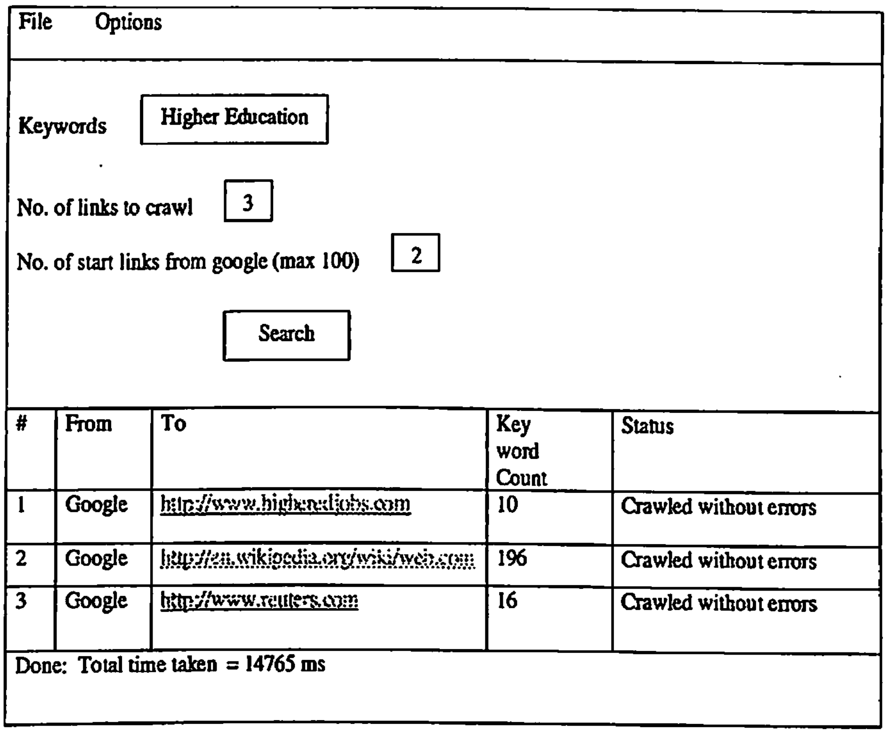
Figure 5.3 shows *Test Case 1.*

The keyword entered is **Higher Education.**
Number of links to crawl is **3.**
Number of start links from google (max 100) is **2.**
The total time taken for execution is **14765ms.**
Three links are extracted from Google and the keyword count for the links are displayed.
All the links are crawled without errors.

B. Links Traversal: Test Case 2

Figure 5.4 shows *Test Case 2.*

The keyword entered is **Conference Alerts.**
Number of links to crawl is **5.**
Number of start links from google (max 100) is **3.**
The total time taken for execution is **7703ms.** Three links are crawled successfully without errors and two links are ignored because of *XML Parsing error.*



Keywords: Higher Education

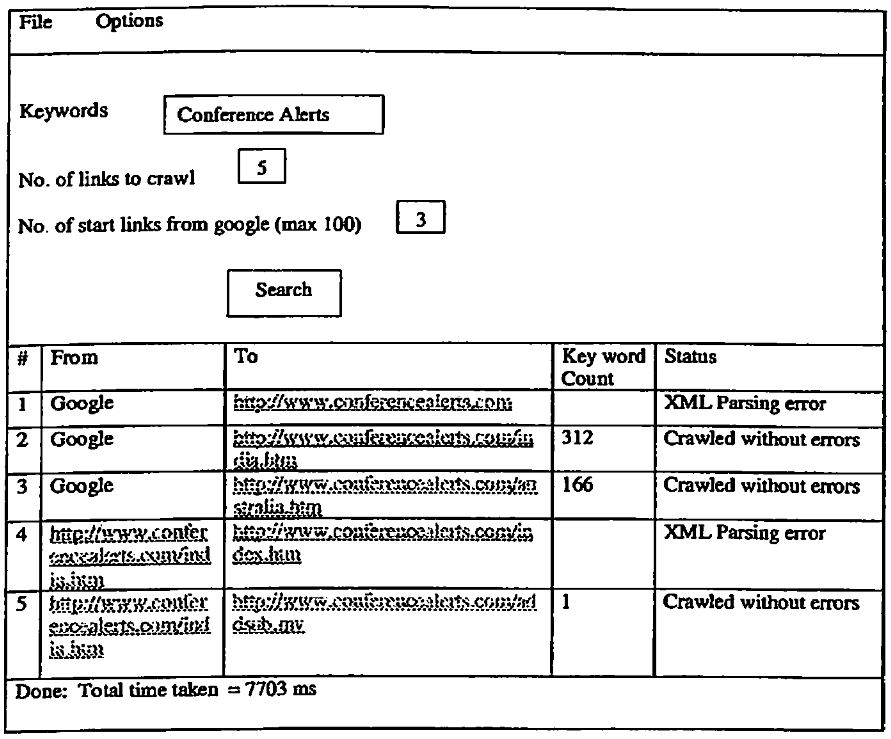
No. of links to crawl: 3

No. of start links from google (max 100): 2

#	From	To	Key word Count	Status
1	Google	http://www.higheredjobs.com	10	Crawled without errors
2	Google	http://en.wikipedia.org/wiki/web.com	196	Crawled without errors
3	Google	http://www.reuters.com	16	Crawled without errors

Done: Total time taken = 14765 ms

Figure 5.3: Links-Traversal Test Case 1



Keywords: Conference Alerts

No. of links to crawl: 5

No. of start links from google (max 100): 3

#	From	To	Key word Count	Status
1	Google	http://www.conferencealerts.com		XML Parsing error
2	Google	http://www.conferencealerts.com/india.htm	312	Crawled without errors
3	Google	http://www.conferencealerts.com/australia.htm	166	Crawled without errors
4	http://www.conferencealerts.com/india.htm	http://www.conferencealerts.com/index.htm		XML Parsing error
5	http://www.conferencealerts.com/india.htm	http://www.conferencealerts.com/adsub.mv	1	Crawled without errors

Done: Total time taken = 7703 ms

Figure 5.4: Links-Traversal Test Case 2

#	From	To	Key word Count	Status
1	Google	http://en.wikipedia.org/wiki/Special Interest Group	44	Crawled without errors
2	Google	http://en.wikipedia.org/wiki/Advocacy Group	111	Crawled without errors
3	Google	http://www.twyman_whitney.com/americancitizen/links/lobbies.com	22	Crawled without errors
4	Google	http://www.bluetooth.com	0	Crawled without errors
5	http://en.wikipedia.org/wiki/Special Interest Group	http://en.wikipedia.org/wiki/Wikipedia:CitingSources	11	Crawled without errors
6	http://en.wikipedia.org/wiki/Special Interest Group	http://en.wikipedia.org/wiki/Wikipedia:Verifiability	10	Crawled without errors

Figure 5.5: Links-Traversal Test Case 3

C. Links Traversal: Test Case 3

***Figure* 5.5** *shows Test Case 3.*

The keyword entered is **Special Interest Group.**

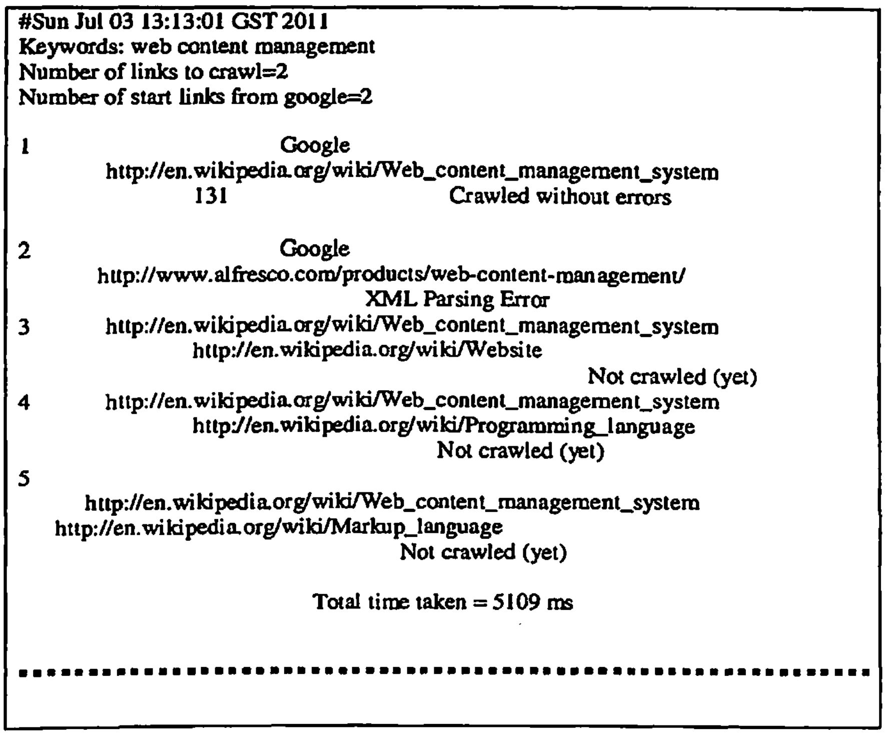
Number of links to crawl is **6**.

Number of start links from google (max 100) is **4**.

The total time taken for execution is **35639ms**.

All the links are crawled successfully without errors. **Figure 5.6** illustrates the *log messages* recorded during the entire search process and the total time taken for web link extraction. When the application is closed, the log messages of the session are saved onto a file for later use. Thus, the complete history of all the searched and downloaded links can be tracked. *The experimental results for Web Link Traversal* are illustrated in **Table 5.2**.

It can be inferred from **Table 5.2** that the execution time for crawling varies between 5109 and 24875 milliseconds depending on the *Number of links to crawl* and *Number of start links* for the network bandwidth of 100Mbps. The detailed analysis of the log



```
#Sun Jul 03 13:13:01 GST 2011
Keywords: web content management
Number of links to crawl=2
Number of start links from google=2

1                       Google
        http://en.wikipedia.org/wiki/Web_content_management_system
                131                     Crawled without errors

2                       Google
       http://www.alfresco.com/products/web-content-management/
                                XML Parsing Error
3       http://en.wikipedia.org/wiki/Web_content_management_system
                http://en.wikipedia.org/wiki/Website
                                                Not crawled (yet)
4       http://en.wikipedia.org/wiki/Web_content_management_system
                http://en.wikipedia.org/wiki/Programming_language
                                        Not crawled (yet)
5
      http://en.wikipedia.org/wiki/Web_content_management_system
   http://en.wikipedia.org/wiki/Markup_language
                                      Not crawled (yet)

                        Total time taken = 5109 ms

■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
```

Figure 5.6: Log Messages for Links-Traversal

messages has been carried out in a subsequent chapter (chapter 7: relating to web log analysis).

Table 5.2: Experimental Results for Web Link Traversal

S.No.	Keywords	No.of links to crawl	No.of start links	Time in ms
1	Web Mining	10	10	20656
2	Academic Search Application	5	3	24875
3	Web Content Management	2	2	5109
4	Web Structure Mining	6	4	5497
5	Web Log Mining	7	5	22785
6	Web Link Extraction	10	5	13883
7	Breadth First Search	4	3	6716
8	Higher Education	3	2	14765
9	IEEE Conferences	4	3	13390
10	Artificial Intelligence Research Group	3	3	6295
11	Conference Alerts	5	3	8032

5.7 Summary

This chapter focuses on the **Links-Traversal** algorithm for implicit graph using *Breadth First Search* in Search Application. The algorithm has been implemented using Javascript and successfully tested for different keywords pertaining to *Higher Education, Conference Alerts* and *Special Interest Group.* The time complexity of the above algorithm has been analyzed. The experimental work for Web links traversal has been performed and the log messages record the time taken during entire search process. The execution time for crawling varies depending on the number of links to crawl and the network bandwidth.

Chapter 6

Design of an Interface for Page Rank Calculation in Web Structure Mining

6.1 Introduction

The net structure of the World Wide Web is constantly changing due to the addition or removal of web pages (nodes) or the increase or decrease in the number of incoming or outgoing links (edges) to or from a web page [61]. It is very much essential to maintain the web structure in an organized way for easier access. The **Page Rank** for the web pages are periodically computed to reflect the current state of the available web pages. This chapter presents an interactive Page Rank calculation algorithm and has been successfully deployed for the **Search Application.**

6.2 Interface Description for Page Rank Algorithm

The present work deals with a Page Rank algorithm called VIS-POS considering two main factors: *Visibility of a link* and *Position of a link within a document.* **Figure 6.1** shows the *Block Schematic of VIS-POS Interface.*

The main activities are stated as follows:

1. Academic users gives query to University application which is submitted to an algorithm VIS-POS that considers *Visibility* and *Position of a link.*

```
2. The algorithm VIS-POS searches the Search Repository stored in the web server
   requested to check if the data is available or not.

3. If  the data is available,
   {
   then the algorithm displays the web pages and their page ranks.
   }
   Else
   {
   the data is extracted from the web using Web Data Extractor.(a program written
   in Java)
   }
```

The recently extracted web data is stored in the Search Repository. The data is checked for availability and the results are displayed to the user.

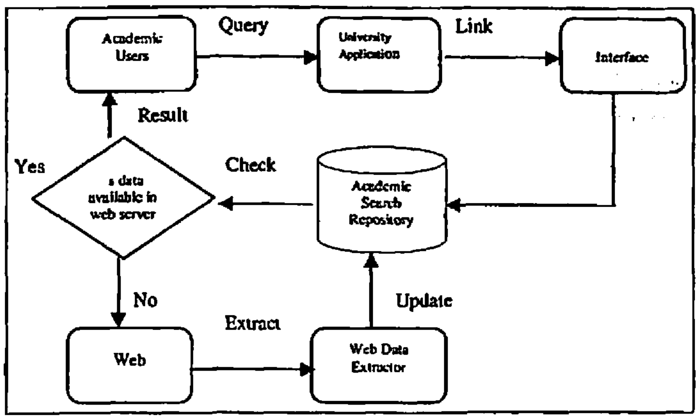


Figure 6.1: Block Schematic of VIS-POS Interface

6.3 Algorithm: VIS-POS

The proposed algorithm **VIS-POS** considers the *Visibility of a link* and the *Position of the link in the page or document* as main criteria. A static web page has been created using Adobe Dream Weaver on which the Javascript is embedded. The prototype is a page rank calculator having 10 * 10 grid where 10 web pages are listed (from A to J). It is designed in such a manner that the user can make hyperlinks from any of these pages to any other. There are also additional options for specifying up to 4 inbound and outbound hyperlinks.

Reading of Inputs:

1. Checkbox values for hyperlinks between pages, outbound and inbound pages.
2. Mode Selection (Simple and Real).
3. Number of iterations (say k).

Generation of Output:

1. Individual ranks of the 10 web pages. (depending on the mode and the hyperlink structure).
2. Total PageRank value.

Procedure:

1. G:= set of pages.

   Action: Total number of pages is in G.

2. for each page p in G do.

   Action: Repeat for all the pages.

   (a) PR[i] = 1.

   Action: Initialize the pagerank array with initial page rank of 1.

3. function **calculate(G).**

   Action: calculate(G) is a function to calculate the page rank of all the pages.

   (a) for step from 1 to k do.

   Action: Run the algorithm for k iterations.

   (b) for each page p in G do.

   Action: For each page get the inbound links initial PR.

   (c) if mode = simple then.

   Action: If the mode is simple, go to step g.

(d) if mode = real then.

   Action: If in real mode check for and clear any orphan pages. (pages that are not linked).

(e) orphan += 1.

   Action: Flag all orphan pages as inactive, including all pages that are linked to, from orphan pages.

(f) for each page q in p.inboundlinks.

   Action: Repeat for all the pages having inbound links.

(g) p += PR[n].

   Action: Calculate PageRank of all the pages.

4. End.

6.4 Implementation

This section describes the implementation aspects of the proposed algorithm VIS-POS.

1. The first thing that is probed is the *Mode.* There are 2 modes, namely **Simple** and **Real** mode. **Real** mode considers *dangling links* and *orphan pages* and **Simple** mode does not consider these.

2. In order to avoid starting errors, the calculator should be **Cleared** before any connections are made or any values are noted.

3. Once the mode is recognized, the checkboxes are probed in the grid to understand the link structure of the 10 pages.

4. Also, the **outbound** and **inbound** links checkboxes are probed for a possibility.

5. Accordingly, the counters are incremented and the page ranks are recursively calculated until the maximum number of iterations is reached.

6. Once **k** reaches its maximum, the calculation process stops and the results are displayed at the corresponding places.

The VIS-POS algorithm has been implemented using Javascript 1.2, JSDK 1.7 and Adobe DreamWeaver.

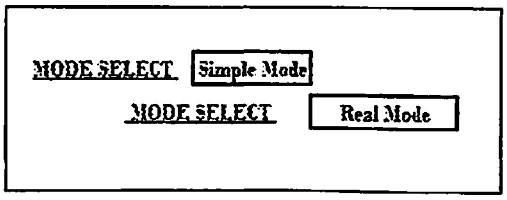


Figure 6.2: Mode Selection

Figure 6.2 shows the *Mode Selection* options. Clicking on **MODE SELECT** causes the mode to change. **Figure 6.3** displays the *Information bar* and the following paragraphs describes in detail the various options.

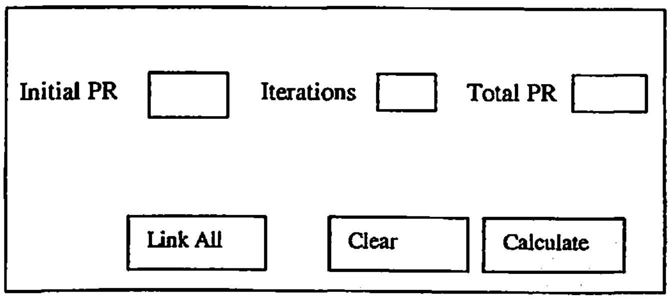


Figure 6.3: Information Bar

1. The **Initial PR** box is used to specify any Initialization of the ranks before the calculation process begins.

2. The number of **Iterations** can also be specified.

3. The **Total PR** specifies the total rank of all the 10 web pages added together (after calculation is done).

4. The **Link All** tab is used to check all the boxes in the 10 X 10 grid and make all possible connections (leaving the inbound and outbound link checkboxes).

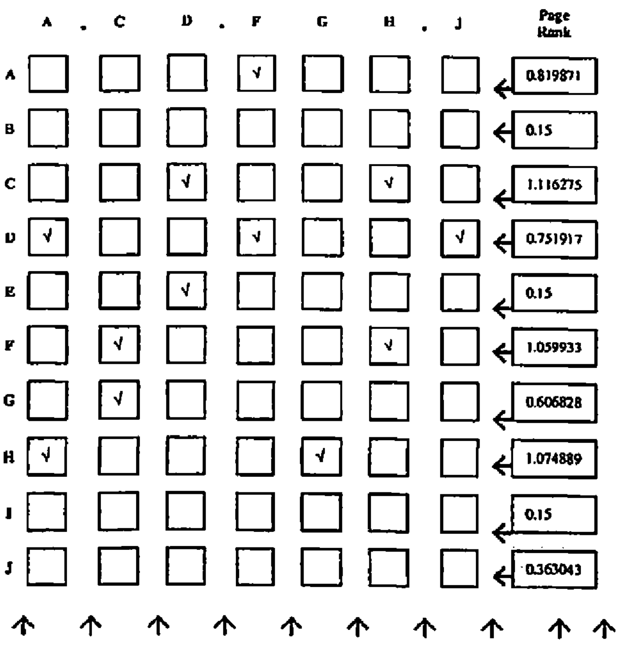
5. The **Clear** tab is used to make the calculator ready for inputs. It must always be used right after selection of the mode and before any connections are made so as to avoid any errors.

6. As the name suggests, the **Calculate** tab is clicked to order the program to perform the calculations after probing the grid for necessary information.

7. <- This arrow is used to check all the boxes in that row (horizontal).

8. -> This arrow is used to check all the boxes in that column (vertical).

Figure 6.4 shows the *Page Rank Calculator* in use.



	A	.	C	D	.	F	G	H	.	J	Page Rank
A	☐		☐	☐		✓	☐	☐		☐	0.819871
B	☐		☐	☐		☐	☐	☐		☐	0.15
C	☐		☐	✓		☐	☐	✓		☐	1.116275
D	✓		☐	☐		✓	☐	☐		✓	0.751917
E	☐		☐	✓		☐	☐	☐		☐	0.15
F	☐		✓	☐		☐	☐	✓		☐	1.059933
G	☐		✓	☐		☐	☐	☐		☐	0.606828
H	✓		☐	☐		☐	✓	☐		☐	1.074889
I	☐		☐	☐		☐	☐	☐		☐	0.15
J	☐		☐	☐		☐	☐	☐		☐	0.363043

Figure 6.4: The Page Rank Calculator

A sample set of 10 web pages have been considered in this experiment (named A to J). The web pages B, E and I have no links pointing to other web pages and are indicated as dots(.). **Calculate** button is used to calculate the Page Rank after making the required links. Links can be created from the pages in the left column to the web pages in the top row but not vice versa.

For example, if page B has a link pointing to page F, check the box in the 6th column of the 2nd row. If page F has a link pointing to page B, check the box in the 2nd column of the 6th row.

Figure 6.5 displays the *inbound links.* The boxes in the figure are used to specify any incoming links (not more than 4) from web pages other than those 10 from A to J. They can have a significant effect on the pointed page as the user is allowed to specify the

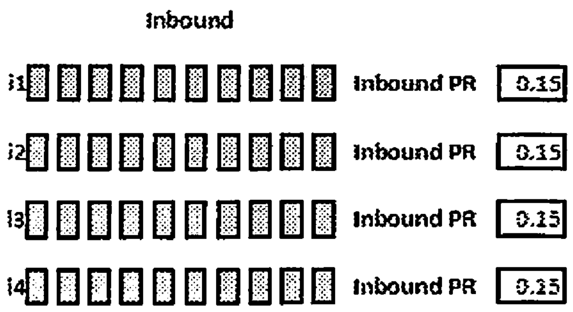


Figure 6.5: Inbound Links

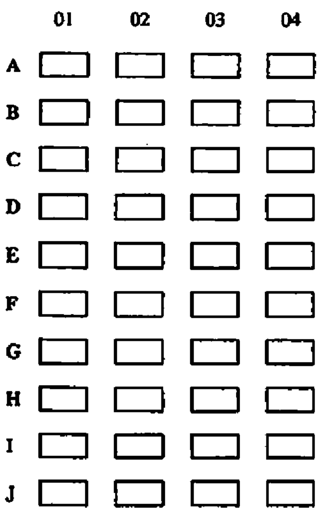


Figure 6.6: Outbound Links

exact rank he wishes those 10 pages to receive. **Figure 6.6** displays the *outbound links*. The boxes in the figure are used to specify outgoing links (not more than 4) to web pages other than those 10 from A to J. **Orphan pages** are those pages that are not linked by any other Web page on the Internet. **Dangling Links** are those links that point to Web pages that do not have a single link emanating from them.

The page rank for a page A [31] is calculated using the formula given below:

$$PR(A) = (1 - d) + d(PR(T1) * L(T1, A) + ... + PR(Tn) * L(Tn, A)) \qquad (6.1)$$

where,

$PR(A)$- PageRank of A.

d - Damping Factor.

$PR(Ti)$ - PageRank of Page Ti.

$L(T1, A)$ - It represents the evaluation of a link which points from $T1$ to A.

The **VIS-POS** algorithm has been tested with the sample inputs of two, three, four and five web pages. The inputs were given using the interface provided through Javascript.

6.5 Test Scenario

Considering two of the criteria for the evaluation of links, namely **Visibility of a link** and the **Position of the link in the page/document**, an example is shown here. Based on the **Random Surfer Model**, these two criteria greatly influence the probability of random clicking on a certain link. In the original PageRank algorithm, this probability is given by the term $(1/C(Ti))$, where equal probability is assumed for each link on one page.

The definitions used in the test cases are outlined in the following paragraph:

P, Q, R, S and T are web pages.

X and Y are evaluation criteria shown in **Tables 6.1** and **6.2** respectively, where

X is Visibility of a link.

Y is Position of a link within a document or page.

Table 6.1: Visibility of a link

X	Visibility of a link
1	Link is not particularly emphasized
2	Link is in bold, italic
3	Link is underlined with large font size

Table 6.2: Position of a link

Y	Position of a link within a document / page
1	Link is in lower half
2	Link is somewhere in the middle
3	Link is in upper half

Case 1: Web Universe with 2 web pages

Consider a web universe consisting of 2 web pages - P and Q. There are outbound and inbound links as shown in **Figure 6.7**.

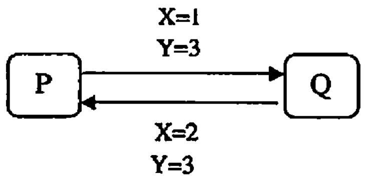


Figure 6.7: Web Universe with 2 web pages

Table 6.3 shows the *Sample inputs for 2 web pages.* **Table 6.4** shows the generalized formulae for Evaluation of links upto five web pages. Based on **Table 6.4**, the links are evaluated for 2 web pages as shown in **Table 6.5**.

Table 6.3: Sample Inputs for 2 web pages

	P		Q	
	X	Y	X	Y
P	-	-	1	3
Q	2	3	-	-

Table 6.4: Generalized formulae for Evaluation of links

Variable	Function	Variable	Function
a	Y(P,Q) * X(P,Q)	k	Y(S,Q) * X(S,Q)
b	Y(P,R) * X(P,R)	l	Y(S,R) * X(S,R)
c	Y(Q,P) * X(Q,P)	m	Y(P,T) * X(P,T)
d	Y(Q,R) * X(Q,R)	n	Y(Q,T) * X(Q,T)
e	Y(R,P) * X(R,P)	o	Y(R,T) * X(R,T)
f	Y(R,Q) * X(R,Q)	p	Y(S,T) * X(S,T)
g	Y(P,S) * X(P,S)	q	Y(T,P) * X(T,P)
h	Y(Q,S) * X(Q,S)	r	Y(T,Q) * X(T,Q)
i	Y(R,S) * X(R,S)	s	Y(T,R) * X(T,R)
j	Y(S,P) * X(S,P)	t	Y(T,S) * X(T,S)

Table 6.5: Evaluation of links for 2 web pages

Variable	Computation steps	Value returned
a	3 * 1	3
c	3 * 2	6

To determine the single factors L, instead of simply weighing the evaluated links with the number of outbound links on a webpage, the total of evaluated links must also be considered. For the single pages Ti, the weighting quotients are shown in **Table 6.6**.

Table 6.6: Weighting quotients for 2 web pages

Weighting quotients	Calculation steps	Value
Z(P)	a	3
Z(Q)	c	6

Now, the evaluating factor L, for a page T1 pointing to T2, is given by:

$$L(T1,T2) = X(T1,T2) * Y(T1,T2)/Z(T1) \qquad (6.2)$$

where,

T1 has a link pointing to T2.

In this example, the calculated values are shown in **Table 6.7**.

Table 6.7: Evaluating factor for 2 web pages

P	Q
-	1
1	-

Considering a d value of 0.50, the following equations are obtained:

$$PR(P) = 0.5 + 0.5(PR(Q)) \tag{6.3}$$

$$PR(Q) = 0.5 + 0.5(PR(P)) \tag{6.4}$$

After evaluating these equations for the solution, the results obtained is tabulated in **Table 6.8**.

Table 6.8: Page Rank for 2 web pages

	Page Rank
P	1
Q	1

Case 2: Web Universe with 3 web pages

Consider a web universe consisting of 3 web pages - P, Q and R. There are outbound and inbound links as shown in **Figure 6.8**.

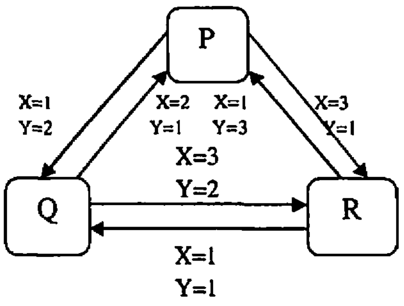


Figure 6.8: Web Universe with 3 web pages

Table 6.9 shows the *Sample inputs for 3 web pages*.

Table 6.9: Sample Inputs for 3 web pages

		P		Q		R
	X	Y	X	Y	X	Y
P	-	-	1	2	3	1
Q	2	1	-	-	3	2
R	1	3	1	1	-	-

Table 6.10: Evaluation of links for 3 web pages

Variable	Computation steps	Value returned
a	2 * 1	2
b	1 * 3	3
c	1 * 2	2
d	2 * 3	6
e	3 * 1	3
f	1 * 1	1

Table 6.10 shows the *Evaluation of links for 3 web pages.* The weighting quotients are shown in **Table 6.11**.

Table 6.11: Weighting quotients for 3 web pages

Weighting quotients	Calculation Steps	Value
Z(P)	a+b	5
Z(Q)	c+d	8
Z(R)	e+f	4

In this example, the calculated values for evaluating factor L are tabulated in **Table 6.12**.

Table 6.12: Evaluating factor for 3 web pages

P	Q	R
-	0.4	0.6
0.25	-	0.75
0.75	0.25	-

Considering a d value of 0.50, the following equations are obtained:

$$PR(P) = 0.5 + 0.5(0.25PR(Q) + O.75PR(R)) \tag{6.5}$$

$$PR(Q) = 0.5 + 0.5(0.4PR(P) + 0.25PR(R)) \tag{6.6}$$

$$PR(R) = 0.5 + 0.5(0.6PR(P) + 0.75PR(Q)) \tag{6.7}$$

After evaluating these equations for the solution, the results obtained is tabulated in **Table 6.13.**

Table 6.13: Page Rank for 3 web pages

	Page Rank
P	1.04416
Q	0.85688
R	1.13886

Case 3 : Web Universe with 4 web pages

Consider a web universe consisting of 4 web pages - P, Q, R and S. There are outbound and inbound links as shown in **Figure 6.9.**

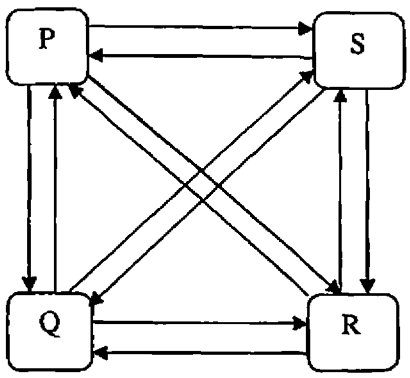


Figure 6.9: Web Universe with 4 web pages

Table 6.14 shows the *Sample inputs for four web pages.*

The links (in the example) are evaluated as shown in **Table 6.15.**

The weighting quotients are shown in **Table 6.16.**

In this example, the calculated values for evaluating factor L are shown in **Table 6.17.**

Table 6.14: Sample Inputs for 4 web pages

	P		Q		R		S	
	X	Y	X	Y	X	Y	X	Y
P	-	-	1	3	1	1	1	2
Q	2	3	-	-	2	1	1	1
R	2	3	2	1	-	-	2	3
S	2	1	2	1	1	3	-	-

Table 6.15: Evaluation of links for 4 web pages

Variable	Computation steps	Value returned
a	3 * 1	3
b	1 * 1	1
c	3 * 2	6
d	1 * 2	2
e	3 * 2	6
f	1 * 2	2
g	2 * 1	2
h	1 * 1	1
i	3 * 2	6
j	1 * 2	2
k	1 * 2	2
l	3 * 1	3

Table 6.16: Weighting quotients for 4 web pages

Weighting quotients	Calculation steps	Value
Z(P)	a+b+c	10
Z(Q)	c+d+h	9
Z(R)	e+f+i	14
Z(S)	j+k+l	7

Table 6.17: Evaluating factor for 4 web pages

P	Q	R	S
-	0.5	0.17	0.33
0.67	-	0.22	0.11
0.43	0.14	-	0.43
0.29	0.29	0.43	-

Considering a d value of 0.50, the following equations are obtained:

$$PR(P) = 0.5 + 0.5(0.67PR(Q) + 0.43PR(R) + 0.29PR(S)) \tag{6.8}$$

$$PR(Q) = 0.5 + 0.5(0.5PR(P) + 0.14PR(R) + 0.29PR(S)) \tag{6.9}$$

$$PR(R) = 0.5 + 0.5(0.17PR(P) + 0.22PR(Q) + 0.43PR(S)) \tag{6.10}$$

$$PR(S) = 0.5 + 0.5(0.33PR(P) + 0.11PR(Q) + 0.43PR(R)) \tag{6.11}$$

After evaluating these equations for the solution, the results obtained is tabulated in **Table 6.18**.

Table 6.18: Page Rank for 4 web pages

	Page Rank
P	1.19309
Q	1.00870
R	0.93134
S	0.96824

Case 4: Web Universe with 5 web pages

Consider a web universe consisting of 5 web pages - P, Q, R, S and T. There are outbound and inbound links as shown in **Figure 6.10**.

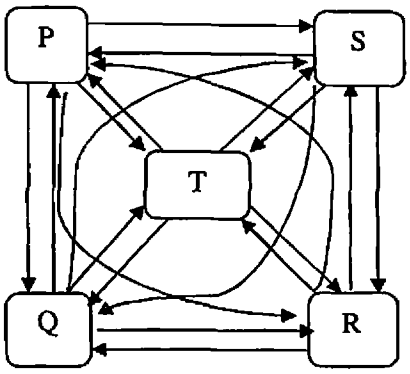


Figure 6.10: Web Universe with 5 web pages

Table 6.19 shows the *Sample inputs for 5 web pages.*

The links (in the example) are evaluated as shown in **Table 6.20**.

The weighting quotients are shown in **Table 6.21**.

In this example, the calculated values for evaluating factor L is shown in **Table 6.22**.

Table 6.19: Sample Inputs for 5 web pages

		P		Q		R		S		T
	X	Y	X	Y	X	Y	X	Y	X	Y
P	-	-	1	3	1	1	1	2	2	1
Q	2	3	-	-	2	1	1	1	3	2
R	2	3	2	1	-	-	2	3	2	3
S	2	1	2	1	1	3	-	-	1	1
T	1	1	1	1	1	1	1	2	-	-

Table 6.20: Evaluation of links for 5 web pages

Variable	Computation steps	Value returned
a	3 * 1	3
b	1 * 1	1
c	3 * 2	6
d	1 * 2	2
e	3 * 2	6
f	1 * 2	2
g	2 * 1	2
h	1 * 1	1
i	3 * 2	6
j	1 * 2	2
k	1 * 2	2
l	3 * 1	3
m	1 * 2	2
n	2 * 3	6
o	3 * 2	6
p	1 * 1	1
q	1 * 1	1
r	1 * 1	1
s	1 * 1	1
t	1 * 1	1

Considering a d value of 0.50, the following equations are obtained:

$$PR(P) = 0.5 + 0.5(0.4PR(Q) + 0.3PR(R) + 0.25PR(S) + 0.2PR(T)) \tag{6.12}$$

$$PR(Q) = 0.5 + 0.5(0.38PR(P) + 0.1PR(R) + 0.25PR(S) + 0.2PR(T)) \tag{6.13}$$

$$PR(R) = 0.5 + 0.5(0.13PR(P) + 0.13PR(Q) + 0.38PR(S) + 0.2PR(T)) \tag{6.14}$$

$$PR(S) = 0.5 + 0.5(0.25PR(P) + 0.07PR(Q) + 0.3PR(R) + 0.4PR(T)) \tag{6.15}$$

$$PR(T) = 0.5 + 0.5(0.25PR(P) + 0.4PR(Q) + 0.3PR(R) + 0.13PR(S)) \tag{6.16}$$

Table 6.21: Weighting quotients for 5 web pages

Weighting quotients	Calculation steps	Value
Z(P)	a+b+g+m	8
Z(Q)	c+d+h+n	15
Z(R)	e+f+i+o	20
Z(S)	j+k+l+p	8
Z(T)	q+r+s+t	4

Table 6.22: Evaluating factor for 5 web pages

P	Q	R	S	T
-	0.38	0.13	0.25	0.25
0.4	-	0.13	0.07	0.4
0.3	0.1	-	0.3	0.3
0.25	0.25	0.38	-	0.13
0.2	0.2	0.2	0.4	-

After evaluating these equations for the solution, the results obtained is tabulated in **Table 6.23.**

Table 6.23: Page Rank for 5 web pages

	Page Rank
P	1.11610
Q	1.00095
R	0.95189
S	1.04009
T	1.06087

The **PR** values computed by **VIS-POS** algorithm can be used for the generic ranking of web pages. **Table 6.24** highlights the various functions used for implementing *Page Rank Calculator.*

Table 6.24: Page Rank Calculator: Function and their Actions using Java

S.No	Function	Actions
1	calculate()	To calculate the page rank of web pages
2	linkAll()	To check all the boxes in the grid to represent linking of all internal pages
3	clearAll	To clear all the values so as to make the calculator ready for inputs

6.6 Formal Verification using Hoare's logic

This section deals with formal verification of the VIS-POS algorithm. Formal verification is a systematic process that uses mathematical reasoning to verify that a design satisfies the requirements.

Hoare's triple [62] is given as:

$$\{P\}Q\{R\} \tag{6.17}$$

where

P - Precondition.

Q - Program statement.

R - Postcondition.

The program statement Q is executed in a store. Before the execution of the above program statement Q, the precondition P should be satisfied. After execution of the program statement, the final store should satisfy the postcondition R. The triple asserts that for any terminating run of a program, if P holds before then R holds afterwards.

Hoare's logic makes use of four inference rules as listed below:

- D0 - Axiom of Assignments

$$P = \{x = f\}R \tag{6.18}$$

where P is derived from R by replacing all occurrences of x with f.

- D1 - Consequence rule

$$(i)\ P\{Q\}R \tag{6.19}$$

$$R \rightarrow S/P\{Q\}S\ (\textit{make a postcondition weaker}) \tag{6.20}$$

$$(ii)\ P\{Q\}R \tag{6.21}$$

$$S \rightarrow P/S\{Q\}R \ (make\ a\ precondition\ stronger) \tag{6.22}$$

- D2 - Composition rule

$$P\{Q1\}R1 \tag{6.23}$$

$$R1\{Q2\}R/P\{Q1;Q2\}R \tag{6.24}$$

- D3 - Iteration rule

  Consider the statement: while B do S

  D3: Inference rule for Iteration

$$PB\{S\}P \tag{6.25}$$

$$P\{\ while\ B\ do\ S\}\ not\ B\ or\ P \tag{6.26}$$

Consider the following axioms from arithmetic

$$x = x \tag{6.27}$$

$$x + 0 = x \tag{6.28}$$

$$X * 0 = 0 \tag{6.29}$$

Hoare's logic is applied to VIS-POS algorithm to formally verify its correctness and discussed in the following paragraphs.

Lemma:

$$x = x + y * 0 \tag{6.30}$$

In the present work, the lemma refers to Eq 6.30. It is taken as true, using the above three axioms from arithmetic. Proving the lemma is the initial step for any proof.

Consider a web universe consisting of four web pages A, B, C and D. There are outbound and inbound links as shown in **Figure 6.11.**

Let **PR** indicate the **Page Rank** of any web page.

Initially

$$total\ PR = 1 \tag{6.31}$$

$$PR(A) + PR(B) + PR(C) + PR(D) = 1 \tag{6.32}$$

$$PR(A) = PR(B) = PR(C) = PR(D) = x = 0.25 \tag{6.33}$$

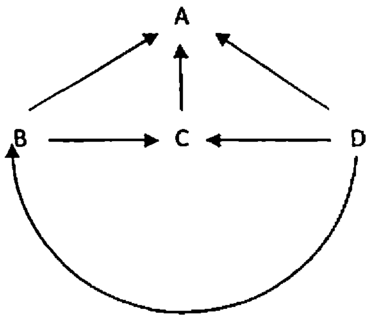


Figure 6.11: Sample Web with 4 web pages

Page A has inbound links from pages B, C and D.

$$[From\ Eq\ 6.32\ \&\ Eq\ 6.33]\ PR(A) = PR(B) + PR(C) + PR(D) = 0.75 \tag{6.34}$$

Page B has two outbound links to A and C. B votes 0.125 to A and 0.125 to C.

Page C has one outbound link to A. C votes 0.25 to A.

Page D has three outbound links to A, B and C. D votes 0.0833 to A, B and C.

$$PR(A) = PR(B)/2 + PR(C)/1 + PR(D)/3 \tag{6.35}$$

Generalizing,

$$[From\ Eq\ 6.35]\ PR(A) = PR(B)/O(B) + PR(C)/O(C) + PR(D)/O(D) \tag{6.36}$$

where, $O(B)$ - Outbound links to Page B.

$$[From\ Eq\ 6.36]\ PR(u) = \sum(PR(V)/O(V)) \tag{6.37}$$

To prove equation *d*, axioms of arithmetic are considered in the following paragraphs.

Taking axioms A0

$$[From\ Eq\ 6.27]\ x = x. \tag{6.38}$$

Multiply both sides by 4 and given $x = 0.25$

$$4x = 1 \tag{6.39}$$

$$[From\ Eq\ 6.39]\ x + x + x + x = 1 \tag{6.40}$$

$$[From\ Eq\ 6.32\ \&\ Eq\ 6.40]\quad x = PR(A) = PR(B) = PR(C) = PR(D) \tag{6.41}$$

$$x = PR(A) + PR(B) + PR(C) + PR(D) = 1. \tag{6.42}$$

$$True \rightarrow PR(A) + PR(B) + PR(C) + PR(D) = 1. \tag{6.43}$$

From **Figure 6.11** A has three inbound links from B, C and D and A has no outbound links.

$$x = 0.75\{I(A) = 3,\ O(A) = 0\} \tag{6.44}$$

where,

$I(y)$ - inbound links.

$O(y)$ - outbound links.

$$[From\, Eq\, 6.34]\; PR(A) = 0.75 \tag{6.45}$$

$$[From\, Eq\, 6.45] \quad x = PR(A) \tag{6.46}$$

$$[From\, Eq\, 6.33]\; PR(A) = 0.75 \tag{6.47}$$

$$\{x = 0.25 = PR(B) = PR(C) = PR(D)\} \tag{6.48}$$

$$PR(A) = PR(B)/1 + PR(C)/1 + PR(D)/1 \tag{6.49}$$

$$[From\, Eq\, 6.34]\; True- > PR(A) = PR(B) + PR(C) + PR(D) \tag{6.50}$$

$$\{O(B) = 2, O(C) = 1, O(D) = 3\} \tag{6.51}$$

$$[From\, Eq\, 6.36] \quad PR(A) = PR(B)/2 + PR(C)/1 + PR(D)/3 \tag{6.52}$$

$$True \rightarrow PR(A) = PR(B)/2 + PR(C)/1 + PR(D)/3 \tag{6.53}$$

From **Figure 6.11** B has two outbound links to A and C.

C has one outbound link to A and D has three outbound links to A, B and C.

$$\{O(B) = 2, O(C) = 1, O(D) = 3\} \tag{6.54}$$

$$[From\, Eq\, 6.36]\; PR(A) = PR(B)/O(B) + PR(C)/O(C) + PR(D)/O(D) \tag{6.55}$$

Hence, the above algorithm has been verified for the correctness of page rank calculation as mentioned in steps Eq 6.35 and Eq 6.36.

6.7 Summary

This chapter dealt with an interactive page rank calculation algorithm. The proposed algorithm calculates the **Page Rank** using web links attribute information such as **Visibility of a link** and **Position of a link within a document**. The algorithm has been deployed for Academic Search database and tested successfully for web universe varying from two to five web pages.

Chapter 7

Web Usage Log Analysis using Descriptive Data Mining

7.1 Introduction

Web Usage Mining involves analysis of information pertaining to web log file. It can discover the browsing patterns of the web users. It is used for modeling the navigation behavior of web users. Analysis of log file gives the user's preferences [63]. The data present in the log file cannot be used as it is, for the mining process. The contents of the log file should be preprocessed and put in a structured format. Web Usage Mining helps to improve user navigation, web design and customer satisfaction. This chapter focuses on design and implementation of an algorithm for Web Usage Log Analysis in Search Application. The algorithm accepts web log file and user queries as input and generates output containing the web log details as well as summary data.

7.2 Problem Description

The Block Schematic of Web Usage Log Analysis algorithm PRE-USE is shown in **Figure 7.1**.

The web usage log file (as mentioned in chapters 3 and 5) is taken as input for Preprocessing. The input file consists of *messages, status information* and *log data.* The preprocessing phase makes use of unix tool awk. The preprocessed file in text format is

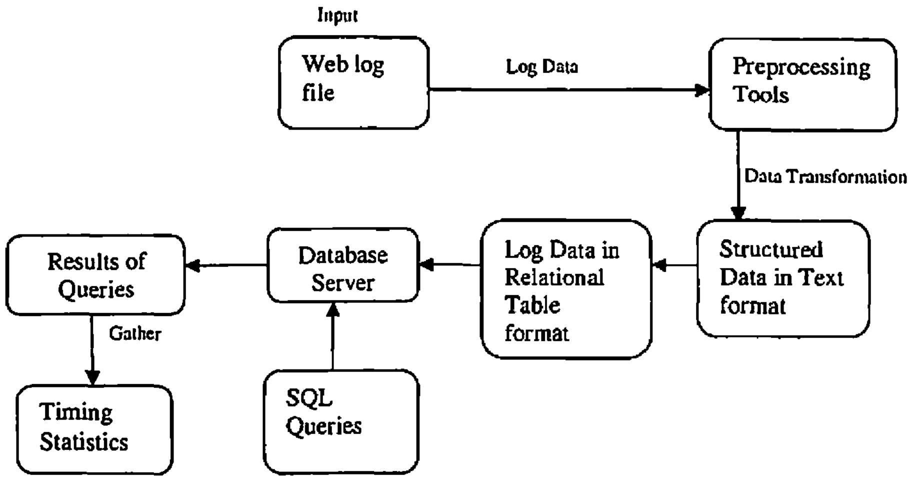


Figure 7.1: Block Schematic of Web Usage Log Analysis algorithm

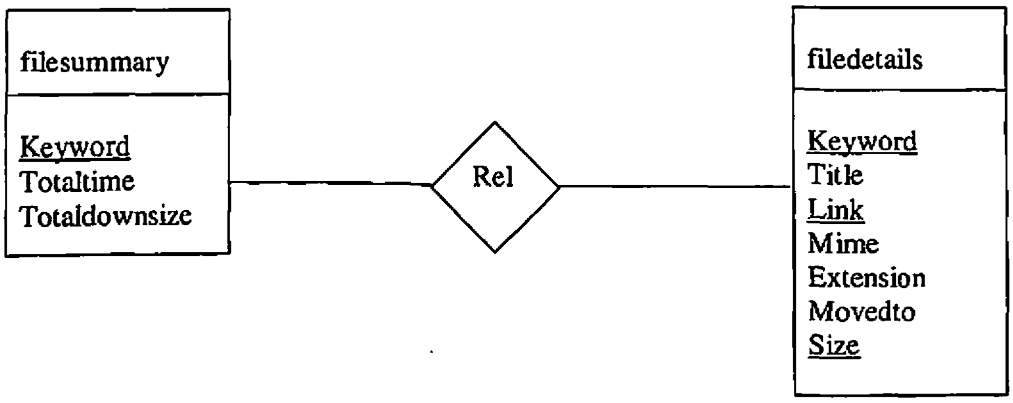


Figure 7.2: ER Diagram

transformed into a *web log file* with *log data*, in relational table. The SQL queries are submitted to the database server in the form of query file. The results of the queries are used for gathering timing statistics.

Figure 7.2 shows the ER diagram for the proposed system. *Rel* denotes the 1-to-many relationship between two tables *filesummary* and *filedetails*.The relation *filesummary* contain the attributes *Key, Time, Downsize*.The relation *filedetails* contain the attributes *Key, Title, Link, Mime, Extension, Moved, Size*.The *keyword* attribute is common to both the entities. The primary key fields are underlined for both the entities. This ER diagram is mapped to relational schema in **Section 7.4**.

7.3 Algorithm: PRE-USE

The algorithm works in two phases namely *Preprocessing* and *Log Analysis.*

Phase I: Preprocessing

Input:

Web log file in text format: contains *messages, status information* and *log data.*

Output:

Web log file in Structured format: attributes separated by delimiter "^".

Procedure:

1. The preprocessing tool is used to convert the web log file into a structured format. The Unix tool awk has been used for the conversion.

2. To populate a database from a file, the format of the file should be such that each attribute value in a row is separated by a delimiter. In this algorithm the delimiter considered has been "^" symbol for the preprocessed file.

3. For each record in the web log file do

   {

   Read current line from input web log file and perform the following actions:

   (a) Locate the string *Keyword* enclosed between two matching "=" symbols and store it in a subscripted variable arr[2].

   (b) Locate the string starting with *Title* after the ":" character and extract the substring and store it in a subscripted variable arr[3].

   (c) Locate the string starting with *Link* after the blank space and extract the substring and store it in a subscripted variable arr[4].

   (d) Locate the string starting with *File extension* after the blank space and extract the substring and store it in a subscripted variable arr[5].

   (e) Locate the string starting with *File moved to* after the "-" character and extract the substring and store it in a subscripted variable arr[6].

   (f) Locate the string starting with *File size* after the " ~" character and extract the substring and store it in a subscripted variable arr[7].

}

4. After extracting all the fields, the preprocessed file is filled with all the columns separated by the delimiter "^".

Phase II: Log Analysis

Input:

A text file containing SQL query, web log file in structured format (obtained from previous phase).

Output:

Relational table with web log data, the results of the query.

Procedure:

1. Populate the relational table using the web log file in structured format.
2. The user submits the SQL query from the query file.
3. Connect to Database server.
4. The SQL server executes the queries.
5. Assemble the query results in an output file.
6. Gather timing statistics.
7. The output file can be viewed and printed progressively.

Figure 7.3 shows the *Web log file with messages, status and log data.*

Web log file *filedetails* contains the following fields.

- Keyword
- Title
- Link
- Mime
- File Extension

```
***************Session***************
#Thu Apr 05 00:44:23 GST 2012
Current Directory: K:\Users\Mogral\Desktop\SearchAssist

************Starting download************
Search URL:
http://www.google.com/search?hl=en&safe=active&q=arduino+conference
&num=5&start=0
1. Title: Scottish Ruby Conference 2010 Arduino , Ruby RAD
Link:
http://www.slideshare.net/lostcaggy/scottish-ruby-conference-2010-arduino
-ruby-rad
------------------
mime: text/html
File extension: html
File moved to 'K:\Users\Mogral\Desktop\search data\ca\html\Scottish Ruby
Conference 2010 Arduino , Ruby RAD.html'
File size ~ 70 KB

2. Title: Arduino Forum - AT command sfor Call Conference
Link: http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1252930403
------------------
mime: text/html
File extension: html
File moved to 'K:\Users\Mogral\Desktop\search data\ca\html\Arduino
 Forum - AT command sfor Call Conference.html'
File size ~ 20 KB

3. Title: Arduino Blog » web'n'stuff
Link: http://arduino.cc/blog/category/webnstuff/page/2/
------------------
mime: text/html
File extension: html
File moved to 'K:\Users\Mogral\Desktop\search data\ca\html\Arduino
 Blog » web'n'stuff.html'
File size ~ 57 KB
```

Figure 7.3: Web log file with messages, status and log data

```
arduino conference ^Scottish Ruby Conference 2010 Arduino , Ruby
RAD^http://www.slideshare.net/lostcaggy/scottish-ruby-conference-2010-arduino-ruby-
rad^text/html^html^'K:\Users\Mogral\Desktop\search data\ca\html\Scottish Ruby Conference 2010
Arduino , Ruby RAD.html'^70 KB

arduino conference ^Arduino Forum - AT command sfor Call Conference^http://www.arduino.cc/cgi-
bin/yabb2/YaBB.pl?num=1252930403^text/html^html^'K:\Users\Mogral\Desktop\search
data\ca\html\Arduino Forum - AT command sfor Call Conference.html'^20 KB

arduino conference ^Arduino Blog »
web'n'stuff^http://arduino.cc/blog/category/webnstuff/page/2/^text/html^html^'K:\Users\Mogral\Desktop\
search data\ca\html\Arduino Blog » web'n'stuff.html'^57 KB
```

Figure 7.4: Preprocessed file layout

- File Moved To
- File Size

Figure 7.4 shows the Preprocessed file, with selected records. The awk script for Pre-Use algorithm in web usage mining is shown in **Appendix F**.

7.4 Experimental Setup

POSTGRES is the DBMS software that has been considered here. The queries are assumed to be in standard SQL format and are submitted from query files. Examples used to illustrate in this chapter are drawn from the following three categories of Search Application namely *Higher Education, Conference Alerts* and *Special Interest Group.*

7.4.1 Database Creation

This step creates the relational tables for storing the contents of web usage log files. The schema information is shown below. The following commands were used to create tables under *filedetails* and *filesummary.*

/* Relation to store Higher education *filedetails* */

```
CREATE TABLE flhe(
key varchar,
title varchar,
link varchar,
mime varchar,
extension varchar,
moved varchar,
size varchar);
```

/* Relation to store Conference Alerts *filedetails* */

```
CREATE TABLE f1ca(
key varchar,
title varchar,
link varchar,
mime varchar,
extension varchar,
moved varchar,
size varchar);
```

/* Relation to store Special Interest Group *filedetails* */

```
CREATE TABLE f1sig(
key varchar,
title varchar,
link varchar,
mime varchar,
extension varchar,
moved varchar,
size varchar);
```

/* Relation to store Higher education *filesummary* */

```
CREATE TABLE f2he(
key varchar,
time integer,
downsize varchar);
```

Table 7.1: Database Schema Information: *filesummary*

Field	Type
Key	varchar
Time	integer
Downsize	varchar

```
/* Relation to store Conference Alerts filesummary */

CREATE TABLE f2ca(
key varchar,
time integer,
downsize varchar);

/* Relation to store Special Interest Group filesummary */

CREATE TABLE f2sig(
key varchar,
time integer,
downsize varchar);
```

Table 7.1 shows the *Database Schema Information: filesummary.* This structure is applicable to all the three categories of data namely *Higher Education, Conference Alerts* and *Special Interest Group.*

The relations

- *f1he* and *f2he* corresponds to *Higher Education.*
- *f1ca* and *f2ca* corresponds to *Conference Alerts.*
- *f1sig* and *f2sig* corresponds to *Special Interest Group.*

Table 7.2 shows the *Database Schema Information: filedetails.* This structure is applicable to all the three categories of data namely *Higher Education, Conference Alerts* and *Special Interest Group.*

The following list relates the attributes in the web log file in structured format with the attributes in the relational tables.

Table 7.2: Database Schema Information: *filedetails*

Field	Type
Key	varchar
Title	varchar
Link	varchar
Mime	varchar
Extension	varchar
Moved	varchar
Size	varchar

- *Keyword* in log file corresponds to *Key* in relation.
- *Title* in log file corresponds to *Title* in relation.
- *Link* in log file corresponds to *Link* in relation.
- *Mime* in log file corresponds to *Mime* in relation.
- *File Extension* in log file corresponds to *Extension* in relation.
- *File moved to* in log file corresponds to *Moved* in relation.
- *File size* in log file corresponds to *Size* in relation.
- *Downsize* in log file corresponds to *Total size* of the downloaded files for a given keyword.
- *Time* in log file corresponds to *Time taken* for the downloaded files for a given keyword.

7.5 Test Scenario

The following queries have been successfully run over the web log database, using PSQL under POSTGRES DBMS. The output results of all these queries are stored in separate files and the timing statistics are gathered for each query.

Table 7.3: Results of Query 1

Key	Time	Downsize
arduino conference	18386	185kB
ruby conference	11692	79kB
netbeans conference	11359	107kB
....	....	....

7.5.1 SQL Queries

7.5.1.1 Query 1

(stored in text file: *he*)

Query 1 shows the records of higher education table.

select key, time, downsize from f2he;

Access details:

Site : 172.16.9.101

DBMS(s) & OS(s) : PSQL under LINUX

Databases(s): *filedetails, filesummary*

Table 7.3 shows the *Results of Query 1* (can be viewed on standard output as well as stored in an output file: *hetout*, in text format).

7.5.1.2 Query 2

(stored in text file: *ca*)

Query 2 shows the records of Conference Alerts table.

select key, time, downsize from f2ca;

Access details:

Site : 172.16.9.101

DBMS(s) & OS(s) : PSQL under LINUX

Database(s) : *filedetails, filesummary*

Table 7.4: Results of Query 2

Key	Time	Downsize
highereducation USA	43111	407kB
masters in engineering UK	28415	376kB
masters in business UK	37947	425kB
....	....	....

Table 7.5: Results of Query 3

Key	Title	Link	Mime	Extension	Moved To	Size
MySQL DBMS	MySQL DBMS Downloads	a	text/html	html	b	15kB
TCP IP	TCP IP Networks	c	text/html	html	d	35kB
.....	.....	......	......	....	.....	.....

Table 7.4 shows the *Results of Query 2* (can be viewed on standard output as well as stored in an output file: *catout* in text format).

7.5.1.3 Query 3

(stored in text file: *sig*)

Query 3 shows the records of Special Interest Group table.

select key, title, link, mime, extension, moved, size from f1sig;

Access details:

Site : 172.16.9.101

DBMS(s) & OS(s) : PSQL under LINUX

Database(s) : *filedetails, filesummary*

Table 7.5 shows the *Results of Query 3* (can be viewed on standard output as well as stored in an output file: *sigtout* in text format).

where,

a is http://dev.mysql.com/downloads

b is K: Users Mogral Desktop searchdata html MySQL downloads.html

c is http://www.faqs.org/docs/linux-network/x-087-2-intro.tcpip.html

d is K: Users Mogral Desktop searchdata html TCP IP Networks.html

7.6 Discussion

The query has been executed for varying sizes of relational table storing the log data. The test database considered here is of size 191MB for *Special Interest Group*, 4.47MB for *Higher Education* and 2.23MB for *Conference Alerts*. **Tables 7.6** and **7.7** show the experimental results for the web log file storing summary information and detailed information, in the *relations filesummary* and *filedetails* respectively. The performance graphs for the two relational tables: *filesummary* and *filedetails*, relating to Special Interest Group database is shown in **Figures 7.5** and **7.6** respectively. The X-axis corresponds to Log file size in kB and Y-axis corresponds to total execution time in seconds. From the above mentioned graphs, it can be inferred that the execution time increases almost linearly, with size of web log files. Log file analysis has been done in single user mode.

It helps the web administrator to maintain the log file in an effective way. Though it looks obvious that the execution time increases for bigger log file sizes, it is desirable to measure *<log file size, execution time>* for the experimental setup. This will help a web administrator in the maintenance of web usage log and also in administrating the web site efficiently. By analysis of log file, adequate hosting resources can be planned. The administrator can make use of the log files for the following maintenance activities:

- archiving of log records in separate folders based on keywords or range of records.
- specify the size of the current log file, where the most recent log records are stored. The remaining log records can be kept in the archive folder.

Table 7.6: Experimental Results for Web log storing summary information: *filesummary*

Log file size in kB	Execution time in seconds
823.562	0.420
1646.128	0.511
2468.694	0.617
3291.260	0.648
4113.826	0.718
4936.393	0.786
7404.091	0.897
8226.657	1.046
9049.223	1.184
9871.789	1.222
11516.921	1.314
12339.487	1.558

Table 7.7: Experimental Results for Web log storing detailed information: *filedetails*

Log file size in kB	Execution time in seconds
493.6393	1.257
575.8959	1.395
658.1525	1.451
822.6657	1.581
1151.6921	1.637
1233.9487	1.728
1316.2053	1.744
1398.4619	1.754
1480.7185	1.799

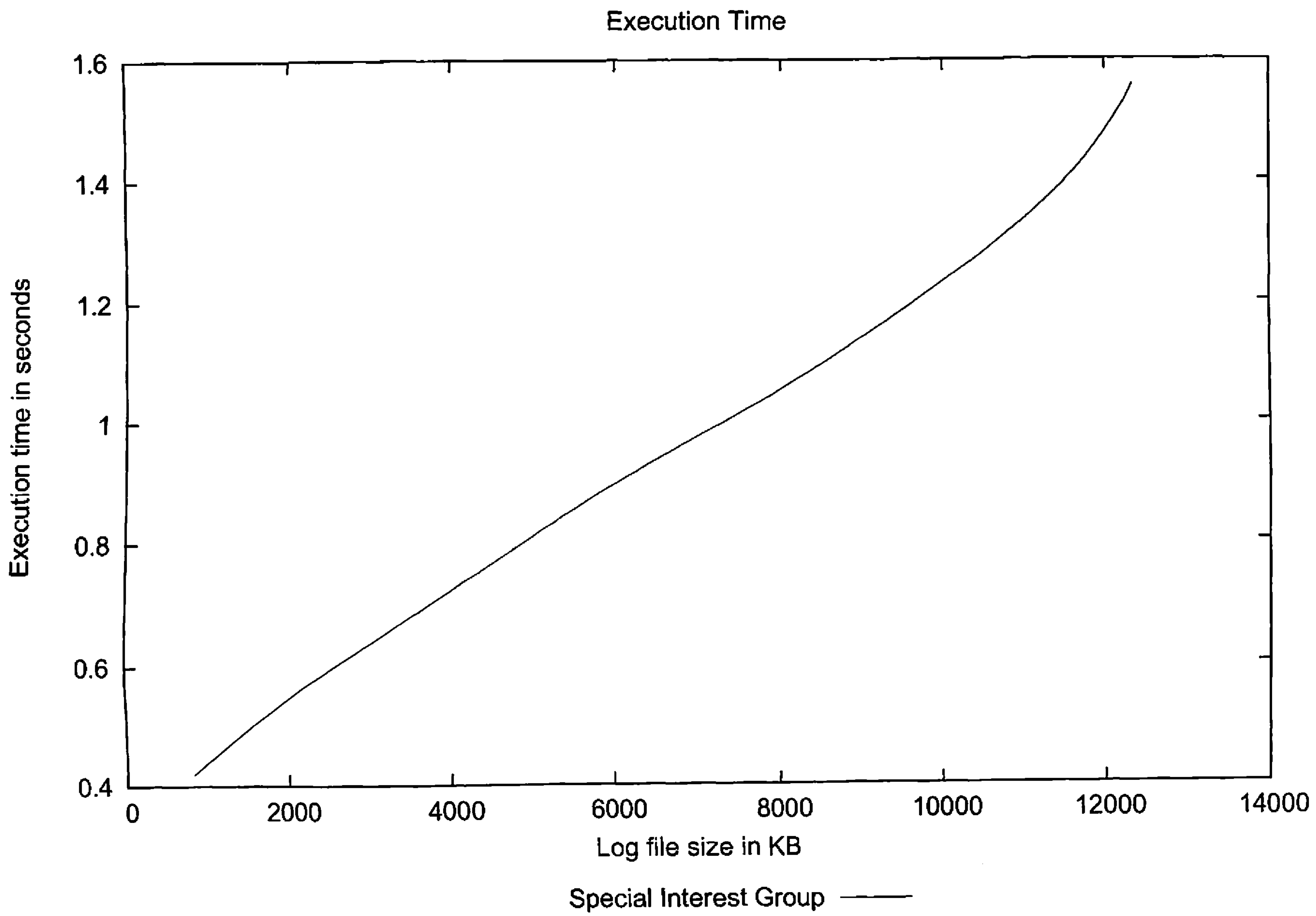


Figure 7.5: Performance Graph of the relation *filesummary*

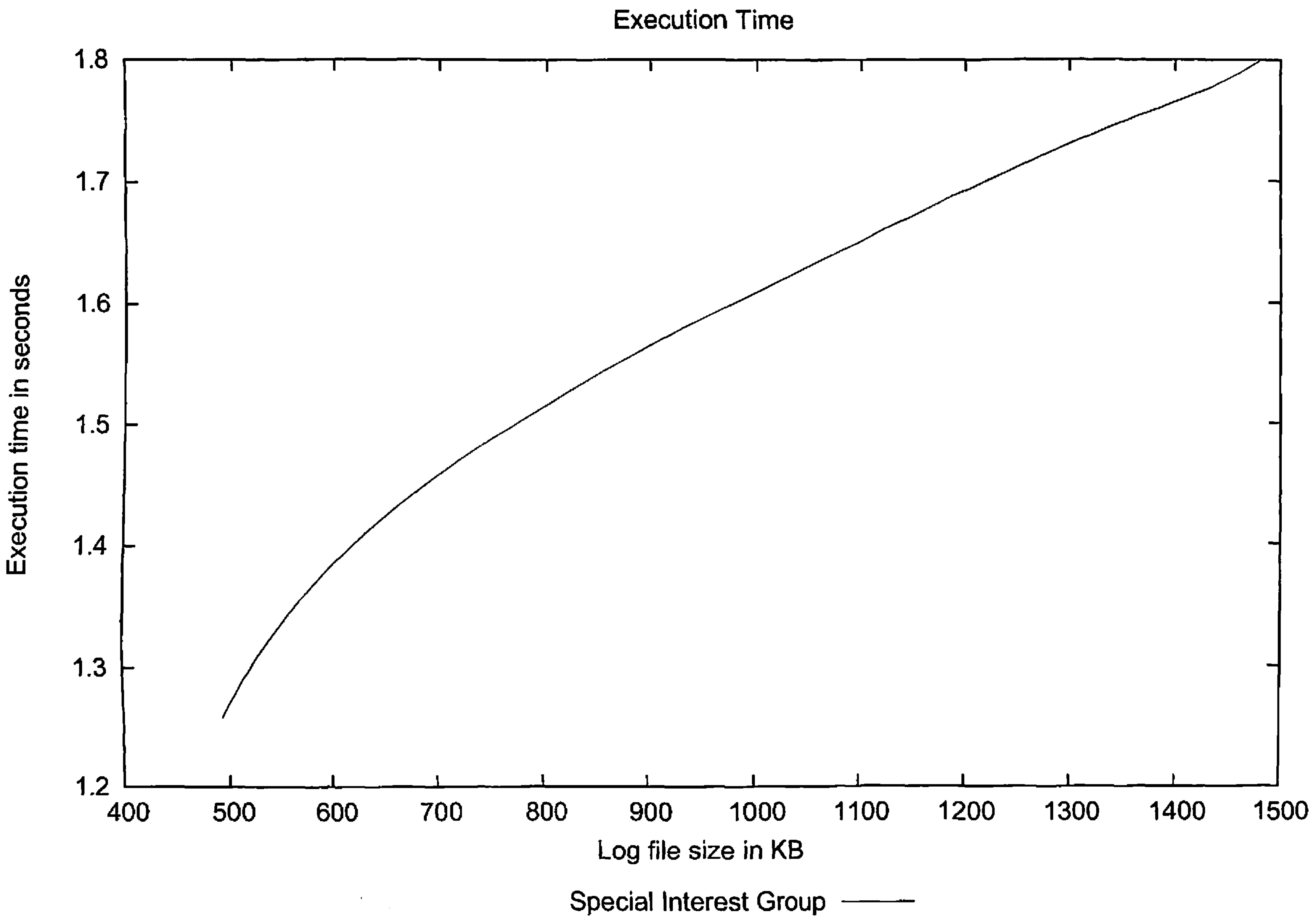


Figure 7.6: Performance Graph of the relation *filedetails*

7.7 Creation of indexes using SQL

The input tables stores the detailed information about the web log files. Indexes are created to provide easy and efficient access to web log information. The SQL queries along with their actions are given below:

7.7.1 Index on Key and Link

Query 4 creates an index on two fields key and link.

create index indexhel on flhe(key,link);

select * from flhe;

This query creates an index on two attributes *key* and *link* over the relation *flhe* corresponding to the log data of higher education. It also saves the output results in a text file *heout1*.

7.7.2 Index on Key, Type and Size

Query 5 creates an index on three fields key, type and size.

create index indexhe2 on flhe(key,extension,downsize);

select * from flhe;

This query creates an index on three attributes *key,extension,size* corresponding to the log data of higher education. It also saves the output results in a text file *heout2*.

7.7.3 Index on Key and Total download size

Query 6 creates an index on two fields key and total download size.

create index indexhe3 on f2he(key,downsize);

select * from f2he;

This query creates an index on two attributes *key, downsize* corresponding to the log data of higher education. It also saves the output results in a text file *heout3*.

7.8 Interestingness measures : t-weight and d-weight

Data mining can be based on two categories: Descriptive and Predictive [7]. Descriptive data mining considers summary data for the data set. Predictive data mining tries to predict the behavior of new data sets with reference to existing set of models [7]. The current work considers descriptive data mining using concept description. The concept in the current context refers to collection data pertaining to web log file. Concept description falls under two types:

- *Characterization:* This provides a brief summary of input data set belonging to a particular class (html/pdf/txt/ppt). In the current work, the summary data refers to attributes such as *file type* and the range of file sizes for input keywords. The interestingness measure calculated here is *t-weight.*

- *Comparison:* This provides descriptions comparing two or more collections of data [7]. In the current work, the classes of data are based on file type (html/pdf/txt/ppt). The interestingness measure calculated here is *d-weight.*

7.8.1 t-weight

The parameters for t-weight are shown below:

t-weight: tuple typicality

q_a: a generalized tuple describing the target class

t-weight can be in the range [0.0,1.0] or [0 %, 100 %]

$$t\_weight = \frac{count(q_a)}{\sum_{i=1}^{n} count(q_i)}$$

n: total number of tuples for the target class in the generalized relation.

$q_1, q_2, q_3 .... q_n$ are tuples of target class in the generalized relation.

q_a is in $q_1 ... q_n$.

t-weight can be specified for 1 or more attributes. t-weight can be set with SUPPORT (minimum threshold value). If t-weight < threshold, the tuple represents negligible portion of database, hence it is uninteresting.

7.8.2 d-weight

The parameters for d-weight are shown below:

d-weight : tuple discriminability

C_j : target class

q_a : a generalized tuple

covers some tuples of target class / contrasting class.

d-weight - range:[0,1] or [0%,100%]

m - total number of *target* and *contrasting* classes.

$$d\_weight = \frac{count(q_a \in C_j)}{\sum_{i=1}^{n} count(q_a \in C_i)}$$

A lower value of t-weight indicates that the generalized tuple in the target class refers to a smaller portion of the database. A larger value of t-weight indicates that the generalized tuple in the target class has significant number of records in the database. A lower value of d-weight indicates that the concept is derived from the contrasting classes. A larger value of d-weight in the target class indicates that the concept represented by the generalized tuple is derived from the target class.

7.8.3 Significance of t-weight and d-weight in the current work

The current work considers the following two parameters w.r.t t-weight and d-weight calculations:

- **Generalized tuple:** Each generalized tuple is identified by *<file size>*. The generalized tuples considered here are as follows: file size < 100kB; file size >= 100kB and <200kB; file size >= 200kB.

- **Class:** The class refers to *<file extension>*. The current work considers the following four classes: html, pdf, txt and ppt.

The t-weight and d-weight values are tabulated in **Table 7.10** and **Table 7.11** respectively. Referring to **Table 7.10**, a lower value for t-weight, say 16.7 indicates that the generalized tuple identified by the file size *<file size >= 100kB>* refers to a smaller portion of the database in the target class *<html>*. A higher value for t-weight, say 46.6

indicates that the generalized tuple identified by the file size *<file size < 100kB>* has significant number of records in the database, within the class *<txt>*.

Referring to **Table 7.11**, a lower value for d-weight, say 5.26 indicates that significant number of records are derived from the contrasting classes (i.e pdf, txt and ppt, the target class here is html). A higher value of d-weight, say 50 indicates that the concept represented by the generalized typle *<file size < 100kB>* is derived from the target class *<txt>*.

To summarize the t-weight of a generalized tuple shows how typical is the tuple in the given class. The d-weight of a generalized tuple shows how distinctive is the tuple in the target class in comparison with its contrasting classes.

7.8.4 SQL queries for extracting summary information

The SQL queries are executed to gather the summary information required for descriptive data mining using *t-weight* and *d-weight.*

7.8.4.1 Query 7

```
select extension, key, size, count(*) as vote
from flsig
where extension = 'html'
group by extension, key, size;
```

7.8.4.2 Query 8

```
select extension, key, size, count(*) as vote
from flsig
where extension = 'pdf'
group by extension, key, size;
```

7.8.4.3 Query 9

```
select extension, key, size, count(*) as vote
from flsig
```

Table 7.8: Cross tab with different keywords

Extension	Keywords	Size	Vote	t-weight
html	Adobe molehill	10	30	10
html	Adobe molehill	79	60	20
html	ARM Development	18	120	40
html	ARM Development	24	90	30
pdf	Google app engine	122	30	33.3
pdf	Introduction to graphic programming	16	60	66.6
txt	Opengl specification	60	30	50
txt	Vertex shaders	119	30	50
ppt	Gimp tutorials	241	30	50
ppt	Web Structure Mining	43	30	50

Table 7.9: Cross tab for file size(count) and file extension

Extension	Count I	Count II	Count III	Total
html	10	15	5	30
pdf	25	20	15	60
txt	70	35	45	150
ppt	35	25	30	90
Total	140	95	95	

where extension = 'txt'

group by extension, key, size;

7.8.4.4 Query 10

select extension, key, size, count(*) as vote

from flsig

where extension = 'ppt'

group by extension, key, size;

The above SQL queries return the results for *keyword, filesize* and *count* (no. of files of a particular file type such as *html, pdf, txt* and *ppt*). *vote* refers to the number of files for a particular keyword.

Count I : File size < 100kB

Count II : File size >= 100 kB and < 200kB

Table 7.10: Cross tab with t-weight values associated for each class

Extension	Count I	Count II	Count III
html	33.3	50	16.7
pdf	41.6	33.3	25
txt	46.6	23.3	30
ppt	38.8	27.8	33.3
Total	140	95	95

Table 7.11: Cross tab with d-weight values associated for each class

Extension	Count I	Count II	Count III
html	7.14	15.8	5.26
pdf	17.8	21.1	15.79
txt	50	36.8	47.37
ppt	25	26.3	31.58
Total	140	95	95

Count III : File size >= 200kB

Table 7.8 shows the *Cross tab with different keywords.* **Table 7.9** shows the *Cross tab for file size(count) and file extension.* The cross tabs have been constructed using the SQL output from the **Tables 7.8 and 7.9.**

Table 7.10 shows the *Cross tab with t-weight values associated for each class.* As a test case, the calculation for one entry corresponding to (html, filesize < 100kB) is shown below:

$$10/30 * 100 = 33.3$$

Table 7.11 shows the *Cross tab with d-weight values associated for each class.* As a test case, the calculation for one entry corresponding to (html,filesize< 100kB) is shown below:

$$10/140 * 100 = 7.14$$

7.9 Complexity

Table 7.12 shows the Complexity of SQL Operations, considered in Web Usage Log Analysis. In the table, n corresponds to number of records in the input relations storing the web log data.

Table 7.12: Complexity of SQL Operations

SQL Operation	Complexity	Explanation
SELECT	O(n)	In the worst case, the select operation can involve the entire relation having n records
PROJECT	O(n)	This also involves a scan of the entire relation with n records, even if few attributes are selected in the output

From **Table 7.12**, it can be seen that the operations SELECT and PROJECT involve retrieval of information from the entire input relations.

7.10 Summary

This chapter dealt with an algorithm for Web Usage Log Analysis. The algorithm works in two phases namely *Preprocessing* and *Log Analysis.* Web usage summary log file is taken as input and preprocessed by converting it into text files with appropriate delimiters between various attributes. The database is populated from the preprocessed file. The relation between two entities *filesummary* and *filedetails* were drawn using a ER diagram. Cross tab with *t-weight* and *d-weight* values associated for each class, have been tabulated. SQL queries are submitted to the database server and the results are recorded. The timing statistics is gathered for each query. The query has been executed for different sizes of web log file. The algorithm can be used by various users in an academic environment.

Chapter 8

Conclusion, Contributions and Future Work

The present work has been carried out to meet the *objectives* and *scope* outlined in the first chapter. The application of web mining algorithms for the search application has been dealt with in greater detail in the successive chapters.

Chapter 2 presented an overview of existing **literature** on *Web Mining.* Web Mining can be broadly divided into three categories: *Web Content Mining, Web Structure Mining* and *Web Usage Mining.* Web Content Mining involves exploration of unstructured and semi-structured data. An overview of three Web Structure Mining algorithms, namely, PageRank, TrustRank, HITS and Sel-HITS have been presented. *Web Usage Mining* focuses on analysis of web log files for extracting information pertaining to access patterns of web users. The relevance of data warehouse functions in web mining has also been discussed.

Chapter 3 dealt with **WDES** algorithm. The proposed algorithm performs data extraction, grouping and storage functions for web users. The algorithm uses the search engines Google and Yahoo at the back end. They can access the web content in an organized way. The above algorithm reduces the time and effort in the search process. The performance of the algorithm has been analyzed. It can be effectively used by various categories of web users, in their search related activities.

Chapter 4 dealt with design of a **user Interface**, to provide navigation options, in order to access the web content. The search database mainly stores the fields *Keyword,* *Title, Link, File extension, File moved to* and *File size* in a relational table. The database is hosted on the web server. The users can access the above database by using a standard interface through web browser. It also presented an algorithm for repository maintenance operations over academic search related data on *Higher Education, Conference Alerts* and *Special Interest Group.* The algorithm has been successfully deployed in a web server using a standard interface to help the web administrator for easy maintenace of repository. Repository has been organized as a collection of folders and file types, stored in the web server. The web administrator can perform a range of options for folders and files, in the interactive mode.

Chapter 5 dealt with **Links-Traversal** algorithm for **implicit graph.** It makes use of Breadth First Search Strategy for traversing the hyperlinks. The timing statistics are gathered from the log messages, during the search process. It has been observed that the execution time for crawling is determined by the number of links and the network bandwidth.

Chapter 6 focused on interactive Page Rank Calculation algorithm for **web structure mining.** It considers two main factors, namely *Visibility of a link* and *Position of a link within a document.* A web user can view the page ranks for a given web universe of web pages using a standard interface.

Chapter 7 presented an algorithm for **web usage log analysis.** The algorithm works in two phases namely, *Preprocessing* and *Log Analysis.* The web usage log files are first converted into text files with appropriate delimiters between various attributes, in the preprocessing phase. In the Log Analysis phase, the *web usage database* is created using the preprocessed text files. SQL queries are applied over the database to provide output results and summary data, on web usage, by various users in an academic environment. Interestingness measures relating to t-weight and d-weight associated for each class, have been tabulated using Descriptive data mining.

8.1 Contributions

The specific contributions relating to the present research work are highlighted here.

1. A WDES algorithm for data extraction, grouping and storage functions has been

proposed. The above algorithm significantly improves the relevance of search results, by specifying the main parameters (such as keywords, no. of links, path name on the local server for storage, file type / extension and file size). It provides search assistance and reduces the time and effort in the search process.

2. Design of **Links-Traversal** algorithm using Breadth First Strategy for **Implicit graph**, helps to narrow down the search space to specified number of links, for user submitted queries. The user is provided with the option to view the content of a link or download the content on demand to the specified folder. Accuracy of the information content pertaining to keywords, can be ascertained by a quick check on search links.

3. An **Interactive Page Rank calculation** algorithm in **web structure mining** has been presented. The algorithm offers a simulation environment for a web user to define a web universe with a given number of incoming and outgoing links in the grid. The connections between the individual pages can be modeled. Then, the page ranks are calculated for each of the web pages and displayed in the output column instantly. This algorithm will help a web administrator in identification of different types of web pages.

4. The **web usage log Analysis** algorithm will greatly help web administrators in their activities relating to analysis and maintenance of web usage logs. It helps in conducting the audit trail on web usage statistics of academic users. Interestingness measures: t-weight and d-weight, relating to descriptive data mining have been calculated for the input classes.

8.2 Directions for Future Work

The possible extensions to the present work may be listed as follows:

1. The **WDES** algorithm can be extended to handle additional media types such as audio, video, images and graphical data. The queries can involve content extraction from the above media types, simultaneously using keywords.

2. The repository can be extended to incorporate classification scheme for the input keywords. The classification scheme can be based on one or more attributes like subject or field of interest.

3. The web provides a dynamic environment where in the number of links corresponding to a given set of keywords change frequently. Newer links may be added or some of the existing links may become obsolete. The repository can be updated incrementally at regular intervals (such as daily, weekly, fortnightly and so on) to reflect the most relevant information about search links.

4. The **web content mining** can be performed over a federated database system. A federated database system provides secure access to multiple databases for its users, using a single query.

5. The **web structure mining** can be extended to analyze the types of web links embedded in web pages. They include interior links, exterior links and links relating to mirrored sites. The interior links of a web page indicate a web document's ability to cross-reference other related web pages within the same document. The exterior links span over different web sites and indicate a web document's ability to cross-reference related web pages across different sites. The links relating to mirrored sites return identical results.

6. The interactive Page Rank Calculation Algorithm can be extended to handle a web universe of any number of input web pages. The present work considers a web universe upto ten pages.

7. The **web usage log analysis** algorithm can be enhanced to analyze all categories of log files such as Common Log Format and Extended Log Format.

References

[1] Cooley R., Mobasher B., Srivastava J. "Data preparation for Mining World Wide Web Browsing Patterns" *Journal of Knowledge and Information Systems.* 1999, 1:5-32.

[2] Madiraju P., Zhang Y. Q., Owen S., Sunderraman., Zhu Y. "Graphical Web Mining Agent for Class Teaching Enhancement" *International Journal for Infonomics.* 2006, 3:243-249.

[3] Kosala R., Blockeel H. "Web Mining Research: A Survey" *ACM SIGKDD Explorations.* 2000, 2:1-15.

[4] Mobasher B., Jain N., Han E.H., Srivastava J. *Web Mining: Pattern Discovery from World Wide Web Transactions.* Minneapolis (MS) : University of Minnesota, Department of Computer Science; 1996 Feb Report No.: TR 96-050, DA|DAAH04-95-1-0538. NSI grant ASC-9634719.

[5] Galeas P. *"Web Mining".* [Online]. Available: `http://www.galeas.de/webmining.html`. [accessed: 18 Sep 2009].

[6] Cooley R., Mobasher B., Srivastava J. "Web Mining: Information and Pattern Discovery on the World Wide Web" *ACM SIGKDD Explorations Newsletter.* 2000, 1:12-23.

[7] Han J., Kamber M. *"Data Mining Concepts and Techniques"* 2nd ed. Morgan Kaufmann, San Fransisco, USA. 2001, 435-441.

[8] Gore M.M., Mishra A.K. *Algorithm for Data Mining.* [CD-ROM]. Allahabad, India. Proceedings of Winter School on Data Mining; 2001.

[9] Zhou X., Li Y., Bruza P., Wu S.T., Xu Y. "Using Information Filtering in Web Data Mining Process", *Proceedings of the ACM International Conference on Web Intelligence*, Silicon Valley, California, USA, 2007, Nov 2-5, pp. 163-169.

[10] Pol K., Patil N., Patankar S., Das C. "A Survey on Web Content Mining and Extraction of Structured and SemiStructured Data", *First International Conference on Emerging Trends in Engineering and Technology* ICETET, Nagpur, India, 2008, pp. 543-546.

[11] Jeyalatha S., Vijayakumar B., Hazarika E.A. "Design of an Interface for Page Rank Calculation using Web Link Attribute Information" *Informatica Economica.* 2010, 14:142-152.

[12] Jeyalatha S., Vijayakumar B., Dhingra S. "Design and Implementation of Data Warehouse Maintenance Operations in an Academic Environment using Java", *International Conference on Computing,* New Delhi, India, 2010, Dec 27-28, pp. 27-28.

[13] Jeyalatha S., Vijayakumar B. "Web Mining Functions in Academic Search Application" *Informatica Economica.* 2009, 13:132-139.

[14] Ou J.C., Lee C.H., Chen M.S. "Efficient algorithms for incremental Web log mining with dynamic thresholds" *The VLDB Journal - The International Journal on Very Large DataBases.* 2008, 17:827-845.

[15] Ding C., Zhou J. "Log Based Indexing to Improve WebSite Search". *Proceedings of the ACM Symposium on Applied Computing,* Seoul, Korea, 2007, Mar 11-15, pp. 829-833.

[16] Zhou J., Ding C., Androutsos D. *"Improving Web site search using web Server Logs".* [Online]. Available: `http://delivery.acm.org/10.1145/119000/1188996/html`. [accessed: 20 Nov 2009].

[17] Sendhilkumar S., Geetha T.V. "Personalized ontology for Web Search Personalization", *Proceedings of the 1st Bangalore Annual Computer Conference,* Bangalore, India, 2008, Jan 18-20, pp. 1-7.

[18] Markellos K., Markellou P., Rigou M., Sirmakessis S. "Web Mining: Past, Present and Future", *Proceedings of the NEMIS Launch Conference*, Patras, Greece, 2003, Apr 5, pp. 26-36.

[19] Srivastava J., Cooley R., Deshpande M., Tan P.N. "Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data" *ACM SIGKDD Explorations.* 2000, 2:12-23.

[20] Pujari A.K. *"Data Mining Techniques"* 1st ed. Universities Press(India) Private Limited, Hyderabad, India. 2001, 231-239.

[21] Borzemski., Leszek. "The Usage of Data Mining to predict Web performance" *Cybernetics & Systems.* 2006, 31:587-608.

[22] Blumberg R., Atre S. The Problem with Unstructured Data, [Monologue from the Internet]. Information Management Magazine; Feb 2003. Available: `http://www.information-management.com/issues/20030201/6287-1.html`.

[23] Sharker S. "Semistructured Data". [Online]. Available: `http://www.cis.upenn.edu/~db/abstracts/semistructured.html`. [accessed: 15 May 2009].

[24] Enhong C., Xufa W. "Semi-structured Data Extraction and Schema Knowledge Mining", *25th Euromicro Conference (EUROMICRO '99)*, Volume 2, Milan, Italy, 1999, pp. 310-317.

[25] Ribeiro N.B., Laender H.F., A. S. da Silva. "Top Down Extraction of Semi-Structured Data", *String Processing and Information Retrieval Symposium & International Workshop on Groupware*, 1999, pp. 176.

[26] Li Z., Keong W. "WICCAP: From Semi-structured Data to Structured Data", *11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'04)*, Singapore, 2004, pp. 86-93.

[27] Mooney R.J., Nahm U.Y. "Text Mining with Information Extraction", *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, Austin, Texas, 2009, pp. 1-16.

[28] Bourdon R. "Wamp Server", [Online]. Available: `http://www.wampserver.com/en/`. [accessed: 11 Apr 2011].

[29] Hall., Brown. *"Core Web Programming"* 2nd ed. Prentice Hall, Palo Alto, California, 2004, 3-29.

[30] Cunningham H. "Information Extraction", *Elsevier B.V.*, 2004.

[31] Sullivan S. "Google Scholar offers Access to Academic Information". [Online]. Available: `http://searchenginewatch.com/article/2048646/Google-Scholar`. [accessed: 18 Nov 2009].

[32] S. Chakrabarti et.al. *"Mining the Web: Analysis of Hypertext and Semi Structured Data"* 1st ed. Morgan Kaufmann, San Francisco, USA, 2002, 45-57.

[33] Jamsa K., Konrad., Anderson A. *"HTML and Web Design Tips and techniques"* 4th ed. Wiley Computer Publishing, 2004, 1-688.

[34] Chakrabarti S., and et.al, "Mining the Web's Link Structure" *Computer.* 1999, 32:60-67.

[35] Page L., Brin S., Motwani., Winograd. *The Page Rank Citation Ranking: Bringing Order to the Web*, Technical Report, Stanford University, Stanford, CA, 1999.

[36] Langville A., Meyer C. "A Survey of Eigenvector Methods for Web Information Retrieval",*SIAM Review*, Vol. 47, pp. 135-161, 2005.

[37] Larson R.R. "Bibliometrics of the World Wide Web: An Exploratory Analysis of the Intellectual Structure of Cyberspace", *Proceedings of the ASIS'96 Annual Conference*, Berkeley, USA, 1996, pp. 71-78.

[38] Search Engine Optimization services and articles, inc Page Rank explained, search engine optimization forum. UK based, worldwide clients, [Online]. Available: `http://www.searchenginegenie.com` [accessed: 11 Jul 2009].

[39] Richardson M., Domingos P. "The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank", *Proceedings of Advances in Neural Information Processing Systems*, Washington, USA, 2002, pp. 673-680.

[40] Brin S., Page L. "The Anatomy of a Large Scale Hypertextual Web Search Engine", *Proceedings of 7th International WWW Conference*, Brisbane, Australia, 1998, pp. 107-117.

[41] Kleinberg J.M. "Authoritative sources in a hyperlinked environment", *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, USA, 1998, pp. 668-677.

[42] Puig A., Ripolles O., Chover M. "Surveying the Identification of Communities" *International Journal of Web Based Communities*. 2008, 4:334-347.

[43] McGlohon M., Akoglu L., Faloutsos. "Weighted Graphs and Disconnected Components: Patterns and A Generator", *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Las Vegas, Nevada, USA, 2008, Aug 24-27, pp. 524-532.

[44] Boullossa J.R.F., Xexeo G. *An Architecture for Web Usage Mining*. [Online]. Available: `http://www.boullosa.org/artigos/Arch-Web-Mining-2002.pdf` [accessed: 16 Jul 2010].

[45] Choenni S. "Design and Implementation of a Genetic Based Algorithm for Data Mining", *Proceedings of the 26th International Conference on Very Large DataBases*, Cairo, Egypt, 2000, Sep 10-14, pp. 33-42.

[46] Punin J.R., Krishnamoorthy M.S., Zaki M.J. *Web Usage Mining: Languages and Algorithms*. Troy(NY): Rensselaer Polytechnic Institute, Department of Computer Science ; 2001 Jan. Report No.: 99-10.

[47] Ceddia J., Sheard J., Tibbey G. "WAT - A Tool for Classifying Learning Activities from a Log File", *Proceedings of the Ninth Australasian Computing Education Conference (ACE2007)*, Ballarat, Australia, 2007, Jan 30 - Feb 2, pp. 11-18.

[48] Mobasher B., Cooley R., Srivastava J. "Automatic Personalization Based on Web Usage Mining" *Communications of the ACM*. 2000, 43:142-151.

[49] Loton T. *"Web Content Mining"* 1st ed. Wiley, New Jersey, USA. 2002, 1-15.

[50] Niksic H., GNU Wget for windows, [Online]. Available: http://gnuwin32.sourceforge.net/packages/wget.htm. [accessed: 12 Aug 2011].

[51] Niksic H., Scrivano G. "GNU WGet", [Online]. Available: http://www/gnu.org/software/wget/ [accessed: 2 Sep 2010].

[52] HTML Cleaner Team, "HTML Cleaner". [Online]. Available: http://htmlcleaner.sourceforge.net/ [accessed: 21 Feb 2011].

[53] Bray T., Paoli J., Sperberg C.M., Maler E., Yergeau F. "Extensible Markup Language 1.0". [Online] Available: http://www/w3.org/TR/xml. [accessed: 16 May 2011].

[54] XPath Tutorial, [Online]. Available: http://www.w3schools.com/xpath/default.asp [accessed: 30 Jun 2009].

[55] Freed N., Borenstein N. MIME Format Specification, [Online]. Available: http://www.ietf.org/rfc/rfc2045.txt [accessed: 15 Jan 2011].

[56] Giada de Laurentis., List of MIME Types, [Online]. Available: http://reference.sitepoint.com/html/mime-types-full. [accessed: 26 Sep 2009].

[57] Refsnes Data 2011, W3Schools Online Web Tutorials. [Online]. Available: http://www.w3schools.com/php/default.asp. [accessed: 23 Oct 2010].

[58] SQL Tutorial - Learn SQL. [Online]. Available: http://www.sql-tutorial.net. [accessed: 27 Mar 2011].

[59] Gal A., John M. "Toward Web Based Application Management Systems" *IEEE Transactions on Knowledge and Data Engineering.* 2001, 13:683-702.

[60] Clark J., DeRose S. W3C XPath Specifications [Online]. Available: http://www.w3.org/TR/Xpath. [accessed: 14 May 2011].

[61] Albert R., Jeong H., Barabasi A.L. "Diameter of the World Wide Web" *Nature.* 1999, 401:130-131.

[62] Hoare's method for proving correctness of program, [Online]. Available: `ie.technician.ac.il/~ofers/Courses/verification/c2.1-Hoare.ppt`. [accessed: 18 Sep 2010].

[63] Grace L.K., Maheshwari V., Nagamalai D. "Analysis of Web logs and Web User in Web Mining" *International Journal of Network Security and Its Applications.* 2011, 3:1-12.

[64] Michael T.G., Tamassia R. *"Algorithm Design: Foundations, Analysis and Internet examples"* 4th ed. John Wiley & Sons, New Jersey, USA. 2002, 253-255.

[65] Liu B. *"Web Data Mining: Exploring Hyperlinks, Contents and Usage Data"* 2nd ed. Springer, New York, USA. 2007, 452-454.

[66] Ramakrishnan R., Gehrke J. *"Database Management Systems"* 3rd ed. McGraw Hill, New York, USA. 2002, 284-285.

[67] Liu B. *Personal Evaluations of Search Engines: Google, Yahoo and Live(MSN).* [Online]. Available: `http://www.cs.uic.edu/~liub/searchEval/SearchEngineEvaluation.htm`. [accessed: 21 Apr 2012].

Appendix A

System Configuration

The present research work has been carried out using the following system configuration:

Computer Programming Laboratory

72 nodes in LAN, comprising of Acer / IBM PCs, Print Server, Windows and Linux OS, Internet access using 100Mbps and Software tools under Linux and Windows.

The typical configuration for each PC is as follows:

Intel Pentium 4: 2.8 GHz ACPI.

512 MB RAM.

40 GB Hard Disk.

1.44MB Floppy Disk.

CD ROM Drive.

Broadcom Net Xtreme Gigabit Ethernet Network Card.

InterR 82865 Graphics Controller Display Adapter.

Webhostingpad.com (204.93.183.17)

The following software are hosted on this server:

PHP, MySQL, Java Development Environment, CGI, SSI, Perl, Python.

Software

RedHat Enterprise LINUX 4.7, Apache Web Server, PHP, MySQL, POSTGRES SQL, HTML, XML and Java2SDK tools.

Document Preparation : LATEX, Visual Editor, GNUPLOT and MS-Office Tools.

Appendix B

Functions and their Actions for WDES: Implementation using Google

Functions	**Description**
makeConfigFileTemplate	This function is invoked to set the default properties and their values before loading the configurations from file or before creating a new configuration file.
SaveConfigFile	Save configurations to config.ini(file).
SaveData	Invokes saveConfigFile and also takes all log messages from UI and appends onto log file.
Search	Search button from UI invokes this function to create a new thread and a new instance of downloadThread for initiating the search.
DownloadThread	Class that deals with the search.
makeSearchURL	From the options given by user through the UI and the search query specified, this function creates a partial URL to google search and it is called from the *run* function and the result of this function is saved to downloadThread.url.

Cont. on next page

Functions	**Description**
SearchthisURL	Using downloadThread.url, this function manages the search. It splits and limits the search to 100 links per request. It calls *downloadLinks* for every 100 links with the required parameters.
Run	Manages the search, one search query at a time. For each search query, it first calls makeSearchURL and saves the result to downloadThread.url and then it calls *searchThisURL.*
downLoadLinks	With the given XPath expression, offset and number of results to show as its parameters, this function does the following: 1. Adds the offset and number of results to the URL in downloadThread.url. 2. Calls Wget to download Google search page. 3. Uses HTMLCleaner to convert html to xml. 4. Build DOM Document from the xml and use the specified XPath expression to find the search results (which are just links) from the page. 5. Traverses through the links and downloads each link. For each link, the following is done: a. Determine their MIME type of the download link. b. Maps the MIME to a file extension from a predefined set of MIME-to-file extension mapping saved in a file. c. Renames the file to required file extension and moves the file to a folder based on the file extension.

Appendix C

Functions and their Actions for WDES: Implementation using Yahoo

Functions	**Description**
search	It creates the complete search query for a yahoo search and invokes the getLinks() function.
getLinks	This function loads the yahoo search result page in a document object using Jsoup, an external HTML parser library. Jsoup is used to parse the HTML document obtained to extract the links in the page that correspond to the search result. The links extracted are stored in a queue and then removed one by one and passed to the savePage() function.
savePage	It receives the link to be saved from the getLinks() function. The link is then converted to a URL object which is then used to open a connection to the web page. The file corresponding to the link is read byte by byte and then stored in a file in the appropriate folder.
savePageImage	Function to take the screenshot of the web page.
saveSearchResults	Function to save the search result page in the local hard disk.

Appendix D

Functions and their Actions for Links-Traversal Algorithm : Java Implementation

Functions	Description
LinksTraversalView	This function is used to set the default properties and values and also to initialize status bar, message timeout and connecting action tasks to status bar.
SaveConfigFile	Save configurations to config.ini(file).
SaveResults	It takes all log messages from the User Interface and appends onto log file.
Search	Search button from the User Interface invokes this function to initiate search by creating a new thread, downloadThread.
DownloadThread	Class that deals with the search.
SearchKeywords	This function creates a partial URL to google search and it is called from the *run* function.
DownloadLinks	It goes through Wget output file and checks the links that have been downloaded.
SearchthisURL	Using DownloadThread.url, this function manages the search.
	Cont. on next page

Functions	**Description**
Run	This function does the following: 1. Begins with links specified by the user and process download queue. 2. Calls Wget to download google search page. Extracts the links. 3. Converts html to xml using HTMLCleaner. 4. Builds DOM out of XML file. 5. Goes through the links and adds them to the queue.

Appendix E

Functions and their Actions for Maintain-Repository Algorithm : Java Implementation

Functions	**Description**
showContent	Content of the directory will be shown.
filetime	To get the file information.
filesize	To get the file size information.
glob	Returns the file names into an array and supports pattern matching so it can be very easy to find files in a particular directory.
Checkfile	Checks the file type.
Post	To upload content in the directory.
Readdir	To list all the contents of the directory.
check-move	Checks if uploads are allowed in the directory.
array	Sort by date.

Appendix F

Awk Script for Pre-Use Algorithm in Web Usage Mining

Program 1

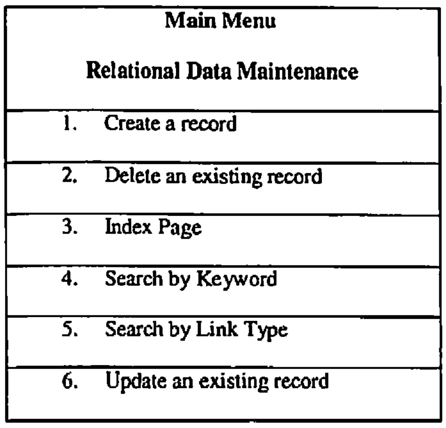
```
BEGIN \{ \}
/Search URL:/ \{
  split($0, arr, "&");
  keyword = arr[3];
  split (keyword, arr, "=");
  keyword = arr[2];
\}

/Total time taken/ \{
  split($0, arr, "=");
  ttime = arr[2];
\}

/Total download size/ \{ split ($0, arr, "=");
     dsize = arr[2];
     print keyword, "~", ttime, "~", dsize, "\\n";
 \}
```

```
END \{
}

-------------------

Program 2

 BEGIN \{
 \}
 /Search URL:/ \{
   split($0, arr, "&");
   keyword = arr[3];
   split (keyword, arr, "=");
   keyword = arr[2];
   flag = 1;
 \}

 /Title:/ \{ delim = ":";
   split($0, arr, delim);
   title = arr[2];
 \}

 /Link:/  \{ split ($0, arr, " ");
   link = arr[2];
 \}

 /File extension:/ \{ split ($0, arr, ":");
     ext = arr[2];
 \}
```

```
/File moved to/ \{ sub(/to/, "_", $0);
     split ($0, arr, "_");
     fmovedto = arr[2];
\}

/File size/ \{ n = split ($0, arr, "~");
     fsize = arr[2];
     if ( flag == 1 ) \{
        print keyword, "^", title, "^", link, "^", ext, "^", fmovedto, "
^", fsize, "\\n";
        flag = 0;
     \} else \{
        print "\\t", "\\t", title, "^", link, "^", ext, "^", fmovedto, "^"
, fsize, "\\n";
     \}
 \}

END \{
\}
```

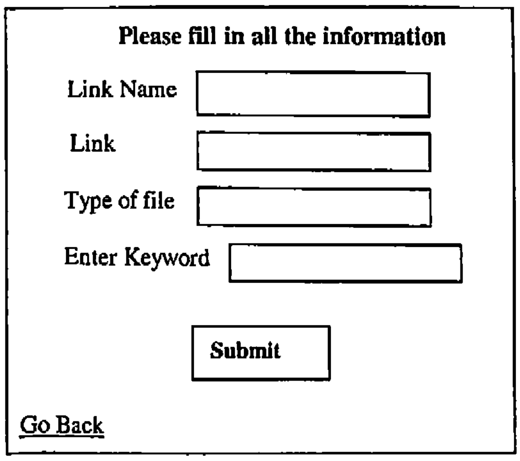
Appendix G

Layout of forms/menus for web based tool for Web user and Administrator



Main Menu Relational Data Maintenance
1. Create a record
2. Delete an existing record
3. Index Page
4. Search by Keyword
5. Search by Link Type
6. Update an existing record

Figure G.1: Main Menu for the Proposed Academic Search Application



Please fill in all the information

Link Name	
Link	
Type of file	
Enter Keyword	

Figure G.2: Insert a Record(for Interactive mode)

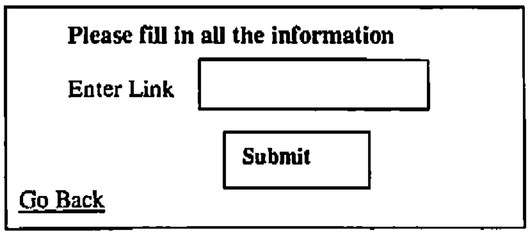


Figure G.3: Delete an existing Record

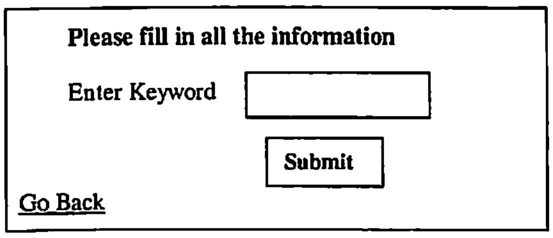


Figure G.4: Search by Keyword

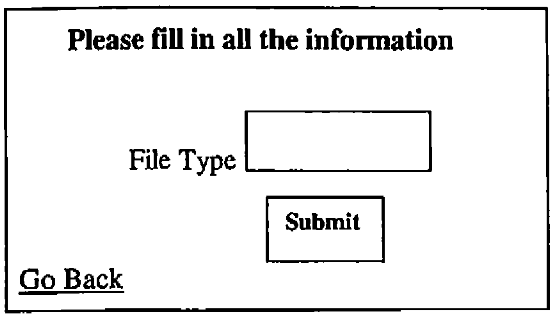


Figure G.5: Search by File Type

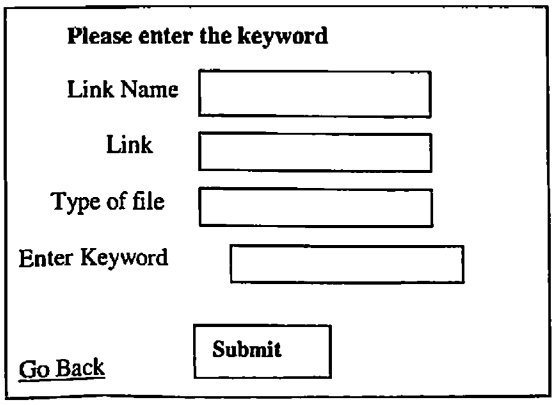
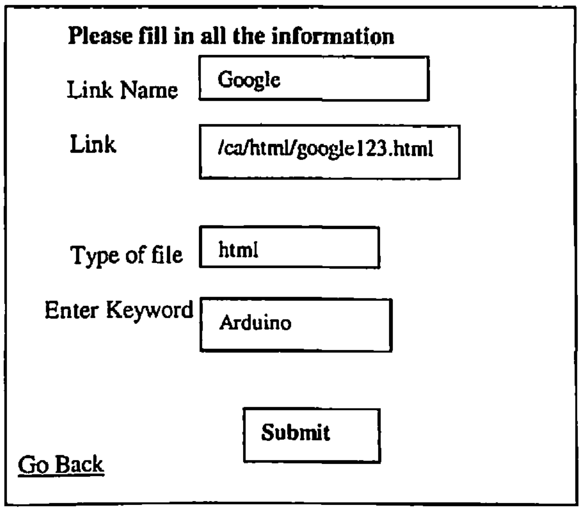


Figure G.6: Update an existing record


Please fill in all the information

Link Name	Google
Link	/ca/html/google123.html
Type of file	html
Enter Keyword	Arduino

Figure G.7: Record Insertion

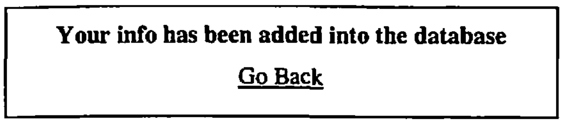
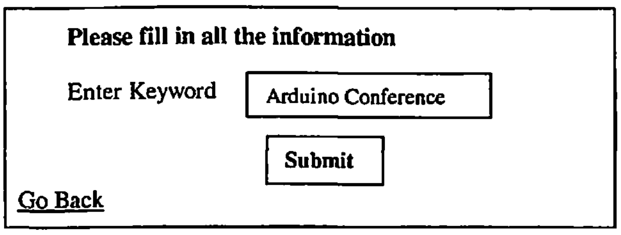
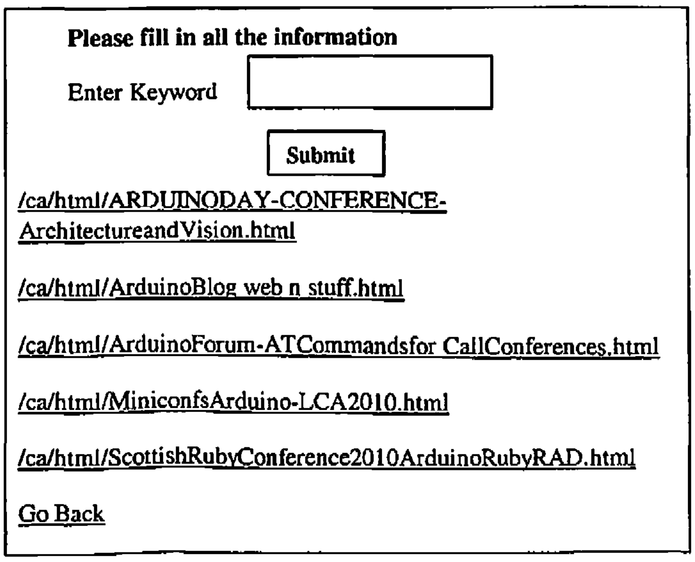

Your info has been added into the database

Figure G.8: Page after Creation


Please fill in all the information

Enter Keyword: Arduino Conference

Figure G.9: Enter Keyword


Please fill in all the information

Enter Keyword

/ca/html/ARDUINODAY-CONFERENCE-ArchitectureandVision.html

/ca/html/ArduinoBlog web n stuff.html

/ca/html/ArduinoForum-ATCommandsfor CallConferences.html

/ca/html/MiniconfsArduino-LCA2010.html

/ca/html/ScottishRubyConference2010ArduinoRubyRAD.html

Figure G.10: List of links by searching Keyword

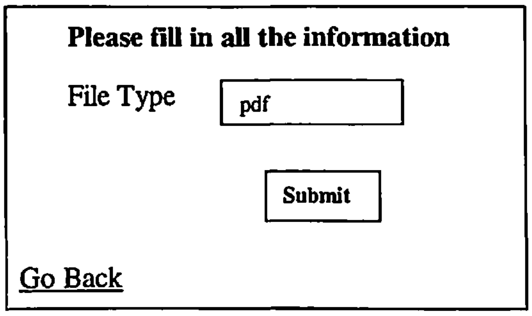


Figure G.11: Enter file type

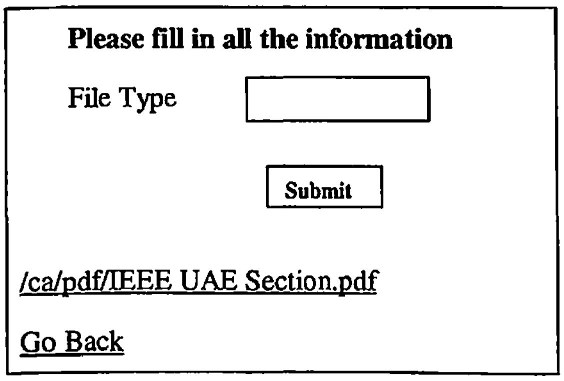

Please fill in all the information
File Type
Submit
/ca/pdf/IEEE UAE Section.pdf
Go Back

Figure G.12: List of links by searching through type

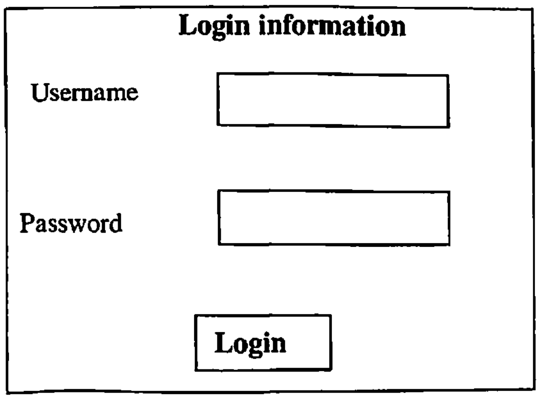
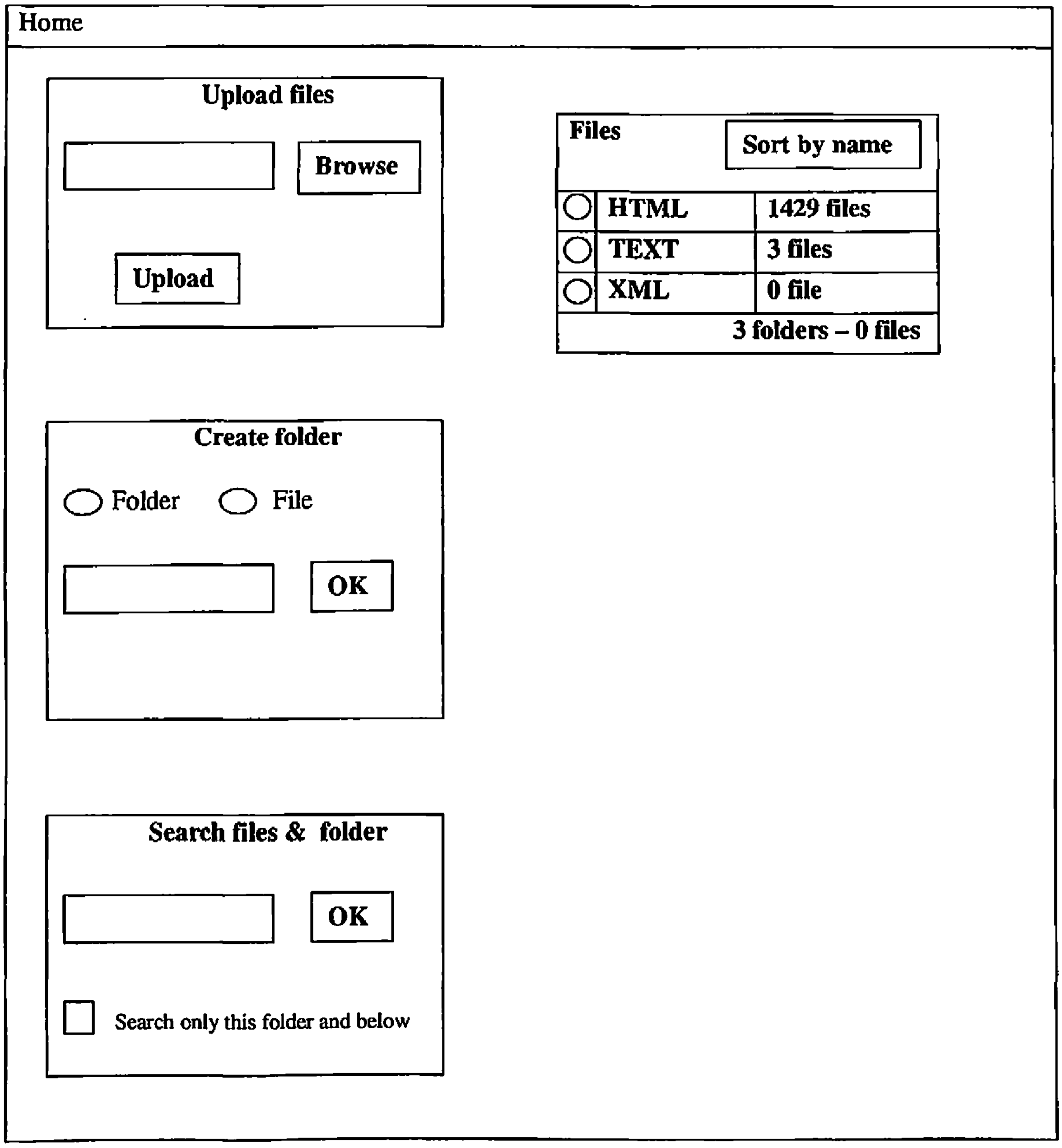
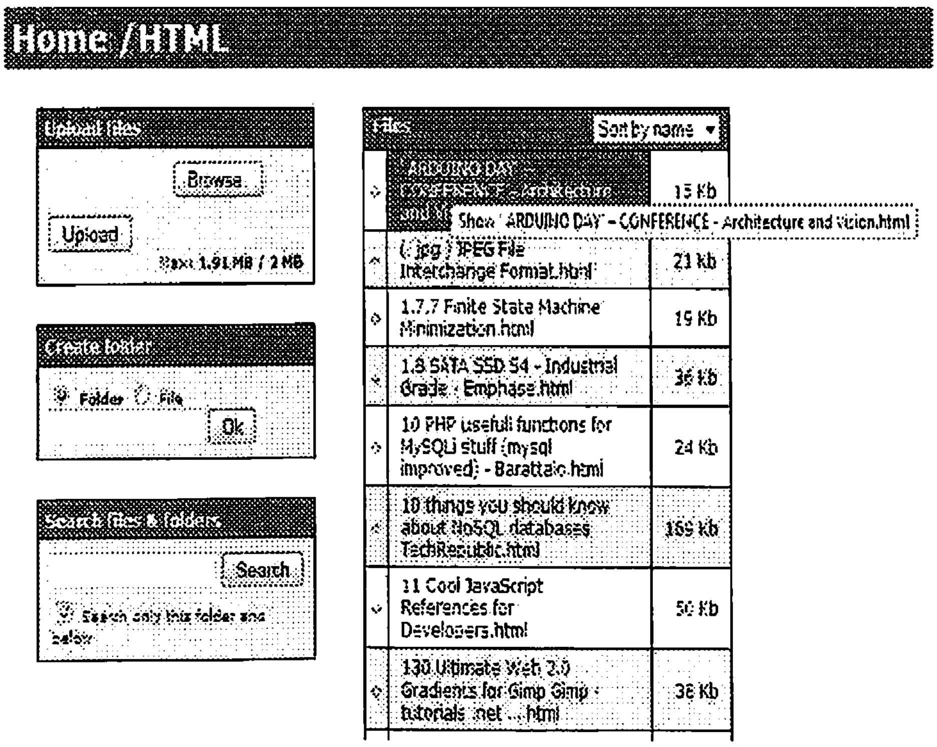


Figure G.13: Maintain-Repository: Authentication page



○	HTML	1429 files
○	TEXT	3 files
○	XML	0 file
		3 folders – 0 files

Figure G.14: Maintain-Repository: Home page


Home /HTML

Upload files

Max: 1.91 MB / 2 MB

Create folder

Folder [✓ Folder] File

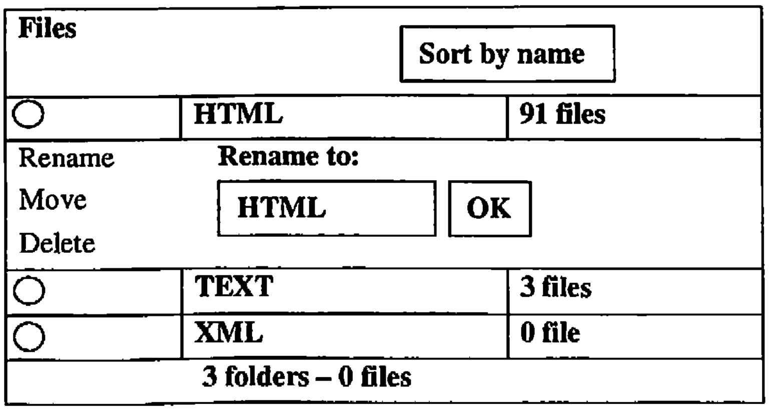
Search files & folders

Search only this folder and below

Files	Sort by name
ARDUINO DAY – CONFERENCE - Architecture and vision.html	15 Kb
(.jpg) JPEG File Interchange Format.html	21 Kb
1.7.7 Finite State Machine Minimization.html	19 Kb
1.8 SATA SSD S4 - Industrial Grade - Emphase.html	36 Kb
10 PHP usefull functions for MySQLi stuff (mysql improved) - Barattalo.html	24 Kb
10 things you should know about NoSQL databases TechRepublic.html	169 Kb
11 Cool JavaScript References for Developers.html	50 Kb
130 Ultimate Web 2.0 Gradients for Gimp Gimp - tutorials .net ....html	38 Kb

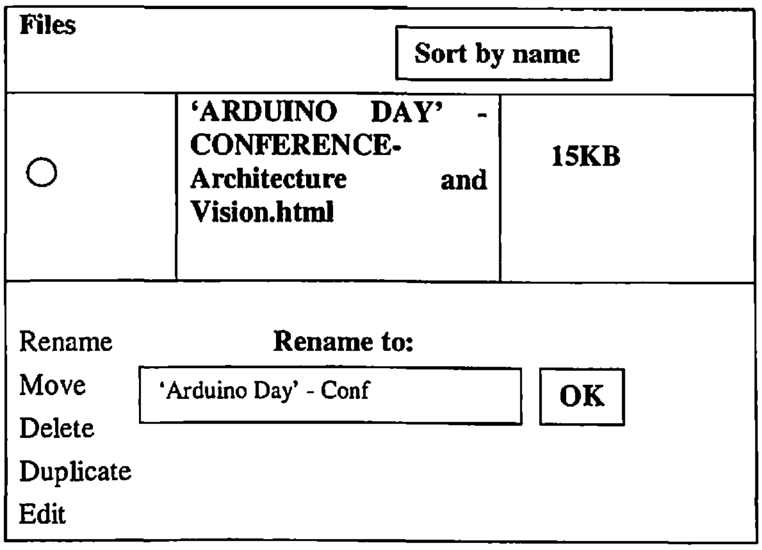
Show 'ARDUINO DAY' – CONFERENCE - Architecture and vision.html

Figure G.15: Maintain-Repository: HTML/XML/Text page



Files		Sort by name
○	HTML	91 files
Rename Move Delete	Rename to: HTML OK	
○	TEXT	3 files
○	XML	0 file
	3 folders – 0 files	

Figure G.16: Maintain-Repository: Folder options



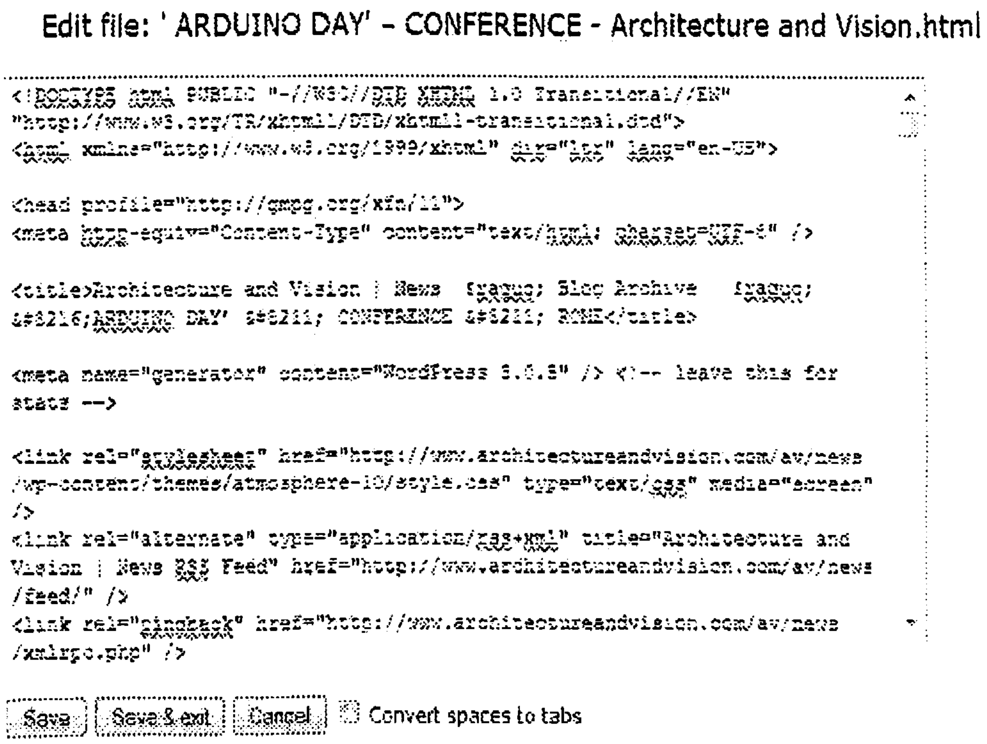
Files		Sort by name
○	'ARDUINO DAY' - CONFERENCE- Architecture and Vision.html	15KB

Rename
Move
Delete
Duplicate
Edit

Rename to:

'Arduino Day' - Conf

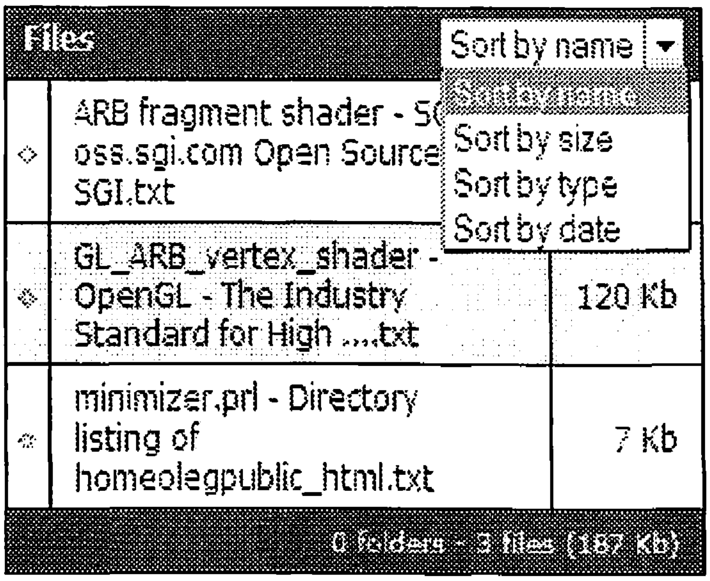
Figure G.17: Maintain-Repository: File options



Edit file: ' ARDUINO DAY' - CONFERENCE - Architecture and Vision.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" dir="ltr" lang="en-US">

<head profile="http://gmpg.org/xfn/11">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Architecture and Vision | News  &raquo; Blog Archive   &raquo;
&#8216;ARDUINO DAY' &#8211; CONFERENCE &#8211; ROME</title>

<meta name="generator" content="WordPress 3.0.5" /> <!-- leave this for
stats -->

<link rel="stylesheet" href="http://www.architectureandvision.com/av/news
/wp-content/themes/atmosphere-10/style.css" type="text/css" media="screen"
/>
<link rel="alternate" type="application/rss+xml" title="Architecture and
Vision | News RSS Feed" href="http://www.architectureandvision.com/av/news
/feed/" />
<link rel="pingback" href="http://www.architectureandvision.com/av/news
/xmlrpc.php" />
```

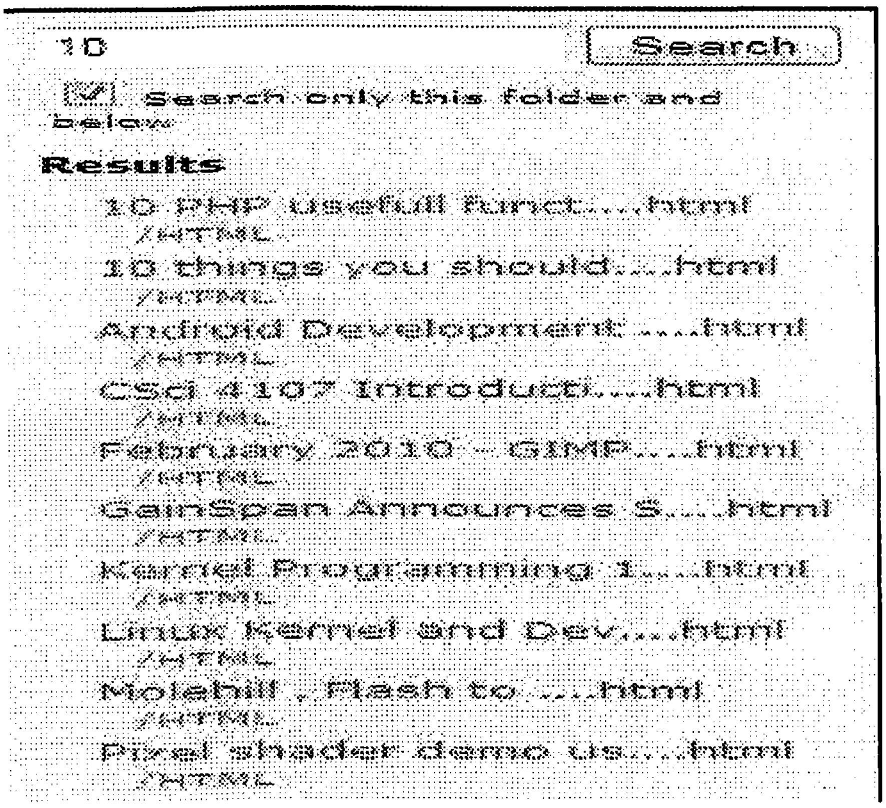
Figure G.18: Maintain-Repository: Edit options



◇	ARB fragment shader - SG oss.sgi.com Open Source SGI.txt	
◇	GL_ARB_vertex_shader - OpenGL - The Industry Standard for High ....txt	120 Kb
◇	minimizer.prl - Directory listing of homeolegpublic_html.txt	7 Kb

0 folders - 3 files (187 Kb)

Figure G.19: Maintain-Repository: Sorting options



10

[✓] Search only this folder and below

Results

10 PHP usefull funct....html
/HTML
10 things you should....html
/HTML
Android Development ....html
/HTML
CSci 4107 Introducti....html
/HTML
February 2010 - GIMP....html
/HTML
GainSpan Announces S....html
/HTML
Kernel Programming 1....html
/HTML
Linux Kernel and Dev....html
/HTML
Molehill , Flash to ....html
/HTML
Pixel shader demo us....html
/HTML

Figure G.20: Maintain-Repository: Search options

Appendix H

List of Publications

Journal Papers

1. Jeyalatha S., Vijayakumar B. "Web Mining Functions in an Academic Search Application". *Informatica Economica,* Vol. 13, No. 3, pp. 132-139, Sep 2009.
2. Jeyalatha S., Vijayakumar B., Anamika Datta. "Interactive Web Data Extraction and Retrieval System using MySQLand PHP in an Academic Environment". *International Journal of Advanced Computing,* Vol. 2, No. 3, pp. 118 - 123, July 2010.
3. Jeyalatha S., Vijayakumar B., Ehtesham A. Hazarika. "Design of an Interface for Page Rank Calculation using Web Link Attributes Information". *Informatica Economica,* Vol. 14, No. 3, pp. 142-152, Sep 2010.
4. Jeyalatha S., Vijayakumar B., Munawwar Firoz. "Design and Implementation of a Tool for Web Data Extraction and Storage using Java and Uniform Interface". *International Journal of Computer Applications,* Vol. 22, No. 4, May 2011.

Conference Papers

5. Jeyalatha S., Vijayakumar B., Zainab A. Shikari. "Design Considerations for a Data Warehouse in an Academic Environment", *International Conference on Computer Science, World Academy of Science, Engineering and Technology (WASET),* Paris, France, 2010, Oct 27-29, pp. 27-29.

6. Jeyalatha S., Vijayakumar B., Sahil Dhingra. "Design and Implementation of Data Warehouse Maintenance Operations in an Academic Environment using Java", *International Conference on Computing,* New Delhi, India, 2010, Dec 27-28, pp. 7-12.

7. Jeyalatha S., Vijayakumar B., Sachin K.Thomas. "Design and Implementation of a Web Extraction Tool for Hypertext Documents", in *Proceedings of the 2nd Conference on Technology Management (MCTM-2011),* Dubai, U.A.E, 2011, March 24, pp. 48-53.

8. Jeyalatha S., Vijayakumar B., Gurpreet Singh Wadhwa. "Design and Implementation of Web Based Application for Relational Data Maintenance in an University Environment", in *Proceedings of the 2nd Conference on Technology Management (CTIT 11),* Dubai, U.A.E, 2011, Oct 26-27, pp. 105-112.

9. Jeyalatha S., Vijayakumar B. "Design and Implementation of a Web Structure Mining Algorithm using Breadth First Search Strategy for Academic Search Application", in *Proceedings of the 6th International Conference on Internet Technology and Secured Transactions (ICITST 2011),* Abu Dhabi, U.A.E, 2011, Dec 11-14, pp. 648-654.

Appendix I

Biography of the Candidate

K. Jeyalatha holds a M.E., in Computer Science from Anna University, Chennai, India. She is working as Senior Lecturer, in Department of Computer Science, at BITS Pilani, Dubai Campus since 2005. Earlier she had worked with affiliated engineering colleges of Anna University, as Lecturer during 1998-2004. Her areas of interest include Web Mining, Data Mining and Database Systems. She is a member of Computer Society of India.

Appendix J

Biography of the Supervisor

B. Vijayakumar holds a Ph.D. in Computer Science from BITS Pilani, India in 2001. He has 20 years of University teaching experience in CSE (National Institute of Technology, India and BITS Pilani, Dubai Campus, U.A.E) and 6 years of experience in computer industry. Presently, he is working as Associate Professor, Computer Science, BITS Pilani, Dubai Campus. His areas of interest include Distributed Database Systems, Component Based Software Engineering, Web Mining, Multimedia Systems and Open Source Software Development. He is a member of Professional bodies ISTE (Indian Society for Technical Education), World Enformatica Society and Staff Advisor for Linux User Group, BITS Pilani, Dubai Campus. He is actively involved as organizing and judging committee member in annual students technical event TECHNOFEST at BITS Pilani, Dubai Campus. He has been involved in co-ordination and coaching the students of BITS Pilani, Dubai Campus for annual UAE National Programming Contest since 2005.